Sudoku Using Parallel Simulated Annealing

Zahra Karimi-Dehkordi, Kamran Zamanifar, Ahmad Baraani-Dastjerdi, and Nasser Ghasem-Aghaee

Department of Computer Engineering, University of Isfahan (UI), P.O. Code 81746-73441, Hezar Jerib Avenue, Isfahan, Iran
{zahra.karimi,zamanifar,ahmadb,aghaee}@eng.ui.ac.ir

Abstract. Parallel Simulated Annealing was applied to solving Sudoku puzzle. Simulated annealing is a stochastic search strategy which is best known for not getting trapped at the local optimums. Although SA suffers from low efficiency, it has been recognized as one of successful solutions in Sudoku. Sudoku puzzles in this study were formulated as an optimization problem in multi-agent environments. Variants of parallel SA could successfully solve this optimization problem. In this paper we implemented 3 different parallel SA in JADE and compared them. The results show that parallel search with periodic jumps gets better efficiency and success rate.

Keywords: Sudoku, Parallel Simulated Annealing, Multi-Agent System.

1 Introduction

Sudoku, which is Japanese term meaning "singular number," has gathered international popularity. The Sudoku puzzle consists of 9 × 9 grid of 81 cells which partitioned in 9 blocks of 3*3. (See Fig 1) Each puzzle has some cells that have already been filled in. The objective of the puzzle is to fill in the remaining cells with the numbers 1 through 9 so that the following three rules are satisfied:

- Each horizontal row should contain the numbers 1 - 9, without repeating any.
- Each vertical column should contain the numbers 1 - 9, without repeating any.
- Each 3 × 3 block should contain the numbers 1 - 9, without repeating any.

Researchers have applied variety of methods to solve the Sudoku puzzle. As this puzzle can be modeled as an optimization problem, several stochastic search techniques have been used in that. Perez [1] has solved Sudoku using variations of PSO, Simulated Annealing and Genetic algorithm and concluded that combination of Simulated Annealing and Genetic algorithm is the best one. He hast analyzed that the parallel nature of genetic algorithm with flexibility of Simulated Annealing is the reason. In this paper we are applying parallel simulated annealing in 3 different variants and comparing the results.

The rest of this paper is organized as follows. In the next section Sudoku problem is presented, then applying simulated annealing on Sudoku is explained and parallel simulated annealing algorithms are reviewed. At last our algorithm and implementation points are presented.

Y. Tan, Y. Shi, and K.C. Tan (Eds.): ICSI 2010, Part II, LNCS 6146, pp. 461–467, 2010.

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

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

Fig. 1. Sudoku puzzle (left), solved Sudoku puzzle (right)

2 Sudoku Using Simulated Annealing

Simulated Annealing (SA) is an optimization technique which finds the optimal point by running series of moves under different thermodynamic condition. During each move, a neighboring state is determined by randomly changing the current state of the individual. The new state is evaluated via a cost function, if a state with a lower cost is found then the individual moves to that state. Otherwise, if the neighboring state has a higher cost then the individual will move to that state only if an acceptance probability condition is met. If it is not met, then the individual remains at the current state. The acceptance probability is proportional with a temperature which is redoing during run. High temperature increases the chance of worse move. A variant of SA called homogenous SA groups moves in markov chains where each Markov chain is generated at a fixed value of t, and t is then altered in-between subsequent chains. [2] The snapshot of this algorithm is shown in Fig 2.

We applied SA at solving Sudoku. Our approach is based on Lewis algorithm[3] who has first applied SA for this puzzle. The algorithm uses a direct representation of candidate solutions. It means each solution is encoded as a matrix. To form initial candidate solution each non-fixed cell gets a random value in such a way that every block contains the values 1 to 9 exactly one. During the run, the neighborhood operator repeatedly chooses two different non-fixed cells in the same block and swaps them. The method of producing initial candidate solution and neighborhood ensures the third rule always met.

The cost function looks at each row and column individually and calculates the number of values between 1 through 9 that are not present. The total cost is simply sum of these 18 cost values. Reevaluating the solutions in each move gets optimized by recalculating at most 2 rows and 2 columns.

Their algorithm takes homogenous variant of SA using markov chain which its length equals with square of the number of non-fixed cells so the maximum of it is 81^2.

Perez[1] uses Quantum Simulated Annealing(QSA) for solving Sudoku. The major difference between QSA and SA is that in SA the temperature determines the probability of moving from one state to the next, while the neighborhood remains

constant. In QSA, the neighborhood radius or the distance between the current state and the neighboring state is decreases when temperature is getting low. In the Sudoku context, Quantum simulated annealing is like simulated annealing with markov chains. But quantum simulated annealing has markov chains with different lengths. However QSA sounds better than SA with markov chains Perez[1] does not compare his result with Lewis[3] one

```
C=Choose an initial solution
T=Choose an initial tempreature
For i = 1 to n
Begin
 For j = 1 to m
 S'= Generate a neighbour of the solution C
 ΔE = Cost(S') - Cost(C)
 If (ΔE>0) then // S' better than C
       C = S'
 Else with probaility Exp (ΔE/T)
       C= S'
 End if
 End for m
T = α * T (α = cooling factor)
End for n
```

Fig. 2. Homogenous simulated annealing algorithm

3 Review of Parallel Simulated Annealing Algorithms

Chen [4] reviewed five strategies of parallelizing SA algorithm. The first approach uses divide and conquer to decompose search space into smaller one. Unpromising sub-spaces are excluded from further search and promising ones are recursively decomposed and evaluated. The second approach is to run several solutions simultaneously; they start with the same random point and synchronize to continue the best one. The third approach further improves the second approach by changing a single-start solution to multistart. The fourth and fifth approaches use a combination of SA with genetic algorithm. In both of them, search agents select an initial configuration randomly and evolve it with genetic operators. Then a synchronizer selects best solution, broadcast it and all continue search in a simulated annealing manner. In the forth approach genetic algorithm just applied at the initial step but in the fifth approach it is applied before every synchronization step to select another best.

Minerals[5] classifies parallel SA differently in 3 categories named low level parallelization, domain decomposition and parallel search. The first one is about parallelizing algorithm for example evaluation part. Domain decomposition corresponds with the second approach in Chen classification and search parallelization is a general category of all others of Chen[4].

So parallel simulated annealing means searching all or different part of search space with a single or muli, improved or produced start point. Our approach uses a combination of all these approaches as explained in the next section.

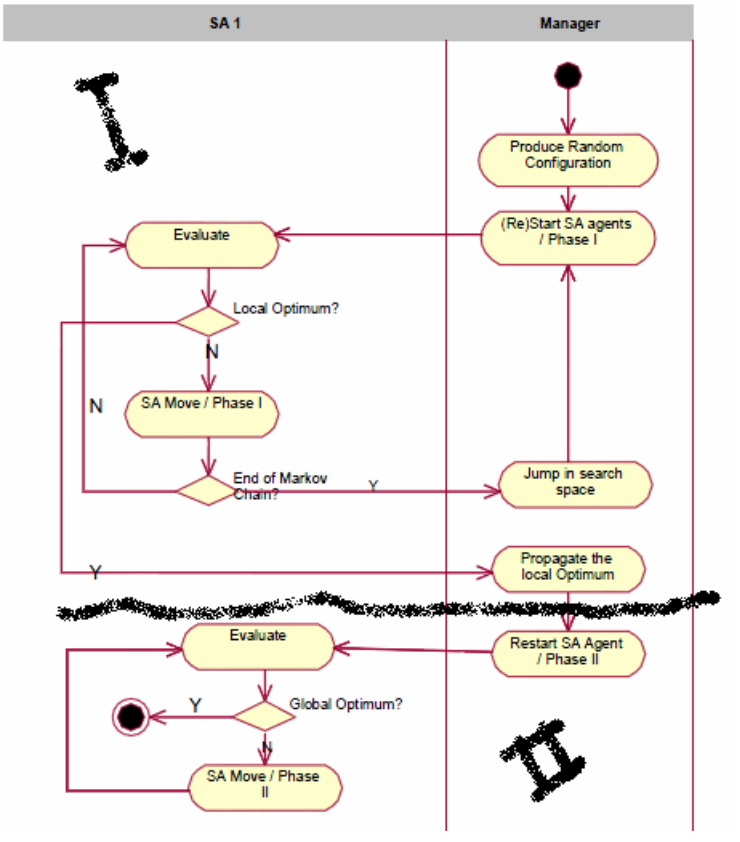

Fig. 3. Domain Composition parallel simulated annealing: The top part presents phase I, and the bottom part shows phase II

4 Sudoku Using Parallel Simulated Annealing

We applied 3 variants of parallel simulated annealing, domain decomposition, parallel search with jumps, independent parallel search. All these variants use single random start configuration.

Domain Decomposition: This 2-phase approach applies a variation of domain decomposition in the first phase and then runs parallel search(Second approach in [4]) during the second phase. Sudoku as a search problem is a permutation one. Decomposing this large permutation space (9^81) sounds impossible. For managing this difficulty, we use 3 parallel search agent each responsible for 3 rows (or blocks) of Sudoku puzzle. It means search agent 1 is moving through the three first rows and search agent 2 is exploring rows 4, 5, 6 and the last search agent works on other rows. Figure 3 shows the activity diagram of presented approach.1.2 Additional Information Required by the Volume Editor.

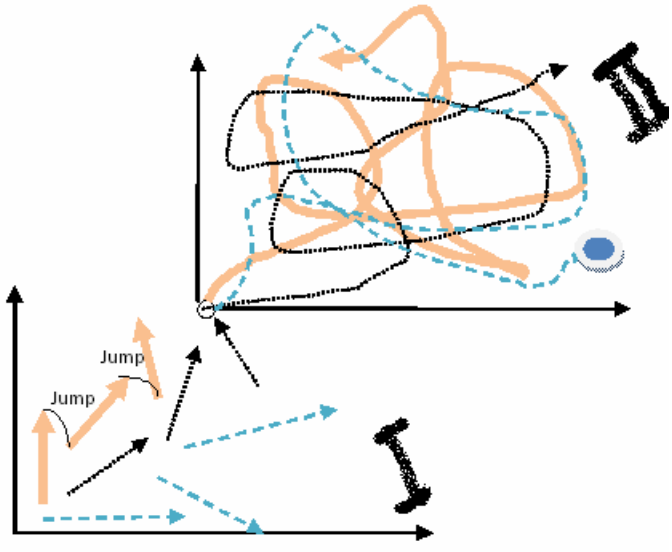

Fig. 4. Moving snapshot of the proposed approach, phase I consists of move-jump of 3 search agents and phase II is parallel search of individual agents

If you have more than one surname, please make sure that the Volume Editor knows how you are to be listed in the author index.

When a search agent is searching through its own part, does not see others' moves. This mechanism is a kind of partitioning but don't cover all the space. We introduce a jump at the end of each markov chain to increase exploration. Jump method is explained in the next section. Fig 4 shows a snapshot of 3 agents moving. This move-jump continues till getting a local optimum. Then second phase starts, local optimum is propagated to all agents and each agent start searching all space independently.

Parallel search with jumps: In this variant we use search agents to search all the puzzle but jumps are used in the search space as previous.

Independent parallel search: In this variant we use search agents without any jumps.

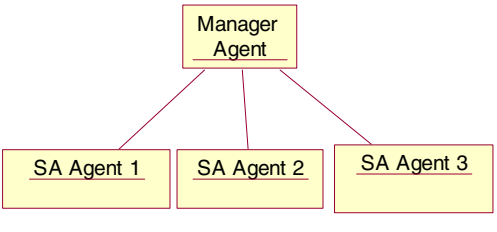

Fig. 5. Architecture of Multi-Agent system, SA Agents interacts with each other indirectly using Manager Agent

5 Experimentation

We has implemented this approach in JADE [6]. It is a mulitagent framework which lets easy implementation of interacting agents. There are 4 agents, one manager and 3 simulated annealing agents. (Fig 5) Manager agent creates and starts SA Agents, manages jumps, propagates found local optimum and maintains the last received search point. SA Agent does evaluating, moving and interacting with manager. Other implementation points are discussed in the following paragraph.

- How jump method is implanted: The manager maintains a point (as a matrix) which is initially filled by produce random configuration. When it receives a new configuration from one of SA agents, It constructs 4 different configurations with swapping their partitions, evaluates them and selects one of them using roulette wheel.
- Simulated Annealing Parameters based on Lewis[3]
  - Cooling Factor(α) = 0.75
  - Number of Markov Chains(n) = 25
  - Length of Markov Chain(m) = (number of unfixed cells) ^ 2
  - Initial temperature(T) = 2.5
- Local Optimum for beginning phase II: Cost = 4

We applied these 3 variants on a Sudoku of Fig 6, each variant runs 30 times. The summary of results has been shown in table 1. The results show that parallel search with jumps has better efficiency, mean, standard deviation an max are the best. Also the last column, success, indicates the number of successful runs which parallel search with jump is the best

Table 1. Summary of results of running algorithm

Time / Variant	Mean	Standard Deviation	Min	Max	Success
Domain Decomposition	23.79167	136.6093	12	58	24
Parallel search with jump	16	13.6161	10	24	28
Parallel Search independently	17.57692	72.0801	9	51	26

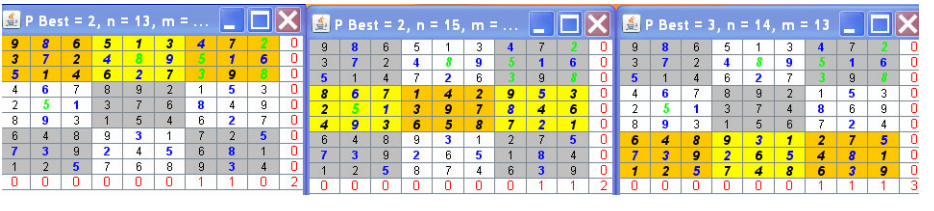

Fig. 6. Snapshot of running program (black numbers are fixed one)

6 Conclusion

In this paper, we have presented parallel version of Simulated Annealing Sudoku Solver[3] to improve its efficiency. However there are lots of parallel Sudoku solvers using genetic algorithm[1], all use parallel search through all the search space. We compared 3 variants of parallel simulated annealing. Domain decomposition makes search space smaller which obviously results in shorter markov chains, but it could not solve Sudoku efficiently. The best one is parallel search through whole space with jumps.

Also the implementation, to our knowledge is the first multi-agent implementation of solving Sudoku which provides synchronization capabilities needed to our approach.

We are going to evaluate the results with more Sudoku sample and multistart variants of parallel simulated annealing.

References

1. Perez, M., Marwala, T.: Stochastic optimization approaches for solving Sudoku, Arxiv preprint arXiv:0805.0697 (2008)
2. Laarhoven, P., Aarts, E.: Simulated annealing: theory and applications. Springer, Heidelberg (1987)
3. Lewis, R.: Metaheuristics can solve sudoku puzzles. Journal of heuristics 13, 387–401 (2007)
4. Chen, D., et al.: Parallelizing simulated annealing algorithms based on high-performance computer. Journal of Global Optimization 39, 261–289 (2007)
5. Minerals, D., Arabia, S.: Evaluating Parallel Simulated Evolution Strategies For VLSI Cell Placement
6. Bellifemine, F., et al.: Jade administrator's guide, TILab (February 2006)