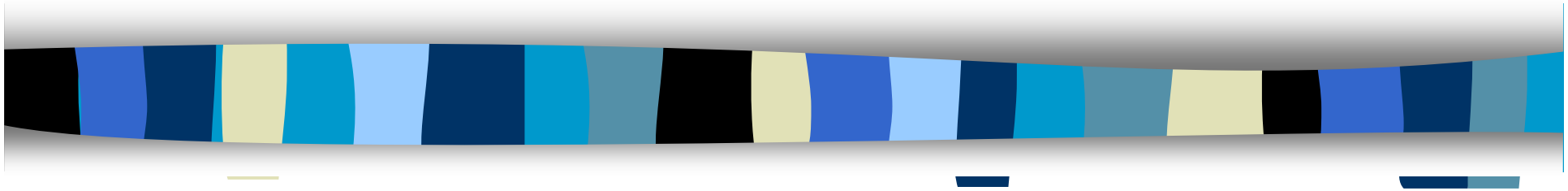
Acceleration of Genetic Algorithms for Sudoku Solution on Many-core Processors



Yuji Sato*[1],
Naohiro Hasegawa*[1],
Mikiko Sato*[2]

*[1]: Hosei University, *[2]: Tokyo University of A&T

Outline

- Background
- Sudoku Solution Accuracy by GA
- Accelerating Genetic Computation with Many-core Processors (GPU/ MCP)
- Evaluation Tests
- Conclusion

Background: Objective

- Evolutionary Computation

  + Parallel Processing

  + Many-core architecture

  → Practical processing time

Bench mark: Sudoku puzzle

- As the first step towards that objective, we take the problem solving Sudoku puzzles and investigate acceleration of the processing with a GPU/ MCP.

The reasons for this approach (1)

- Sudoku puzzles are popular throughout the world.

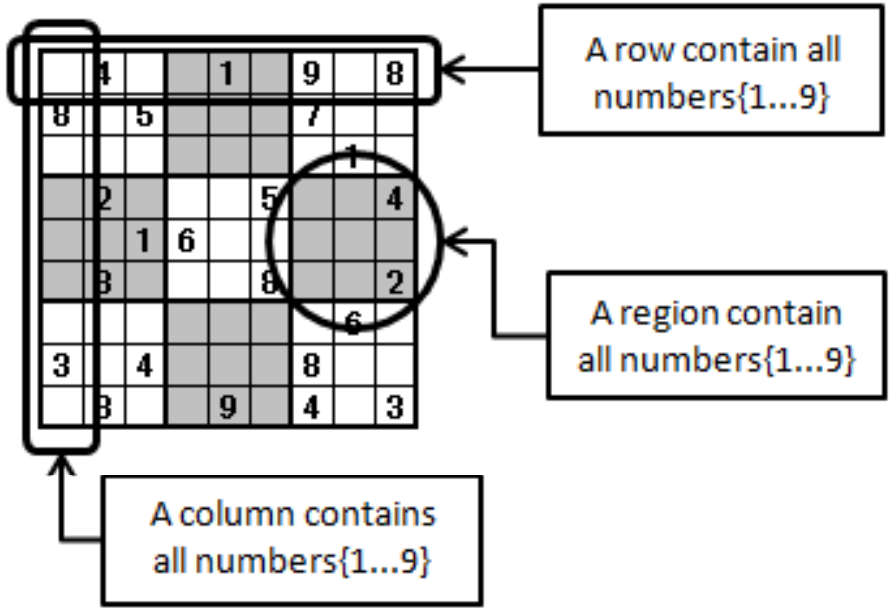
The reasons for this approach (2)

- Genetic computation is suitable for parallelization.
- Therefore, increasing the number of core-processors may make the processing time for GAs equal to that for backtracking algorithms.

The reasons for this approach (3)

- GPUs are designed for the processing of computer graphics in games.
- But, research on General-Purpose computation on Graphics Processing Units (GPGPU) has begun, and GPUs can be used to support solving a logical game.

Sudoku Solution by GA:
An example of Sudoku puzzles



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

Fig. 1. An example of Sudoku puzzles, 24 positions contain a given number, the other position should be solved. A Sudoku puzzles is completed by filling in all of the empty cells with numerals 1 to 9.

Research Example

conventional design of the chromosome

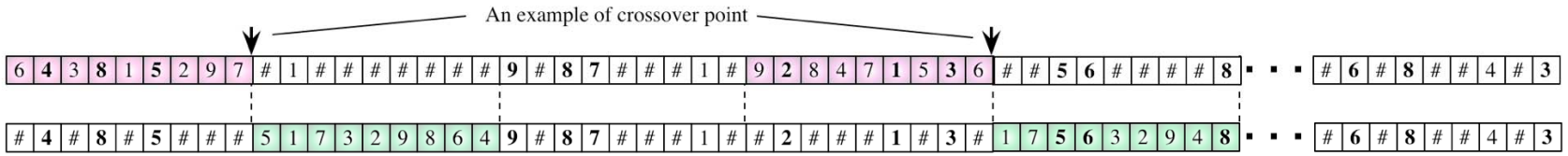


Fig. 4. An example of conventional design of the chromosome and the crossover operation. The chromosome is defined as one-dimensional array of 81 numbers that is divided into nine sub blocks and the crossovers points can only appear between sub blocks.

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

The problem addressed here

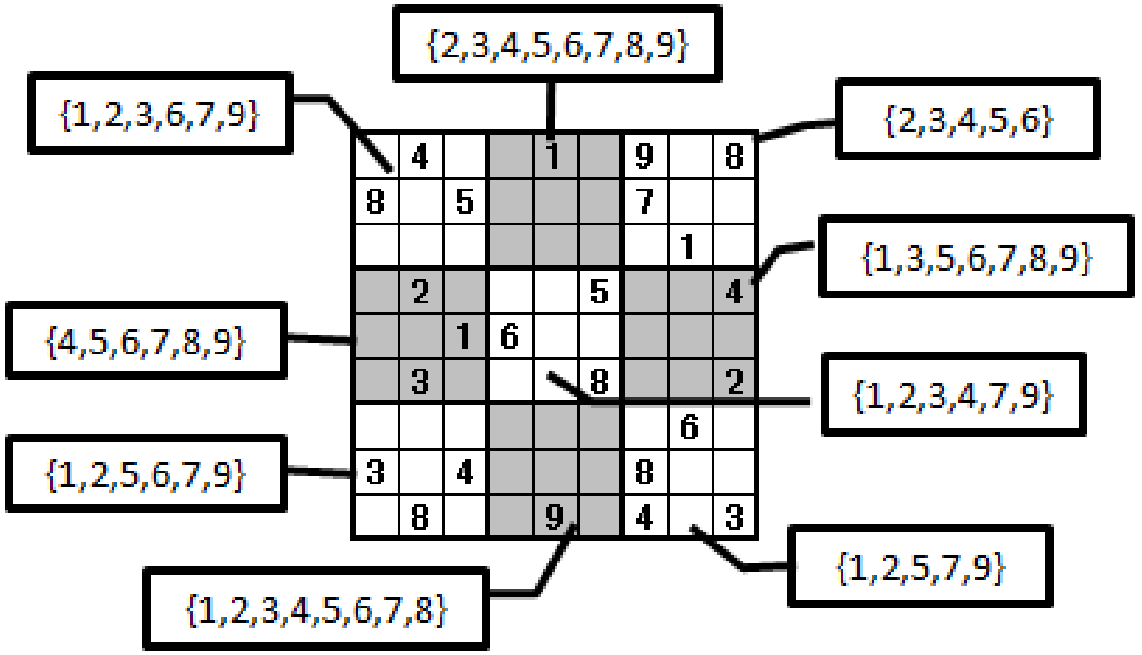
- This design generate chromosomes comprises highly fit schema of long length that is constructed from cell rows or columns in sub-blocks, and this highly fit schemata (BBs) are easy to be destructed by the crossover operation.

Basic Concept

- Genetic operations that emphasize preservation of BB.
- Improve local search function.

Method of Applying GAs

Definition of Chromosomes



We define this 9 x 9 two-dimensional array as the GA chromosome. Fill in the cell that do not contain given values with random numerals.

The fitness function

$$f(x) = \sum_{i=1}^{9} g_i(x) + \sum_{j=1}^{9} h_j(x)$$

$$g_i(x) = \left|x_i\right| \quad h_j(x) = \left|x_j\right|$$

The score is the number of different elements in a row (g_i) or column (h_j), and the sum of the row and column scores is the score for the individual.

The fitness function

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

7
8
8
23

9
9
9
27

Score of the row that constitute the sub-blocks

Crossover

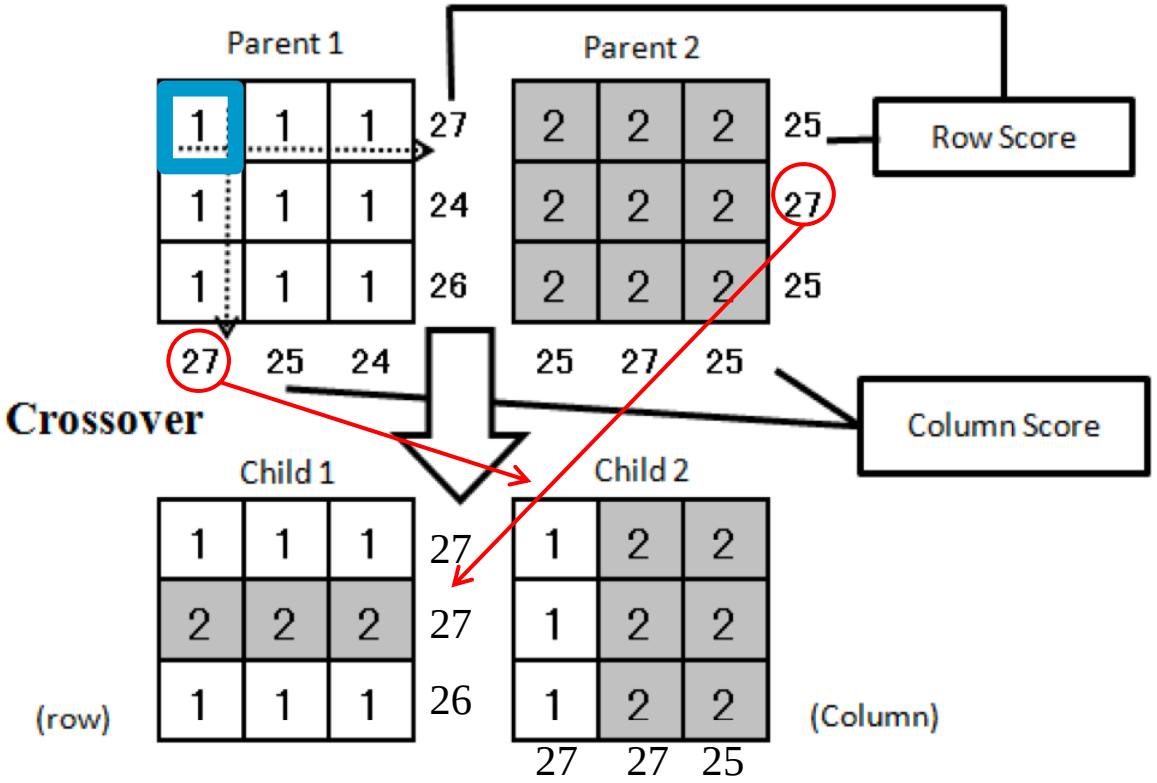


Fig. 3. An example of the crossover considered the rows or the columns that constitute the sub-blocks. The child inherits the ones with the highest score.

Mutation

Swap mutation inside the sub block

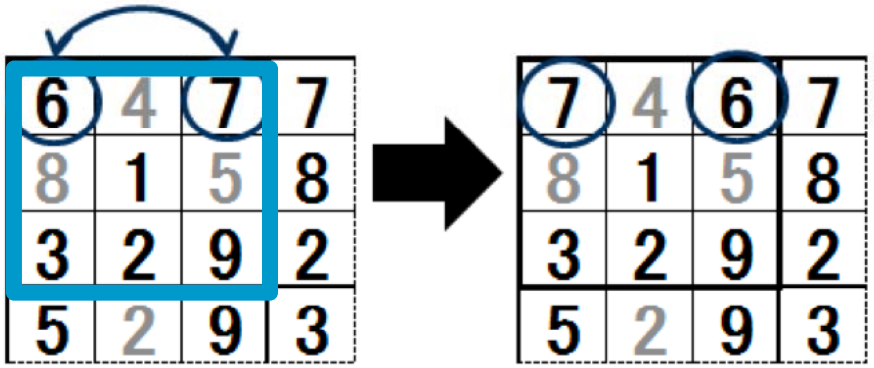
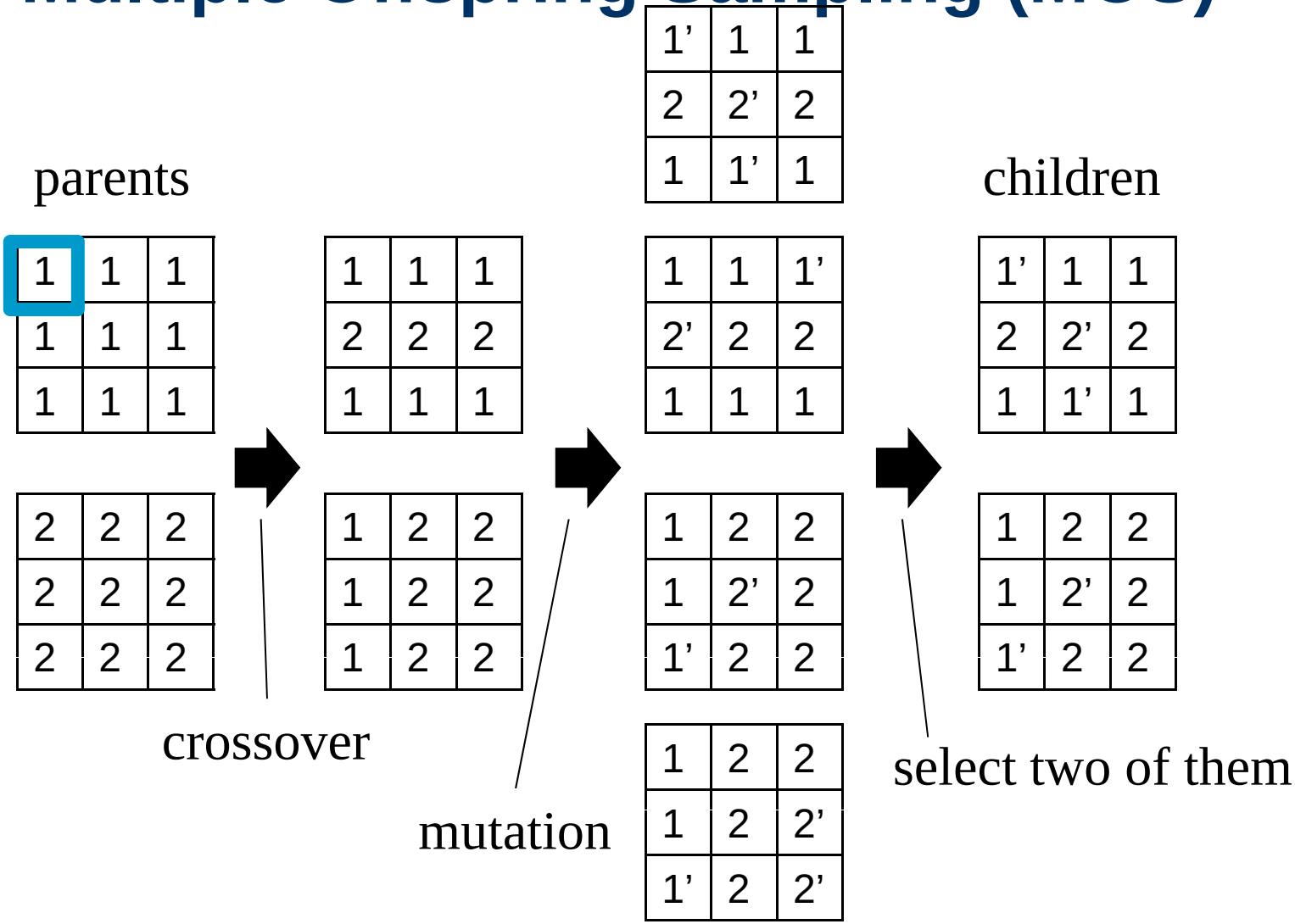


Fig. 5. An example of the swap mutation. Two numbers inside the sub block are selected randomly if the numbers are free to change.

Local Search:
Multiple Offspring Sampling (MOS)



The experimental parameters

- [Population size] 150
- [Number of child candidates/Parents] 2
- [Crossover rate] 0. 3
- [Mutation rate] 0. 3
- [Tournament size] 3

Evaluation Experiments

The puzzles used for evaluations

- We selected two puzzles from each level of difficulty in the puzzle set from a book.
- For comparison with the conventional examples, we also used the particularly difficult Sudoku puzzles introduced by Timo Mantere in reference.

The puzzles used for evaluations - Easy level

No.1(38)

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

No.11(34)

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

The puzzles used for evaluations - Intermediate level

No.27(29)

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

No.29(30)

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

The puzzles used for evaluations - Difficult level

No.77(28)

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

No.106(24)

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

The puzzles used for evaluations - Super Difficult level

24

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

23

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

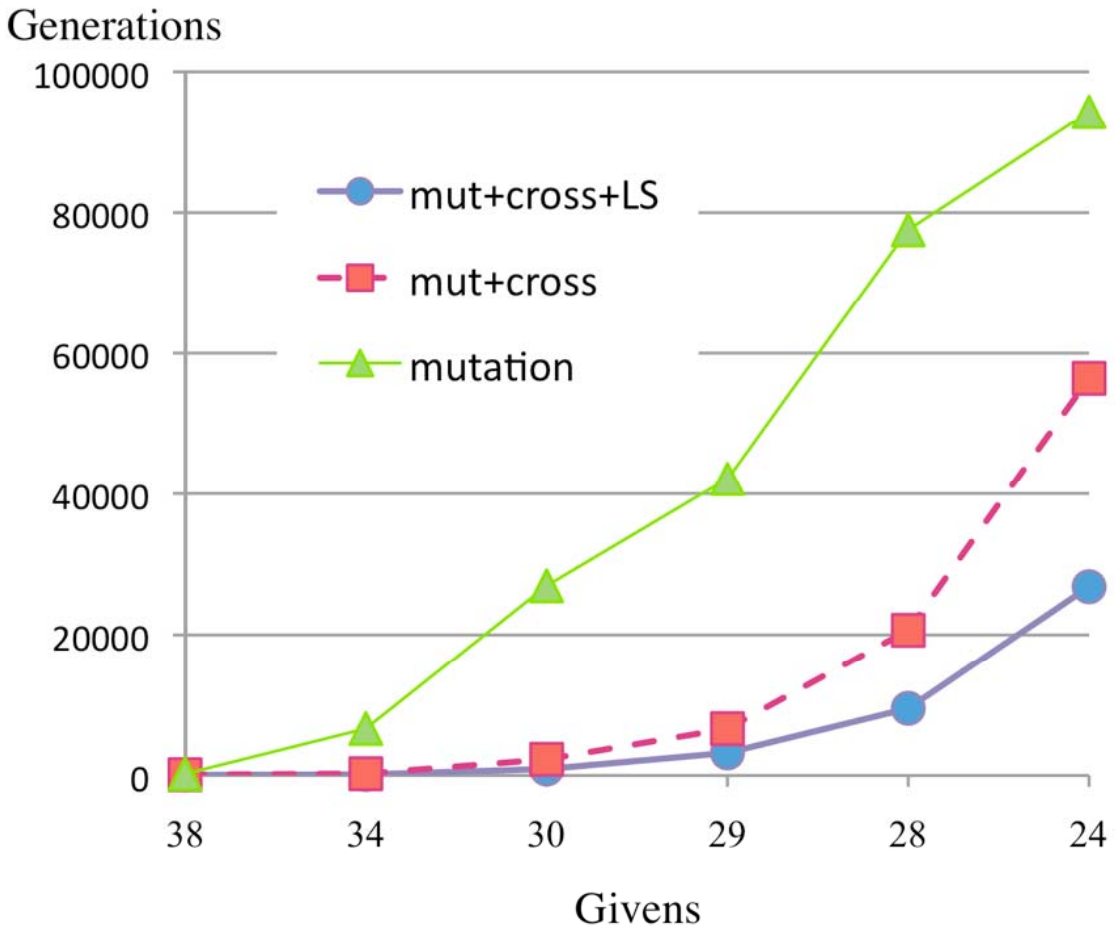
22

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

Super difficult Sudoku's. Available via WWW: http://lipas.uwasa.fi/~timan/sudoku/EA_ht_2008.pdf#search='CT20A6300%20Alternative%20Project%20work%202008' (cited 8.3.2010).

Experimental results

Benchmark test



Experimental results

Benchmark test

Table. 1 The comparison of how effectively GA finds solutions for the Sudoku puzzles with different difficulty ratings.

Difficulty rating	Givens	mut+cross+LS		mut+cross		Swap mutation	
		Count	Average	Count	Average	Count	Average
Easy (No. 1)	38	100	62	100	105	100	223
Easy (No. 11)	34	100	137	100	247	96	6627
Medium (No. 27)	30	100	910	100	2274	86	26961
Medium (No. 29)	29	100	3193	100	6609	66	42141
Difficult (No. 77)	28	100	9482	100	20658	35	77573
Difficult (No. 106)	24	96	26825	74	56428	9	94314

Experimental results

Comparison with previous research

Table. 3 Our result and the result represented in [7]

Sudoku puzzle	Our proposed GA 100, 000 trials	Mantere-2008 [7] 100, 000 trials
Al Escarcot	83 /100	5/100

[Population size] *1: 150, *2: 11

Our approach: GA (+ Local Search)
Mantere etc. : GA + Cultural Algorithm

Comparison with previous research

	Improve efficiency	Speed up
Mantere etc.	Cultural Algorithm (CA)	Small population size
Our GA	Properly GA design + LS	Parallel processing on GPU

Experimental results

- The results show the proposed genetic operation was relatively improved the optimum solution rate.
- On the other hand, the processing time was still completely poor compared to the backtracking algorithm.

Accelerating Genetic Computation with GPU: GTX460 specifications

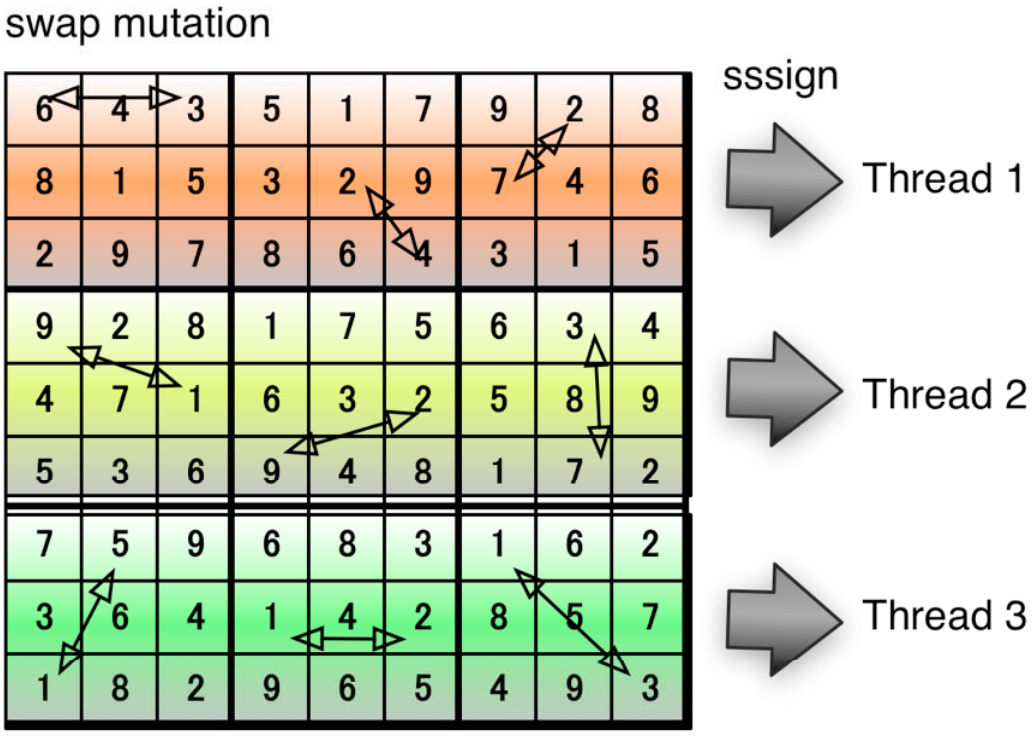
Board	ELSA GLADIA GTX460
#Core	336 (7 SM X 48 Core / SM)
Clock	675 MHz
Memory	1 GB
Shared memory / SM	48 KB
#Register / SM	32768
#Thread / SM	1024

The parallelization of genetic computing must be implemented with full consideration given to the feature.

Parallel processing for individuals

- The genetic computing programs running in the SMs using threads are executed in parallel, and the execution of the same program in each SM with different initial values is considered to serve as a measure against initial value dependency.

Parallel processing for genetic manipulation



An example of the swap mutation within a sub-block and the thread assignment.

Estimated execution time

- Single-core：

  T_{exe} x N x G

- Parallel processing for individuals：

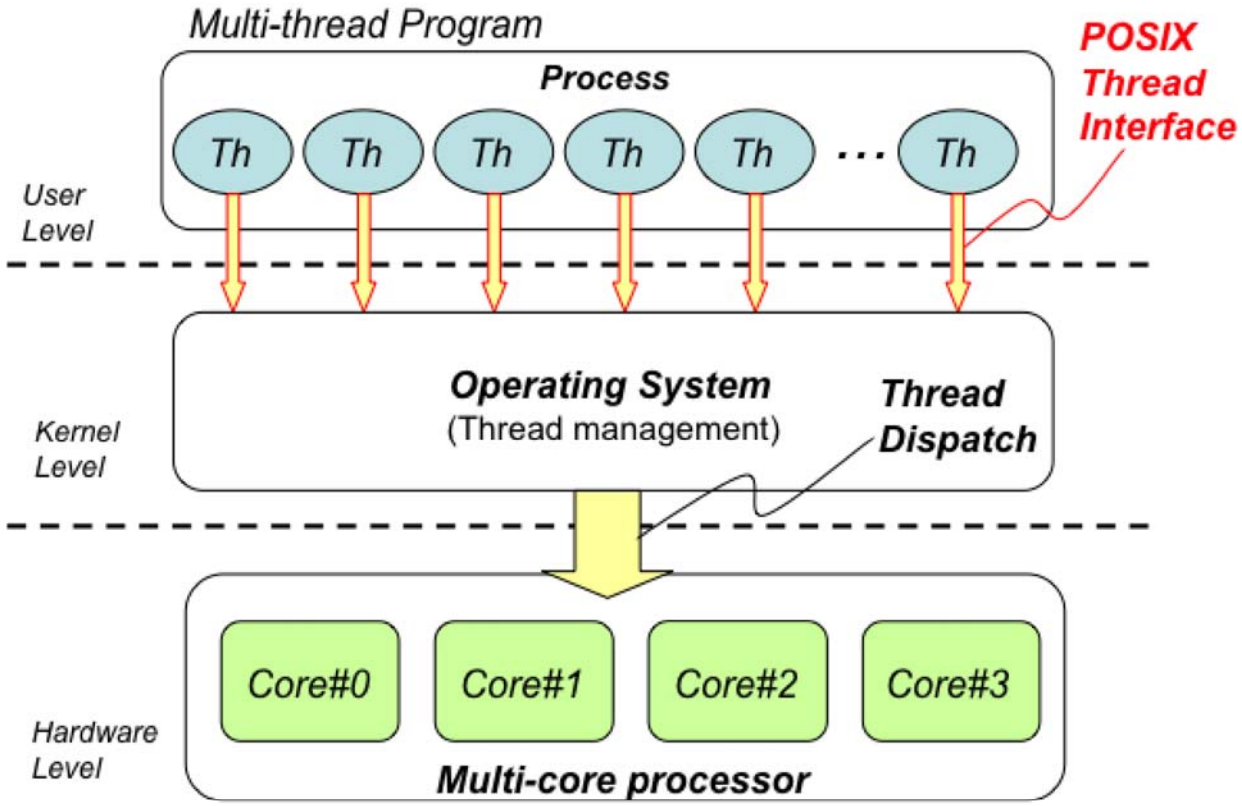
  T_{exe} x N/α x G $\quad (48 < \alpha < N)$

- Parallel processing for manipulation ：

  $[\ (1 - k) + k/\beta]\ T_{exe}$ x N/α x G

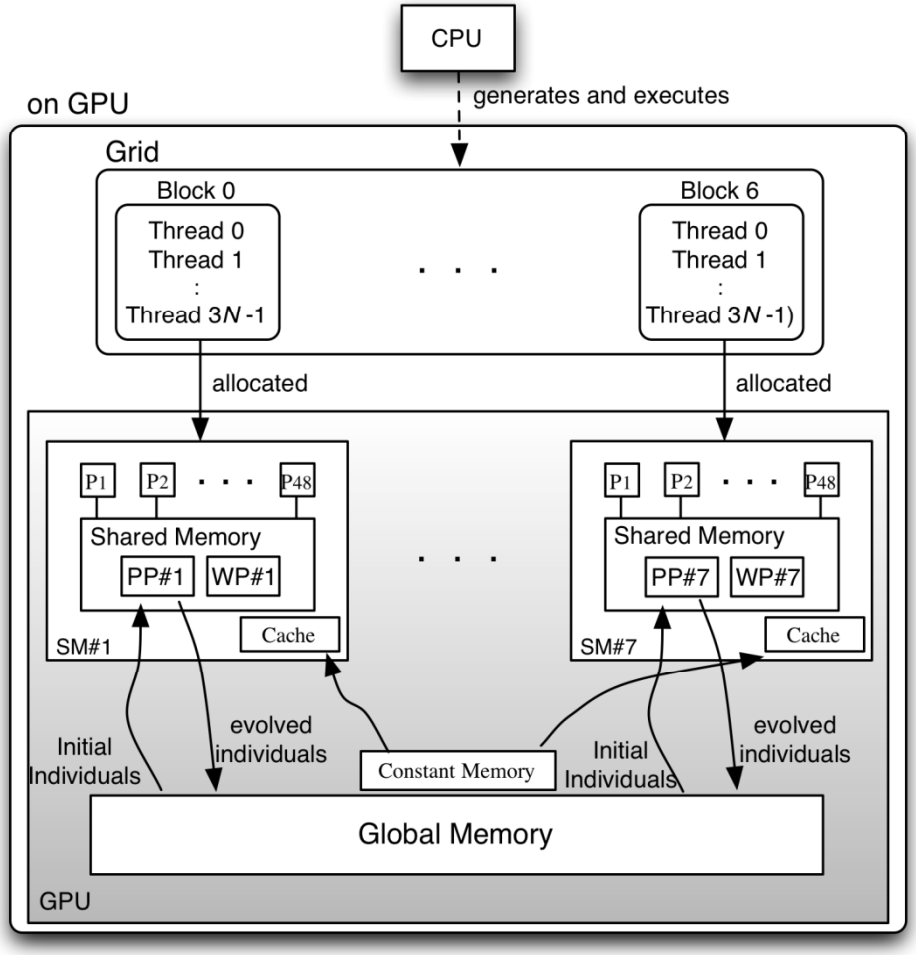
  $(0 < k < 1,\ 0 < \beta < 3)$

The system architecture for multi-core processors



(Intel Core i7)

Accelerating Genetic Computation with GPU



7 blocks / grid
3 x *N* threads / block

Evaluation Tests:
Execution Environment

CPU	MCP: Intel Corei7 920 (2.67GHz, 4 cores) GPU: Phenom Ⅱ X4 945 (3 GHz, 4 cores)
OS	Ubuntu 10.04
C Compiler	gcc 4.4.3 (optimization " –O3")
CUDA Toolkit	3.2 RC

Evaluation Tests: Test Data

- The evaluation results for problems classified as Super Difficult-1 (SD1), Super Difficult-2 (SD2), and Super Difficult-3 (SD3).

  (Super difficult Sudoku's. Available via WWW: http://lipas.uwasa.fi/~timan/sudoku/EA_ht_2008.pdf#search='CT20A6300%20Alternative%20Project%20work%202008' (cited 8.3.2010).)

Evaluation Tests: Acceleration Effect

- Table 6. The acceleration effect of using the GPU/MCP (SD2, Givens: 23)

	Count [%]	Average Gen.	Execution time	
Java	83	45,468	7m 50s 678	x 74
C	86	44,250	1m 26s 320	x 14
Core i7 #Thread: 8	100	5,992	12s 12	
GTX460 #SM: 7	97	22,142	6s 391	

Cutoff set: 100,000 generations, Population size: 150

Evaluation Tests: Minimum Time (GPU)

- Table 13. The minimum numbers of generations and the execution times required to solve SD1 through SD3

Sudoku	Minimum Gen.	Execution time
SD1	83	25 ms
SD2	158	47 ms
SD3	198	76 ms

Evaluation Tests: Scalability (MCP)

- Table 7. The number of generations until the correct solution was obtained, the execution time, and the rate of correct answers (SD2, Givens: 23)

	Count [%]	Average Gen.	Execution time
#Th: 1	82	42,276	28s 41
#Th: 2	98	25,580	22s 48
#Th: 4	100	13,261	21s 47
#Th: 8	100	5,992	12s 12

Evaluation Tests: Scalability (GPU)

- Table 10. The number of generations until the correct solution was obtained, the execution time, and the rate of correct answers (SD2, Givens: 23)

	Count [%]	Average Gen.	Execution time
#SM: 1	50	70,067	20s 199
#SM: 2	69	58,786	16s 958
#SM: 3	82	41,757	12s 630
#SM: 4	93	31,254	9s 260
#SM: 5	95	28,709	8s 287
#SM: 6	97	22,065	6s 368
#SM: 7	97	22,142	6s 391

Evaluation Tests:
Appropriate Population Size (GPU)

- Area for individual data：

  1 byte (char) x 81 x *N* x 2
- Area for selection： 4 bytes (int) x *N*
- Area for crossover： 4 bytes (float) x *N*/2
- Area for mutation： 1 byte (char) x 81*N*

  → Total: 249*N*
- Maximum number of *N* which can be stored in the 48 KB shared memory： 192

Evaluation Tests:
Appropriate Population Size

- Table 14. The execution time and the correct solution rates for when the number of individuals is set to 192.

Sudoku	Count [%]	Average Gen.	Execution time	
SD1	100	9072	2s 751	- 5%
SD2	100	13,481	4s 530	- 29%
SD3	100	22,799	6s 862	- 21%

Evaluation Tests:
Appropriate Population Size (MCP)

Table 12. The result on increasing the number of individuals (SD2)

	Count [%]	Ave. Gen.	Exec. Tim	Best Gen.
100	100	8,641	11s 63	644
150	100	5,992	12s 12	243
200	100	7,115	19s 20	229
300	100	9,441	38s 29	123
400	98	15,441	84s 76	86

MCP v.s. GPU

- These experiments show that the GPU can find solutions faster than the multi-core processor by making use of a higher degree of parallelization.

MCP v.s. GPU

- At the same time, it is more difficult to use a GPU than a multi-core processor which can execute programs in parallel without having to worry about limitations in number of threads or shared memory capacity.

Conclusion

- We have used the problem of solving Sudoku puzzles to show that parallel processing of genetic algorithms in a many-core processor can solve difficult problems in practical time.

Conclusion

- Specifically, we implemented parallel genetic computing on the NVIDIA GTX 460 and Intel Core i7, and showed that execution acceleration factors of from 7 to 25 relative to execution of a C program on a CPU are attained and a correct solution rate of 100% can be achieved, even for super-difficult problems.

Future works

- We want to try another parallel GA implementation on many-core processors.
- We need to investigate another approach to avoid initial value dependency.
- We want to show that EC (+ GPU) can solve super difficult Sudoku puzzles in one second.

- Thank you for your attention!