Solving the Quadratic Assignment Problem Using a Hybrid Genetic Algorithm and Simulated Annealing Approach

Mazilu Cosmin-Alexandru

UAIC
Computer Science

January 12, 2025

Abstract

The Quadratic Assignment Problem (QAP) is one of the most challenging combinatorial optimization problems, with applications ranging from facility layout design to electronics and logistics. Due to its NP-hard nature, exact methods become computationally expensive for large problem instances. In this report is explored a hybrid solution combining Genetic Algorithm (GA) and Simulated Annealing (SA) to solve QAP efficiently. The hybrid approach leverages the global search capabilities of GAs and the local refinement strength of SA, aiming to balance exploration and exploitation. Experimental results on benchmark instances demonstrate the effectiveness of the proposed hybrid method in achieving high-quality solutions within reasonable computational time.

Introduction

The Quadratic Assignment Problem (QAP) is a classical optimization problem first introduced by Koopmans and Beckmann in 1957. It models real-world scenarios where a set of facilities must be assigned to a set of locations in a manner that minimizes the total cost, which is determined by the distances between locations and the flow between facilities.

This report focuses on a hybrid approach that combines the strengths of Genetic Algorithms and Simulated Annealing to address the limitations of standalone methods. Genetic Algorithms are inspired by the principles of natural evolution, using selection, crossover, and mutation operators to explore the solution space. While GAs excel in maintaining diversity and avoiding local optima, they may struggle with fine-tuning solutions. Simulated Annealing, on the other hand, is a probabilistic local search technique inspired by the annealing process in metallurgy. SA starts with an initial solution and iteratively explores neighboring solutions, accepting worse solutions with a probability that decreases over time, governed by a temperature parameter. This feature allows SA to escape local optima early in the search process while focusing on refinement as the temperature lowers.

Formal description of the QAP

Let n be the number of facilities and locations and denote by N the set $N = \{1, 2, 3, 4, ..., n\}$.

$$\min_{\phi \in S_n} \sum_{i=1}^{n} \sum_{j=1}^{n} f_{ij} d_{\phi(i)\phi(j)} + \sum_{i=1}^{n} b_{i\phi(i)}$$

where S_n is the set all the permutations $\phi : N \to N$. Each individual product $f_{ij} d_{\phi(i)\phi(j)}$ is the cost of assigning facility i to location $\phi(i)$ and facility j to location $\phi(j)$. [6]

Methods

In a genetic algorithm, a population of candidate solutions is developed, termed individuals or phenotypes, to improve their fitness for a given optimization problem. Each candidate solution is characterized by a set of properties, known as chromosomes, which can undergo alterations such as mutation or crossover. The process begins with a randomly generated population and proceeds iteratively, with each iteration referred to as a generation. During each generation, the fitness of each individual is assessed, typically by evaluating the objective function of the optimization problem. Based on their fitness, individuals are stochastically selected to contribute to the next generation. With the best chromosome of the generation is initialize the SA algorithm, The chromosome after SA will replace the worst chromosome from generation .Their genetic information may be recombined and occasionally mutated to produce a new population. This process repeats until a stopping criterion is met, such as reaching a predetermined number of generations or achieving a satisfactory fitness level.

Initialization of the Genetic Algorithm

An initial population of candidate solutions is created by generating a random permutation of the set $\{1, 2, 3, 4, ..., n\}$ representing the order of locations. [1]

Selections

The chromosomes for next generation are selected using Tournament Selection. This type of selection consists in choosing random k chromosomes from the generations, and selecting for next generation the with the best fitness from all k. Values of k is usually between 2 and 10. A smaller value of k promotes diversity for generation, meanwhile a higher value of k make the genetic algorithm to tend to be greedy and help for converging. [5]

Crossover operator

In this genetic algorithm is used two point crossover. Two cutting point are generated randomly. The offsprings are formed by swapping the locations from parents between the cutting points.

Mutation operator

To prevent the entire population from converging to a local optimum in the problem, mutation introduces random changes in the offspring. In this case, every chromosome has a probability to have two positions swapped.

Hypermutation

If the value of the best fitness of population doesn't improve after a specific number of generations, the probability of mutation increase drastically for one generation to explore new chromosomes. [4]

Simulated annealing

(Metropolis algorithm) is a metaheuristic trajectory optimization algorithm, used in finding optimal solutions in large search spaces. In addition, Simulated Annealing has a new mechanic, called temperature. The algorithm starts with an initial temperature value which decrease overtime while the algorithm is running. The purpose of the temperature is to make the algorithm accept worse solutions in order to escape from a local optima. The higher the temperature the higher is the chance that a worse solution in accepted. Temperature schedule, which will be presented later, it's a trade-off between the computational time and the quality of solutions. The algorithm ends after a maximum number of repetitions or if after a certain number of iterations the optimal solution doesn't improve. [2]

Tested Instances

The instances teste are: chr12a, chr18b, chr25a, nug12, nug20, nug30, tai20a, tai80a, tai256c. tho30, tho40, tho150, sko100a. The number from their name represents the size, number of locations and facilities. [3]

Experiments description

Every instance will be runned 30 times with the following parameters and values:

- Max generation: 1000;
- Population size: 200;
- Crossover rate: 0.3;
- Mutate rate: 0.15;
- Hypermutation: after 50 generations of no improvement, mutation rate 0.6;
- Temperature (for Simulated Annealing): the initial temperature is set to 0.99 and minimal temperature is 10e-10. Temperature is modeled with the following formula: $T_n = T_{n-1} \cdot 0.99$.

0.1 Experimental results

chr12a	
	Value
Known optimimum	9552
Mean	9552
Standard deviation	0
Min value	9552
Max value	9552
Mean Exec Time	7.49001

Table 1: GA Statistical Metadata for chr12a

chr18a	
	Value
Known optimimum	1534
Mean	1534
Standard deviation	0
Min value	1534
Max value	1534
Mean Exec Time	15.17184

Table 2: GA Statistical Metadata for chr18b

chr25a	
	Value
Known optimimum	3796
Mean	3970.47619
Standard deviation	150.75291
Min value	3796
Max value	4254
Mean Exec Time	24.64862

Table 3: GA Statistical Metadata for chr25b

nug12	
	Value
Known optimimum	578
Mean	578
Standard deviation	0
Min value	578
Max value	578
Mean Exec Time	7.43418

Table 4: GA Statistical Metadata for nug12

nug20	
	Value
Known optimimum	2570
Mean	2570
Standard deviation	0
Min value	2570
Max value	2570
Mean Exec Time	16.38443

Table 5: GA Statistical Metadata for nug20

nug30	
	Value
Known optimimum	6124
Mean	6142.09523
Standard deviation	11.37208
Min value	6124
Max value	6164
Mean Exec Time	33.49198

Table 6: GA Statistical Metadata for nug30

tai20a	
	Value
Known optimimum	703482
Mean	706348.36363
Standard deviation	1273.08039
Min value	703482
Max value	708468
Mean Exec Time	15.87497

Table 7: GA Statistical Metadata for tai20a

tai80a	
	Value
Known optimimum	13499184
Mean	14102860.76923
Standard deviation	19044.17916
Min value	14077758
Max value	14139036
Mean Exec Time	247.53337

Table 8: GA Statistical Metadata for tai80a

tai256c	
	Value
Known optimimum	13499184
Mean	44964170.66666
Standard deviation	20840.10463
Min value	44936120
Max value	45002152
Mean Exec Time	2632.70483

Table 9: GA Statistical Metadata for tai256c

tho30	
	Value
Known optimimum	149936
Mean	150403.6
Standard deviation	363.60396
Min value	149936
Max value	151078
Mean Exec Time	33.78586

Table 10: GA Statistical Metadata for tho30

tho40	
	Value
Known optimimum	240516
Mean	242880.66666
Standard deviation	894.60954
Min value	241882
Max value	244350
Mean Exec Time	64.43272

Table 11: GA Statistical Metadata for tho40

tho150	
	Value
Known optimimum	8133398
Mean	8444079.6
Standard deviation	29271.74275
Min value	8384812
Max value	8481034
Mean Exec Time	996.65036

Table 12: GA Statistical Metadata for tho150

sko100a	
	Value
Known optimimum	152002
Mean	156049.0
Standard deviation	170.53856
Min value	155844
Max value	156296
Mean Exec Time	431.92342

Table 13: GA Statistical Metadata for sko100a

For smaller instances, the algorithm consistently achieved optimal results with zero standard deviation, highlighting its reliability in these cases. Larger instances showed a slight deviation, but the solutions remained competitive and close to optimal.

The hybrid approach proved efficient, with execution times scaling predictably with the problem size. Smaller instances were solved in mere seconds, while larger ones like tai256c required extended computation but remained practical.

The algorithm maintained a balance between exploration and exploitation, as evidenced by its ability to avoid local optima and achieve competitive solutions even in challenging instances.

Conclusion

In conclusion, in this paper is addressed the complex Quadratic Assignment Problem (QAP) by integrating Genetic Algorithms (GA) and Simulated Annealing (SA). The experimental results demonstrate the algorithm's ability to consistently deliver high-quality solutions. Its performance on several challenging instances, where the solutions closely approached the known optimal costs, underscores the practical applicability of our approach.

Additionally, the algorithm's rapid computational times enhance its practicality, making it both theoretically robust and suitable for real-world applications. The synergistic use of genetic operations and SA within the algorithm has proven highly effective in tackling the intricate optimization challenges posed by the QAP.

Overall, these findings highlight the potential of hybrid algorithms in solving complex optimization problems. This research establishes a strong foundation for the effectiveness of such techniques and contributes to the growing body of knowledge in heuristic optimization strategies.

Bibliography

[1] Hosein Azarbonyad and Reza Babazadeh. A genetic algorithm for solving quadratic assignment problem (qap). *arXiv preprint arXiv:1405.5050*, 2014.

[2] Roberto Battiti and Giampietro Tecchiolli. Simulated annealing and tabu search in the long run: a comparison on qap tasks. *Computers & mathematics with applications*, 28(6):1–8, 1994.

[3] CORAL. A quadratic assignment problem library. Accessed: 2025-01-12.

[4] Lect. Dr. Eugen Croitoru. Genetic Algorithms. https://profs.info.uaic.ro/eugen.croitoru/teaching/ga/#Notions02, 2024.

[5] Anupriya Shukla, Hari Mohan Pandey, and Deepti Mehrotra. Comparative review of selection techniques in genetic algorithm. In *2015 international conference on futuristic trends on computational analysis and knowledge management (ABLAZE)*, pages 515–519. IEEE, 2015.

[6] Eranda Çela. The quadratic assignment problem: Theory and algorithms, 2008. Accessed: 2025-01-12.