MZZ-GA Algorithm for Solving Path Optimization in 3D Printing

Mateusz Wojcik, Leszek Koszalka, Iwona Pozniak-Koszalka and Andrzej Kasprzak
Department of Systems and Computer Networks
Wroclaw University of Technology
Wroclaw, Poland
e-mail: wojcik.mateusz991@gmail.com, {leszek.koszalka, iwona.pozniak-koszalka, andrzej.kasprzak}@pwr.edu.pl

Abstract— **This paper is focused on fused deposition process which is one of the technologies that can be used in rapid prototyping process. This process is divided into four different stages, one of which is path planning. This stage has a remarkable impact on the overall timing of the printing process. In this paper the implemented algorithms for solving the path optimization problem are presented. The properties of the implemented MZZ-GA algorithm are investigated, with the use of the designed two-stage experimentation system. Basing on the obtained results, we can conclude that the proposed approach seems to be promising.**

Keywords-algorithm; pathfinding; printing; optimization; simulation experiments

I. INTRODUCTION

Nowadays rapid prototyping is one of the fastest growing technologies. This process allows for creating a solid object without any specific tooling. The main advantage of this process is the ability to create a very complex object, in short time. There are several different systems which can be used in this technology, including: (i) Stereo-lithography (SL), (ii) Selective laser sintering (SLS), and (iii) Fused deposition modeling (FDM). In this paper, we use FDM which belongs to the so-called Layered Manufacturing (LM) technology, in which a solid object is produced by the deposition of material layer. The object in LM has to be processed before printing. According to [1] this process requires the completion of the four main tasks:

- Object orientation - the best orientation for the object is determined.
- Support generation – the additional element is generated in order to holds the parts of the object (after printing these additional elements can be dissolved).
- Slicing – a special algorithm extracts the layers (in the vector form) from the object. The 3D model is converted into the 2D images.
- Path planning – the algorithm plans the moves of the extruder.

For the purposes of this paper we have focused on the last task. The path planning problem can be divided into two different sub-problems: (i) path generating, and (ii) path optimization. To solve the first sub-problem an algorithm should generate and group the tool path segments into individual sub-paths. The paths can belong to one of the two groups: the contour paths or the raster paths. The raster paths are filling the interior section of the layer (always after contours). To solve the second sub-problem, an algorithm should optimally link the sub-paths which were found previously. The criteria for the optimality that should be met include: best possible surface quality, minimum tool wear, shortest machining time achieved, or minimum machining cost.

There are several different algorithms for solving these sub-problems. The algorithms proposed for solving path generating problem are based on strategies, such as: Raster [2], ZigZag [3], Contour [4], Spiral [5], and Fractal space curves [6]. The algorithms proposed for solving path optimization problem are based on approaches, such as: Genetic Idea [7], Adaptation of TSP [8], and Neural Network [9].

In this paper, the algorithm called MZZ-GA is proposed for solving path planning problem. This algorithm enables path generating with the designed Modified Zig Zag (MZZ) algorithm, and next, using the obtained result, it solves path optimization problem with the implemented Genetic Algorithm (GA). The properties of MZZ-GA are evaluated with a designed and implemented experimentation system.

This paper is organized as follows. In Section II, the related works are discussed. In Section III, the formal model of the considered problem is stated. Section IV presents the proposed algorithms for a solution to the problem. Findings from computational experiments are presented in Section V. Conclusion and plans for further research in the area appear in Section VI.

II. RELATED WORK

A. Paths generating algorithms

In paper [10], it is showed how to improve planning process for Rapid Prototyping / Manufacturing for the complex product models, such as biomedical models. That work contains a description of several different algorithms, including a path generating algorithm used to generate contour tool–paths along the boundary. Also, the zig-zag tool-paths used to generate paths for the internal area of the layer are described. The most interesting is the proposition for the solution of the path optimization problem with the use of zig-zag algorithm in which the variable for the optimization is the slope of the zig-zag direction. The most important conclusion of that work is that the mixed tool-path algorithm can balance the geometrical quality and the effectiveness.

 ISBN: 978-1-61208-399-5

Paper [11] presents the Zig-Zag algorithm. This algorithm was developed for 3-axis computer numerical control (CNC) machine. In this algorithm the tool moves in a straight line in a feed-forward direction. Also, an algorithm for finishing operation on the machines is described. Because that algorithm was developed for CNC machines, the authors of [11] have also developed a tool holder collision detection algorithm.

B. Path optimization algorithms.

In [8] two different methods for path optimization in Layered Manufacturing are presented. One of them is based on genetic approach, and the other one is based on the strategy which is used for solving a combination of Asymmetric Traveling Salesman Problem and Assignment Problem (TSP-Assign). The authors of [8] formulated the problem of path finding as constructing a set of curves on a layer, from which the algorithm should find the optimum sequence and direction of curves. They conclude that GA optimization can improve path planning tremendously, but this method is computationally expensive. Moreover, they state that TSP-Assign algorithm for path optimization was even more time-consuming than the GA approach. Also, an approach which combines GA and local search approach is proposed. This technique may improve path planning by reducing the jump distance by up to 50%.

The authors of paper [12] propose two different algorithms. The first is based on a simple greedy option (nearest neighbor procedure). A heuristic algorithms starts from the upper right corner and in every step adds points which belong to the counter that has not been visited yet. After the algorithm has visited all corners it begins searching for the path in the internal area. The second algorithm is based on the combination of the nearest and the farthest insertion method. At the beginning, it adds a point which belongs to the first corner. After that it checks if it should add a point which belongs to the second corner. The authors of [12] draw the conclusion that depending on geometry different methods can give better or worse results. The first algorithm produces better results when there are only a few counters and a few continuous raster segments. If the number of those objects increases, the second algorithm can produce better results.

III. PROBLEM STATEMENT

In this paper, we concentrate mainly on the path planning problem, in particular on the path optimization on the single layer. In the mathematical form, the problem can be stated as follows:

Given:

- Printing layer as a set of the binary points (an example of the layer can be seen at Figure 1):

$$X_{ij} = \begin{cases} 1 \; if \; printing \; point \\ 0 \; otherwise \end{cases} \; i \in n \; j \in m$$

(1)

- Lengths between two points defined by (2):

$$L_{X_{ab}\, X_{cd}} = \sqrt{(a-c)^2 + (b-d)^2}$$

(2)

where:

- n – Lengths of the printing layer,
- m – Width of the printing layer.

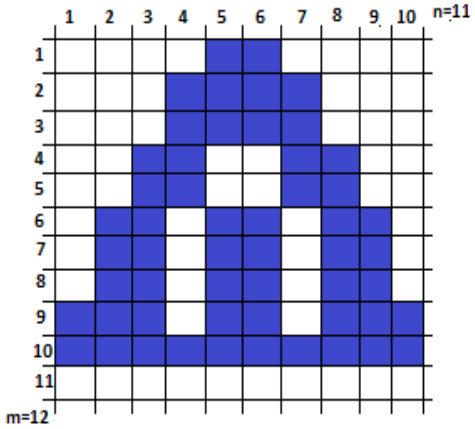

Figure 1. Example of the printing layer (grey points refer to printing area).

To find:

- V – order of points that will be visited:

$$V = [X_{ab}\, X_{cd}\, X_{ef\ldots}]$$

(3)

- The decision about using the tool to extrude the plastic on the single point:

$$U_i = \begin{cases} 1 - if \; V_i will \; be \; printed \\ 0 - otherwise \end{cases}$$

(4)

- The total length of the final path:

$$L = \sum_{k=1}^{p} ((V[k] - V[k-1]) * U_i)$$
$$= L_{printed} + L_{switching\ path},$$

(5)

where p is the number of the visited points.

Such that:

- *Lopt = min(L).*

Subject to constraints:

- Each point can be visited only once:

$$V[i] \neq V[j].$$

IV. ALGORITHMS

There were implemented two different algorithms for paths generating and one algorithm for path optimization. In this section, the both algorithms are presented in detail.

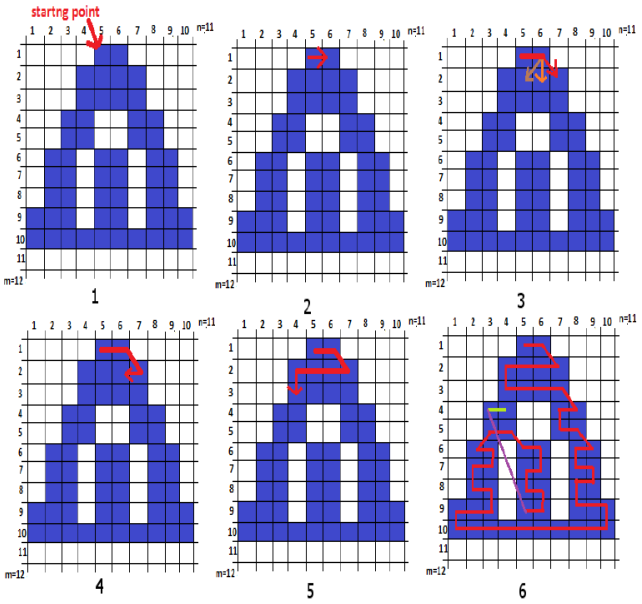

Figure 2. An example - the MZZ algorithm in action.

A. Modified Zig Zag algorithm

The MZZ algorithm works in the following way:

1. Start from top, left corner (Figure 2.1) – extruder's home position is point (0,0); because of that top left corner will be the closest point from extruder.
2. Move into the right corner (Figure 2.2) – next step is to select all points which lay to the right of the first pixel.
3. Check the bottom pixels (Figure 2.3) – when there are no more pixels on the right, the algorithm checks starting from the right corner below if there are pixels available.
4. Check the pixels above – when previous step ends with no results, algorithm checks the pixels above. Otherwise the algorithm skips this step.
5. Choose the pixel in the rightmost position (Figure 2.3) - if any of those pixels is available – i.e. it has not been visited yet – the rightmost pixel is selected.
6. Check the pixels to the left (Figure 2.4) – after step 4 the algorithm starts checking and selecting pixels which lay to the left of the pixel which has been chosen in step 4.
7. Check the pixels below (Figure 2.5) – when there are no more pixels to the left, the algorithm checks if the pixels below are available. It starts from the lower left corner.
8. Check the pixels above– when the previous step ends with no results, the algorithm checks the pixels above. Otherwise the algorithm skips this step.
9. Choose the pixel in the leftmost position (Figure 2.5) - if any of those pixels is available – i.e. it has not been visited yet – the lower leftmost pixel is selected.
10. Repeat the procedure – the algorithm repeats the steps from step 2 to 9. This creates the zig-zag-shaped paths.
11. When there are no more pixels available to select in the neighborhood of the previous pixel, but there are some available in the whole layer, the algorithm goes to step 1.

The important difference as compared to the standard Zig-zag algorithm appears, when the MZZ algorithm finishes checking available pixels on the bottom. Standard Zig-Zag algorithm will now start a new path, while the MZZ algorithm will also check the pixels above. If there are any pixels available there, it will continue adding those pixels (Figure 2.6).

B. Genetic Algorithm for path generating

GA as path generating algorithm works in the same way as the standard GA [7]. It starts with generating population. The single solution in this population is the path which contains all the points of the layer. The initial solutions in the population are generated randomly. After this GA selects the best solutions from the population, i.e., the algorithm selects the "m" solutions with the shortest path lengths. The next step is to crossover the paths to make another population. The crossover process is not so complicated. At first the algorithm randomly chooses two parents – two different paths from which the crossover will be made. Then algorithm selects "l" random points from the first parent. The rest of the points belong to the second parent. As in standard GA, in "p" percent of the new paths random mutations occur. If the mutation occurs, two points are swapped. The whole procedure is repeated "t" times.

C. Genetic Algorithm for path optimization

This algorithm is responsible for optimizing the path that is already found by MZZ algorithm. The standard MZZ algorithm connects sub-paths as they are found. This means that at the beginning the MZZ algorithm finds the first path, then the second, and so on. After that it connects the last point from the first path with the first point from the second path, and so on.

In the proposed implementation the GA looks for the best linking of the sub-paths found by MZZ algorithm. An example of the implemented GA can be seen in Figure 3, where 3(a) shows sub-paths which were found by the MZZ algorithm; 3(b), 3(c) and 3 (d) are possible final paths which appear after linking of the sub-paths.

The process of this algorithm goes like that of the standard Genetic Algorithm. At first it generates population. The single solution in this population is the path which contains all sub-paths. The first population is generated

 ISBN: 978-1-61208-399-5

randomly. After that the "m" best solutions are selected. In this case the better solution is when the path is shorter. The next stage is to crossover the paths to make the next population. This process is done as follows. At first it selects two parents – paths that will be used to make a new solution.

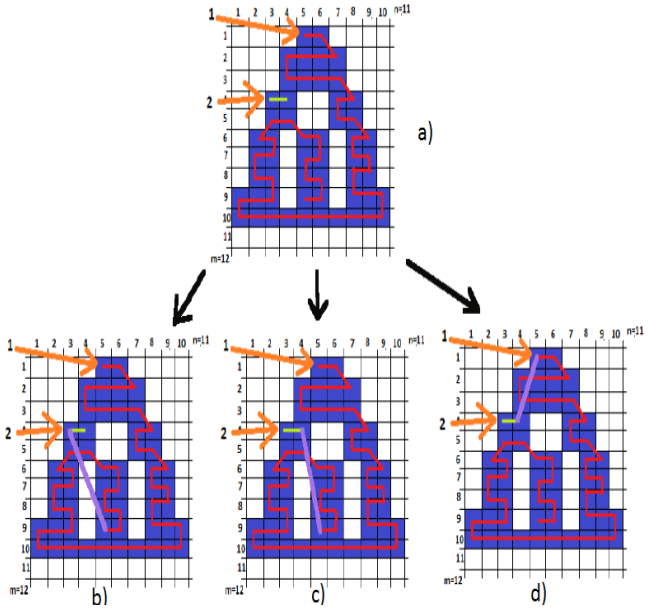

Figure. 3. An example of possible paths made from linking sub-paths.

The new solution contains the "l" sub-paths from the first parent, and rests of the sub-paths belong to the second parent. The parameter "l" is chosen randomly. The next process is mutation. A small percentage of the whole population passes through this process. In this process the randomly chosen sub-paths are swapped.

V. INVESTIGATION

A. Software

The experimentation system, i.e. the application for testing algorithms has been designed by the authors of this paper and implemented in Java. The experiments were carried out on computer with Intel Core i5-3210M CPU 2.50GHz.

B. Experiment design

We took into consideration four different objects denoted as Pic. 1, Pic. 2, Pic. 3, and Pic. 4. All objects can be seen in Figure 4. The main differences between the objects were in the number of pixels and complexity. The first three objects had a small number of pixels and were not complex. The last object had a much bigger number of pixels and was much more complex than the other objects.

Two-stage experiments were carried out in the following way:

1. In the first stage of the experiment the algorithms GA path generating and MZZ path generating were tested for all the objects. In order to adjust the internal parameters, the GA path generating algorithm was tested in respect of two internal parameters: the number of population and the number of epoch.

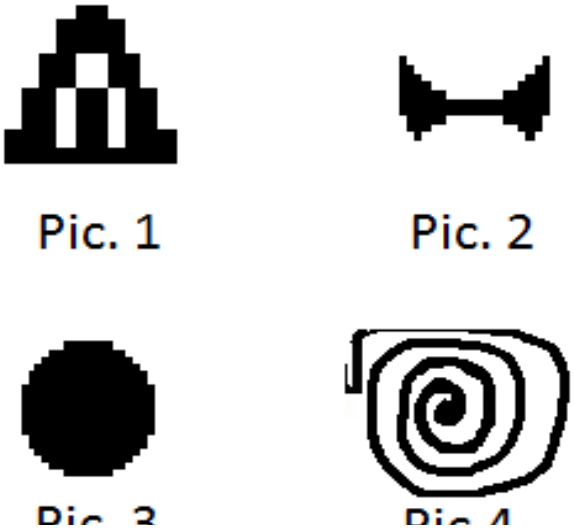

Figure. 4. Four different objects which are tested during the experiment.

2. In the second stage of the experiment the optimization algorithm MZZ-GA was tested. In this experiment GA was linking sub-paths which were found by the MZZ algorithm. Also, the influence of the number of population and the number of epoch on the obtained results was tested.

The experiments with GA path generating algorithm and MZZ-GA path optimization algorithm were carried out with different number of epoch (NP) and different number of population (NE). Three different values of NP and NE for each of the tested objects were planned.

As the result of a single experiment two values were taken:

- The execution time of the algorithm,
- The length of the founded path.

Because the GA contains random operations, the single experiments were repeated ten times. Therefore, the averaged values of two metrics were considered as the indices of the performance. Those metrics are:

- The average execution time (AET),
- The average length of the founded path (ALP).

Moreover, the standard deviation was calculated (sdLP).

C. Results of experiments

Table 1 shows the results of the experiments in the first stage for GA path generating algorithm. Table 2 shows the final results of MZZ-GA, i.e. the results of GA path optimization connected to MZZ.

It may be observed in Table 1 that GA path generating gave better results when the number of epoch (NE) was bigger than the number of population (NP).

The same conclusion can be drawn from Table 2. However, in this case, for objects with low number of pixels and less complexity, there is no need to use such a big number of populations and of epoch. For the objects named

 ISBN: 978-1-61208-399-5

as Pic. 1, Pic. 2, and Pic. 3, the same results of ALP were obtained.

TABLE 1. RESULTS OF GENETIC ALGORITHM AS PATH GENERATING ALGORITHM.

	GA PATH GENERATING				
	AET [MS]	SDLP	ALP	NP	NE
PIC. 1	139	4.5	105.9	100	100
	1783	0.9	108.25	1000	100
	939	5.4	84.9	100	1000
PIC.2	176	2.2	130.4	100	100
	1062	2.8	135.0	500	100
	912	1.8	126.5	100	500
PIC. 3	1216	3.4	538.1	100	100
	4170	1.1	541.6	300	100
	3996	3.1	555.7	100	300
PIC. 4	606	8.7	10903.0	10	10
	8698	1.8	10912.0	100	10
	7201	25.2	10873.0	10	100

In the case of the object denoted as Pic. 4, the one with a high number of pixels and high complexity, using bigger number of epoch gave better results than using bigger number of population. Moreover, for those parameters the solution was found in shorter time.

TABLE 2. RESULTS OF GENETIC ALGORITHM AS PATH OPTIMIZATION ALGORITHM.

	GA PATH OPTIMIZATION				
	AET [MS]	SDLP	ALP	NP	NE
PIC. 1	93	0	57.2	100	100
	1416	0	57.2	1000	100
	542	0	57.2	100	1000
PIC. 2	71	0	108.7	100	100
	1392	0	108.7	1000	100
	583	0	108.7	100	1000
PIC. 3	55	0	304.0	100	100
	1325	0	304.0	1000	100
	534	0	304.0	100	1000
PIC. 4	781	51.4	4016.6	100	100
	4244	33.4	4015.7	500	100
	4065	37.3	3798.9	100	500

Table 3 shows the results of both indices of performance (ALP and AET) for all algorithms. The best results obtained for GA path generating (from Table 1) are shown in column GA-GEN, and the best results obtained for MZZ-GA optimization (from Table 2) are shown in the column GA-OPT. The column MZZ presents the results obtained with MZZ path generating algorithm.

TABLE 3. COMPARISON OF ALL ALGORITHMS

No. OF PIXELS	GA-GEN		MZZ		GA-OPT	
	AET	ALP	AET	ALP	AET	ALP
55	939	84.9	0	5.5	93	57.2
89	912	126.5	0	129.5	71	108.7
294	1216	538.1	1	484.2	55	304.0
2335	7201	10873.0	12	4158.8	4065	3798.9

It can be seen that the GA path generating produced the worst results – ALP and AET are the highest. The path length for the complex object (Pic. 4) is twice as big as the length of the path found by MZZ algorithm. From the results of MZZ algorithm it can be seen that work time in this case is the shortest, and that path lengths are almost in every case better than in cases when they were given by GA path generating algorithm. The results for GA path optimization algorithm are the best. It can be seen that for any object the GA significantly improved path lengths found by MZZ algorithm. It is also evident that the execution times are much shorter than those produced by the GA as path generating, but longer than for simple MZZ algorithm.

VI. CONCLUSION

On the basis of the obtained results of experiments, it can be concluded that MZZ algorithm is not the best algorithm that can be used for path finding individually. Also, the classic genetic path generating algorithm should not be used as a path generating algorithm.

The genetic algorithm certainly should be recommended (as the path optimization algorithm) when it is used together with MZZ path generating algorithm as a combined MZZ-GA algorithm.

When using MZZ-GA algorithm, it is necessary to bear in mind that its merits are governed by the reasonable number of epoch and the proper number of population (see Table 2). However, the user may face some difficulties - for the bigger number of pixels it will be difficult to use the same number of epoch and the same number of population as for smaller data.

In further research in the area the authors are planning to consider more algorithms for path generating and path optimization based on other evolutionary approaches, e.g. the one presented in [13]. In particular, using the hybrid approach and contour approach in constructing effective algorithms seems to be very promising.

There are also several interesting issues that might be discussed in future work in this area, such as designing and implementing experimentation systems to conduct the multistage experiments in the automatic manners, along with the issues presented in [14].

 ISBN: 978-1-61208-399-5

ACKNOWLEDGMENT

This work was supported by the statutory funds of the Department of Systems and Computer Networks, S40029_K0402, Wroclaw University of Technology, Poland.

REFERENCES

[1] K. P. Venuvinod and M. Weiyin, Rapid prototyping Laser-based and Other Technologies, Springer, 2004.

[2] M. R. Dunlavey, "Efficient polygon-filling algorithms for raster displays", ACM Trans. Graph., 1983, pp. 264–273.

[3] S. C. Park and B. K. Choi, "Tool-path planning for direction-parallel area milling", Comput Aided Design, vol. 32, 2000, pp. 17–25.

[4] R. Farouki, et. al., "Path planning with offset curves for layered fabrication processes", J. Manuf. Syst., vol. 14, 1995, pp. 355–368.

[5] H. Wang, et. al., "A metric-based approach to two-dimensional tool-path optimization for high-speed machining", J. Manuf. Sci. Eng., 2005, pp. 127-133.

[6] P. Kulkarni, et. al., " A review of process planning techniques in layered manufacturing", Rapid Prototyp. J., vol. 6, 2000, pp. 18–35.

[7] J. Balic, A. Nestler, and G. Schulz, "Prediction and optimization of cutting conditions using neural networks and genetic algorithm", J. Mech. Eng., Assoc. Mech. Eng. Tech. Slovenia, ISSN 0039-2480, 1999, pp. 192–203.

[8] P. K. Wah, K. G. Murty, A. Joneja, and L. C. Chiu, "Tool path optimization in layered manufacturing", IEE Trans., vol. 34, 2002; pp. 335–347.

[9] J. Balic and M. Korosec, "Intelligent tool path generation for milling of free surfaces using neural networks", International Journal of Machine Tools & Manufacture, vol. 42, no. 10, 2002, pp. 1171-1179.

[10] G. Q. Jin, W. D. Li, and L. Gao, "An adaptive process planning approach of rapid prototyping and manufacturing", Robotics andComputer-Integrated Manufacturing, vol. 29, 2013, pp. 23–38.

[11] D. Misra; V. Sundararajan, and P. K. Wright, "ZigZag tool path generation for sculptured surface finishing", DIMACS Series in Discrete Mathematics and Theoretical Computer Science, 2003.

[12] N. Volpato, R. T. Nakashima, L. C. Galvão, A. O. Barboza, P.F. Benevides, and L. F. Nunes, "Reducing repositioning distances in fused deposition-based processes using optimization algorithms," Advanced Research in Virtual and Rapid Prototyping. London: CRC Press - Taylor and FrancisGroup, vol. 1. 2013, pp. 417-422.

[13] D. Ohia, L. Koszalka, and A. Kasprzak, "Evolutionary algorithm for solving congestion problem in computer network", Lecture Notes in Computer Science, Springer, vol. 5711, 2009, pp. 112-121.

[14] L. Koszalka, D. Lisowski, and I. Pozniak-Koszalka, "Comparison of allocation algorithms for mesh structured networks using multistage simulation", Lecture Notes in Computer Science, Springer, vol. 3984, 2006, pp. 58-67

 ISBN: 978-1-61208-399-5