COMPARISON OF IMPROVED ACO ALGORITHMS FOR TOOL PATH OPTIMIZATION IN MULTI-HOLE DRILLING

Luka OLIVARI

Abstract: Optimization of the tool path in the multi-hole drilling process is crucial for reducing the overall cost of production as most of the time is consumed by non-cutting tool movement. Metaheuristics are often used for tool path optimization, with novel and improved algorithms proposed regularly. The observed problem is that proposed algorithms are often compared only to the non-improved versions of metaheuristics or to solutions obtained by CAM software. As expected, improved and novel approaches produce better results than their non-improved counterparts, but it is not known how they compare to the other improved algorithms. Also, the Chebyshev distance matrix is more appropriate for calculating distances in multi-hole drilling. This paper aims to provide data that describes the performance of several improved versions of the ACO algorithm using the Chebyshev distance matrix. By doing so, researchers could easily compare their proposed algorithms to more than just basic ACO to get better feedback about their algorithm performance for tool path optimization.

Keywords: tool path, Ant Colony Optimization, ACO, multi-hole drilling

1 INTRODUCTION

There is a constant effort to optimize different aspects of the CNC machining process like minimizing machining time, airtime, tool path, computational time, tool changing time, and cost, or increasing productivity and surface quality. All of these can be achieved with an improved machine and tool design, parameter, and tool path optimization. [1]

Drilling is an essential operation in CNC machining and multi-hole drilling is a process of drilling a large number of holes in a product. The hole-drilling processes can be divided into single-tool hole drilling (ST) and multi-tool hole drilling (MT) problems. As the name says, in ST only one type of tool is used for drilling, and in MT holes on the workpiece require different tools to be produced. As non-cutting time in the multi-hole drilling process can take up most of the total process time, it is one of the key aspects of optimization in the production industry. That is the main reason why authors often propose new or improved optimization methods for the multi-hole drilling process. Both ST and MT problems are essentially the same from the optimization point of view, because both problems may be represented by Traveling Salesman Problem (TSP). In the ST problem, the cost of the tool moving from hole to hole is represented with a cost matrix, and in the MT problem the cost is the sum of tool change and tool movement which can also be represented with a single cost matrix. [2]

The authors propose three kinds of improvements over the existing optimization algorithms. Completely new algorithms that often draw inspiration or copy some natural process like the mating or hunting habits of animals. Enhanced versions of existing algorithms with improvements to achieve superior performance over the original algorithm. Hybrid algorithms combine multiple algorithms to produce a new algorithm with better performance than its initial components.

Two problems were observed. One problem is that improved or hybrid algorithms for optimization of tool path in multi-hole drilling process are often compared to the basic versions of classical algorithms like Ant Colony Optimization, simple heuristics, or to the tool paths obtained with CAM software, and as expected, achieve better performance. That approach doesn't show how those algorithms compare to the improved versions of existing algorithms. The second problem is that the Euclidian distance matrix doesn't accurately represent travel times or costs in CNC machining [2]. It is because CNC machines use two motors, one for movement along X and the other for the Y axis. If motors are sequentially activated a rectilinear distance matrix would be appropriate. As that is usually not the case, a Chebyshev distance matrix is the most suitable for multi-hole drilling problems because both motors are simultaneously activated, and actual travel time is determined by the longest distance along X or Y axis. As stated in [2] only 19 out of 53 reviewed papers on multi-hole drilling tool path optimization used the Chebyshev distance matrix.

This paper aims to provide comprehensive data on the performance of the improved ACO algorithms applied to the benchmark TSP instances using the Chebyshev distance matrix which can be calculated using expression (1). The goal of this paper is to enable researchers quickly and easy comparison of proposed algorithms for tool path optimization in multi-hole drilling to more than just the basic Ant System.

$$d_{ij} = \max\left(\left|x_1 - x_2\right|, \left|y_1 - y_2\right|\right) \tag{1}$$

Data will also be provided for the performance of the improved ACO algorithms on the same benchmark problems using the Euclidean distance matrix. There are two reasons for that. First, the optimal or best-known solutions for these benchmark problems are known for the Euclidean distances between nodes, but not for the Chebyshev distances. That makes it hard to calculate the relative maximum deviation from the optimum for the optimization algorithm. It is reasonable to expect similar values for the relative maximum deviation for Euclidean and Chebyshev distances, so by providing data for one case, the other can be approximated. The second reason is that a lot of authors use the Euclidean distance matrix in their algorithms, and we want to provide data for easy comparison in that case too.

The following paragraph contains a short literature survey to showcase a comparison of new and improved algorithms to basic heuristics, meta-heuristic, or CAM software results. In [3] authors propose a novel Bat Algorithm (BA) that draws inspiration from bat hunting behavior. The BA algorithm was compared to Genetic Algorithm, Particle Swarm Optimization, Ant Colony Optimization, and Artificial Bee Colony algorithms. In [4] authors apply Genetic Algorithm with a modified cross-over method to a workpiece with 28 holes to optimize the tool path for energy-efficient machining. Results were compared with the tool path generated by Autodesk Inventor. In [5] authors use Artificial Neural Networks and a hierarchical refinement process for fast tool path optimization. Results were compared to those obtained with the greedy algorithm followed by 2-opt. In [6] authors use a simulated annealing algorithm combined with an adaptive large neighborhood search to optimize the laser cutting path. The problem is represented with generalized TSP and results were compared to commercial CAM software and similar scientific papers.

After an introduction, in chapter 2 Ant System, its improvements, and the differences between them are described, test results and discussion are presented in chapter 3, and concluding remarks in chapter 4.

2. ANT COLONY OPTIMIZATION

Ant Colony Optimization (ACO) is a metaheuristic optimization framework that encompasses a variety of algorithms inspired by the stigmergy of the ant colonies in search of food. Ants in nature use the stigmergy mechanism, a form of indirect interaction, by modifying their environment with pheromone trails. Ants, unaware of each other, lay and “read” the pheromone trail to obtain information. An ant that has found the food source deposits a pheromone trail which increases the probability that other ants will follow the same path to the food source. The more ants use that path, and lay their pheromone, the chance for other ants to follow that path increases.

Artificial ants in ACO algorithms have a similar approach which can be used to solve a wide range of optimization problems, especially problems that involve finding the shortest path in a graph, such as the Traveling Salesman Problem (TSP) and Vehicle Routing Problem (VRP). Ant System (AS) was the first algorithm in the ACO metaheuristic optimization framework. Although AS didn’t perform as well as other algorithms for solving TSP at that time, its main value is that it inspired many other ACO algorithms with significantly improved performance. Some of those improved algorithms are Elitist AS, Ranked AS, and MAX-MIN AS. The main difference between an original AS and its “classical” improvements is in the pheromone update mechanism. [7]

Pheromone is updated after all ants in the current iteration have completed their tours. The first step of pheromone update is pheromone evaporation, i.e. lowering pheromone levels by a constant percentage, after that, new pheromone levels are added. Pheromone evaporation is regulated with parameter ρ to stop pheromones from rising infinitely and to deter ants from exploring bad tours and edges which accumulated pheromones by chance. Parameter ρ can be set in a range from 0 to 1, and it represents the percentage of how much of accumulated pheromone evaporates between iterations.

After evaporation, ants deposit pheromones on the edges used for constructing their tours. The amount of pheromone Δτk deposited by ant k depends on the tour quality. The shorter the tour the more pheromone is deposited. Edges used by many ants with short tours receive a lot of pheromones so there is a higher probability for that edges to be chosen in the next iterations. Pheromone evaporation and deposition are modeled according to the expression (2).

$$\tau_{ij}^{k} = \left(1-\rho\right)\tau_{ij} + \sum_{k=1}^{m} \Delta\tau_{ij}^{k} \tag{2}$$

Elitist Ant System (EAS) increases amounts of pheromone deposition on the edges that make the best-so-far solution Tbs created since the beginning of the process. This tour can be viewed as an additional elite ant that deposits a pheromone in each iteration. The length of Tbs is denoted by the Cbs. The amount of additional pheromone added to the best-so-far solution is calculated by e/Cbs, where e is the weight given to the best-so-far solution.

Ranked-based Ant System (RAS or ASrank) is an extension of the Elitist Ant System in which amounts of deposited pheromone depends on the ant’s rank r, i.e. the quality of the solution obtained by each ant. As in EAS the ant with the best-so-far solution deposits the most pheromones in each iteration, the second best-so-far ant deposits fewer pheromones, etc. The number of ants that deposit pheromone is denoted with w and the amount of pheromone deposited by the best-so-far is calculated by w·1/ Cbs and by the rest of the ranked ants is (w-r) ·1/ Cr, where Cr is the length of the solution by rth ant.

High-quality solutions are obtained by combining ACO algorithms with Local Search because they complement each other. In fact, the definition of the ACO framework allows using local search. There are lots of ways to combine Local Search with ACO algorithms, a general rule is the better solution comes at the price of longer computation time. We used the 2-opt algorithm, as one of the basic Local Search algorithms. 2-opt switches two pairs of edges in all possible combinations for the given tour to eliminate any cross-overs of edges. For in-depth descriptions of the Ant System, its extensions and possible combinations with the Local Search reader are directed to [7].

3 EXPERIMENT AND DISCUSSION

As both ST and MT problems can be represented with Traveling Salesman Problem, Ant System and its improved versions were applied to TSP benchmark problems eil51 and eil101 found in TSPLIB [8]. The best-known solution for eil51 is 426, and for eil101 is 629. The parameters used for each algorithm are shown in table 1. Used parameters are recommended parameters for general good performance from [7], except for the number of ants. Preliminary testing showed a significant increase in computation time at the price of slightly worse solutions. Using 101 ants for the eil101 Ant System produced an average solution of 696,5 in 85 seconds, while Rank-based AS produced an average solution of 651,5 in 80 seconds. Parameter β has recommended value in the range 2 – 5, we used the upper limit $\beta = 5$, as it produced better results in preliminary testing. Initial pheromone levels τ_0 are calculated according to the equation in the table, where C^{nn} is the length of tour obtained with the Nearest Neighbor algorithm.

Table 1 ACO parameters

	α	β	ρ	m	τ_0	
AS	1	5	0.5	25	m/C^{nn}	-
EAS	1	5	0.5	25	$(e+m)/\rho C^{nn}$	$e = m$
RAS	1	5	0.1	25	$0.5r(r-1)/\rho C^{nn}$	$w = 6$

As mentioned before, there are lots of ways to combine ACO with Local Search algorithms. In our case, we used the least computationally intensive Local Search algorithm, 2-opt, and apply it after all ACO iterations to improve only the final solution. No special speed-up techniques were used for ACO algorithms nor 2-opt which means that faster performance is possible.

Testing was performed on Windows 10 64-bit operating system using MATLAB R2021a. The computer configuration was Intel(R) Core(TM) i5-10210U CPU 2.10 GHz, installed RAM 8 GB. Code used can be found on the GitHub repository [9]. The stopping criteria were 300 iterations, and every algorithm was run 20 times to average out results.

Algorithm performance was analyzed using the following metrics: 1) best solution overall, 2) average solution quality, 3) relative standard deviation, 4) relative maximum deviation, and 5) average execution time.

The best solution overall is the best solution produced by the given algorithm in all runs. Average solution quality is calculated by the mean of the best solutions produced in all 20 runs by a single algorithm. Relative standard deviation is calculated by dividing the standard deviation from the optimal solution and given in form of a percentage. It shows how precisely the algorithm produces solutions close to the optimum. The maximum deviation is calculated by dividing the worst final solution in all runs by the optimal solution and given in form of a percentage. It shows the worst expected performance of the algorithm with given parameters. It is used only for the Euclidean distance matrix because the optimal solution using the Chebyshev distance matrix isn't known. Average execution time is wall-clock time for a single run of the algorithm i.e., time to compute all iterations until stopping criteria are met, including the Local Search. In table 2. and table 3. test results are presented. The numbers in the first column correspond to the numbers before the metrics in the text above.

Table 2 Test results, Chebyshev distance matrix

eil51	AS	EAS	RAS	AS + 2-opt	EAS + 2-opt	RAS + 2-opt
1)	378	**377**	**377**	**377**	**377**	**377**
2)	388	382,2	379,3	381,5	379	**378,2**
3)	0,8%	0,86%	**0,54%**	1,17%	0,61%	0,6%
4)	-	-	-	-	-	-
5)	8,1 s	7,9 s	7,9 s	8,1 s	**7,8 s**	7,9 s
eil101	AS	EAS	RAS	AS + 2-opt	EAS + 2-opt	RAS + 2-opt
1)	595	569	570	576	569	**565**
2)	623	591,3	587,8	590	579,2	**576**
3)	2,32%	3,04%	2,86%	2,16%	1,6%	**1,39%**
4)	-	-	-	-	-	-
5)	20,8 s	21,1 s	**19,4 s**	21,7 s	21,1 s	19,5 s

Table 3 Test results, Euclidean distance matrix

eil51	AS	EAS	RAS	AS + 2-opt	EAS + 2-opt	RAS + 2-opt
1)	449,6	429,5	430,4	431,3	**429**	429,1
2)	457,9	437,4	435,4	439,2	433,1	**432,8**
3)	1,02%	1,28%	0,87%	1,15%	**0,71%**	0,72%
4)	8,77%	5,35%	4,67%	4,92%	**4,16%**	**4,16%**
5)	8,1	8,1	**7,6 s**	8,3 s	8,1 s	7,7 s
eil101	AS	EAS	RAS	AS + 2-opt	EAS + 2-opt	RAS + 2-opt
1)	691,3	642,5	649	657,4	643,1	**642**
2)	705,7	669,1	661,3	671	653,2	**649,5**
3)	1,25%	2,31%	1,59%	1,3%	1,14%	**1,13%**
4)	14,74%	11,95%	9,05%	9,13%	5,93%	**5,79%**
5)	20,5 s	20,5 s	**19 s**	21,2 s	21,3 s	19,7 s

As expected, Rank-based AS produced slightly better average results than Elitist AS, which in turn produced better results than the basic Ant System. Applying 2-opt improved all average solutions in all three algorithms. The same best overall solution for eil51 and Chebyshev distance matrix was achieved by all algorithms except AS, while the best average solution was produced by RAS and 2-Opt. The best overall solution for eil101 was produced with RAS combined with 2-opt, while the best overall solution for eil51 and Chebyshev distance matrix RAS and 2-opt. The worst maximum deviation for the Euclidean distance matrix was produced by

Ant System, and it can be concluded that the worst maximum deviation for the Chebyshev distance matrix could also be expected from Ant System. Rank-based AS, without 2-opt, had the lowest computational time. The reason for that is the shorter pheromone update calculation, because in RAS only the top *w* number of ants deposit pheromones, while in other algorithms all ants deposit pheromones.

Coupling ACO algorithms with better Local Search algorithms like 3-Opt or Lin-Kernighan Heuristic could produce much better results at the price of longer computation time. Longer computation could be somewhat countered with the use of speedup techniques for ACO and Local Search algorithms.

4. CONCLUSIONS

Data on the performance of the Ant System and improved ACO algorithms, Elitist Ant System, and Rank-based Ant system using the Chebyshev distance matrix for TSP is provided. Data can be used for comparison of newly proposed or improved algorithms for tool path optimization in multi-hole drilling. Of course, used algorithms could achieve much better performance if parameters were fine-tuned for specific problems, but as the main purpose of this data is to enable quick and easy comparison of newly proposed algorithms to the more than just Ant System, it is more appropriate to use recommended parameters for generally good performance.

For future research, algorithms could be applied to other benchmark and industrial problems, especially larger ones. More improved ACO algorithms could be included for easy comparison, as well as improved versions of other heuristics and metaheuristics. Also, a framework for comparison of meta-heuristics would be very useful as well as benchmark problems specific to multi-hole drilling.

5. REFERENCES

[1] Narooei KD, Ramli R. Application of artificial intelligence methods of tool path optimization in CNC machines: A review. Res J Appl Sci Eng Technol. 2014;8(6):746–54.

[2] Dewil R, Küçükoğlu İ, Luteyn C, Cattrysse D. A Critical Review of Multi-hole Drilling Path Optimization. Arch Comput Methods Eng [Internet]. 2019 Apr 22;26(2):449–59. Available from: http://link.springer.com/10.1007/s11831-018-9251-x

[3] Diyaley S, Burman Biswas A, Chakraborty S. Determination of the optimal drill path sequence using bat algorithm and analysis of its optimization performance. J Ind Prod Eng. 2019;36(2):97–112. Available from: https://doi.org/10.1080/21681015.2019.1585974

[4] Raja B, Saravanan M. Tool Path optimization by Genetic algorithm for Energy Efficient Machining. 2018;14:1670–9.

[5] Fok KY, Ganganath N, Cheng CT, Iu HHC, Tse CK. Tool-path optimization using neural networks. Proc - IEEE Int Symp Circuits Syst. 2019;2019-May(c):1–5.

[6] Hajad M, Tangwarodomnukun V, Dumkum C, Jaturanonda C. Solving the laser cutting path problem using population-based simulated annealing with adaptive large neighborhood search. Key Eng Mater. 2020;833 KEM:29–34.

[7] Dorigo M, Stützle T. Ant colony optimization. 2004. 319 p. Available from: https://mitpress.mit.edu/books/ant-colony-optimization

[8] http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/ (accesed: 16.1.2023.)

[9] https://github.com/l-olivari/ICIL2023 (accesed: 1.4.2023.)

Authors' contacts:

Luka Olivari, mag. ing. mech.
Polytechnic of Šibenik
Trg Andrije Hebranga 11, Šibenik, Croatia
lolivari@vus.hr