Optimization Project - An Alternative Solution for PCB Drilling Using Traveling Salesman Problem

Işıl Altınışık - 150220308, Emre Aydoğmuş - 150220323, Ömer Atmaca - 150220335

May, 2024

1 Abstract

Traveling salesman Problem (TSP) is one of the best-known NP-hard problems, which means there is no algorithm to solve it in polynomial time. This project represents a comparative study of three different optimization algorithms (Genetic Algorithm(GA), Simulated Annealing (SA) and Particle Swarm Optimization (PSO)) applied to a variant of TSP in the context of printed circuit board (PCB) drilling path optimization.

Contents

2 Introduction

The Traveling Salesman Problem (TSP) is one of the best-known NP-hard problems, which means that there is no guaranteed way to obtain the optimal route and no exact algorithm to solve it in polynomial time [1]. It can simply be stated as: if a traveling salesman wishes to visit exactly once each of a list of cities and then return to the home city, what is the least costly route the traveling salesman can take?

TSP has many practical applications including regular distribution of goods or resources, planning public transportation, planning tour routes, drilling of printed circuit boards, overhauling gas turbine engines, x-ray crystallography analysis,dna sequencing etc. The method which would definitely obtain the optimal solution of TSP is the method of exhaustive enumeration and evaluation. This procedure begins by generating the possibility of all the tours and evaluate according length of the tour. The tour with the smallest length chosen as the best, and guaranteed to be optimal [2]. However, the approximation algorithms to solve TSP has reached highly efficient results.

In this paper, the aim is to solve TSP in the context of drilling Printed Circuit Boards (PCB) with different approximate optimization algorithms. PCB drilling is a critical aspect of manufacturing process of circuits where numerous holes must be drilled at precise locations on the board.

3 Problem Definition

To connect a conductor on one layer with a conductor on another layer, or to position the pins of integrated circuits, holes have to be drilled through the board. The holes may be of different sizes. To drill two holes of different diameters consecutively, the head of the machine has to move to a tool box and change the drilling equipment. This is quite time consuming. Thus it is clear that one has to choose some diameter, drill all holes of the same diameter, change the drill, drill the holes of the next diameter, etc. Thus, this drilling problem can be viewed as a series of TSPs, one for each hole diameter, where the 'cities' are the initial position and the set of all holes that can be drilled with one and the same drill. The 'distance' between two cities is given by the time it takes to move the drilling head from one position to the other. The aim is to minimize the travel time for the machine head [3].

4 Background

Printed Circuit Boards (PCBs) are integral components of modern electronic devices, serving as the foundation for connecting various electronic components and facilitating their functionality. These holes are essential for mounting electronic components, establishing electrical connections, and ensuring the proper functioning of the final product. Optimizing the drilling process for PCBs is crucial because efficient drilling reduces manufacturing time, leading to higher productivity in PCB production facilities. Minimizing the production time is particularly important in industries where time-to-market is a critical factor, such as consumer electronics and telecommunications.

Optimizing the drilling path also helps to minimize material wastage and resource consumption. By identifying the most efficient route for drilling holes, manufacturers can reduce the amount of raw material required for each PCB, resulting in cost savings and improved resource utilization. This aspect is especially relevant in environmentally conscious industries, where reducing waste and conserving resources are top priorities.

The times it takes for a brute force algorithm to find the shortest path based on the number of nodes is shown in Table 1. As can be inferred from this table, more efficient approximation algorithms, as done in this paper for solving the TSP problem, are required. It shows exactly why do we need optimization in our life.

Table 1: Tame it takes to compute all possible permutations of path

Amount of Node	Time (second)
3	0.00001990004.
4	0,00005899998
5	0,0003123
6	0,0022327
7	0,01774
8	0,1696
9	1,783
10	19,956
11	250,96
30	576,000,000,000 (years)
48	1,399.$10^{28}(years)$
76	3,355.$10^{53}(years)$

5 Assumptions

Every point to be drilled is considered as nodes and path that the drilling machine follows is the path to minimized. To determine coordinates of every hole on the PCB, imaginary grid lines applied on the surface of PCB which is shown in Figure 1.

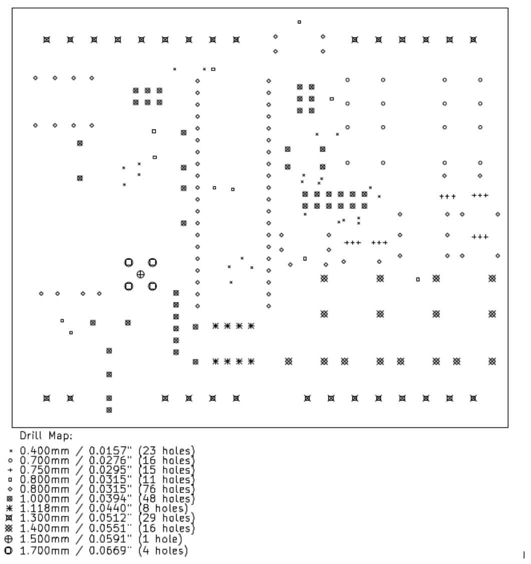

Drill Map:
× 0.400mm / 0.0157" (23 holes)
○ 0.700mm / 0.0276" (16 holes)
+ 0.750mm / 0.0295" (15 holes)
□ 0.800mm / 0.0315" (11 holes)
◇ 0.800mm / 0.0315" (76 holes)
⊠ 1.000mm / 0.0394" (48 holes)
✳ 1.118mm / 0.0440" (8 holes)
⌧ 1.300mm / 0.0512" (29 holes)
▩ 1.400mm / 0.0551" (16 holes)
⊕ 1.500mm / 0.0591" (1 hole)
O 1.700mm / 0.0669" (4 holes)

Figure 1: Map of holes on the surface of PCB with their information.

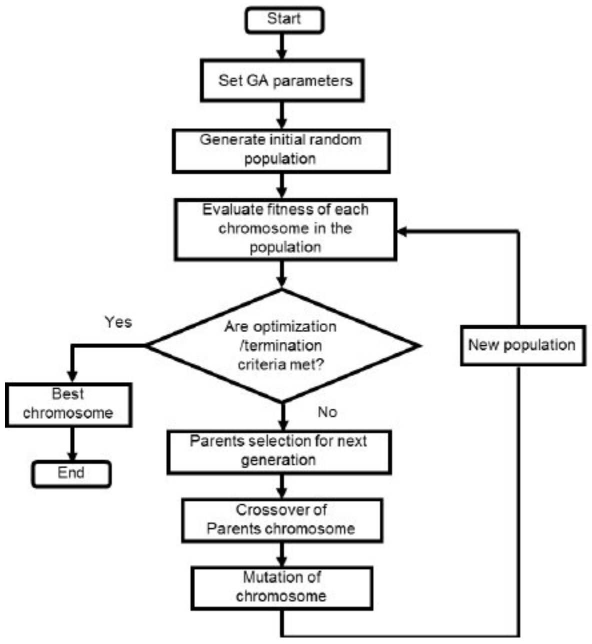

Start
Set GA parameters
Generate initial random population
Evaluate fitness of each chromosome in the population
Are optimization /termination criteria met?
Yes
No
Best chromosome
End
Parents selection for next generation
Crossover of Parents chromosome
Mutation of chromosome
New population

Figure 2: Scheme of genetic algorithm for TSP

6 Methods

6.1 Genetic Algorithm

The GA is a population-based optimization technique inspired by the process of natural selection. In this project, a population of drilling path solutions is iteratively evolved using genetic operators such as crossover, mutation and selection. The fitness of each solution is evaluated based on the total drilling distance. The GA aims to converge to an optimal or near-optimal solution by evolving the population over multiple generations. After the termination of genetic algorithm, an optimal solution is obtained. If the termination condition is not satisfied then algorithm continues with new population[4].

An overall generalized genetic algorithm scheme for TSP is shown in the Figure 2.[7]

6.2 Simulated Annealing

SA is a probabilistic optimization technique that starts with an initial solution and iteratively explores the solution space by accepting probabilistic improvements and occasional worse solutions. At each iteration, SA evaluates the fitness of the current solution. Unlike traditional optimization methods, SA allows for escaping local optima by occasionally accepting worse solutions, thereby preventing premature convergence. This stochastic acceptance criterion enables SA to explore the search space more effectively and discover potentially better solutions that may lie beyond local optima.

As SA progresses, the temperature parameter decreases according to a predefined cooling schedule, gradually reducing the probability of accepting worse solutions. Thusi it controls the balance between exploration and exploitation, ensuring that the optimization process converges towards an optimal or near-optimal solution while avoiding premature convergence to suboptimal solutions. An overall generalized genetic algorithm scheme for TSP is shown in the Figure 3.[6]

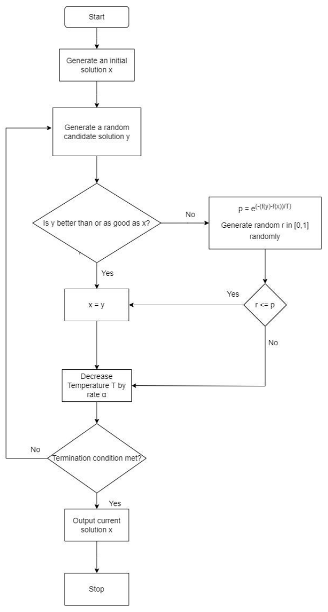

Start
Generate an initial solution x
Generate a random candidate solution y
Is y better than or as good as x?
No
$p = e^{(-(f(y)-f(x))/T)}$
Generate random r in [0,1] randomly
Yes
x = y
Yes
r <= p
No
Decrease Temperature T by rate α
No
Termination condition met?
Yes
Output current solution x
Stop

Figure 3: Scheme of simulated annealing algorithm for TSP

6.3 Particle Swarm Optimization

PSO is a population-based optimization technique inspired by the social behavior of bird flocking or fish schooling. In this project, a population of particles represents potential drilling paths. Each particle adjusts its position based on it's own best-known solution and the best solution found by the entire swarm. PSO iteratively updates particle positions to converge towards an optimal solution. At the beginning, a swarm of particles is randomly initialized within the solution space. Each particle's position represents a potential solution to the optimization problem. Additionally, each particle maintains information about its own best-known solution (referred to as "personal best") and the best solution found by any particle in the swarm (referred to as "global best"). During each iteration, particles update their positions based on both personal and global information. This update is guided by two main factors: the particle's velocity and its position. The velocity determines the direction and magnitude of the particle's movement in the solution space, while the position represents the actual solution being evaluated. By adjusting their velocities and positions iteratively, particles gradually converge towards an optimal solution. Through interaction and information sharing among particles, PSO iteratively refines the search process.

7 Implementation

In this section, we describe the implementation details of our approach to solving the PCB drilling path optimization problem using three different algorithms: Genetic Algorithm (GA), Simulated Annealing (SA), and Particle Swarm Optimization (PSO). Our objective is to find the most efficient path for drilling holes on a PCB, minimizing the total travel distance of the drilling head.

The problem is modeled as a Traveling Salesman Problem (TSP), where each hole on the PCB represents a city. The goal is to determine the shortest possible route that visits each hole exactly once. The coordinates of the holes are provided as input data in a CSV file, which is read into a list of city coordinates.

Since to minimize the time lost when changing drill sizes the most efficient path for the drilling machine is to drill all the holes of the same diameter first, then change the drill size and proceed to the next set of holes, the datatset is divided into 11 part and the algorithms are applied to each part seperately.

8 Working Mechanism

8.1 Genetic Algorithm (GA)

In our implementation:

- **Initialization:** We generate a random population of candidate solutions, where each candidate represents a possible drilling path.
- **Crossover and Mutation:** We apply crossover to combine segments of two parent solutions to generate offspring. Mutation is introduced by randomly swapping two cities in the path to maintain diversity in the population.

- **Selection:** Tournament selection is used to choose individuals for reproduction based on their fitness, which is evaluated by the total drilling distance.
- **Termination:** The algorithm runs for a predefined number of generations or until convergence to a satisfactory solution is achieved.

8.2 Simulated Annealing (SA)

SA is a probabilistic optimization technique that starts with an initial solution and iteratively explores the solution space. In our implementation:

- **Initialization:** We generate an initial solution randomly, representing a potential drilling path.
- **Annealing Schedule:** The temperature starts high and gradually decreases. At each step, a new solution is generated by swapping two cities in the current solution.
- **Acceptance Criteria:** New solutions are accepted if they result in a shorter path or with a certain probability if they are worse, based on the current temperature. This helps in escaping local optima.
- **Termination:** The process continues until the temperature drops below a threshold, at which point the best solution is considered the final solution.

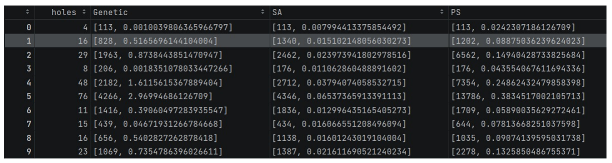
8.3 Particle Swarm Optimization (PSO)

PSO is a population-based optimization technique inspired by the social behavior of bird flocking or fish schooling. In our implementation:

- **Initialization:** A swarm of particles is initialized with random positions, each representing a potential drilling path.
- **Velocity and Position Update:** Each particle adjusts its velocity based on its personal best position and the global best position found by the swarm. The positions of the particles are then updated accordingly.
- **Convergence:** The swarm iteratively updates the particles' positions, converging towards an optimal solution over time.
- **Termination:** The process continues for a fixed number of iterations or until the improvement in the best solution falls below a threshold.

8.4 Performance Evaluation

We evaluate the performance of each algorithm by comparing the total drilling distance and the computational time required to find the optimal or near-optimal solution. The results are summarized in Figure 3. The shortest paths found by each algorithm and the time they take is printed as a data frame using "pandas" library.



	holes	Genetic	SA	PS
0	4	[113, 0.0010039806365966797]	[113, 0.007994413375854492]	[113, 0.0242307186126709]
1	16	[828, 0.5165696144104004]	[1340, 0.015102148056030273]	[1202, 0.08875036239624023]
2	29	[1963, 0.8738443851470947]	[2462, 0.023973941802978516]	[6562, 0.14940428733825684]
3	8	[206, 0.0018351078033447266]	[176, 0.011062860488891602]	[176, 0.043554067611694336]
4	48	[2182, 1.6115615367889404]	[2712, 0.03794074058532715]	[7354, 0.24862432479858398]
5	76	[4266, 2.96994686126709]	[4346, 0.06537365913391113]	[13786, 0.3834517002105713]
6	11	[1416, 0.39060497283935547]	[1836, 0.012996435165405273]	[1709, 0.05890035629272461]
7	15	[439, 0.04671931266784668]	[434, 0.016066551208496094]	[644, 0.07813668251037598]
8	16	[656, 0.5402827262878418]	[1138, 0.01601243019104004]	[1035, 0.09074139595031738]
9	23	[1069, 0.7354786396026611]	[1387, 0.021611690521240234]	[2278, 0.1325850486755371]

Figure 4: Results of three different algorithms for different number of holes

8.5 Visualization

The best drilling path found by each algorithm is visualized using Matplotlib, illustrating the sequence of holes to be drilled on the PCB. This visualization helps in comparing the efficiency of the paths generated by different algorithms.

The shortes path found in each algorithm is plotted using matplotlib library and shown in the Figure 5, 6 and 7 for the holes that has radius of 0.8 mm and in the Figure 8, 9 and 10 for the holes that has radius of 1.3 mm. All the figures are the representation of the PCB sample given in the Figure 1 in cartesian coordinate system.

In summary, we implemented and compared three different algorithms for optimizing the drilling path on a PCB.

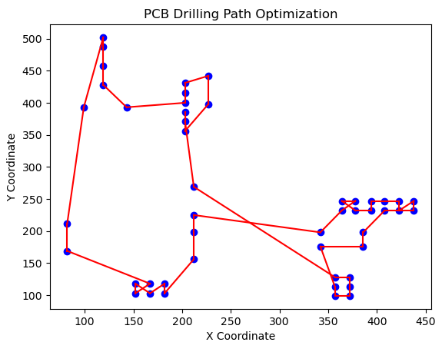


Figure 5: Genetic Algorithm for 76 Holes

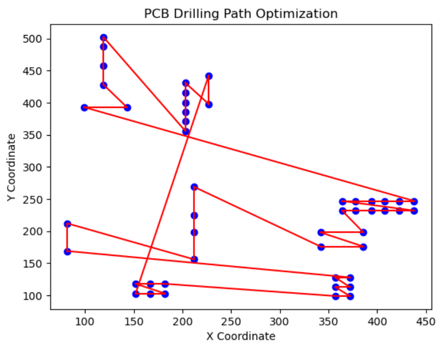


Figure 6: Simulated Annealing Algorithm for 76 Holes

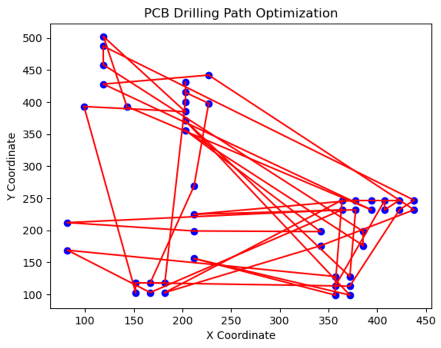


Figure 7: Particle Swarm Optimization for 76 Holes

9 Results and Discussion

As mentioned and shown in the Table 1, the only way to find the exact answer of TSP is only possible

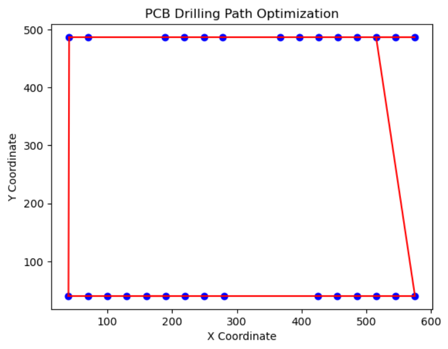


Figure 8: Genetic Algorithm for 29 Holes

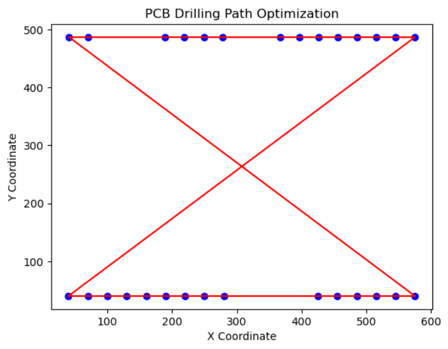


Figure 9: Simulated Annealing Algorithm for 29 Holes

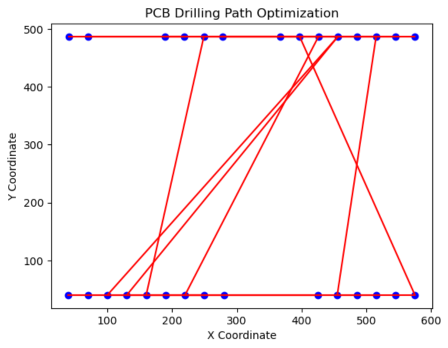


Figure 10: Particle Swarm Optimization for 29 Holes

with brute force algorithms which is not efficient for large datasets. However, the optimization algorithms used in this project is converges to real answer rapidly and helps us to find an approximation solution in a considerably more efficient way. Each algorithm has its strengths and weaknesses. While, according to Plot Figures, genetic algorithm gives the best results (the shortest path), it is also the slowest algorithm among others. On the other hand, Particle Swarm Algorithm is the fastest one. However, it gives the worst results. To conclude, choice of algorithm can be guided by the specific requirements of the application in terms of solution quality and computational efficiency.

10 References

[1] I. Berninger, “Vehicle Routing Problem on Android/iOS,” Bachelor Thesis, Institute of Computer Science Research Group DPS, University Innsbruck, 2014.

[2] A. Homaifar, S. Guan, and G. E. Liepins, “Schema Analysis of the Traveling Salesman Problem Using Genetic Algorithms,” Complex Systems 6, pp. 533-552, 1992.

[3] R. Matai, S. Singh, M.L. Mittal, ”Traveling Salesman Problem: an Overview of Applications, Formulations, and Solution Approaches”, p.2, 2010

[4] T. Narwadi, Subiyanto, ”An Application of Traveling Salesman Problem Using the Improved Genetic Algorithm on Android Google Maps”, 2017 [5] M.J: Kochenfender, T.A. Wheeler, Algorithms for Optimization, the MIT Press [6] https://medium.com/@francis.allanah/travelling-salesman-problem-using-simulated-annealing-f547a71ab3c6 [7] N.M. Razali, J. Geraghty, ”Genetic Algorithms Performance Between Different Selection Strategy in Solving TSP”, 2010