DOI: 10.1111/cgf.14968
Pacific Graphics 2023
R. Chaine, Z. Deng, and M. H. Kim
(Guest Editors)

COMPUTER GRAPHICS *forum*
Volume 42 (2023), Number 7

Dissection Puzzles Composed of Multicolor Polyominoes

N. Kita

Tokyo University of Agriculture and Technology, Japan

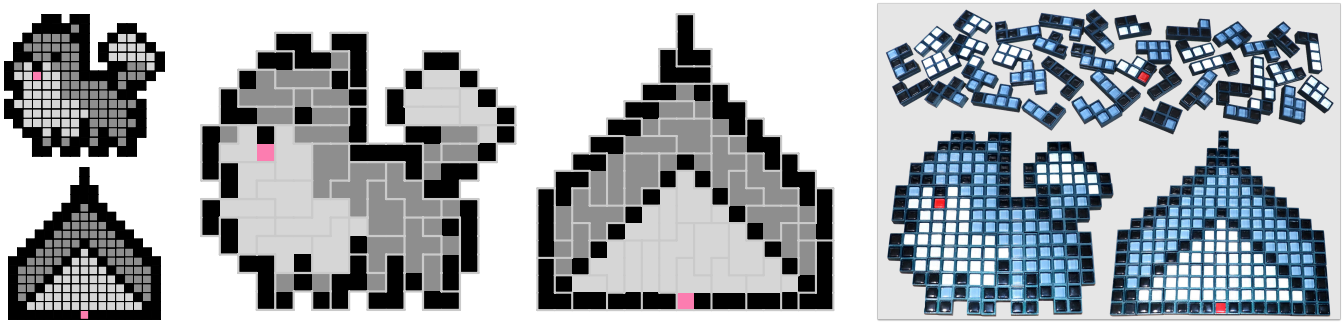


Figure 1: *In this paper, we propose a method for generating dissection puzzles composed of multicolor polyominoes. Given target patterns as inputs (left), we find a set of multicolor polyomino pieces that dissect the inputs (middle). We observed the Pomeranian could transform into a house and vice versa with the 3D-printed pieces (right).*

Abstract

Dissection puzzles leverage geometric dissections, wherein a set of puzzle pieces can be reassembled in various configurations to yield unique geometric figures. Mathematically, a dissection between two 2D polygons can always be established. Consequently, researchers and puzzle enthusiasts strive to design unique dissection puzzles using the fewest pieces feasible. In this study, we introduce novel dissection puzzles crafted with multi-colored polyominoes. Diverging from the traditional aim of establishing geometric dissection between two 2D polygons with the minimal piece count, we seek to identify a common pool of polyomino pieces with colored faces that can be configured into multiple distinct shapes and appearances. Moreover, we offer a method to identify an optimized sequence for rearranging pieces from one form to another, thus minimizing the total relocation distance. This approach can guide users in puzzle assembly and lessen their physical exertion when manually reconfiguring pieces. It could potentially also decrease power consumption when pieces are reorganized using robotic assistance. We showcase the efficacy of our proposed approach through a wide range of shapes and appearances.

CCS Concepts

• ***Computing methodologies*** → ***Computer graphics;*** • ***Mathematics of computing*** → ***Combinatorial optimization;***

1. Introduction

Puzzles have long fascinated people, with dissection puzzles—geometric puzzles that involve assembling a set of pieces into different geometric shapes—serving as a captivating example [Fre97, Fre02]. These puzzles, such as the *Tangram* and *T puzzle*, challenge users to rearrange pieces to form distinct shapes (Figure 2). A key mathematical insight underpinning these puzzles is the Wallace-Bolyai-Gerwien theorem, which posits that two 2D polygons can be dissected into one another if and only if they have an equal area [Gar85]. This theorem has ignited a line of research into finding minimal dissections, i.e., solutions involving the fewest possible pieces. Despite proving that minimizing the number of pieces is an NP-hard problem [BDD*15], and approximating this minimum is equally challenging [MRY16], the pursuit continues.

In this paper, we bring a novel perspective to dissection puzzles by considering not only the geometric shapes but also their appearances, specifically their coloration (Figure 1). Our task is more nuanced than simply finding a geometric dissection between two 2D polygons with the fewest possible pieces. We strive to find a common set of polyominoes—each with colored faces—that can be assembled into multiple, distinct shapes and appearances. While finding a minimal number of polygonal pieces for dissection may

Figure 2: *Tangram (left) and T puzzle (right) are well-known dissection puzzles.*

be more challenging, our problem also presents its unique difficulties, particularly in achieving a perfect match for the silhouette and appearance of two or more target shapes. To tackle this problem, we adopt integer programming (IP), an appropriate method due to its inherent suitability for dealing with discrete variables and its proven efficiency in solving combinatorial optimization problems.

In creating these multicolor dissection puzzles, we noted the challenge puzzle designers face in designing shapes that are visually appealing and balanced in color distribution. In response, we developed a support tool to generate symmetric patterns—a preference supported by studies highlighting the human predilection for symmetry [Wey52, CH06]—given a specific area for each color. Our tool not only helps in generating aesthetically pleasing patterns but also works in tandem with our method to find dissections between them.

Furthermore, we introduce a method to determine an optimized sequence for rearranging pieces from one shape to another, thereby minimizing the total moving distance of the pieces. This optimized rearranging sequence (ORS) could offer tangible benefits: guiding users during puzzle assembly if the users want to know how to rearrange the pieces, reducing physical exertion during manual reconfiguration, and potentially lowering power consumption if robotic assistance is used for reorganization.

We showcase the efficacy of our proposed approach by making the puzzles through an array of shapes and appearances, demonstrating its potential for both puzzle enthusiasts and professional designers.

2. Related Work

Geometric Puzzle Design. Geometric puzzle design has been studied for decades [Cof06]. While puzzle solvers are required to arrange polyomino tiles to cover target 2D shapes in traditional polyomino tiling puzzles [Gol94], Lo et al. presented a method to construct three-dimensional (3D) polyomino puzzles [LFL09]. Song et al. proposed a constructive approach to generate recursive interlocking puzzles from given voxel models [SFCO12]. Recently, Chen et al. presented a computational design method of high-level interlocking puzzles [CWSB22], where multiple moves are required to take out the first subassembly from the puzzle. While those are bottom-up constructive approaches, Kita and Miyata proposed a top-down partitioning approach in designing polyomino puzzles to improve the controllability of the puzzle designers [KM21].

In the computer graphics community, researchers have worked on generalizing traditional puzzle mechanisms. Xin et al. generalized the traditional six-piece cuboid orthogonal burr puzzles to design burr puzzles from 3D models [XLF*11]. Sun and Zheng generalized Rubik's Cube mechanism to design complex twisty joints and puzzles of general 3D models [SZ15]. While the traditional centrifugal puzzles are played on a tabletop due to their mechanism, Kita and Saito proposed a computational method to design generalized centrifugal puzzles [KS20].

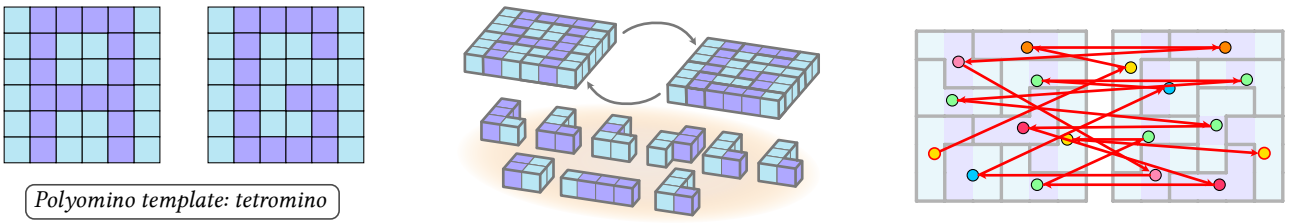
Geometric Dissections. Geometric dissection puzzles are one of the famous geometric puzzles, where puzzle solvers are required to arrange a set of pieces to two or more distinct forms [Fre97]. Zhou and Wang proposed a computational method to design geometric dissection puzzles [ZW12]. However, the method was demonstrated only on coarse discrete grids with simple target shapes. On the other hand, Tang et al. presented a method to design steady 3D dissection puzzles with more complex targets by accepting slight modifications [TSW*19]. In addition to those methods that target 2D or 3D shapes represented as a discrete grid, Duncan et al. proposed a dissection design technique that approximately dissects naturalistic 2D shapes [DYYT17].

Hinged dissection is a special case of dissection, where all pieces are connected in a chain at hinged points and one shape transforms into another shape by swinging the chain continuously [Fre02]. Li et al. proposed a construction method for reversible inside-out transform (RIOT) [LMAH*18]. RIOT is a reversible hinged dissection, where two 2D shapes can be inverted inside-out and transformed into each other. Similar to hinged dissection, computer graphics researchers have studied transformable objects. *Boxelization* transforms a 3D object into a cube or a box with a continuous folding sequence [ZSMS14]. Yuan et al. proposed a computational method to design transformable robots that shape-shift to different forms [YZC18]. Yu et al. presented *LineUp*, which computes physical transformations between 3D models with different shapes and topology with a chain structure [YYL*19].

We propose a novel dissection puzzle composed of multicolor polyominoes. We find a common set of *colored* polyomino pieces that can be rearranged into two or more distinct shapes. The most relevant to our puzzles are the *Checkerboard puzzles* [SH97] and the *Chequers Puzzles* [Hof93]. The Checkerboard puzzles consist of a dissected standard 8×8 checkerboard, while the Chequers Puzzles consists of a miniature (5 in. square) chessboard divided into 14 pieces, each consisting of three to five squares. For the Checkerboard puzzles, the puzzle solvers try to reassemble the pieces into an 8×8 square with the proper checkering. Unlike the Checkerboard puzzles and the Chequers Puzzles that focus only on checkerboard patterns, we focus on coloring not restricted to checkerboard patterns, i.e., designers can design their desired coloring on target shapes.

3. Solving Dissection Problem

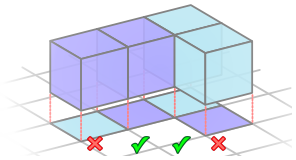
Representation. Given target patterns, i.e., shapes with coloring $\{T_1, \ldots T_N\}$, polyomino templates P_i $(i = 1, \ldots, |N|)$, and the number of colors $k = 1, \ldots, |K|$, our goal is to find a set of common *colored* polyomino tiles (puzzle pieces) that can be rearranged from



(a) Target patterns & polyomino templates (b) Find a set of pieces that dissects the targets (c) Find the optimized rearranging sequence

Figure 3: *Overview of the proposed method. Given target patterns and polyomino templates (a), our goal is to find a common set of colored polyomino tiles that can be rearranged from one target to the other by translations, rotations, and/or flip (b), and we also find the optimized rearranging sequence (c).*

one target to another by translations, rotations, and/or flip (Figure 3 (a, b)). We assume a polyomino tile consists of voxels, each of which has a single color, and the target patterns are represented on a voxel grid.



Variables. We introduce variables used in our 0-1 IP formulation in advance. The binary variable $P_{i,j,k}$ takes 1 if a polyomino tile of type i, index j with coloring pattern k is used and 0 otherwise. The index is assigned to each type of polyomino tile according to where to place it on a discrete grid. To be exact, we represent the index on target T_n as j_n since the index may differ depending on targets. We set a weight $W_{i,j_n,k}$ to each $P_{i,j_n,k}$ by summing up the mismatches between $P_{i,j_n,k}$ and the corresponding coloring pattern of T_n (inset). We compute the mismatch by calculating the sum of the absolute difference for each voxel color of $P_{i,j_n,k}$ and the corresponding target voxel color. Furthermore, we introduce a binary 3D array $\mathbf{A}^{(n)} \in \{0,1\}^{|T_n|\times|J_n|\times|K|}$, where if $P_{i,j_n,k}$ covers the ℓ-th voxel $v_\ell \in T_n$, $\mathbf{A}^{(n)}_{\ell,j_n,k}$ is set to 1 and 0 otherwise.

Objective Function. We formulate our combinatorial optimization problem as a 0-1 IP problem. One trivial solution is to use the *monominoes* (1×1 tiles) as puzzle pieces. Upon our problem configuration and assumptions, we can always find the solution for dissection puzzles using *monomino* tiles. However, such a solution spoils the entertainment of dissection puzzles. To avoid such trivial solutions and make the generated puzzles more enjoyable, we set our objective to minimize the number of pieces:

$$\min \sum_{i\in I}\sum_{j_1\in J_1}\sum_{k\in K} P_{i,j_1,k}. \tag{1}$$

In the following paragraphs, we describe the constraints of our IP formulation.

Exact Cover Constraint. We have to exactly cover the target voxels. That is, every voxel should be covered by a single polyomino tile exactly once without overlap. We add the following constraint to achieve the exact cover tiling:

$$\sum_{i\in I}\sum_{j_n\in J_n}\sum_{k\in K}\sum_{\ell\in T_n} \mathbf{A}^{(n)}_{\ell,j_n,k}\cdot P_{i,j_n,k} = 1 \qquad \forall n\in N. \tag{2}$$

Common Set Constraint. Since we have to find a common tile set between given target patterns, we add the following constraints:

$$\sum_{j_1\in J_1} P_{i,j_1,k} = \sum_{j_2\in J_2} P_{i,j_2,k} = \cdots = \sum_{j_n\in J_n} P_{i,j_n,k} \qquad \forall i\in I, \forall k\in K. \tag{3}$$

With this constraint, we represent that the same number of polyomino templates should be used in each target.

Pattern Match Constraint. Finally, we add the following constraint for the exact pattern (appearance) matches:

$$\sum_{i\in I}\sum_{j_n\in J_n}\sum_{k\in K} W_{i,j_n,k}P_{i,j_n,k} = 0 \qquad \forall n\in N. \tag{4}$$

Enumerating Intrinsic Patterns. When enumerating polyomino tiling $P_{i,j,k}$, we compute all possible rotations and/or flips. For each tile with a coloring pattern, there are eight operations, i.e.,

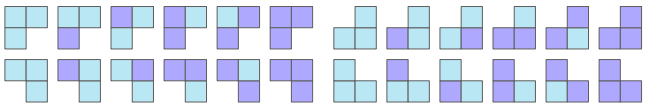


Figure 4: *Possible intrinsic patterns. For instance, 24 patterns are listed for the case of bicolor L-tromino.*

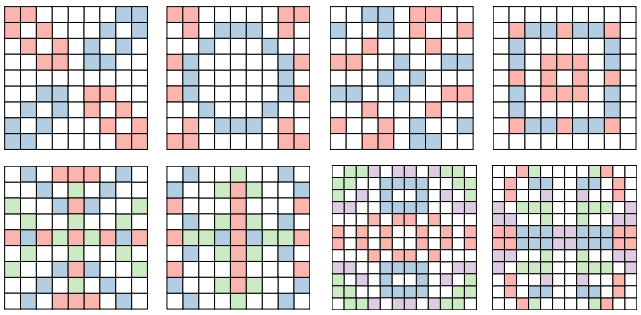


Figure 5: *Example symmetric patterns generated using our tool. Since designing geometric patterns with the same number of colors is a demanding task, we support such design tasks by providing a tool to generate symmetry patterns from the given number of target areas and colors procedurally.*

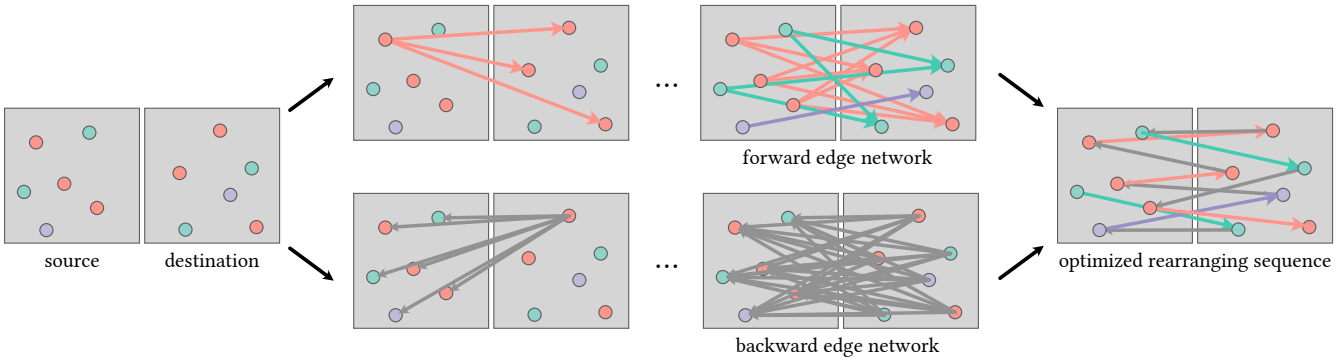


Figure 6: *Given two target tilings, namely, the source board T_m and the destination board T_n (left), we construct a forward edge network by connecting each puzzle piece (node) in T_m to the nodes with the same type (represented as circles with the same colors) in T_n, while a backward edge network is constructed by connecting each node in T_n to all nodes in T_m (middle). We then find the optimized rearranging sequence by solving an IP problem (right).*

$\{0, \pi/2, \pi, 3\pi/2\}$-planar rotations and those for flipping. That is, we have to compute a total of $8 \cdot |T_n|^{|K|}$ patterns for each polyomino template type i of index j. However, identical patterns may exist as a result of applying those operations. For example, Figure 4 shows the case of L-type bicolor tromino. Potentially, there are $8 \times 2^3 = 64$ patterns. However, we only consider the combinations of 4 posing and 6 coloring, i.e., 24 *intrinsic* patterns, to reduce the computational cost.

Generating Symmetric Patterns. Designing geometric patterns with the same number of colors is a demanding task. To tackle this issue, we support such design tasks by providing a tool to procedurally generate symmetry patterns from the user-specified number of target areas and colors (Figure 5). Since symmetric patterns are preferable to those of nonsymmetric patterns for humans [Wey52, CH06], such a choice is reasonable. Currently, we consider reflectional symmetries in our tool. Now users can design patterns manually or use our symmetric pattern generator, or combine both approaches to obtain preferable pattern pairs.

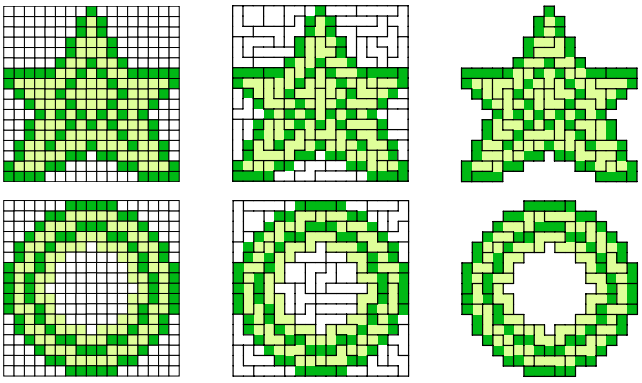


Figure 7: *Non-rectangular dissection puzzles (right) as well as rectangular ones (middle) can be generated from the same targets (left) without any modifications to the formulation by specifying a specific color as a transparent background.*

4. Optimized Rearranging Sequence

We can identify a common set of polyomino tiles that can dissect distinct targets using the approach described in the previous section. However, such an approach only finds a set of puzzle pieces and cannot find a sequence of rearranging those pieces.

In this section, we provide another IP formulation to find an ORS (Figure 3(c)). The ORS could help guide users if they want to know how to rearrange the pieces. There may be several ways to rearrange pieces depending on the user's objective. Here, we minimize the total moving distances of puzzle pieces when rearranging from one target to another. This choice of objective is reasonable since we can reduce the user's physical load of manual rearrangement. Potentially, the shortest rearranging sequence can also reduce power consumption when rearranging with robot hands in an automated situation. The ORS is a variant of the shortest Hamiltonian Path or the Traveling Salesman Problem (TSP). However, our problem involves additional constraints that are difficult to apply directly in the state-of-the-art TSP solvers, which led to the development of a dedicated solver.

Figure 6 shows the overview of the proposed approach. We first construct the forward/backward edge network and then solve the IP problem to find the optimized sequence. In the following paragraphs, we introduce the decision variables, objective, and high level descriptions of the constraints. For the detailed descriptions of the constraints, please see Appendix A.

Variables. We denote the path of rearranging a piece from target T_m to T_n, i.e., an outgoing edge as a binary variable $e_{ij} \in \overrightarrow{E}$, where i and j represent nodes (pieces) in T_m and T_n, respectively. If we use edge e_{ij}, it is set to 1 and 0 otherwise. Similarly, $e_{ji} \in \overleftarrow{E}$ represents an incoming edge. We denote $\mathcal{E} = \overrightarrow{E} \cup \overleftarrow{E}$. Edge e_{ij} has a weight w_{ij} and its value is the Euclidean distance between nodes i and j. We can also consider the costs for piece flipping and rotation by scaling the weights or adding extra costs. $o_k \in \mathcal{O}$ is an integer

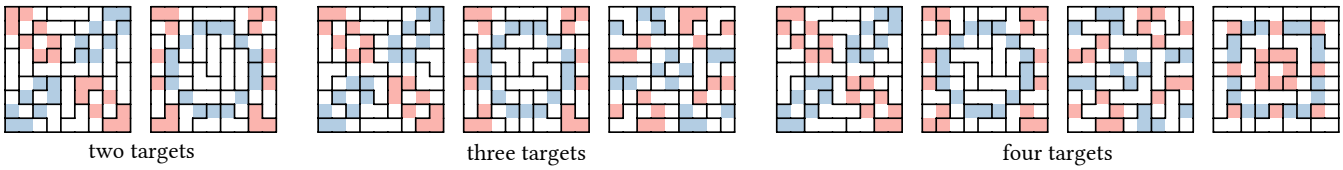


Figure 8: *Dissection puzzles of tricolor polyomino pieces with two, three, and four targets. The target patterns are created using our tool (cf. Figure 5).*

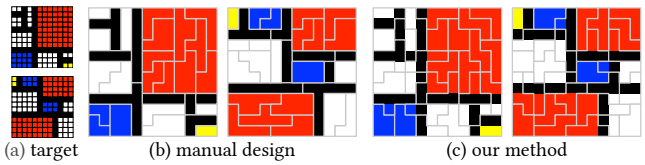


Figure 9: *Comparison of manual dissection and result found by our solver. Given target patterns inspired by Mondrian's Composition with Red Blue and Yellow (a), we can manually design dissection pieces, where each piece comprises single colors, whereas we can find nontrivial dissection pieces with our solver (c). Note that both (b) and (c) have 30 pieces.*

variable that represents the order of rearranging sequence, where $k \in \{0, 1, \ldots, 2N-1\}$ and N is the number of nodes in T_n. Binary variables $x_i,\ x_j \in \mathcal{X}$ represent whether the node i is the first node ($x_i = 1$) or not ($x_i = 0$) and node j is the terminal node ($x_j = 1$) or not ($x_j = 0$), respectively.

Formulation. Our IP Problem that minimize the total moving distance of rearranging pieces from one target to another can be formulated as follows:

$$\begin{aligned} \text{minimize} \quad & \sum_{e_{ij} \in \overrightarrow{E}} w_{ij} e_{ij} + \sum_{e_{ji} \in \overleftarrow{E}} w_{ji} e_{ji}, \\ \text{subject to} \quad & C_{\text{indegree}}(\mathcal{X}, \mathcal{E}), \\ & C_{\text{outdegree}}(\mathcal{X}, \mathcal{E}), \\ & C_{\text{exclusivity}}(\mathcal{X}, \mathcal{E}), \\ & C_{\text{order}}(\mathcal{X}, \mathcal{E}, \mathcal{O}). \end{aligned} \tag{5}$$

This minimizes the sum of outgoing and incoming edge weights under several constraints. For the detailed descriptions about the indegree constraint C_{indegree}, the outdegree constraint $C_{\text{outdegree}}$, the exclusivity constraint $C_{\text{exclusivity}}$, and the order constraint C_{order}, please refer to the Appendix A.

5. Experimental Results

Figure 1 shows the typical multicolor dissection puzzle designed by the proposed method. Figure 7 shows the rectangular and non-rectangular dissection puzzles generated by the proposed method. We can generate non-rectangular dissection puzzles by treating the background as a transparent color; thus, users can design diverse shapes for the puzzles.

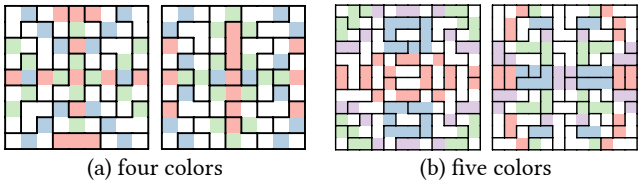


Figure 10: *Most of the results shown in this paper are two- and three-color targets, but we can also find dissections for (a) four- and (b) five-color targets. The target patterns are created using our tool (cf. Figure 5).*

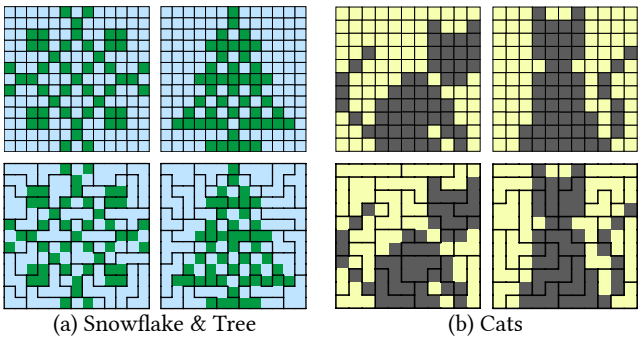


Figure 11: *Aside from geometric patterns, we can also create dissection puzzles for other motifs, such as animals and plants.*

Comparison to Manual Design. As shown in Figure 9, we can manually design the decomposition of targets with single-color polyomino pieces (b). However, such trivial dissection spoils the satisfaction of solving the puzzle. On the other hand, we can create *nontrivial* dissection puzzles using our method (c).

Versatility. Figure 8 shows that the proposed method can also accept more than two target patterns. Most of the results shown in this paper have two or three colors. Figure 10 shows the dissection puzzles with four and five colors, demonstrating the versatility of the proposed method. As shown in Figure 11, we can create multicolor dissection puzzles using various motifs, including plants and animals as well as geometric patterns.

Fabrication. We have physically made several dissection puzzles designed by the proposed method using a 3D printer and glass tiles (Figure 1 right and Figure 12). We printed the base tile holder using

Table 1: *Timing statistics for solving dissection puzzles. † indicates suboptimal solutions the solver found at the reported timing. The closeness (gap) to the optimal solutions is provided.*

Target	Figure	#Colors	Tile set (n-omino)	#Tiles	First sol.	Shown sol.	Gap †
Pomeranian & House	1	4	$n=4,5$	39	1313 s	22252 s	-
Geometric pattern (two targets)	8 left	3	$n=3$	27	0.14 s	0.15 s	-
Geometric pattern (three targets)	8 middle	3	$n=3$	27	0.29 s	0.31 s	-
Geometric pattern (four targets)	8 right	3	$n=2,3,4$	27	7.06 s	27 s	-
Star & Circle	7 middle	2	$n=2,3,4$	76	22 s	264 s	4.93%
Star & Circle (transparent background)	7 right	2	$n=2,3,4$	44	3.43 s	408 s	-
Mondrian's Composition	9 (c)	4	$n=3,4,5$	30	622 s	789 s	4.00%
Geometric pattern (four colors)	10 (a)	4	$n=2,3,4$	24	3.68 s	225 s	-
Geometric pattern (five colors)	10 (b)	5	$n=2,3,4$	43	9.07 s	1634 s	-
Snowflake & Tree	11 (a)	2	$n=3,4,5$	36	979 s	1282 s	6.11%
Cats (side & front)	11 (b)	2	$n=3,4,5$	27	33 s	64 s	10.4%
PACIFIC GRAPHICS 2023	14	2	$n=2,3,4$	19	4998 s	12067 s	5.26%
Dissection Fonts 0–9	15	2	$n=2,3,4$	19	533 s	1093 s	15.8%
QR Codes	16	2	$n=3,4$	78	2174 s	3553 s	5.77%

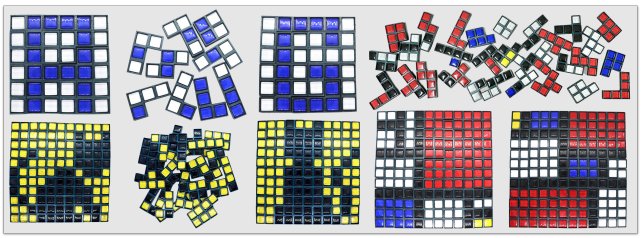


Figure 12: *Fabrication results.*

a 3D printer and attached a glass tile for each tile face of the holder with an adhesive.

Evaluation of Dissection Puzzle Solver. We used the Gurobi Optimizer [Gur23] to solve the IP problems described in Sections 3 and 4. All experiments were done on an Intel Core i9, 2.4 GHz personal computer. Table 1 lists the statistics. In our experiments, we have tested various cases, i.e., changing the number of colors from 2 to 5, and tile sets (domino, tromino, tetromino, pentomino, and their combinations). The first sol. column reports the time that the solver finds the first solution and the shown sol. column reports the time of the results shown in the figures.

Evaluation of the ORS Solver. We compared the proposed ORS solver and the greedy approach regarding the ORS quality (objective value) and timing using synthetic configurations (Figure 13). Our greedy approach iteratively selects the target node exhibiting the minimal edge weight, continuing this process until all nodes are traversed. We generated N node positions of M-class (node types) in source and destination targets located side by side (Figure 13 left). As shown in Figure 13, the node colors represent the node types. The middle and right of Figure 13 show the results found by the greedy and our IP approaches, respectively. Table 2 presents the statistics for the greedy and our IP approaches. As the number of nodes increases, the greedy approach takes a long time to find

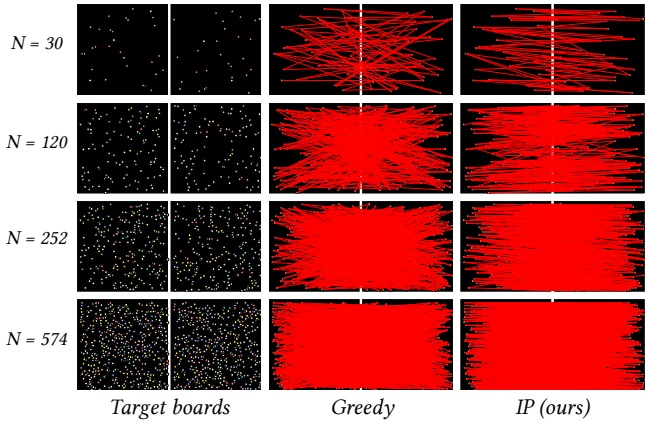


Figure 13: *Comparison of the rearranging sequence of the greedy and our IP approaches. We can visually observe that our IP-based solver finds better (shorter) sequences than the greedy approach.*

Table 2: *Timing statistics for solving the ORS problem.*

N	M	Greedy	IP (ours)	Obj val. (IP/greedy)
30	5	0.0129 s	0.0677 s	0.930
120	10	0.9211 s	1.8782 s	0.924
252	15	13.9152 s	11.5262 s	0.933
574	20	332.9344 s	130.4043 s	0.938

the ORS due to combinatorial complexity, while our IP solver finds the ORS in reasonable time budgets. Furthermore, the ratios listed in the right column show that the greedy approach finds approximately 7% longer sequences than our IP-based approach.

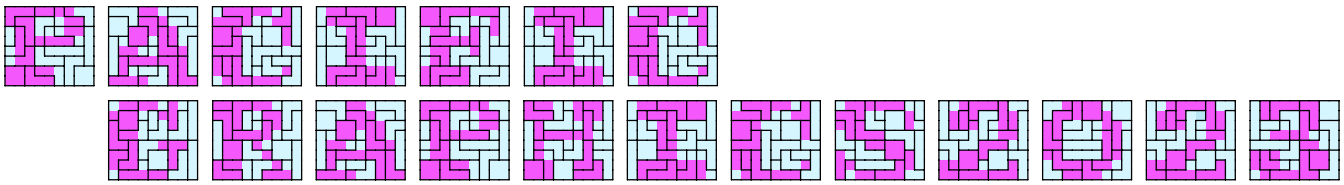


Figure 14: *PACIFIC GRAPHICS 2023 synthesized with our dissection font (19 pieces).*

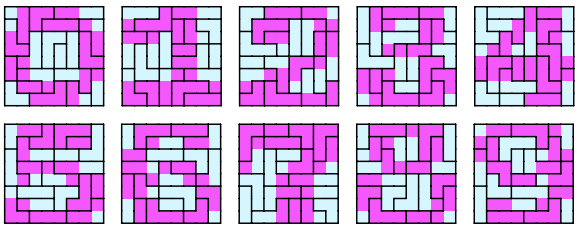


Figure 15: *Dissection font 0–9.*

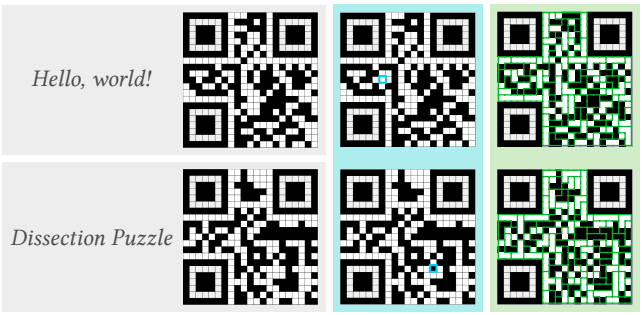


Figure 16: *Dissection QR codes. Given two data; "Hello, world!" and "Dissection Puzzle" texts, we first generate QR codes from these data (left). We then modify these QR codes to have the same number of black and white cells (middle). We reverse the randomly chosen white and black cells at the enclosed positions with blue rectangles. Finally, we apply our method to solve the dissection problem (right). We exclude three finder patterns located at the upper corners and bottom left from the target regions.*

6. Applications

Dissection Font. Demaine et al. proposed the dissection font [DDKU18]. Knuth has also presented *Font 36* [Knu20]. Inspired by the *Font 36*, we have also created our dissection fonts 0–9 (Figure 15). We can rearrange polyomino pieces to make digits 0–9. Unfortunately, we could not find the dissection pieces of all the 36 characters using *domino*, *tromino*, and *tetromino* within a reasonable time amount. However, we can find partial dissections. Figure 14 shows the synthesized result for the text *PACIFIC GRAPHICS 2023* using the dissection font designed by our method.

Dissection QR Codes. Our method can apply to generate dissection *Quick Response* (QR) codes (Figure 16). Besides the entertainment value of solving dissection puzzles, the dissection QR code can encode additional information into the pattern. Given two data, we first generate QR codes from these data (Figure 16 left). These QR codes can have different numbers of black and white (BW) cells. Thus, we modify the patterns so that the pair of QR codes has the same number of BW cells. We randomly select a pair of cells and reverse the BW colors until two patterns have the same number of BW cells (Figure 16 middle). Finally, we solve the dissection problem to generate the dissection QR codes (Figure 16 right). We exclude three finder patterns located at the upper corners and bottom left from the target regions during optimization. In Figure 16, we use the QR code version 1 (21×21) with error correction level L (low). We can apply our method to other 2D codes, such as *Micro QR codes* and *Aztec codes*, as well as other versions and/or error correction levels of QR codes.

7. Discussions

Dissection Problem. While we have found the optimal dissection solutions on several targets, we could not find the optimal solutions on the other cases reported with gaps in Table 1. Unfortunately, it is difficult to estimate which case is harder, i.e., takes a long time to solve due to its combinatorial nature, even if the targets look easier to solve intuitively. However, we observed that using small n-ominoes, such as dominoes and trominoes, we can find solutions relatively easier than using larger n, such as tetrominoes and pentominoes.

While the general problem of finding a minimal dissection is acknowledged to be NP-hard, we recognize that our specific formulation with a common set of puzzle pieces has not been rigorously proven to share this complexity. Indeed, there exists a possibility that a polynomial-time solution might be applicable to our scenario. A comprehensive exploration of the complexity of this specific problem forms an interesting avenue for future work.

Moreover, although solving the dissection problem to find the optimal solution for Figure 1 takes over six hours, we can find the first solution that meets the constraints Eqs. 2–4 less than half an hour. Even if the solution is locally optimal, it is an advantage in geometric problem solving to obtain a solution that satisfies the constraints in a relatively short time (cf. [ZW12, DYYT17]). More performance improvement could be required if building interactive applications, such as an interactive design system.

ORS. Our experimental findings indicate an optimal sequence that is 7% shorter than a solution derived using a greedy approach. While this might appear modest, it becomes increasingly significant as the problem size grows.

Further delving into the contrast between 2D and 3D puzzles, it is essential to highlight our formulation's adaptability. Though our study is rooted in 2D puzzles, the inherent versatility of our formulation allows for a straightforward extension to 3D dissection puzzles. In a 3D scenario, optimizing the rearranging sequence is likely to result in even more pronounced efficiencies. Moreover, such a transition introduces a series of compelling challenges that could form the basis for future exploration. In this light, our current research not only provides insights into the 2D domain but also sets the stage for more complex and multi-dimensional investigations.

8. Conclusion and Future work

In this paper, we introduced novel dissection puzzles that utilize multicolor polyominoes. We proposed a 0-1 IP formulation to find optimal dissection solutions, a versatile approach that accommodates puzzles with multiple shapes and color combinations. Moreover, we developed an IP solver specifically tailored to find the ORS for these dissection puzzles, which provides users guidance on how the pieces be rearranged.

One possible future research would extend the methodology to 3D multicolor polycube dissection puzzles. In our current model, we utilized polyomino templates and sought to minimize the number of puzzle pieces subject to various constraints. However, this approach does not allow for the identification of dissection puzzles composed of multicolor polyominoes with a global minimum number of pieces. This limitation represents an exciting challenge to be addressed in future work as it could further optimize puzzle complexity and solvability. Second, while we demonstrated two applications; dissection fonts and QR codes that can be used as entertainment and educational applications, our dissection puzzles can be used for architectural, interior decoration, and reconfigurable furniture and robots. Another interesting future direction is to extend our dissection puzzle formulation to dissection tiling problems. Since tiling such as zellij has broad applications, extending our dissection puzzle formulation to dissection tiling problems would open novel exciting problems to explore. Third, while we try to find common polyomino tiles among target patterns in this paper, it would also be interesting to generate diverse patterns from a common set of polyomino tiles. Finally, our investigation into the rearranging sequences of dissection puzzles has uncovered interesting parallels with path-finding problems. This connection presents intriguing possibilities for applications such as reconfigurable robot design within constrained environments.

Acknowledgements

Pomeranian pixel art in Figure 1 is courtesy of Pixilart user: PastaPenguin (https://www.pixilart.com/art/pomeranian-998a524873e2669), where a different palette has been applied. The black cat in Figure 11 is courtesy of Pixiv user: Warsomo (https://www.pixiv.net/artworks/97840604).

References

[BDD*15] BOSBOOM J., DEMAINE E. D., DEMAINE M. L., LYNCH J., MANURANGSI P., RUDOY M., YODPINYANEE A.: k-piece dissection is np-hard. In *18th Japan Conf. on Discrete and Computational Geometry and Graphs* (2015), vol. 2. 1

[CH06] CÁRDENAS R. A., HARRIS L. J.: Symmetrical decorations enhance the attractiveness of faces and abstract designs. *Evolution and Human Behavior 27*, 1 (2006), 1–18. doi:https://doi.org/10.1016/j.evolhumbehav.2005.05.002. 2, 4

[Cof06] COFFIN S.: *Geometric Puzzle Design*, 1st ed. A K Peters/CRC Press, 2006. 2

[CWSB22] CHEN R., WANG Z., SONG P., BICKEL B.: Computational design of high-level interlocking puzzles. *ACM Trans. Graph. 41*, 4 (jul 2022). doi:10.1145/3528223.3530071. 2

[DDKU18] DEMAINE E., DEMAINE M., KNUTH D. E., UNO Y.: Dissection font. https://erikdemaine.org/fonts/dissect/, 2018. (Accessed on 11/13/2022). 7

[DYYT17] DUNCAN N., YU L.-F., YEUNG S.-K., TERZOPOULOS D.: Approximate dissections. *ACM Trans. Graph. 36*, 6 (Nov. 2017), 182:1–182:13. URL: http://doi.acm.org/10.1145/3130800.3130831. 2, 7

[Fre97] FREDERICKSON G. N.: *Dissections: Plane and Fancy*. Cambridge University Press, 1997. 1, 2

[Fre02] FREDERICKSON G. N.: *Hinged Dissections: Swinging and Twisting*. Cambridge University Press, 2002. 1, 2

[Gar85] GARDNER R.: A problem of sallee on equidecomposable convex bodies. *Proceedings of the American Mathematical Society 94*, 2 (1985), 329–332. 1

[Gol94] GOLOMB S. W.: *Polyominoes: Puzzles, Patterns, Problems, and Packings Revised and Expanded Second Edition*, 2nd ed. Princeton University Press, New York, NY, 1994. 2

[Gur23] GUROBI: Gurobi optimization, 2023. URL: http://www.gurobi.com/. 6

[Hof93] HOFFMANN: *Puzzles Old and New*. Frederick Warne, 1893. 2

[KM21] KITA N., MIYATA K.: Computational design of polyomino puzzles. *The Visual Computer 37*, 4 (2021), 777–787. doi:10.1007/s00371-020-01968-5. 2

[Knu20] KNUTH D. E.: *The Art of Computer Programming: Volume 4B Combinatorial Algorithms Part 2*, vol. 4B. Addison-Wesley Professional, October 2020. 7

[KS20] KITA N., SAITO T.: Computational design of generalized centrifugal puzzles. *Computers & Graphics 90* (2020), 21 – 28. URL: https://doi.org/10.1016/j.cag.2020.05.005. 2

[LFL09] LO K.-Y., FU C.-W., LI H.: 3d polyomino puzzle. *ACM Trans. Graph. 28*, 5 (Dec. 2009), 157:1–157:8. URL: http://doi.acm.org/10.1145/1618452.1618503. 2

[LMAH*18] LI S., MAHDAVI-AMIRI A., HU R., LIU H., ZOU C., VAN KAICK O., LIU X., HUANG H., ZHANG H.: Construction and fabrication of reversible shape transforms. *ACM Trans. Graph. 37*, 6 (Dec. 2018). URL: https://doi.org/10.1145/3272127.3275061. 2

[MRY16] MANURANGSI P., RUDOY M., YODPINYANEE A.: Dissection with the fewest pieces is hard, even to approximate. In *Discrete and Computational Geometry and Graphs: 18th Japan Conf.(JCDCGG 2015)* (2016), vol. 9943, Springer, p. 37. 1

[SFCO12] SONG P., FU C.-W., COHEN-OR D.: Recursive interlocking puzzles. *ACM Trans. Graph. 31*, 6 (Nov. 2012), 128:1–128:10. URL: http://doi.acm.org/10.1145/2366145.2366147. 2

[SH97] SLOCUM J., HAUBRISH J.: *Compendium of Checkerboard Puzzles*. Slocum Puzzle Foundation, Beverly Hills, 1997. 2

[SZ15] SUN T., ZHENG C.: Computational design of twisty joints and puzzles. *ACM Trans. Graph. 34*, 4 (July 2015), 101:1–101:11. URL: http://doi.acm.org/10.1145/2766961. 2

[TSW*19] TANG K., SONG P., WANG X., DENG B., FU C.-W., LIU L.: Computational design of steady 3d dissection puzzles. *Computer Graphics Forum 38*, 2 (2019), 291–303. URL: https://onlinelibrary.wiley.com/doi/abs/10.1111/cgf.13638. 2

[Wey52] WEYL H.: *Symmetry*. Princeton University Press, 1952. 2, 4

[XLF*11] XIN S., LAI C.-F., FU C.-W., WONG T.-T., HE Y., COHEN-OR D.: Making burr puzzles from 3d models. *ACM Trans. Graph. 30*, 4 (July 2011), 97:1–97:8. URL: http://doi.acm.org/10.1145/2010324.1964992. 2

[YYL*19] YU M., YE Z., LIU Y.-J., HE Y., WANG C. C. L.: Lineup: Computing chain-based physical transformation. *ACM Trans. Graph. 38*, 1 (Jan. 2019). URL: https://doi.org/10.1145/3269979. 2

[YZC18] YUAN Y., ZHENG C., COROS S.: Computational design of transformables. *Computer Graphics Forum 37*, 8 (2018), 103–113. URL: https://onlinelibrary.wiley.com/doi/abs/10.1111/cgf.13516. 2

[ZSMS14] ZHOU Y., SUEDA S., MATUSIK W., SHAMIR A.: Boxelization: Folding 3d objects into boxes. *ACM Trans. Graph. 33*, 4 (July 2014), 71:1–71:8. URL: http://doi.acm.org/10.1145/2601097.2601173. 2

[ZW12] ZHOU Y., WANG R.: An algorithm for creating geometric dissection puzzles. In *Proceedings of Bridges 2012: Mathematics, Music, Art, Architecture, Culture* (Phoenix, Arizona, USA, 2012), Bosch R., McKenna D., Sarhangi R., (Eds.), Tessellations Publishing, pp. 49–56. 2, 7

Appendix A: Finding Optimized Rearranging Sequence

This section provides detailed descriptions for the constraints of ORS IP formulation in Section 4.

Indegree Constraints. Here we provide a detailed description for $C_{\text{indegree}}(\mathcal{X},\mathcal{E})$ constraint. The indegree for each node in T_m except for the first node is 1. This constraint can be written using the big-M notation as follows:

$$\begin{cases} \sum_{e_{ji}\in\overleftarrow{E}} e_{ji} \le 1 + Mx_i \\ 1 \le \sum_{e_{ji}\in\overleftarrow{E}} e_{ji} + Mx_i \end{cases} \quad \forall i \in I, \tag{6}$$

where I is a set of indices assigned to the nodes in T_m, J is a set of indices assigned to the nodes in T_n, and M is a sufficiently large constant value. In addition, the indegree of the first node is 0, which is written as follows:

$$\sum_{e_{ji}\in\overleftarrow{E}} e_{ji} \le M(1-x_i) \quad \forall i \in I. \tag{7}$$

Similarly, the indegree constraint for each node in T_n except for the terminal node can be written as follows:

$$\begin{cases} \sum_{e_{ij}\in\overrightarrow{E}} e_{ij} \le 1 + Mx_j \\ 1 \le \sum_{e_{ij}\in\overrightarrow{E}} e_{ij} + Mx_j \end{cases} \quad \forall j \in J, \tag{8}$$

and the indegree of the terminal node is 1:

$$\begin{cases} \sum_{e_{ij}\in\overrightarrow{E}} e_{ij} \le 1 + M(1-x_j) \\ 1 \le \sum_{e_{ij}\in\overrightarrow{E}} e_{ij} + M(1-x_j) \end{cases} \quad \forall j \in J. \tag{9}$$

Outdegree Constraints. Here we provide a detailed description for $C_{\text{outdegree}}(\mathcal{X},\mathcal{E})$ constraint. The outdegree for each node in T_m except for the first one is 1. This constraint can be written as follows:

$$\begin{cases} \sum_{e_{ij}\in\overrightarrow{E}} e_{ij} \le 1 + Mx_i \\ 1 \le \sum_{e_{ij}\in\overrightarrow{E}} e_{ij} + Mx_i \end{cases} \quad \forall i \in I, \tag{10}$$

and the outdegree of the first node is 1:

$$\begin{cases} \sum_{e_{ij}\in\overrightarrow{E}} e_{ij} \le 1 + M(1-x_i) \\ 1 \le \sum_{e_{ij}\in\overrightarrow{E}} e_{ij} + M(1-x_i) \end{cases} \quad \forall i \in I. \tag{11}$$

Similarly, the outdegree constraint for nodes in T_n except for the terminal node can be written as follows:

$$\begin{cases} \sum_{e_{ji}\in\overleftarrow{E}} e_{ji} \le 1 + Mx_j \\ 1 \le \sum_{e_{ji}\in\overleftarrow{E}} e_{ji} + Mx_j \end{cases} \quad \forall j \in J, \tag{12}$$

and the outdegree of the terminal node is 0, i.e.,

$$\sum_{e_{ji}\in\overleftarrow{E}} e_{ji} \le M(1-x_j) \quad \forall j \in J. \tag{13}$$

Exclusivity Constraint. Here we provide a detailed description for $C_{\text{exclusivity}}(\mathcal{X},\mathcal{E})$ constraint. There should be one first and one terminal node:

$$\begin{cases} \sum_{i\in I} x_i = 1 \\ \sum_{j\in J} x_j = 1 \end{cases} \tag{14}$$

and also for edges, i.e., e_{ij} and e_{ji} should be mutually exclusive:

$$e_{ij} + e_{ji} \le 1 \quad \forall e_{ij} \in \overrightarrow{E}, e_{ji} \in \overleftarrow{E}. \tag{15}$$

Order Constraints. Here we provide a detailed description for $C_{\text{order}}(\mathcal{X},\mathcal{E},\mathcal{O})$ constraint presented in Section 5. The order of the first node is 0, which can be written as follows:

$$o_i \le M(1-x_i) \quad \forall i \in I, \tag{16}$$

and the order of the terminal node is $2N-1$:

$$\begin{cases} o_j \le 2N-1+M(1-x_j) \\ 2N-1 \le o_j + M(1-x_j) \end{cases} \quad \forall j \in J. \tag{17}$$

Finally, the order should be monotonically increasing, i.e.,

$$\begin{cases} o_i + 1 \le o_j + M(1-e_{ij}) \\ o_j \le o_i + 1 + M(1-e_{ij}) \\ o_j + 1 \le o_i + M(1-e_{ji}) \\ o_i \le o_j + 1 + M(1-e_{ji}) \end{cases} \quad \forall e_{ij} \in \overrightarrow{E}, e_{ji} \in \overleftarrow{E}. \tag{18}$$