A Robust Fast Bridging Algorithm for Laser Cutting

Qirui Hu

State Key Laboratory of Fluid Power Transmission and Control

Zhiwei Lin (✉ zjtzhylin1986@163.com)

State Key Laboratory of Fluid Power Transmission and Control

Jianzhong Fu

State Key Laboratory of Fluid Power Transmission and Control

Research Article

Keywords: Laser cutting, Bridging, Piercing reduction, Delaunay triangulation

Posted Date: October 18th, 2021

DOI: <https://doi.org/10.21203/rs.3.rs-970883/v1>

License: This work is licensed under a Creative Commons Attribution 4.0 International License.

[Read Full License](#)

Abstract

Bridging the different parts together is considered a simple but effective strategy to reduce the number of piercing operations during laser cutting. However, fast bridging is never an easy task. In this paper, we present a near-linear bridging algorithm for the input parts with the shortest total bridge length. At first, the input part contours are discretized into a point cloud, then the point cloud is triangulated with the Delaunay standard. The shortest line segments between any two adjacent parts are found in the triangles connecting the two parts. These segments are finally extended into bridges. To solve the problem of the damages to the contour characteristics caused by the bridges, some restrictions are set on the screening of the discrete point cloud and the Delaunay triangles. This algorithm not only ensures the minimum total distance of all bridges, but also avoids the problem of generating bridge loops. Computational experiments show that the proposed bridging algorithm is much faster than that in existing commercial software. The feasibility and superiority of the algorithm are verified by actual lasering cutting experiments.

1 Introduction

Laser cutting is one of the main modern technologies in sheet metal product manufacturing industry. The use of laser cutting brings many benefits, such as minimizing the cutting width and reducing thermal deformation [1]. A typical production process consists of designing parts, nesting the parts on metal sheets, and cutting the parts from the sheets [2]. Although it seems pretty neat, there are many cutting processes to consider between nesting parts and cutting plan, like piercing, bridging and common cut. In recent years, many researches in the field of laser cutting have considered the influence of cutting processes. Huang et al. [3] studied the layout and path planning of rectangular parts with the influence of piercing. Rao et al. [4] designed a system on parts layout and path planning for irregular shape parts. Raggenbass et al. [5] and Pan et al. [6] also studied on layout and path planning system for irregular shape parts. Fig. 1 illustrates an example of nesting parts on a metal sheet. As shown in Fig. 1, a part is defined as an outer contour with possibly a set of inner contours forming holes in the part. A contour itself is composed of a set of geometry elements which can be line segments, polylines, circles, arcs, ellipses, B-spline curves etc. Each element is defined by its end nodes and can be cut in both directions (unless explicitly specified otherwise). To minimize the material used on the metal sheet, nesting algorithms attempt nesting parts within parts (circular parts nested inside ring parts in Fig. 1). Most researches on laser cutting focus on cutting path planning after nesting, while few consider the influences of piercing, bridging and common cut [7].

Piercings are operations where the laser penetrates the metal sheet. This is required when the laser starts to cut in an area of the metal sheet where it has not cut before. To ensure the quality of the processed parts, piercings should be arranged in the position on metal sheet where no parts have been arranged [7]. Fig. 2 shows an example of cutting path with piercing. The complete cutting path for part in Fig. 2 can be represented as $\tau(P_1, P_2, P_3, P_4, P_5, P_6, P_2, P_7, P_8)$. P_1 is the piercing point as well as the start point of this cutting path. The distance from P_2 to P_7 is the excessive cutting segment to make sure this cutting path

been cut completely. The time of piercing is related to the thickness of the metal sheet and laser power. For thicker metal sheet, piercing usually takes more time. In order to reduce piercing operations and processing time, two approaches, common cut and bridging are often used, as shown in Fig. 3.

Suppose $part_1$ and $part_2$ are the two parts on the metal sheet that need to be cut, as shown in Fig. 3. Fig. 3a shows the conventional approach. The laser beam begins to cut from point P_1 , goes around the shape contour, and ends at point P_2 . Then the beam jumps to the piercing point P_3 , goes around $part_2$ and ends at point P_4 . The second approach in Fig. 3b adopts common cut. Unlike the first approach, the laser beam does not need to pierce again after it reaches point P_2 . Instead, it takes advantage of a common edge shared by $part_1$ and $part_2$ (p_3 - p_4) and cuts only three rectangular edges of $part_2$. Fig. 3c shows another approach which completes two parts in one cutting path by a bridge. Although common cut and bridging can reduce piercing, these two methods are still inadequate.

For the common cut approach, on the one hand, it limits the positions of the jump points (only P_3 or P_4 is allowed in Fig. 3b); on the other hand, it also limits the cutting sequence between parts (complete contour with pierce must be cut first, and residual detached contour later). These two limitations make the jump distance of the common cut longer than that of other cutting approaches, and also make the cutting path planning task more difficult.

For the bridging approach, it is realized by connecting adjacent separate cutting paths to reduce the piercing operations, as well as the number of jumps and the air move. Rather than adding restriction like common cut, bridging will simplify the cutting path planning. However, as shown in Fig. 4, bridging may damage the contour characteristics of the parts. Fig. 4a and Fig. 4b demonstrate the situations that how bridge can be harmful to the geometry precision of the parts when it is connected to the corner or curve of the part contour. Fortunately, the problem can be solved by optimizing the location of bridges [8].

Although piercings are very expensive, air moves are considerably cheaper than cut moves. It follows that a bridge cannot be longer than a certain length before a combination of piercing and air move becomes cheaper [7]. Generally, in order to reduce the processing time and the part stress caused by the bridge, the bridge should better be short. Based on this, finding the location to build bridge between parts is similar to calculate the shortest distance between contours. Manber et al. [9] used bridge to minimize the number of pierce point for contour groups connected through common cuts. Dewil et al. [10–13] made a lot of contributions in laser cutting field and proposed that a bridge can be used to connect two sub-regions to reduce the number of piercing. Although the bridging concept is popular in laser cutting, the bridging algorithm itself is seldom researched. To the best of our knowledge, no public literature can be found on this aspect.

In this work, an efficient bridging algorithm based on Delaunay triangulation is proposed and implemented in the laser cutting application. Specifically, the part contours that have been previously nested are discretized into a point cloud. With the help of Delaunay triangulation, the point cloud is triangulated into multiple triangles, then some invalid triangles are screened out to decrease

computational complexity, and the shortest distance between adjacent parts are calculated by the remaining triangles. Eventually, the target bridge is built from selected shortest distance segments. The whole bridging algorithm proceeds quickly since the time complexity of Delaunay triangulation is near linear. It can bridge all parts in $O(n \log n)$ time, in which n is the size of the discretized point. Consider that bridging may damage the contour characteristics of the parts, some limitations are set during discretizing part contours and screening out invalid triangles. Besides, the problem of bridging into a ring (bridge loop) is solved while calculating the shortest length of all bridges.

The remainder of this paper is organized as follows. In Section 2, the structure of the bridge and the algorithm input are introduced. Section 3 introduces the method of discretizing nested parts into a point cloud. In Section 4, constructing triangles and building bridges by selected triangles are presented. Comparison of bridge effect with commercial software and actual cutting experiments are given in Section 5. The last section concludes this paper.

2 Bridge Construction And The Algorithm Input

Before illustrating our algorithm, the structure of the bridge and the algorithm input will be introduced in this section. As shown in Fig. 5, $part_1$ and $part_2$ are connected by a bridge, which can be represented as $u(V_1, V_2, V_3, V_4)$. There are two vertexes on each connected part (V_1, V_2 on $part_1$ and V_3, V_4 on $part_2$). The distance between V_1 and V_2 or V_3 and V_4 is called the bridge width (D_W). D_W will affect the stress of the connection and is usually set according to the size of parts. Generally speaking, D_W will not exceed the length of the shortest edge of parts. In this paper, D_W is an appropriate input value. Apart from D_W , the maximum length of the bridge (D_L) is also an input value used in our algorithm. As mentioned in Section 1, the bridge should be as short as possible and D_L will help screen out the overlong bridges.

According to the definition of parts in Section 1, each contour of a part is represented by a series of endpoints. But the contours of parts may include curves like circles, arcs, B-spline curves and even elliptic arcs, which means that they cannot be replaced by points as accurate as line segments or polylines. And these curves need to be subdivided into many points by angle (as shown in Fig. 6a) or parameter (as shown in Fig. 6b) to participate in the calculation. In order to ensure the cutting accuracy of these curves, the distance between lines composed of these discrete points should be very short, which is an essential guarantee to filter out specific line segments in our algorithm.

Overall, part contours may contain curves when designed, which will be replaced by series of points in nesting and bridging calculations, as shown in Fig. 6. The relationship of all parts participate in bridging is shown in Fig. 7. All parts are in a part set (S_p) and each part is also a contour set (S_c). Each contour is an endpoint set (S_e) and two adjacent endpoints form an edge.

As shown in Fig. 8a, the cutting paths for $Part_1$ and $Part_2$ are connected by a bridge, which makes the two parts also connected after cutting, as shown in Fig. 8b. Through post-processing, two required parts can be obtained after the bridge is removed, as shown in Fig. 8c. As mentioned in Section 1, if the bridge is

built on the corner of the part or on the curve contour (replaced by many points), it will inevitably destroy the contour characteristics of the part when the bridge is removed.

3 Part Edge Decertation

Finding the shortest distance between parts is fundamental to bridge all parts. To realize that, in this work, the Delaunay triangulation method is applied, which will be presented in Section 4. Before that, all nested parts defined in Section 2 need to discrete into a point cloud as the input of Delaunay triangulation.

There are two problems to be noticed in the construction of bridges. One is that, naturally, the bridge should be as short as possible. The other problem is that, if the bridge is adjacent to any curves or endpoints of the contour, the bridge will inevitably damage the contour characteristics, as shown in Fig. 4. As shown in Fig. 9, if the contour is made up of both curves and line segments, priority should be given to bridging the line segments, even if the length of the bridge is longer than that of the curve.

In order to solve the above two problems, a distance value D_m is set when all parts are discretized into a point cloud. As shown in Fig. 10, D_m is the minimum allowed distance between two adjacent discrete points on each edge, which is a mathematical tool to determine how many points can be produced at most for a given edge.

D_m is set to screening and discretizing these part edges if the edge length is larger than twice D_m . Fig. 10 shows the relationship between D_m and edge length. In Fig. 10, if the length of edge is shorter than twice D_m , the distance between the generated discrete point and the endpoint of the edge will be less than D_m (like L_1 in Fig. 10) and that may affect the bridge when extending segments into bridges. Therefore, only edges longer than twice D_m will be discretized and endpoints are not added to the point cloud to avoid the bridge adjacent to endpoints. For the edge that larger than twice D_m , attention should be paid to starting from the middle of the edge to ensure that the distance between the discrete point and the endpoint is not less than D_m (like L_3 in Fig. 10).

As described in Section 2, curves in parts are discretized by angles or parameters into multiple points. In order to approximately replace the curve, the length of each line segment made up of these points must be short enough (shorter than D_m). Therefore, the discrete points of the curve can be identified and excluded from the discrete point cloud by D_m . As a result, if the entire contour is made up of curves and all edges of this contour do not meet the requirements, this contour will no longer bridge with other parts.

To determine the value of D_m , we need to consider the value of D_w and the effect on Delaunay triangulation. To be specific, a smaller D_m will help find more discrete points and make it easier to find a shorter distance between parts. However, if the D_m is too small, the width of the generated bridge will be affected when widening lines into bridges, which will be presented in Section 4.

All discrete points will be stored in the set of points S_d . To make it easier to understand, the point structure which is used throughout the paper is defined first.

Definition 1 Struct Point {

double x, y;

int partIndex, polyIndex, edgeIndex ;

}.

In this structure, the real numbers x and y are the coordinates of the point in Euclidean space; the integer partIndex is the index of S_p ; the integer polyIndex is the index of S_c ; and the integer edgeIndex is the index of S_e . After all the discrete points been saved, the next step is Delaunay triangulate.

4 Building Bridges By Delaunay Triangulation

Triangles constructed by Delaunay triangulation have good properties such as empty circle [14] and is often used to solve the shortest distance between contours [15–17]. In this paper, we use this method to triangulate all discrete points in S_d . The incremental algorithm proposed by Berget al. [18] is employed. The time complexity of their algorithm is $O(n \log n)$, in which n is the size of the point set. Fig. 11 shows a result of Delaunay triangulation for discrete points.

In our previous research, the shortest distance between contours is found by screening triangles and sorting them in sequence [19]. But in this paper, the shortest distance between adjacent parts is the target. To reduce the number of triangles that need to be sorted in sequence, two rules are set to eliminate triangles. Firstly, if all three points of a triangle are on the same part or on three different parts, this triangle is not helpful for us to find the distance between two parts. Secondly, for parts with multiple contours, if two points in a triangle are on different contours of the part, the triangle should be eliminated, too. Triangles mentioned above can be easily filtered by partIndex and polyIndex of triangle points, that is why we need these two parameters in point structures. Fig. 12 shows the result of deleting triangles through these two steps.

All the remaining triangles are two points (E_1, E_2) on the same contour of one part, while the third point (E_3) on another part. But there are still some triangles that contain or cross endpoints of parts need to be eliminated. If the shortest distance is taken on these triangles, the bridges will either damage the contour characteristics of the parts or make bridging mistakes as shown in Fig. 13. Such triangles can be deleted by judging whether E_1 and E_2 are on the same edge, which can be implemented by edgeIndex parameter of triangle points.

In summary, through the three-step screening of the Delaunay triangles, all the remaining triangles satisfy that E_1 and E_2 on the same edge of one part and E_3 is on another part. Next, all remaining triangles are

sorted to find the shortest distance between adjacent parts. Fig. 14 shows the method of finding the shortest distance from a triangle. If the height (E_3-H) from E_3 to edge E_1-E_2 is in the triangle, the height is used as the shortest edge. If not, the shorter edge is selected as the shortest line between E_1-E_3 and E_2-E_3 . The shortest line structure is defined as:

Definition 2

Struct Shortest line {

Point pt1, pt2;

}.

The structure of the shortest line needs two points, and the structure of points is introduced in Section 3. If the shortest line contains point H, the properties partIndex, polyIndex and edgeIndex will be the same as E_1 or E_2 . All shortest lines are saved in a set denoted as S_L .

The shortest distance between adjacent parts calculated by Delaunay triangles is shown in Fig. 15 and these lines in S_L need to be extended to bridges with width. Before that, some lines need to be deleted to avoid bridge loops. As we have mentioned in Section 1, the purpose of bridging is to reduce the number of piercings. However, if the bridging produces loops, each loop will need a new piercing point, which goes against the original intention. To eliminate these loops, all lines in S_L need to be sorted by length and selectively deleted according to the D_L that we set initially. The shortest one in S_L is used to connect parts firstly and the connected parts will be composed into one part. Subsequently, each time the remained shortest line in S_L is used to connect the parts until all parts are connected to a part or all lines in S_L are used. All the lines used will be transformed into bridges. Because we avoid selecting points close to the endpoints of the parts, all used lines can be widened to bridges without exceeding the border, as shown in Fig. 16.

5 Experiment

In this section, two experiments are presented. One is to compare the effect and running time of proposed bridging algorithm with the existing commercial software, the other is the actual processing experiment. We use the proposed algorithm to implement the actual laser cutting to verify the feasibility and superiority of this algorithm.

5.1 Computational examples and comparisons

The proposed bridging algorithm has been programmed in the C++ language and run on a personal computer with a dual-core CPU @2.70GHZ and 8GB RAM. The proposed algorithm succeeds in many examples that have been tested.

It has been mentioned that we did not find a detailed study on the laser cutting bridging algorithm in the existing research, so we compare the method with the existing commercial software. However, this means that we have to implement their own nesting function in laser cutting. But the nesting results are different from commercial software because of the difference of the nesting algorithm, which makes the amount and location of bridging parts somewhat different (The calculation amount is of the same order of magnitude). While this may have some slight impact on the nesting results, it turns out that the running time of the two bridging algorithms is significantly different. Moreover, the bridge algorithm in this paper avoids the damage to the contour characteristics of the bridged parts, while there is no corresponding setting in the commercial software. Therefore, in order to reduce the difference in calculation amount and make the results relatively objective, only one curve part in experiments is selected to place.

Fig. 17 shows an example of the result of bridging algorithm introduced in this paper. It can be found from **Fig. 17** that the bridge algorithm proposed in this paper can bridge all the parts with the global shortest distance without damaging the contour characteristics of the parts, unless the part is not suitable to bridge like the circular part. In contrast, **Fig. 18** shows the bridging results of commercial software. The bridge algorithm of commercial software does not consider the contour characteristics of parts, and some parts are not bridged for some unknown reasons. In addition, the proposed algorithm also gains an edge on the calculation speed, as shown in **Table 1**. It can be seen from the difference in running time in **Table 1** that the proposed algorithm is able to quickly complete the bridge of a large number of parts, while the bridge of all parts is a time-consuming operation for commercial software.

Table 1. Comparison of running time between proposed algorithm and commercial software for different parts.

Part amount	time for proposed algorithm (s)	time for commercial software (s)
100	0.078	0.62
200	0.14	2.19
300	0.219	5.68
400	0.328	9.46
800	0.766	88.15
1000	1.109	More than 300

5.2 Actual cutting experiment

The actual cutting experiment uses a 40 W desktop laser cutting machine, and cutting metal sheet is 3 mm thickness of acrylic metal sheet. First, the parts group in the previous case is cut and verified. In this case, there are five parts with six contours. After the proposed algorithm finishes bridging, the number of contours decreases to two. The overall cutting length is 2.84 m with two piercing points. **Fig. 19a** shows the cutting process with previous case. And **Fig. 19b** shows the result on control panel that the total

process costs 5 minutes and 58 seconds with 8 mm/s cutting speed. The finished cutting product is shown in Fig. 20.

Considering that this example is too simple to highlight the feasibility and superiority of the proposed algorithm, a more complex part group is selected for the second actual machining experiment. Fig. 21 shows the nesting results for this part group. There are 24 parts with 28 contours, among which there is a circular part. The cutting result on control panel is shown in **Fig. 22a** that the cutting speed is the same as the previous experiment, set at 8mm / s, and the total processing time is 11 minutes and 46 seconds. The finished product is shown in **Fig. 22b**. Parts in this experiment are complex in shape and large in number, but the algorithm proposed in this paper avoids the damage to the contour characteristics of the parts caused by bridge successfully. At the same time, the proposed algorithm finds the global shortest bridge position in a very short time.

The feasibility and superiority of the proposed algorithm has been verified by two actual cutting experiments in this section. When the shape of parts is complex and the number is large, the algorithm can still avoid the damage to the contour characteristics of the parts caused by bridge and ensure that all parts are bridged at the shortest global distance.

6 Conclusions

In laser cutting processing, cutting parts needs piercing, which is usually a time-consuming operation for thicker metal sheet. Bridging is an effective method to reduce the number of piercing points but usually damage the contour characteristics of parts. To solve this problem, a new bridging algorithm for laser cutting is proposed in this paper. This algorithm can calculate the shortest distance of all bridged parts without damaging the contour characteristics of the parts, with a considerable calculation speed. The input of the algorithm is all the parts that need to be bridged and some necessary parameters (minimum distance D_m , bridge width D_w and the longest bridge length D_l). Firstly, all the parts to be bridged are discretized into a point cloud by D_m . In order to ensure that the bridge generated later will not damage the contour characteristics of the parts, some points need to be screened out. Next, all discrete points are constructed into a series of triangles by Delaunay triangulation. To reduce the amount of calculation, some restrictions are set to filter out invalid triangles. The remaining triangles are sorted to calculate the shortest distance between adjacent parts. In order to avoid the bridging loop problem, all the calculated line segments obtained are sorted by length from short to long and bridge parts from the shortest one. So far, the global shortest distance to bridge all parts has been found. Finally, all lines are extended into bridges according to D_w . The output of the algorithm are all bridges and the time complexity of the proposed algorithm is $O(n \log n)$, in which n is the total point number of the input point cloud. Comparison between proposed algorithm and the existing commercial software proves that the proposed algorithm can complete the required functions in a much shorter time. Two actual cutting experiments verify the feasibility and superiority of the proposed algorithm.

Declarations

Funding

This work was financially supported by the Natural Science Foundation of Zhejiang Province, China (No. LGG19E050027) and the University Laboratory Research Project of Zhejiang Province, China (N0. ZD202001).

Competing Interests

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Availability of data and materials

Not applicable

Code availability

Not applicable

Ethical Approval

The authors declare that they have no conflict of interest.

Consent to Participate

Consent to Participate

Consent for publication

Consent for publication

Authors Contributions

Not applicable

Acknowledgment

This work was financially supported by the Natural Science Foundation of Zhejiang Province, China (No. LGG19E050027) and the University Laboratory Research Project of Zhejiang Province, China (N0. ZD202001).

Data Availability

Not applicable

References

1. Makarovskikh T, Panyukov A. Software for the Problem of Constructing Cutting Tool Paths in CAD/CAM Systems for Technological Preparation of Cutting Processes[J]. Automation and Remote Control 2021; 82(3)
2. Makarovskikh T, Panyukov A. Development of Routing Methods for Cutting out Details[C]// OPTA-SCL 2018. 2018
3. Huang X, Xi F, Li J et al (2009) Optimal layout and path planning for flame cutting of sheet metals. Int J Comput Integr Manuf 22(1):30–41
4. Rao Y, Huang G, Li P et al (2006) An integrated manufacturing information system for mass sheet metal cutting. Int J Adv Manuf Technol 33(5):436–448
5. Raggenbass A, Reissner J (1991) Automatic generation of NC production plans in stamping and laser cutting. CIRP Ann Manuf Technol 40(1):247–250
6. Pan M, Rao Y (2009) An integrated knowledge based system for sheet metal cutting–punching combination processing. KnowlBased Syst 22(5):368–375
7. Reginald D, Pieter V, Dirk C (2016) A review of cutting path algorithms for laser cutters[J]. The International Journal of Advanced Manufacturing Technology 87:5–8
8. Raggenbass A, Reissner J (1989) Stamping - laser combination in sheet processing. Ann CIRP 38(1):291–294
9. Manber U, Israni S (1984) Pierce Point Minimization and Optimal Torch Path Determination in Flame Cutting[J]. Manuf Syst 3(1):81–89
10. Dewil R, Vansteenwegen P, Cattrysse D (2014) Construction heuristics for generating tool paths for laser cutters. Int J Prod Res 52(20):5965–5984
11. Dewil R, Vansteenwegen P, Cattrysse D, Vossen T, Laguna M (2015) An improvement heuristic framework for the laser cutting tool path problem. Int J Prod Res 53(6):1761–1776
12. Dewil R. On generating tool paths for laser cutters, vol 232. PhD dissertation 2014, Leuven
13. Dewil R, Vansteenwegen P, Cattrysse D (2015) Sheet metal laser cutting tool path generation—dealing with overlooked problem aspects. Key Eng Mater 639:517–524
14. Lee D, Schachter B. Two algorithms for constructing a Delaunay triangulation[J]. International Journal of Computer & Information Sciences 1980; 9(3)

15. Toussaint G, Bhattacharya B. Optimal algorithms for computing the minimum distance between two finite planar sets[J]. North-Holland 1983; 2(2)
16. Prosenjit B, Pat M. Online Routing in Triangulations[J]. SIAM Journal on Computing 2012; 33(4)
17. Jan G.. Mechatronics (2014) An $O(n \log n)$ Shortest Path Algorithm Based on Delaunay Triangulation[J]. IEEE/ASME Transactions on 19(2):660–666
18. Berg M, Cheong O, Kreveld M et al. Computational geometry: algorithms and applications. 2008
19. Lin Z, Fu J, He Y et al. A robust 2D point-sequence curve offset algorithm with multiple islands for contour-parallel tool path[J]. Computer-Aided Design 2013; 45(3)

Figures

Figure 1

An example of nesting parts.

Figure 2

An example of cutting path with piercing

Figure 3

Two approaches to avoid piercing; (a) normal cutting with two piercing points; (b) common cut with one piercing point; (c) bridging with one piercing point.

Figure 4

An example that the bridge damages part contour characteristic.

Figure 5

Structure of bridge

Figure 6

Two ways of discrete curves

Figure 7

Structure of all parts.

Figure 8

Part bridging and post-processing.

Figure 9

An example that shorter bridge might damage the part surface characteristic.

Figure 10

Selecting discrete points by D_m .

Figure 11

A result of Delaunay triangulation for discrete points.

Figure 12

Remaining triangles through two steps delete.

Figure 13

Triangles contain or cross endpoints of parts.

Figure 14

Two situations of finding the shortest line.

Figure 15

All lines in SL and bridge loops built by them.

Figure 16

The shortest bridge to connect these parts.

Figure 17

Bridging result for proposed algorithm.

Figure 18

Bridging result for commercial software.

Figure 19

Cutting experiment for previous case.

Figure 20

Finished product.

Figure 21

Nesting result without bridge.

(a)

(b)

Figure 22

Cutting experiment for previous case.