ORIGINAL ARTICLE

A Novel Approach to Optimize Numerical Control Codes Using a Systematic Block Management Method

Uon Leapheng, Calvin Alexander Ng, Francis Gregory Ng*, Sharaful-Ilmi Paduman, and Alvin Chua

Mechatronics Research Laboratory, Mechanical Engineering Department, De La Salle University - Manila

(**Received** 05 January 2018; **Accepted** 29 March 2018; **Published on line** 1 March 2019)

*Corresponding author: francis_gregory_ng@dlsu.edu.ph

DOI: 10.5875/ausmt.v9i1.1728

Abstract: The numerical control (NC) codes generated from Computer Aided Manufacturing (CAM) software follow the sequence of adding elements to the design. As an alternative, it would be quite beneficial in cost-reduction to optimize the manufacturing sequence to minimize the run time. Accordingly, this paper introduces a novel approach to optimize the numerical control codes generated from a CAM software package using a systematic block management method. To improve the drilling sequence, contours are also considered as block entities in a traveling salesman problem (TSP) with modifications to systematically manage the blocks of code that define the contours. A MATLAB program was created to automatically optimize the numerical codes from a CAM software translator. Because of easier usage and good performance, a partitioned approach for the λ-opt algorithm was implemented instead of the Lin-Kernighan Heuristic (LKH) that has been extensively proven to be effective in optimizing the traveling salesman problem. The new approach was simulated on sample designs and has been shown to achieve at least 15.52% reduction in airtime and 11.32% reduction in tool travel distance. Experimental data showed a 13.45% decrease in total milling time.

Keywords: Computer-aided manufacturing; Numerical Control; Optimization; Traveling salesman problem.

Introduction

Computer Numerical Control (CNC) milling machines are some of the most fundamental and important manufacturing machines. Interestingly, the production of holes and curves makes up a significant majority of the processes that are required to produce industrial parts. Additionally, the time taken to travel between them is significant [1]. This means that it is beneficial to generate a connecting path that that is as short as possible, in order to minimize the time spent in moving from one position to the next.

The problem of generating the shortest possible route to connect multiple points is called the Traveling Salesman Problem (TSP). However, the TSP is categorized as NP-hard, which means that the direct approach to generate an optimal solution becomes impractically difficult when given a large number of points [2].

Exact algorithms such as the cutting-plane and facet-finding algorithms are quite complex and would require codes with tens of thousands of lines [3]. Even though these would be faster than a brute-force method, such exact algorithms would normally take quite long to process due to the difficult nature of the problem. To remedy this, numerous heuristic-based approaches have been proposed which converge very closely to the optimal solution but require significantly less computational effort.

Numerous studies have used various metaheuristics on the traveling salesman problem [4] [5] [6] [7] [8] [9]. However, these mostly just attempt to prove the effectiveness of the algorithms in solving NP-hard problems which could then be applied elsewhere after some reconfigurations.

For actually determining the solution to the TSP, the Lin-Kernighan heuristic (LKH) is deemed as one of the best methods, in solving it. It is a relatively old TSP heuristic algorithm and has been tailored specifically for the TSP. It was implemented by Lin and Kernighan in 1971 and

afterwards, was definitively improved by Helsgaun [3]. A modified version is available online [10] but it is comprised of thousands of lines of code written in ANSI C. It is therefore quite portable but the underlying process would be difficult to understand. Still, the basis of the modified Lin-Kernighan remains the same which is to manage λ-opt moves.

A basic version would be to merely implement the fundamental 2-opt algorithm. This would be computationally faster than using LKH in its entirety and would be more appropriate for investigating the block management method of this study. The 2-opt algorithm functions by dividing the path into two links and puts them back together in a new way by reversing one of the links. λ=2 and λ=3 are the most commonly used but it is difficult to know the effect on the running time and on the quality of the solution. This is what LKH addresses by being a variable λ-opt algorithm [3]. Major refinements have been done, particularly the introduction of a set of rules that avoids ineffective exchanges.

For the application of TSP in terms of NC code optimization, printed circuit board (PCB) drilling models have been optimized by [11] and their results show that the tool path can be reduced by around 70%. The process made use of the LKH but it is evidently limited to drilling operations.

For general milling, there have been commercial software solutions that optimize the NC code, but only the feedrate of tool operations that are done completely in air [12] [13]. Studies have also been done on the spindle speed [14] and material removal rate [15].

Uon Leapheng is a student of De La Salle University-Manila, earning a Master of Science degree in Mechanical Engineering. He is a scholar supported by the ASEAN University Network (AUN)

Calvin Alexander Ng is a student of De La Salle University-Manila, earning both a Bachelor of Science degree and a Master of Science degree in Mechanical Engineering through a ladderized program. He is a scholar supported by the Engineering Research and Development for Technology (ERDT) of the Department of Science and Technology (DOST).

Francis Gregory Ng is a student of De La Salle University-Manila, earning both a Bachelor of Science degree and a Master of Science degree in Mechanical Engineering through a ladderized program. He is a scholar supported by the Engineering Research and Development for Technology (ERDT) of the Department of Science and Technology (DOST).

Sharaful-Ilmi Paduman is a student of De La Salle University-Manila, earning a Master of Science degree in Mechanical Engineering. He is a scholar supported by the Commission on Higher Education (CHED).

Alvin Chua is a full professor and the current chairman of the Mechanical Engineering Department of De La Salle University-Manila. He earned his BS, MS, and Ph.D. in Mechanical Engineering at De La Salle University-Manila. As a scholar under the Department of Science and Technology-Engineering and Science Education Project (DOST-ESEP), he conducted research at the University of New South Wales, Australia. He received a special citation for the 2003 NAST-DuPont Talent Search for Young Scientists (Mechanical Engineering). He has published several papers in international conferences like Conference on Decision and Control (CDC), and Advanced Intelligent Mechatronics (AIM). His research interests are on mechatronics, optimal estimation (Kalman Filters), automation, controls, and robotics.

Evidently, the neglected aspect of optimization concerns the length of the toolpath itself. Despite the availability of heuristic-based algorithms for solving TSP, the built-in toolpath generation for Computer Aided Design (CAD) or Computer Aided Manufacturing (CAM) programs are far from optimal. Specifically, the toolpath generated by the spectraCAD software is based on machining the features in the order in which they were drawn, which is clearly not optimal. A major reason for this is the absence of a method to widen the reach of TSP outside the domain of points to incorporate more complicated operations such as contouring.

The only studies on TSP that have been done so far are in the application to drilling operations using various algorithms. [11] [16] [17] [18] [19] [20]. This paper describes a novel approach on systematic block management using λ-opt moves to optimize the NC codes generated from a CAM software. The paper extends the previous studies as it considers other entities like line segments, arcs and circles in the optimization process which was not implemented in previous literatures. The new method developed is a systematic block management method. It designates the NC codes into blocks so that each block would be a series of operations that make a contour, be it points, open contours, or closed contours. Each type of contour is managed differently, and all have to be connected more effectively than the original to result into a final improved NC file. The assumptions that were independently devised to handle such information are given in the subsequent discussion on the methodology.

λ-opt Algorithm

λ-opt (or k-opt) is a heuristic algorithm for solving the TSP problem. A tour is considered λ-optimal when it is impossible to shorten the tour by replacing any set of λ segments [3]. The higher the value of λ, the more likely that the tour is optimal. Equation 1 below shows the number of moves to be assessed for a simple k-opt algorithm.

$$n_{moves} = (k-1)!\,2^{k-1} \tag{1}$$

The naive implementation of k-opt is shown in Figure 2. First, k number of points are randomly selected which are the boundaries of k segments of the path, since TSP makes use of a closed path. In preparing the segments, the selected points are sorted in ascending order to aid in the programming of the rearrangement function. Then, for the rearrangement itself, one of the segments is fixed since the placements are just relative to each other. Making the segment containing the starting block as the

fixed segment would avoid discontinuity problems with the ending block, thus it is much easier to handle. As such, these segments are reconnected in all the possible ways and the sequence with the shortest length of tours is accepted. All the possible moves for k-opt is determined by the formula taken from [21].

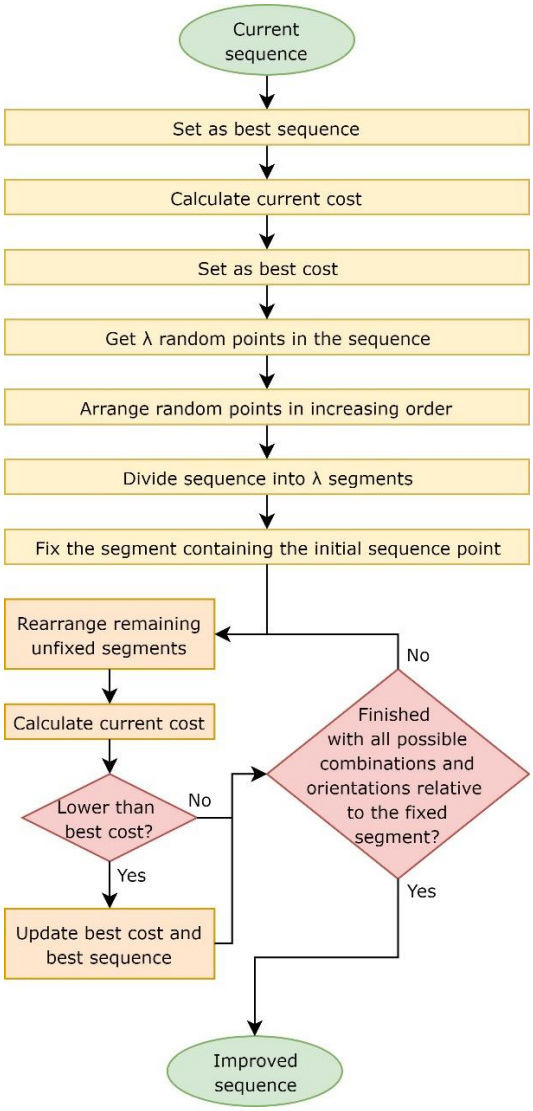

Figure 1. Flow Chart of the λ-opt Algorithm.

In Equation (1), the factorial term considers the number of combinations of (k-1) segments, while the exponential term considers the flipping of the segments. The flipping can be represented as all the binary numbers with a length not greater than (k-1) such that the 0/1 value indicates whether a particular segment has been flipped.

While there are only 2 possible ways for rearranging points using 2-opt, the possibilities for 3-opt and 4-opt are 8 and 48, respectively. However, since the "no action" choice would not make sense, these are reduced by one to give 1, 7, and 47 possible unique rearrangements for 2, 3, and 4-opt, respectively. The most basic version, the 2-opt algorithm, is one of the most used local search algorithms in TSP optimization. It was first proposed by Croes in 1958 [22]. In some cases, the 2-opt encounters difficulty in unraveling a sequence. Thus, 3-opt and 4-opt become necessary. Despite this, they are not used frequently due to the increased computational load associated with the large number of permutations.

Methodology

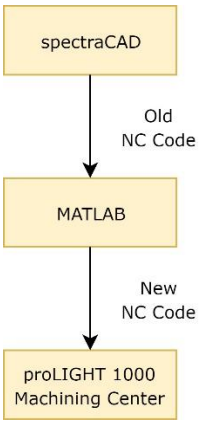

Figure 2. General Flowchart for NC-code Optimization.

Figure 2 shows the procedure for the systematic block management method which utilizes three software applications. SpectraCAD is used to make the design and generate the NC code. Next, MATLAB is the central program where the algorithms are made and utilized to improve an NC code file. Finally, proLIGHT 1000 Machining Center is used to run the simulations and the actual milling process.

SpectraCAD Engraver

This software converts CAD drawings into NC files that can be used on a CNC milling center. It features an integrated CAD drawing package that allows a seamless and easy working environment. However, its drawing options are primarily limited to arcs, circles, points, rectangles, lines, and text. It also has a few basic editing features but for the ease of use, a modern CAD software can be used to develop the design which would be saved in the Drawing Exchange Format (DXF) extension.

For generating the NC code, the engraving setup must be previously specified to make sure that the postprocessor matches the proLIGHT 1000 Machining Center, and that the stock size corresponds to the drilling plate dimensions. Afterwards, the Engrave command is used to generate the NC code for the design.

proLIGHT 1000 Machining Center

The proLIGHT 1000 Machining Center is a three-axis

tabletop milling machine which can be run directly from a personal computer. The CNC Base for proLIGHT 1000 Machining Center is the control program which accepts NC codes.

The tool settings and the verification setup should be matched with the NC code and the hardware. Then, the Verify command will run a simulation of the process. Estimating the runtime and distance can be directly run using a single dedicated command in the program.

MATLAB

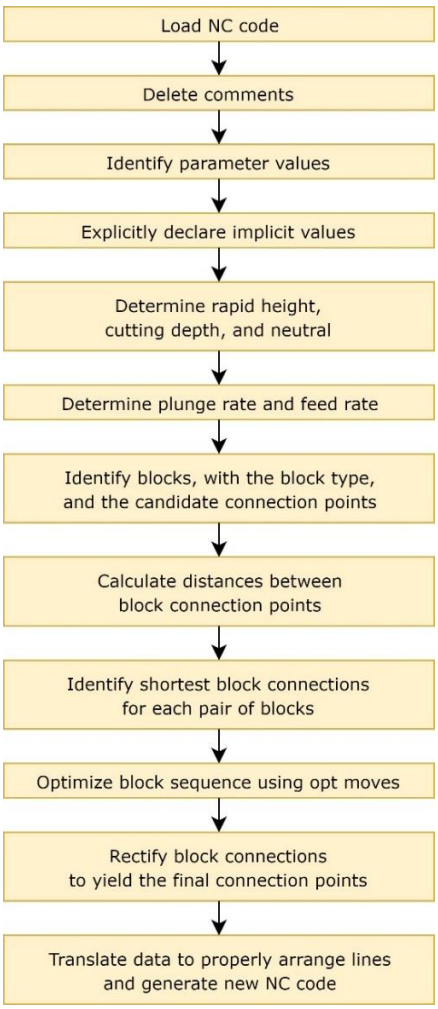

Figure 3. Overview of MATLAB NC Code-Processing.

Functions are made in MATLAB to handle the data from the original NC file and process them to result in a new NC file for the same design but with an improved total airtime. The overview of the process is illustrated in the flowchart in Figure 3. This would be useful in understanding the more detailed explanations that will follow afterwards.

A very important requirement for the program is that the codes should be in absolute coordinates, including the rotation center of the arcs. Failure to meet this requirement would surely result in a failed output.

NC Code Input and Initial Processing

First, an NC code is read by the MATLAB program and entered into a cell array. The comments in the lines which are indicated by the semicolon symbol are removed from the code as they are irrelevant and may result in processing issues. The remaining values are the actual useful code, and these are stored based on the line number and letter codes. However, MATLAB starts indexing at 1 while the line numbers start with 0. This results in a mismatch in the indexing which can be fixed by subtracting 1 from the index value when translating back the line number for the improved NC file.

After obtaining the values in the proper fields corresponding to the letter codes, the implicitly-defined values based on previous lines are explicitly assigned.

The three elevation (Z) values are then identified, namely the cutting depth, the rapid height, and the neutral. Respectively, these values indicate how deep the contour is, how high the tool traverses between contours, and how close to the surface it approaches before slowing the feed rate to plunge into the material. When there are more than three Z values identified, default values are provided to simplify the subsequent processes.

Then, the two feedrate (F) values are determined, which are the plunge rate and the feedrate. The feedrate is used for all Z operations except for the plunging process. Similar to before, when there are more than two F values identified, default values are provided to simplify the subsequent processes.

Block Classification and Preparation for Optimization

Blocks are identified based on the height changes, where each block is the sequence of lines with the Z value at the cutting depth. A special case is the starting position at [0, 0] which is considered a block because the tool would start from that position. After identifying the blocks, they are classified into three types: (1) point, (2) open contour, (3) closed contour. By means of the block length, the points can be identified. Then to determine whether the block is an open or a closed contour, the coordinates of the start and the end are compared. After classification, appropriate candidate connection points are identified. The local endpoints of the segments and arcs which comprise the closed contour are candidate connection points, except for the endpoint of the entire block which is redundant with the start point of the block. For open contours, the start and end points of the block are the only candidate connection points.

The distance data between the candidate connection points are prepared in a matrix of distances. Intra-block connections are invalid so their values in the matrix are set to “Inf”. This serves as an assurance that no unwanted pairing makes it through the program. Based on the distance matrix that was obtained, the closest

connection points per block pair are identified.

Optimization Process and Underlying Assumptions

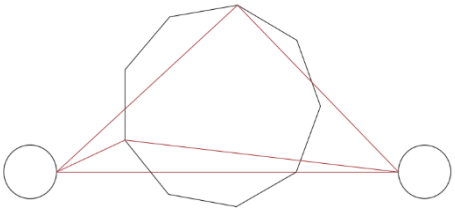

Figure 4. Minimization Assumption for Inter-Block Connections through Reducing the Total Triangle Length by Minimizing One Side.

The optimization process relies on the assumption that in connecting two entities through another entity in the middle, the shortest path would be one where a segment is minimized while the other is much closer to the direct length between the two entities. This concept is based on the thought experiment that when one side of the triangle cannot be adjusted, the shrinking of another side to a minimum will minimize the total length. This is illustrated in Figure 4 where the bottom two lines approximate the shortest distance between the two circular entities at the both ends of the horizontal line. The assumption was followed which arrived at the minimal possible combined length through a valid connection point of the nonagon entity at the middle.

However, this is sometimes not the case as could be seen in Figure 5 where shorter pairs can be made outside of the assumption. As an observation of these figures, it can be seen that a more accurate minimization is actually to choose the segments which form an angle that is closest to 180°. However, this concept is not simple to implement and may be done in a future study.

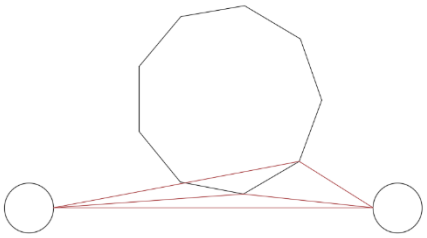

Figure 5. Example Failure of the Assumption.

The use of the closest connection points is merely an approximation of the best sequence when all points are actually considered. Otherwise, if all points are considered, the calculation requirement will be on the order of n^3 because each combination of the points in the previous, current, and next blocks would have to be inspected.

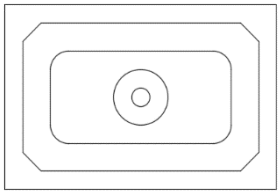

Figure 6. Symmetrical Design.

Centroidal distances are not used for assessing block distances due to the existence of designs where blocks have the same centroid, largely brought about by design symmetry. An example of this is shown in Figure 6 where the program will be blind to select which block sequence to choose and which points to connect.

The main optimization is done on the block sequence. The cost function for this optimization process is the inter-block distance to be minimized. The process, which is called a partitioned [2, 3, 4]-opt algorithm, is done by partitioning the sequence where each partition has a minimum of 7 blocks. Otherwise, without the partitioning, the program will easily be overwhelmed if the number of blocks is large. On each of the partitions, the 2-opt algorithm is used as the main algorithm due to its speed. The 3-opt is inserted for every five 2-opt operations whereas the 4-opt is inserted for every twenty.

The maximum number of partitions is determined based on the block length. The number of partitions start from one and increases to the maximum for the first set of operations, then decreases back to one for the second set. This continues for 5 sets which has been empirically determined to be effective. The process can handle a TSP, albeit an open one because the cost function does not include the connection between the start and end points as this would be detrimental in optimizing the sequence when dealing with partitions.

Data Translation into a Working NC File

Because of the approximation laid out earlier, rectification would have to be done to make sure that the open and closed contours are properly connected. This is with respect to the entry and exit points. The open contours are constrained to enter through one point and to exit through another, whereas closed contours only make use of one point for both entry and exit. The approach mostly makes use of two connection points per block, assuming that the previous and next blocks connect at different points. Thus, each would have to be assessed as to which would yield a lower cost. This rectification is done sequentially starting from the first block. Since the entry and exit points are dependent variables, it is only necessary to identify one of them. In the code, the entry points were identified.

Finally, the finalized NC code is made by translating

back the data. This process is more complex than the input of the NC code since there has to be manipulations based on the block sequences and the connection points. Lines for the rapid traverse to the block connection point and the rapid descend to the neutral height are assembled prior to each of the main block lines, except for the first block which is a special case because it is a non-drill starting point. After the main block lines, the lines for rapid ascent are assembled.

The difficulty with the translation is mostly on the main block lines. For points, there would be no problem. But for open and closed contours, different approaches have to be done to assure that the proper sequence is followed and the same design would be milled.

For open contours, when the starting point is the same as the starting point of the initial code, there would be no problem as everything would just be copied. If the other end is used first, however, the coordinates would have to be flipped. Simply doing this to the entire lines including the G values (type of operation) and the *I, J, K* values (rotation center) would be disastrous since they would be positioned in the wrong places. The way this is managed is to flip only the [x y] coordinates of the block. Then, the G values of the very first line is maintained at 1. The rest of the lines would flip the G values and the *I, J, K* values. G2 is then mutated to G3 and vice versa. Figure 7 illustrates an example and Table 1 shows the correct solution where the original sequence is A-B-C-D-E-F which is then flipped to F-E-D-C-B-A. The motions, whether (1) linear, (2) clockwise, or (3) counter-clockwise are given by the G value.

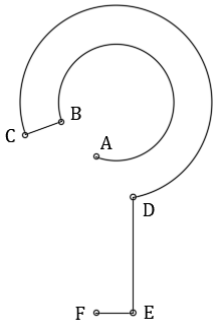

Figure 7. Example Open Contour for Flipping

In the example, letters are used to represent the coordinates. Moreover, a G1 code indicates a straight line to the coordinate whereas G2 and G3 indicate clockwise and counterclockwise rotations respectively. As shown in the example where F is selected as the start of the flipped sequence, the rest of the coordinates follow the flipped order. However, the G value is maintained at G1 as the initial point is for the plunging of the tool. Then, the rest of the G values from the original sequence are flipped. The G2 and G3 values indicated in pink are mutated to be the opposite rotation, as shown in the flipped sequence in violet.

Table 1. Solution to Open Contour Flipping.

Original Sequence		**Flipped Sequence**	
Point	G value	Point	G value
A	1	F	1
B	3	E	1
C	1	D	1
D	2	C	3
E	1	B	1
F	1	A	2

On the other hand, for closed contours, having the same starting point as the original will also be convenient since there would be no change required. However, since there are usually more candidate connection points in a closed contour, it is likely that changes have to be made, unlike the case for open contours. For closed contours, circular shifts have to be made where the list moves down and the last in the list goes back to the top. Similarly, with the open contour, this could not be simply done to the entire block lines. The circular shift has to be done on the lines excluding the very first line. The initial line will have the [x y] coordinates of the specified connection point. The circular shift will proceed until the last line has the same coordinate as the first line. Then, the G value of the first line is fixed to 1. Figure 8 and Table 2 show the validity of this process where the original sequence is A-B-C-D-E-F-A which is then changed so that E comes first, making the sequence E-F-A-B-C-D-E. The motions, whether (1) linear, (2) clockwise, or (3) counter-clockwise are given by the G value.

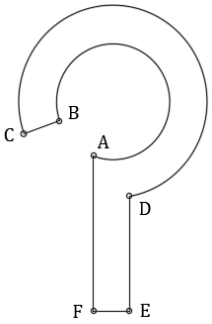

Figure 8. Example Closed Contour for Shifting

In the example, the initial G value is maintained at 1, similar to the open contour case. The rest of the lines are shifted until the chosen coordinate from the original sequence goes to the last element of the list.

Table 2. Solution to Closed Contour Shifting.

Original Sequence		Flipped Sequence	
Point	G value	Point	G value
A	1	E	1
B	3	F	1
C	1	A	1
D	2	B	3
E	1	C	1
F	1	D	2
A	1	E	1

After developing the program, to conduct the study, the authors have access to various PCs that roughly range from medium-low to medium-high specifications. However, this is irrelevant as the testing only requires the output of the developed program. Nonetheless, it can be noted that all the solutions were obtained in less than one minute.

Simulation Results

The partitioned [2, 3, 4]-opt algorithm has been applied to generate improved tool paths for different designs such as *circles*, *stars*, *DLSU1*, *DLSU2*, and *tree*.

Along with contours, the code was additionally tested with toolpaths consisting of only points, which makes it very similar to the ordinary TSP. The figures that follow illustrate the example designs:

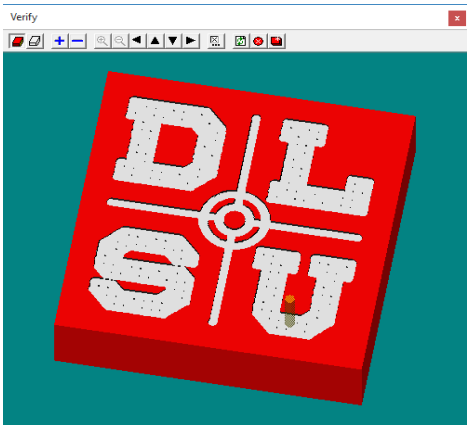

Figure 9. Simulated Output of *DLSU2* Design

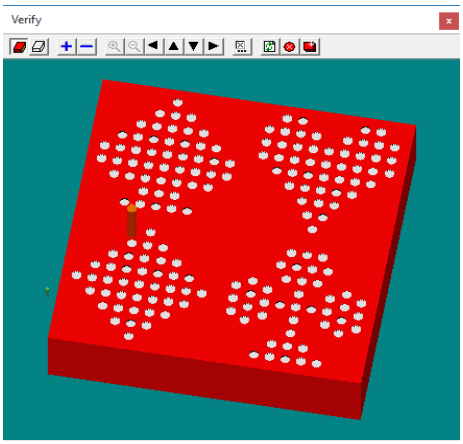

Figure 10. Simulated Output of *Points* Design

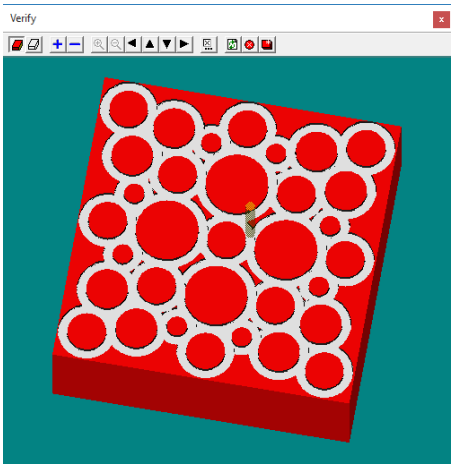

Figure 11. Simulated Output of *Circles* Design

The optimized tool path results are shown in Table 3, 4, and 5. As can be seen, Table 3 shows the distance improvement for the toolpath after applying the algorithms. The percentages time improvements for the various shapes are between 11% and 21%, whereas it is more than 53% for the pure drilling case.

Table 3. Distance Improvement

Shape	**Distance** (original)	**Distance** (improved)	**Percentage Decrease**
Circles	3603.81	2874.21	20.25%
Stars	1432.70	1161.4	18.94%
DLSU1	3961.89	3237.91	18.27%
DLSU2	5942.63	4818.24	18.92%
Tree	6142.62	5447.35	11.32%
Points	11031.80	5097.67	53.79%

Focusing on the airtime where the tool is not in contact with the workpiece, which is the portion of the process being optimized, a much higher improvement could be seen for the cases with contours. For the drilling case, no significant change has been recorded because the distance itself is mostly directly related to the airtime.

Table 4. Airtime Improvement

Shape	**Airtime** (original)	**Airtime** (improved)	**Percentage Decrease**
Circles	1183.69	745.93	36.98%
Stars	408.20	245.42	39.88%
DLSU1	1293.65	859.27	33.58%
DLSU2	2382.20	1707.56	28.32%
Tree	2687.83	2270.67	15.52%
Points	6619.08	3058.60	53.79%

Table 5. Overall Time Improvement

Shape	**Time** (original)	**Time** (improved)	**Percentage Decrease**
Circles	18:32	17:57	3.15%
Stars	08:13	08:00	2.64%
DLSU1	20:35	20:01	2.75%
DLSU2	24:45	23:52	3.57%
Tree	22:57	22:24	2.40%
Points	12:25	7:45	37.58%

However, for the overall time improvement, the change is smaller yet still significant. The degree of this difference with the airtime improvement is based on the length of the blocks as well as the number of blocks since plunging and retraction contribute to the overall time. Thus, the effectiveness of the algorithm is reflected in the airtime improvement and the economic impact is seen in the overall time improvement which is affected by factors that are fixed and cannot be changed by any optimization process. In Table 5, the largest improvements is found in the design consisting purely of points which do not have an associated travel length unlike contours.

Experimental Results

The researchers used Intelitek proLIGHT machining center as the CNC milling machine to perform actual experiment trial to verify the applicability of the devised improvement algorithm.

First, a new etching design was made to resemble a "Printed Circuit Board" (PCB). The PCB has a combination of holes, lines, and curves that will be a balanced test of the different functions of engraving. The design was simulated using the same software used to obtain an estimated theoretical time as found in Figure 12.

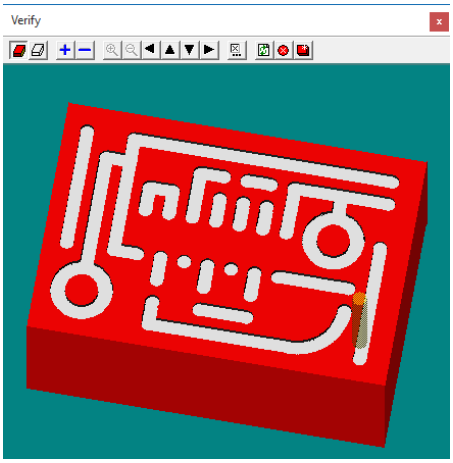

Figure 12. Simulation of PCB

The non-optimized NC code was loaded into the machine and the program was run. This was followed by the optimized NC code was afterwards similarly loaded and run.

Figure 13 shows the experimental set up. The workpiece is a blue machinable wax with a workable surface size of 3 in. × 2 in. This is fastened to the 3-axis CNC milling machine through a pneumatic vise. The specifications do not vary therefore any changes in performance will be largely due to the improvement of the NC codes themselves.

Figure 13. Experimental set up with specimen

The milling times were recorded from the start of

spindle rotation until it automatically turns off. The result was a time reduction from 8:33 to 7:23 which is a 13.35% improvement, consistent with the simulated results. Figure 14 shows the actual machined part using the proLIGHT Machining Center.

Figure 14. Actual Machined Part

Conclusion

In this study, the λ-opt algorithm has been implemented on variably-partitioned sequences to improve the tool path of CNC machine with regard to contours. This has been made possible with a block management method centered on how the distance matr3ix is defined based on an approximate assumption on the minimization of the distance along three entities. The block management also goes up until the translation phase of the program such that the proper order of the code is followed to maintain the desired design.

With performance in terms of time and distance, the improved NC code can be used in milling to reduce the production cost. As illustrated, the time can be improved by more than 3% and the tool travel distance by more than 20% on designs with contours. Therefore, applying this algorithm is an efficient and easy way to improve the tool path for a CNC machine. Just replacing the old NC code with the improved NC code for use in the CNC machine will yield noticeable benefits. Airtime improvement is seen to be upwards of 10% and can reach around 50% depending on the initial design.

A notable improvement would be the possibility of accessing the quadrant points of circles as candidate connection points. This is because the current implementation only makes use of one point from a complete circle. Moreover, as the block management method has already been proven to be effective, it may already be possible to use the complete LKH method in order to achieve a solution that is closer to the optimal. This would be a good improvement over the devised method of partitioned λ-opt manipulations. Lastly, other important milling operations such as facing, pocketing, and multiple passes may also be considered for optimization.

Acknowledgment

The authors would like to acknowledge the support of the ASEAN University Network (AUN), the Engineering Research and Development for Technology (ERDT), and the Commission on Higher Education (CHED) for scholarship grants. The authors also would like to express their gratitude to DOST-PCIEERD and the Mechanical Engineering Department of De La Salle University in providing the facilities and expertise for the fulfillment of this study.

References

[1] F. Kolahan and M. Liang, “Optimization of hole-making operations: a tabu-search approach,” *International Journal of Machine Tools and Manufacture*, vol. 40, pp. 1735-1753, 2000
doi: 10.1016/s0278-6125(03)90018-5

[2] K. Castelino, R. D'Souza, and P. K. Wright, “Toolpath optimization for minimizing airtime during machining,” *Journal of Manufacturing Systems*, vol 22, no. 3, pp. 173-180, 2003.
doi: 10.1016/s0278-6125(03)90018-5

[3] K. Helsgaun, “An effective implementation of the Lin–Kernighan traveling salesman heuristic,” *European Journal of Operational Research*, vol. 126, no. 1, pp. 106–130, 2000.
doi: 10.1016/s0377-2217(99)00284-2

[4] O. Montiel-Ross, N. Medina-Rodríguez, R. Sepúlveda, and P. Melin, “Methodology to Optimize Manufacturing Time for a CNC Using a High Performance Implementation of ACO,” *International Journal of Advanced Robotic Systems*, vol. 9, no. 4, p. 121, 2012.
doi: 10.5772/50527

[5] L.-P. Wong, M. Y. H. Low, and C. S. Chong, “A Bee Colony Optimization Algorithm for Traveling Salesman Problem,” in proceeding of *Second Asia International Conference on Modelling & Simulation*, Malaysia, May 13-15, 2008.
doi: 10.1109/ams.2008.27

[6] Y.-F. Liao, D.-H. Yau, and C.-L. Chen, “Evolutionary algorithm to traveling salesman problems,” *Computers & Mathematics with Applications*, vol. 64, no. 5, pp. 788–797, 2012.
doi: 10.1016/j.camwa.2011.12.018

[7] G. Barach, H. Fort, Y. Mehlman, and F. Zypman, “Information in the Traveling Salesman Problem,” *Applied Mathematics*, vol. 3, no. 8, pp. 926–930, 2012.
doi: 10.4236/am.2012.38138

[8] V. Zharfi and A. Mirzazadeh, “A Novel Metaheuristic

for Travelling Salesman Problem," *Journal of Industrial Engineering*, vol. 2013, pp. 1–5, 2013. doi: 10.1155/2013/347825

[9] K. LI, "Solving The Traveling Salesman Problem Using Efficient Randomized Parallel Approximation Algorithms," *Parallel Algorithms and Applications*, vol. 10, no. 3–4, pp. 271–281, 1997. doi: 10.1080/10637199708915622

[10] K. Helsgaun, "LKH Version 2.0.7", 2012. [Online]. Available: http://www.akira.ruc.dk/~keld/research/LKH/. [Accessed: Dec. 20, 2017].

[11] R. Aciu and H. Ciocarlie, "G-Code Optimization Algorithm and its application on Printed Circuit Board Drilling," in proceeding of 9*th IEEE International Symposium on Applied Computational Intelligence and Informatics (SACI)*, Romania, May 15-17, 2014, pp. 43–47. doi: 10.1109/saci.2014.6840096

[12] SPRING Technologies, "NCSIMUL OPTITOOL", 2017. [Online]. Available: https://www.ncsimul.com/contenu.php?ID=73. [Accessed Dec. 20, 2017]

[13] CGTech, "VERICUT G-CODE OPTIMIZATION", 2017. [Online]. Available: http://www.cgtech.com/services/training-schedule/vericut-g-code-optimization/

[14] X. Zhang, L. Zhu, D. Zhang, H. Ding, and Y. Xiong, "Numerical robust optimization of spindle speed for milling process with uncertainties," *International Journal of Machine Tools and Manufacture*, vol. 61, pp. 9–19, 2012. doi: 10.1016/j.ijmachtools.2012.05.002

[15] S. Warghat and T. R. Deshmukh, "Experimental Investigation and Optimization of Machining Parameters of CNC Milling," *MAYFEB Journal of Mechanical Engineering*, vol. 1, pp. 1-11, 2016.

[16] K. Narooei, R. Ramli, M. Rahman, F. Iberahim, and J. Qudeiri, "Tool Routing Path Optimization for Multi-Hole Drilling Based on Ant Colony Optimization," *World Applied Sciences Journal*, vol. 32, no. 9, pp.Danijela Pezer* 1894-1898, 2014. doi: 10.5829/idosi.wasj.2014.32.09.456

[17] A. Kumar and P. Pachauri, "Optimization Drilling Sequence by Genetic Algorithm," *International Journal of Scientific and Research Publications*, vol. 2, no. 9, pp. 1-7, 2012.

[18] D. Pezer, "Efficiency of Tool Path Optimization Using Genetic Algorithm in Relation to the Optimization Achieved with the CAM Software," I*nternational Conference on Manufacturing Engineering and Materials*, vol.149, pp.374-379, 2016. doi: 10.1016/j.proeng.2016.06.681

[19] N. Al-Sahib and H. Abdulrazzaq, "Tool Path Optimization of Drilling Sequence in CNC Machine Using Genetic Algorithm," *Innovative Systems Design and Engineering*, vol.5, no.1, 2014.

[20] W. Lim, G. Kanagaraj, and S. G. Ponnambalam, "PCB Drill Path Optimization by Combinatorial Cuckoo Search Algorithm," *The Scientific World Journal*, vol. 2014, pp.1-11,2014. doi: 10.1155/2014/264518

[21] A. Blazinskas and A. Misevicius, "Combining 2-Opt, 3-Opt and 4-Opt With K-Swap-Kick Perturbations for the Traveling Salesman Problem," in proceeding of *17th International Conference on Information and Software Technologies*, pp. 27-29, 2011.

[22] G. A. Croes, "A Method for Solving Traveling-Salesman Problems," *Operations Research*, vol. 6, no. 6, pp. 791–812, 1958. doi: 10.1287/opre.6.6.791