SudoSLVRR: A Multithreaded Multi-Colony Ant Optimization with Ring and Random Communication Topologies as Sudoku Solver

John Michael E. Jabuen, Gian Myrl Renomeron, and
John Paul T. Yusiong

Division of Natural Science and Mathematics
University of the Philippines Tacloban College
Magsaysay Boulevard, Tacloban City 6500, Leyte, Philippines

Abstract. A popular logic-based combinatorial problem known as Sudoku is computationally expensive because of its interdependent constraints. From traditional solvers to heuristics approaches, although effective when it comes to smaller puzzles, faces difficulties with more complex puzzles. This paper presents SudoSLVRR, a parallel multi-colony ant optimization as sudoku solver that addresses scalability limitations of the existing solvers. This framework combines constraint propagation and multi-colony ant optimization with dynamic collaborative mechanism (DCM-ACO) inside the multithreaded environment. Each multi-colony ant is executed inside each thread where threads exchange their iteration-best solution and best-so-far solution in ring and random communication topologies, respectively. The three-source pheromone update is performed by each thread based on the received information. Experimental results on 9×9, 16×16, and 25×25 Sudoku puzzles shows that the proposed framework was able to achieve higher solution success rate and less solution completion time even in larger Sudoku puzzles when compared to single-colony ant and non-parallel multi-colony ant solvers.

Keywords: Constraint Satisfaction Problem · Swarm Intelligence · Ant Colony Optimization · Dynamic Collaborative Mechanism · Parallel Optimization

1 Introduction

The rule of the logic-based Sudoku game includes filling the grid with numbers from 1 to n without repetition to row, column, and subgrid. A valid Sudoku solution must simultaneously satisfy multiple interdependent constraints which makes it a good benchmark for constraint satisfaction problems [3, 10]. Backtracking is the popular traditional way of solving Sudoku [13, 20, 22]. It tries to solve the puzzle by filling every cell with valid values then goes back to the previous assignment if it leads to invalid puzzle. It was then followed by heuristic approaches [15, 22] who mimicked human reasoning in solving Sudoku. However, these approaches suffered from exponential increase of time. To overcome this limitation, metaheuristic approaches specifically swarm-based algorithms emerged. Artificial Bee Colony (ABC) [6], Ant Colony Optimization (ACO) [3], and Particle Swarm Optimization (PSO) [9, ?] are a few examples of these approaches. Though ABC's performance showed potential by simulating foraging behaviour, it still struggled on more complex Sudoku puzzles. Unlike PSO and ABC, ACO was able to provide solutions with less execution time thereby indicating that the convergence speed in ACO is faster. Due to its potential, a multi-colony ant optimization was designed to test its performance when there are multiple colonies of ants in the search space. The Dynamic Collaborative Multi-Colony Ant Optimization (DCM-ACO) allows sharing of pheromone information between multiple colonies which improves diversity and convergence speed thus outperforming the single colony [4]. Parallelization of swarm-based algorithms was developed as an effort to improve the performance of the algorithms. One of these is the Parallel Independent Runs (PIR) but no sharing of information between independent agents leads to redundant exploration in the search space [2]. There is another parallelization technique that focuses on ants which leads to synchronization overhead when a single ant updates global pheromone [7].

To address the exponential increase in time and stagnation, this paper proposes SudoSLVRR, a multithreaded sudoku solver that combines constraint propagation and DCM-ACO within threads that communicate according to the communication topologies. SudoSLVRR is designed to balance exploration and exploitation which improves scalability on large-scale Sudoku puzzles.

2 Related Work

We first consider the application of Ant Colony Optimization (ACO) to Sudoku. ACO is a population-based search method inspired by the foraging behaviour of ants [1], and it has been successfully applied to a wide range of computational problems. The basic ACO algorithm uses a population of agents, or "ants", which individually explore a given problem space and incrementally construct a solution, guided by a global pheromone data structure that is used to inform the decisions taken by the ants. Components associated with higher pheromone concentrations are more likely to be selected, creating a positive feedback loop that steers the population towards high-quality solutions. Premature convergence is discouraged through continuous evaporation of pheromone concentrations. Since the publication of the original Ant System [1], several variants have been proposed, including Ant Colony System (ACS) and Max–Min Ant System (MMAS), each differing in how pheromone is deposited, evaporated, and bounded [17].

Lloyd and Amos [3] present an algorithm for Sudoku based on ACS, combining constraint propagation (CP) [13] with ACO-based search. Their CP component prunes the set of possible values for each cell whenever a value is fixed, effectively eliminating large portions of the search space in parallel. The ACO component then explores the remaining possibilities using pheromone-guided value selection: each ant works on its own copy of the board, iterating over cells and probabilistically choosing values according to pheromone concentrations stored in a global matrix. A local pheromone update discourages following ants from making the same choices, while a global update rewards the best-performing ant in each iteration. A key contribution of their work is the introduction of Best Value Evaporation (BVE), a novel anti-stagnation operator that subjects the globally deposited pheromone amount itself to evaporation over time, thereby preventing the algorithm from locking in to suboptimal solutions. Their method does not rely on any problem-specific heuristic information, which makes it potentially applicable to other constraint satisfaction problems beyond Sudoku.

While the single-colony approach of Lloyd and Amos [3] demonstrates the viability of ACO for Sudoku, single-colony methods inherently face a trade-off between convergence speed and solution diversity—improvements that accelerate convergence tend to reduce the exploration of the search space, and vice versa. To address this limitation in the context of the Travelling Salesman Problem, Mo et al. [4] propose a multi-colony ant optimization with dynamic collaborative mechanism and cooperative game (DCM-ACO). Their framework organises two ACS sub-colonies and one MMAS sub-colony into a heterogeneous population, leveraging the fast convergence of ACS alongside the diversity-preserving pheromone bounds of MMAS. To coordinate pheromone accumulation among the ACS sub-colonies, they introduce a cooperative game strategy based on the Shapley value, which redistributes the total pheromone income to each sub-colony in proportion to its contribution—measured as a function of both solution quality and population diversity (quantified via information entropy). In addition, they design a dynamic collaborative mechanism comprising two methods: a pheromone fusion mechanism that resets the pheromone distribution of a stagnating ACS colony by blending it with the MMAS colony's pheromone matrix, and a public path recommendation strategy that transfers commonly selected edges from the ACS colonies to the MMAS colony to accelerate its convergence. Although DCM-ACO was developed for the TSP rather than Sudoku, its multi-colony architecture and inter-colony communication strategies are directly relevant to this work, as they offer a principled way to balance exploration and exploitation across heterogeneous ant populations.

Separately, Yang et al. [2] address ACO scalability from the perspective of parallelism with a Randomly Matched Parallel Ant Colony Optimization (RMACO) algorithm implemented using MPI. RMACO distributes the search across multiple processors, each hosting its own sub-ant colony. A key design decision is the communication topology: each processor exchanges its iteration-best solution with its neighbour along a fixed ring topology, while simultaneously exchanging its best-so-far solution with a randomly matched processor that changes every exchange cycle. This dual-topology approach avoids the rapid assimilation that occurs under fully-connected topologies, while still enabling sufficient information sharing to guide convergence. RMACO further employs a non-fixed exchange cycle: longer intervals in the early iterations allow sub-colonies to explore their respective solution spaces independently, while shorter intervals in later iterations promote active exchange to accelerate convergence. The pheromone update after each exchange cycle incorporates contributions from three sources—the processor's own iteration-best, the ring neighbour's iteration-best, and the randomly matched processor's best-so-far—enabling ants to benefit from a broader set of search experiences. The ring-and-random communication strategy of RMACO is particularly relevant to our work, as it provides the foundation for the inter-thread communication topology adopted in SudoSLVRR.

Building on these three lines of work, our proposed framework, **SudoSLVRR**, integrates the constraint propagation and ACO-based search of Lloyd and Amos [3], the heterogeneous multi-colony architecture and dynamic collaborative mechanism of Mo et al. [4], and the ring-and-random parallel communication topology of Yang et al. [2], within a multithreaded environment tailored for solving Sudoku puzzles of varying sizes.

3 SudoSLVRR: Algorithm Design and Structure

3.1 Dataset

4 Results and Discussion

5 Conclusions and Future Work

References

1. M. Dorigo, "Optimization, Learning and Natural Algorithms," Ph.D. Dissertation, Politecnico di Milano, 1992.
2. Q. Yang and L. Fang, "RMACO: A Randomly Matched Parallel Ant Colony Optimization," *World Wide Web*, vol. 19, no. 6, pp. 1009–1022, 2015. [Online]. Available: https://doi.org/10.1007/s11280-015-0373-2
3. H. Lloyd and M. Amos, "Solving Sudoku with Ant Colony Optimization," *IEEE Transactions on Games*, vol. 12, no. 3, pp. 302–313, 2020. [Online]. Available: https://doi.org/10.1109/TG.2020.2969827
4. Y. Mo, Z. Tang, and L. Zhao, "Multi-Colony Ant Optimization with Dynamic Collaborative Mechanism and Cooperative Game," *Information Sciences*, vol. 608, pp. 892–908, 2022. [Online]. Available: https://doi.org/10.1016/j.ins.2022.06.055
5. Y. Manyam, S. Sundaram, and A. Sahay, "Sudoku Solver Algorithms: A Comparative Analysis," in *2024 First International Conference for Women in Computing (InCoWoCo)*, IEEE, 2024, pp. 1–9. [Online]. Available: https://doi.org/10.1109/InCoWoCo58689.2024.10443752
6. J. A. Pacurib, G. M. M. Seno, and J. P. T. Yusiong, "Solving Sudoku Puzzles Using Improved Artificial Bee Colony Algorithm," in *2009 Fourth International Conference on Innovative Computing, Information and Control (ICICIC)*, 2009, pp. 885–888. [Online]. Available: https://doi.org/10.1109/ICICIC.2009.334
7. M. Randall and A. Lewis, "A Parallel Implementation of Ant Colony Optimization," *Journal of Parallel and Distributed Computing*, vol. 62, no. 9, pp. 1421–1432, 2002. [Online]. Available: https://doi.org/10.1006/jpdc.2002.1838

8. P. Malakonakis, M. Smerdis, E. Sotiriades, and A. Dollas, "An FPGA-based Sudoku Solver based on Simulated Annealing methods," in *2009 International Conference on Field-Programmable Technology*, 2009, pp. 522–525. [Online]. Available: https://doi.org/10.1109/FPT.2009.5377608
9. S. McGerty, "Solving Sudoku puzzles with particle swarm optimisation," *Final Report, Macquarie University*, 2009.
10. H. Simonis, "Sudoku as a constraint problem," in *CP Workshop on modeling and reformulating Constraint Satisfaction Problems*, vol. 12, 2005, pp. 13–27.
11. E. C. Chi, "Techniques for Solving Sudoku Puzzles," *arXiv preprint arXiv:1203.2295*, 2012.
12. T. Yato and T. Seta, "Complexity and completeness of finding another solution and its application to puzzles," *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, vol. 86, no. 5, pp. 1052–1060, 2003.
13. P. Norvig, "Solving Every Sudoku Puzzle," 2006. [Online]. Available: http://norvig.com/sudoku.html
14. D. Eppstein, "Computational Complexity of Games and Puzzles," in *Proceedings of the International Conference on Fun with Algorithms*, 2006, pp. 1–12.
15. J. F. Crook, "A pencil-and-paper algorithm for solving Sudoku puzzles," *Notices of the AMS*, vol. 56, no. 4, pp. 460–468, 2009.
16. T. Mantere and J. Koljonen, "Solving, rating and generating Sudoku puzzles with GA," in *2007 IEEE Congress on Evolutionary Computation*, IEEE, 2007, pp. 1382–1389.
17. M. Dorigo and T. Stützle, *Ant Colony Optimization*. MIT press, 2004.
18. M. Pedemonte, E. Alba, and F. Luna, "Bit-parallel ACO for the traveling salesman problem," *Applied Intelligence*, vol. 35, no. 3, pp. 346–357, 2011.
19. E.-G. Talbi, *Metaheuristics: From Design to Implementation*. John Wiley & Sons, 2009.
20. M. Schottlender, "The effect of guess choices on the efficiency of a backtracking algorithm in a Sudoku solver," in *IEEE Long Island Systems, Applications and Technology (LISAT) Conference 2014*, 2014, pp. 1–6. [Online]. Available: https://doi.org/10.1109/LISAT.2014.6845190
21. A. Moraglio and J. Togelius, "Geometric particle swarm optimization for the sudoku puzzle," in *Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation*, 2007, pp. 118–125. [Online]. Available: https://doi.org/10.1145/1276958.1276975
22. A. Bhattarai, D. Uprety, P. Pathak, S. N. Shrestha, S. Narkarmi, and S. Sigdel, "A Study Of Sudoku Solving Algorithms: Backtracking and Heuristic," *arXiv preprint arXiv:2507.09708*, 2025.