Optimizing QuickSort Pivot Selection Using Simulated Annealing

Bruce Metoyer and Vivek Shandilya

Department of Computer Science

Introduction

QuickSort is a sorting algorithm that picks an element as a pivot and partitions the given array around the chosen pivot by placing the pivot in the correct position in a sorted array. Essential in theoretical computer science for analyzing average-case complexity and developing new techniques. There are traditional pivot selection techniques that include first, last, random, and median. Simulated Annealing is an Optimization algorithm designed to search for an optimal solution in a large solution space. Heat and temperature play a crucial role in simulated annealing. Heat is a measure of the degree of randomness in the search process, which decreases over time. An initial temperature is set, and the higher the temperature, the higher the randomization jumps will be. As the temperature decreases according to the cooling schedule, the jumps will be lower and more precise, reaching global optima.

Methodology

Baseline QuickSort:

- Each traditional pivot method was tested on all array types
- The cost function evaluates each pivot strategy:
  - **Cost** = swaps + comparisons + (balance factor x array size)
- Balance factor:
  - **balance factor** = $\frac{|\text{left\_parirtion\_size} - \text{right\_partition\_size}|}{\text{size}}$

Each pivot strategy displays:

- Final sorted array
- Number of swaps
- Total number of comparisons
- Pivot selection cost

SA Optimization:

- Implemented to explore optimization capabilities
- Objective function:
  - Rastrigin function: a non-convex, multimodal function commonly used for benchmarking global optimizers
- Key steps in the algorithm include:
  1. Define the objective function (Rastrigin)
  2. Generate a random initial solution within bounds
  3. Use a neighborhood function to explore nearby solutions
  4. Apply a probabilistic acceptance rule based on the current temperature
  5. Decrease temperature over 1000 iterations to balance exploration and exploitation
  6. Continuously track the best solution and score

SA-Optimized QuickSort

- **Goal:** Aims to address the limitations of traditional pivot selection strategies by identifying optimal pivots during SA
- **Cooling Schedule:** T = Initial Temperature / Iteration + 1
- **SA Parameters:**
  - **Initial Temp:** 0
  - **Min Temp:** 0.0001
  - **At each recursive step:**
    - Generate pivot candidates
    - Evaluate the cost for each
    - Accept the best or near best based on temperature
  - **Neighbor strategy:**
    - 80% local (+ or – 1 index)
    - 20% global (random index)

Motivation

Is there a more efficient method to select pivots?

The first and last pivot approaches will end up in the worst case when the array is already sorted. Random does not have a pattern for which the worst-case occurs, and the median takes more time on average, as median finding has high computational constraints. By learning from previous sorting operations, Simulated Annealing (SA) can dynamically explore pivot selections by balancing randomization and optimization. By incorporating randomness and systematic refinement, SA offers a novel approach to enhancing sorting algorithms.

Problem Statement

We aim to optimize QuickSort pivot selection using Simulated Annealing (SA), a probabilistic algorithm designed to efficiently escape local minima and converge towards a near-optimal solution.

Proposed Solution

Objective: Improve QuickSort pivot selection by dynamically minimizing sorting inefficiencies. We Implement:

1. **Baseline QuickSort** using traditional pivot selection strategies, tracking metrics such as cost, swaps, and comparisons.
2. **Simulated Annealing Benchmark** applied to the Rastrigin function (used for testing SA implementation)
3. **SA-Optimized QuickSort,** combining QuickSort with the SA approach, where a cost function and the SA strategy guide the pivot selection.

Experiment Setup

Test Data:
All pivot selection methods will be tested against:

- Random arrays
- Sorted arrays
- Partially sorted arrays
- Reverse-sorted arrays
- Three arrays with duplicate values

Comparison Strategies:
Traditional QuickSort with:

- First Element Pivot
- Last Element Pivot
- Median Pivot
- Random Pivot
- Proposed SA-Optimized Pivot Selection

Tools & Environment: Implemented in Python using Jupyter Notebook
Data Structures: Python lists
Tracked Metrics:

- Swaps
- Comparisons
- Pivot Selection Cost

Results

Data used & Input:

1. SA - Sorted Array: [0,1,2,3,4,5,6,7,8,9,10]
2. UA – Unsorted Array: [20, 382, 78, 292, 1000, 889, 2, 10, 92, 3, 11]
3. RSA – Reverse/Inverse Sorted Array: [10, 9, 8, 7, 6, 5, 4, 3, 2, 1]
4. NSA – Nearly Sorted Array: [11, 12, 13, 14, 2, 1, 20, 30, 19]
5. DA1, DA2, DA3 – Duplicate Arrays
   1. [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]
   2. [100, 49, 90, 100, 49, 90, 100, 49, 90, 100, 49, 90, 100, 49, 90, 100, 49, 90, 100, 49, 90, 100, 49, 90, 100, 49, 90, 100, 49, 90]
   3. [10, 4, 10, 4, 10, 4, 10, 4, 10, 4, 10, 4]

Traditional vs SA-Optimized QuickSort Performance

ATB	LS	LCM	LCO	FS	FCM	FCO	RS	RCM	RCO	MS	MCM	MCO	SAS	SACM	SACO
SA	0	55	110.0	20	55	110.0	12	32	52.0	14	22	26.0	8	22	26.0
UA	12	23	30.0	18	26	40.0	14	25	38.0	20	23	32.0	18	22	26.0
RSA	5	45	90.0	13	45	90.0	15	26	42.0	15	19	24.0	13	19	24.0
NSA	7	20	32.0	11	17	24.0	11	20	34.0	9	18	28.0	13	16	20.0
DA1	30	465	930.0	60	465	930.0	56	465	930.0	60	465	930.0	58	465	930.0
DA2	41	134	234.0	70	135	238.0	61	134	234.0	64	134	234.0	58	134	234.0
DA3	14	41	72.0	24	36	62.0	22	41	72.0	23	41	72.0	20	36	62.0

Results

Average Metric Performance Table

	Last	First	Random	Median	SAQS
Swaps	15.5	30.8	27.28	29.28	26.28
Comparisons	111.85	111.28	106.14	103.14	102
Cost	214	213.42	200.28	192.28	188.85

Summary

In conclusion, SA avoids the disadvantages of traditional pivot strategies and adapts well to diverse array configurations due to its randomization and measure of disorder. The simulated annealing approach had the best cost and comparison efficiency, while showing good consistency with the swap data. The SA approach, along with all the other traditional methods, performed poorly, with duplicate arrays consistently showing high values in terms of cost, swaps, and comparisons. Although the SA approach showed the best average metrics, the last swaps method produced the best average results because it had zero total swaps for sorted arrays. SA is also dependent on parameters, including iterations, temperature, and cooling schedule. For future work, we can consider using neural networks to guide SA based on past patterns. Combining SA with other metaheuristics, such as the Genetic Algorithm (GA) or Particle Swarm Optimization (PSO), can also be considered.

References

[1] Bashath, S., & Ismail, A. R. (2019). *Improved Particle Swarm Optimization by Fast Simulated Annealing Algorithm*. **ICAIIT**, pp. 297–301. https://doi.org/10.1109/ICAIIT.2019.8834515

[2] Katsuki, K., Shin, D., Onizawa, N., & Hanyu, T. (2022). *Fast Solving Complete 2000-Node Optimization Using Stochastic-Computing Simulated Annealing*. **ICECS**, pp. 1–4. https://doi.org/10.1109/ICECS202256217.2022.9971124

[3] Nakada, K., Sekii, D., Tamura, K., & Yasuda, K. (2023). *Combinatorial Optimization Based on Hierarchical Structure in Solution Space*. **SMC**, pp. 5058–5063. https://doi.org/10.1109/SMC53992.2023.10394606

[4] Marcellino, M., Pratama, D. W., Suntiarko, S. S., & Margi, K. (2021). *Comparative of Advanced Sorting Algorithms*. **ICCSAI**, pp. 154–160. https://doi.org/10.1109/ICCSAI53272.2021.9609715

[5] Aylaj, B., & Nouh, S. (2022). *Degeneration vs Classical of Simulated Annealing: Performance Analysis*. **CommNet**, pp. 1–5. https://doi.org/10.1109/CommNet56067.2022.9993814

[6] Ćatić, I., Mujić, M., Nosović, N., & Hrnjić, T. (2023). *Enhancing Performance of CUDA Quicksort*. **ICAT**, pp. 1–5. https://doi.org/10.1109/ICAT57854.2023.10171304

[7] Taotiamton, S., & Kittitornkun, S. (2017). *A Parallel Dual-Pivot QuickSort Algorithm*. **ICSEC**, pp. 1–5. https://doi.org/10.1109/ICSEC.2017.8443883

[8] Faujdar, N., & Ghrera, S. P. (2015). *Analysis and Testing of Sorting Algorithms*. **CSNT**, pp. 962 967. https://doi.org/10.1109/CSNT.2015.98

[9] Ćatić, I., Mujić, M., Nosović, N., & Hrnjić, T. (2023). *Enhancing Performance of CUDA Quicksort* (Duplicate). **ICAT**, pp. 1–5. https://doi.org/10.1109/ICAT57854.2023.10171304

[10] Wulandari, R. (2024). *Improving Facility Layout Using Genetic Algorithm and Simulated Annealing*. **ICICyTA**, pp. 876–881. https://doi.org/10.1109/ICICYTA64807.2024.10913089

[11] Cao, X., Li, G., Ye, Q., Zhou, R., Ma, G., & Zhou, F. (2017). *Multi-objective Optimization of PMSM Using Hybrid SA Algorithm*. **ICIEA**, pp. 535–540. https://doi.org/10.1109/ICIEA.2017.8282902

[12] Meng, T. (2024). *Optimization of Warehousing Paths Using SA*. **ISPCEM**, pp. 1068–1072. https://doi.org/10.1109/ISPCEM64498.2024.00188

[13] Wang, H., Xia, J., & Song, Y. (2024). *Crew Allocation via SA and Heuristic Algorithms*. **ICIPCA**, pp. 1564–1568. https://doi.org/10.1109/ICIPCA61593.2024.10709305

[14] Yang, Y., Wang, J., Wu, Z., Luo, M., Wei, L., & Li, Z. (2024). *Tobacco Sorting Optimization Using GA-SA*. **ICEACE**, pp. 928–933. https://doi.org/10.1109/ICEACE63551.2024.10898222

[15] Jangra, A., & Dubran, H. (2021). *SA-Based Load Balancing in Cloud Computing*. **ICRITO**, pp. 1–4. https://doi.org/10.1109/ICRITO51393.2021.9596215

[16] Zheng, J., & Zhang, P. (2024). *Variance Consensus SA for Cargo Sorting*. **IMCEC**, pp. 322–326. https://doi.org/10.1109/IMCEC59810.2024.10575637

[17] GeeksforGeeks. (2014). *Quick Sort Algorithm*. https://www.geeksforgeeks.org/quick-sort-algorithm/

[18] GeeksforGeeks. (2024). *Implement Simulated Annealing in Python*. https://www.geeksforgeeks.org/implement-simulated-annealing-in-python

Acknowledgements

Thanks to the Department of Computer Science at Bowie State University for facilitating this course. This course enabled me to develop research skills while also gaining knowledge about an interesting topic in simulated annealing. I would also like to thank the staff and peers who have assisted me throughout this journey.