A Graph-based Model for Understanding Interlocking Assemblies

ZIQI WANG

Computing a feasible disassembling sequence of parts for a given 3D assembly is a fundamental research topic in geometric reasoning. Its inverse problem which is to create geometry of a 3D assembly according to predefined constraints on the parts disassembly order, attracts more and more attention in computer graphics community. This report focuses on 3D interlocking assemblies where all component parts have to be disassembled after removing a single key part. Though several computational methods for designing interlocking assemblies such as puzzle and furniture has recently been contributed, the interlocking mechanism has not yet been fully understood and the full search space of interlocking configurations has not been fully explored, restricting applicability for the design.

In this report, I propose a graph-based method for modeling the interlocking mechanism. The core idea is to represent part blocking relationships with a family of base *Directional Blocking Graphs (DBGs)*. By utilizing graph analysis tools in classic graph theory, my approach builds a connection between an interlocking assembly and the connectivity of its DBGs. Based on this connection, I propose an efficient algorithm to test interlocking with polynomial time complexity and enable the ability to explore interlocking configurations that are not possible by the state of the art. As a result, my method has potential to lead to a more efficient and flexible computational tool for designing interlocking assemblies.

ACM Reference format:
ZIQI WANG. 2018. A Graph-based Model for Understanding Interlocking Assemblies. *ACM Trans. Graph.* 37, 6, Article xxx (December 2018), 4 pages. https://doi.org/10.1145/8888888.7777777

1 INTRODUCTION

Digital modelling and fabrication, which significantly bridge the gap between the 3D virtual design and physical product, attracts more and more attention in computer graphics community. A 3D assembly is composed of multiple component parts with specific form and/or functionality can be efficiently created in a modern CAD software. The software would give immediate feedback regarding user-defined property of the assembly. By editing the part geometry, designer could improve local performance of the assembly (such as part appearance, connection strength, etc.) with the help of feedback from the CAD software. However, local modification of part geometry is less effective for global properties (like structural stability and disassembling sequence of parts). Specific design tools are demanded by users.

This report focuses on 3D interlocking assemblies where all component parts have to be disassembled after removing a single key part. A general CAD software is inefficient for designing such interlocking assemblies because there is no obvious correspondence between the assembly geometry and disassembling sequence of parts, especially for non-professional designers. Though several computational methods for designing interlocking assemblies such as puzzle and furniture has recently been contributed, the interlocking mechanism has not yet been fully understood and the full search space of interlocking configurations has not been fully explored, restricting applicability for the design.

2018. 0730-0301/2018/12-ARTxxx $15.00
https://doi.org/10.1145/8888888.7777777

Designing a interlocking assembly, which sets partial constraints on disassembling sequence, is an inverse problem of finding a feasible disassembling sequence of parts for a given assembly. The latter is a classic research topics in geometric reasoning. Thus, I choose my first reference paper [Wilson and Latombe 1994] about how to automatically generate a disassembly sequence and select a less complicate way of disassembling process. The input assembly is assumed to have three properties:

- Each part is rigid.
- Neighboring parts have planar surface contact only.
- Input assembly can be disassembled by single-part translational motions, i.e., part rotation is not required and all other parts remain fixed when removing a part.

The 3D model of an assembly from a typical CAD software are appropriate for rendering but it does not directly provide the information that is needed to easily plan assembly algorithms. Therefore, the authors propose a family of *Directional Blocking Graph* (DBGs), where each describes the potential interactions among parts for a given translation direction. It is proved only a finite subset of DBGs are pairwise different, which can be computed with a polynomial time complexity. To remove a part from an assembly, the in/out-degree of this part have to be zero in at least one DBG. Besides, for a given translation direction, any two parts have to be removed together as long as they are in the same strongly connected component(SCC) in the corresponding DBG. This powerful theory leads to the following fact.

> *All directional blocking graphs(DBGs) of an interlocking assembly have to be strongly connected except the key.*

What is strongly connected? How to check strongly connected in a graph? How to make a graph strongly connected? The second reference paper [Tarjan 1972] concludes these problems with efficient algorithms. By utilizing their graph analysis tools, test interlocking can be achieved in polynomial time complexity.

Interlocking has a long history in Chinese and Japanese furniture design but few types of artificial interlocking designs have been invented so far. Verifying interlocking property of an assembly requires testing the immobilization of all the subsets of pieces. Such tests lead to extremely expensive algorithms let alone designing an interlocking assemblies. To avoid exhaustively testing, my last reference paper [Song et al. 2012], focus on a small subclass of interlocking puzzles that are recursive in the sense that the assembly of puzzle pieces (with at least three pieces) remains an interlocking puzzle also after the (sequential) removal of pieces. Still, they are the first who develop a computational method for generating new types of interlocking geometries. As a result, their method could design new voxelized puzzles with more pieces.

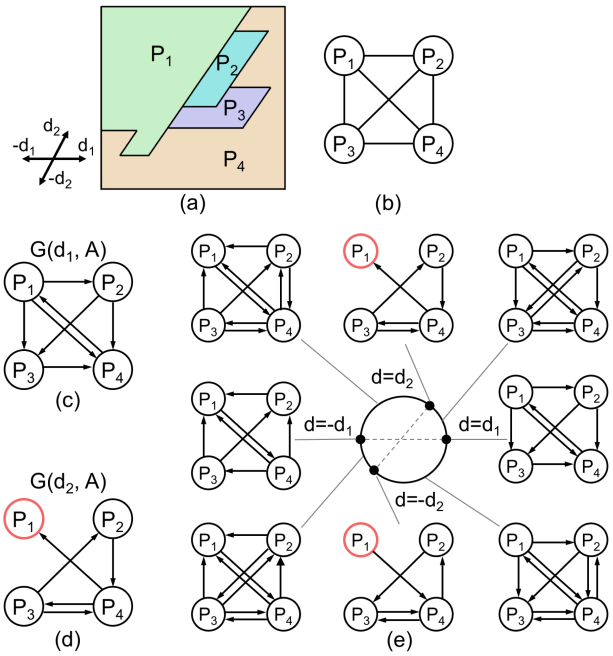


Fig. 1. Example DBGs and NDBG. (a&b) A 2D interlocking assembly and its parts-graph, where the key P_1 is movable along d_2; (c&d) Two DBGs of the assembly; and (e) NDBG of the assembly. A part with zero out-degree or in-degree in a DBG is highlighted with a red circle.

The section 2.1 discuss the DBGs in detail and are followed by the section 2.2 which presents the interlocking testing algorithm. The correctness of the recursive interlocking approach [Song et al. 2012],where maintaining interlocking property in consecutive three parts make the assembly interlocking, can be proved by graph-based representation. Furthermore, there exists non-recursive interlocking which does not cover in [Song et al. 2012]. In addition, an assembly which part-graph have more than one cut-point, cannot be interlocking. By understanding the augmentation problem from [Tarjan 1972], the section 2.4 gives one possibility to re-design the input geometry, such that its part-graph of is biconnected.

2 MODEL INTERLOCKING ASSEMBLIES

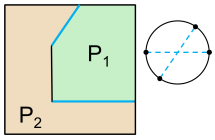
2.1 Graph Model

Consider an assembly **A**, made of n parts P_1, ..., P_n. We make the following assumptions: 1) each part P_i is rigid; 2) neighboring parts have planar surface contact only; and 3) **A** can be disassembled by single-part translational motions, i.e., part rotation is not required and all other parts remain fixed when removing a part.

Directional Blocking Graph (DBG). We denote as $G(d, A)$ the *directional blocking graph* of assembly **A** for translation along direction d. This directed graph has nodes representing the parts of **A** and directed edges $e_{i\to j}$ from P_i to P_j if and only if P_j prevents any translational motion of P_i along d. In other words, $e_{i\to j}$ can be read as "P_i is blocked by P_j" in direction d. See Figure 1(c&d) for two examples.

If $G(d, A)$ is *strongly connected*, i.e. if every node can be reached from every other node, no part or part group is movable along d; see Figure 1(c). A part group **S** of **A** is locally free to translate in direction d ($-d$), if and only if the out-degree (in-degree) of **S** in $G(d, A)$ is zero; see P_1 in Figure 1(d).

Non-directional Blocking Graph (NDBG). We represent the set of all translation directions in 2D by the unit circle denoted as C. For every pair of parts in contact in **A**, we draw the diameter that is parallel with the contact line. The drawn diameters partition C into an arrangement of regions, for which the corresponding DBG $G(d, A)$ remains constant when d varies over a region. For any pair of parts in contact (e.g., P_1 and P_2 in the inset), if there are more than two contact lines, we only retain the two diameters of C (e.g., two contact lines in blue) which bound the cone of directions in which one part is free to translate relative to the other. The arrangement of points and intervals on C and the associated DBGs form the *non-directional blocking graph* of **A**; see Figure 1(e). The NDBG for a 3D assembly can be built similarly by constructing DBGs for each regular region of a unit sphere that represents all possible translation direction in 3D; please refer to [Wilson and Latombe 1994] for more details.



Base Directional Blocking Graphs. An NDBG represents the parts blocking relations with redundancy in two aspects. First, the DBG corresponding to an arc in C can be derived by performing union operations on the DBGs associated with the two end points of the arc; see again Figure 1(e). Second, we can obtain $G(-d, A)$ from $G(d, A)$ easily by reversing the direction of every edge in $G(d, A)$ due to the reciprocity of blocking relations among the parts.

Therefore, it is sufficient to model the blocking relations in **A** by using only a set of *base DBGs* denoted as $\{G(d, A)\}$, which we select as the DBGs corresponding to the end points in a half circle of C. For example, two DBGs in Figure 1(c&d) form $\{G(d, A)\}$. We call the set of directions corresponding to the base DBGs as *base directions*, denoted as $\{d\}$. The number of base DBGs (as well as base directions) is $O(n^2)$ since every pair of parts provides at most two diameters in C.

2.2 Testing Interlocking

In an interlocking assembly, every part and every part group are immobilized for all possible translation directions, except a single key. To test immobilization of a part group **S**, we need to compute blocking relations between **S** and **A** − **S**: the part group **S** is immobilized if **S** is blocked by **A** − **S** in all translation directions. Explicitly testing interlocking by checking immobilization of every part and every part group has exponential time complexity. However, treating each part group **S** independently ignores significant redundancies in the blocking relations across the parts. We exploit these redundancies and propose a more efficient approach to test global interlocking. The key idea is to utilize the blocking relations encoded in the set of base DBGs to implicitly test immobilization of every part and every part group along a finite number of translation directions, i.e., the base directions $\{d\}$. In detail, an assembly with at least three parts is interlocking, if all base DGBs are either

(1) strongly connected, or

(2) have only two strongly connected components one of which has a single part that is identical across all DGBs.

Here the strongly connected component with a single part is the key of the assembly. The direction d associated with each DBG with two strongly connected components is the key part's (reversed) movable direction according to the in-edge (out-edge) of the key in the DBG. For example, the assembly in Figure 1(a) is interlocking and P_1 is the key since its two base DBGs in Figure 1(c&d) satisfy the above requirement.

2.3 Recursive Interlocking

Generating a n pieces recursive interlocking has two steps:

- Select the key part P_1 from the general voxelized shape S and denote the remaing part as R_1
- Iteratively extract pieces, one by one, forming a sequence of extracted pieces P_1, P_2, ..., P_{n-1}, with R_{n-1} , the remaining part of S, as the last piece:

$$S \rightarrow [P_1, R_1] \rightarrow [P_1, P_2, R_2] \rightarrow \cdots \rightarrow [P_1, \cdots, P_{n-1}, R_{n-1}].$$

Recursive interlocking property requires that any consecutive three parts have to be interlocking and the first part is the local key.

$$[\mathbf{P_1}, P_2, P_3], \cdots, [\mathbf{P_i}, P_{i+1}, P_{i+2}], \cdots, [\mathbf{P_{n-2}}, P_{n-1}, R_{n-1}]$$

The local key is marked as bold.

In [Song et al. 2012], they prove that **recursive interlocking is interlocking**. They prove the correctness of theory by discussing in different cases, which can be simpilified by the graph-based tool. For simplification, denote R_{n-1} as P_n

Lemma 2.1. *if* $[\mathbf{P_i}, P_{i+1}, P_{i+2}]$ *is a local interlocking group, and* P_i *is the local key, then* P_{i+1}, P_{i+2} *are in the same strongly connected component for any DBG.*

Proof: Interlocking property means P_{i+1} and P_{i+2} only can be moved together in any direction when the key part P_i is fixed. Therefore P_{i+1} and P_{i+2} are in the same strongly connect component for any DBG.

Theorem 2.2. *Recursive interlocking is interlocking*

Proof: According to the lemma, we have P_i and P_{i+1} are in the same strongly connected component for any DBG. Then $P_2, \cdots, P_n$ are in the same strong connected component because the strongly connected property is transitive. Then the whole assembly is interlocking by the previous definition.

Corollary 2.3. *Consecutive* $K(K \geq 3)$ *pieces interlocking is interlocking*

An non-recursive interlocking example is given in Fig.2. Given a general voxelized shape S, the solution number of recursive interlocking accounts for a small proportion of general interlocking and

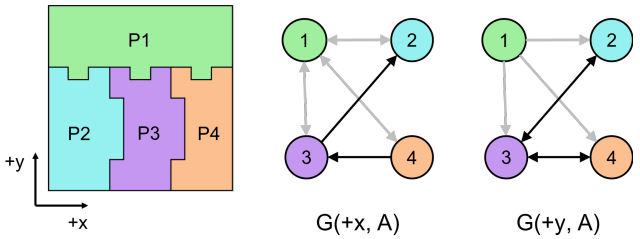


Fig. 2. A non-recursive interlocking 2D puzzle A. The dark black edges in $G(+x, A)$ do not form a cycle, which means $[P_2, P_3, P_4]$ is not interlocking. Therefore, the assembly A is not recursive interlocking.

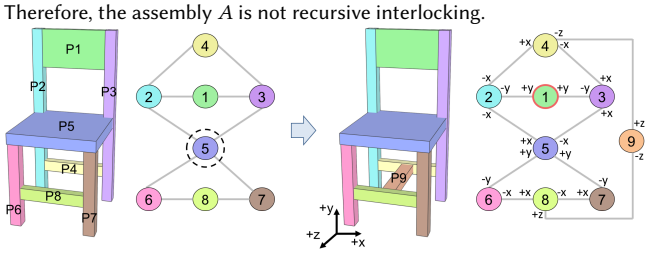


Fig. 3. Left: a Chair and its parts-graph, where a cut point (i.e., P_5) exists. Right: after adding a new part (i.e., P_9), our approach can generate an interlocking joint configuration, where the axial removal direction allowed by each joint is shown in the corresponding edge in the parts-graph.

the proportion will dramatically drop when the number of parts increase. It can be explained in two aspects.

Cycle in DBGs: The recursive interlocking usually has cycles with less than 4 vertices for any DBGs. The local interlocking group $[P_i, P_{i+1}, P_{i+2}]$ constrain the cycle size.

Disassembling sequence: The recursive interlocking only have two (the order of last two pieces can be swapped) possible disassembling sequence. The general interlocking in Fig.2 has four disassembling sequences:

$$(1, 2, 3, 4), (1, 2, 4, 3), (1, 4, 2, 3), (1, 4, 3, 2)$$

2.4 Failure Cases and Modification

Lastly, inspired by our DBG-based representation, we find that a parts-graph with a cut point cannot be interlocking, no matter what kinds of joints are planned; see Figure 3(left) for an example. This observation allows us to modify a given input to make it possible to be interlocking by adding a minimal number of new parts in the parts-graph in oder to remove the cut point; see Figure 3(right) for an example.

3 CONCLUSION

This report takes a specific kind of assembly, the interlocking assembly, as an example. The interlocking property is formulated as a graph-based representation. Testing interlocking then is equivalent to analysing the strongly connected component of each DBG. The next step would be generating general interlocking structure with the help of DBGs.

However, it is better to consider more general topic beyond interlocking. The final goal of my PhD research is to build a general framework for assembly design. The composition of an general assembly includes six properties (see Fig. 4). Among them, fabrication

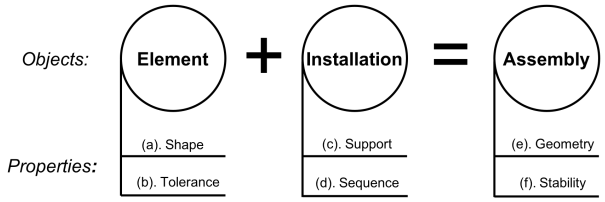


Fig. 4. The general framework of designing an assembly structure. The **element** is the component part of the assembly and the **installation** is the assembling process. (a) The feasible shape of element varies for different fabrication tools. For instance, 2D laser cutter only cuts 2D pattern on a flat plate but 3D printer could fabricate more freeform model. (b) Tolerance is the displacement between the fabrication and virtual model. Machine cannot have infinite precision and the tolerance is inevitable during manufacture. (c) Temporary support such as scaffold stabilise the structure during assembling process. Less support could reduce the waste, which may takes up to 1/3 of the whole cost. (d) The assembling sequences includes the order of parts and the trajectory of placing the part in right positions. (e) User defined target geometry (f) Structural stability cares whether the whole structure will collapse under external load.

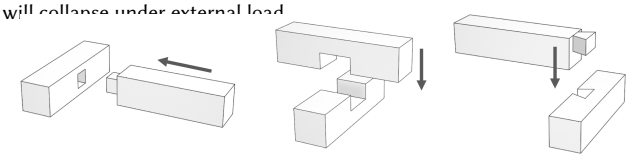


Fig. 5. Example woodworking joints. From left to right: mortise-and-tenon, halved joint, and dovetail joint, where the black arrow shows the single part movable direction allowed by the joint.

tolerance and structural stability are hard constraints and the reset have to optimize according to the demand. Besides, the framework also needs to maintain a sufficient design freedom for users. Considering all properties at same time is extremely challenging and most of the published work try to take a few properties into account.

Properties (a) + (d) + (e): Most of the interlocking paper [Song et al. 2012],[Xin et al. 2011], [Song et al. 2016], [Fu et al. 2015] are belong to this category. The constraints of installation sequence, where the key part is always the last pieces of the installation, is guaranteed by the element's shape. One of the common problem of these works is that without computing structural stability, the interlocking design is unsafe to be used in practice. Interlocking furniture, for instance, have three types of connections (see Fig.5). In practice, the structural strength of dovetail joint is weaker than the tenon joint. How to choose suitable joints according to the force flow while maintaining the interlocking property could be a promising research topic.

Properties (a) + (b) + (e): Tolerance are usually required to accommodate the engineering error during manufacture, this resulted empty space allows each part of the assembly to have a small displacement. The accumulation of these displacements could be amplified greatly in a not well design and severely undermine the structure's stability (see Fig.6). A survey work of tolerance analysis is presented in [Chen et al. 2014]. The tolerance analysis in mechanical engineering often deals with simple geometry and clear accumulation mechanism. However, our objects usually have complicated surface with ambiguous tolerance accumulation mechanism. One possible approach is to find the connection between the tolerance accumulation and cycles in DBGs.

REFERENCES

Hua Chen, Sun Jin, Zhimin Li, and Xinmin Lai. 2014. A comprehensive study of three dimensional tolerance analysis methods. *Computer-Aided Design* 53 (2014), 1–13.

Chi-Wing Fu, Peng Song, Xiaoqi Yan, Lee Wei Yang, Pradeep Kumar Jayaraman, and Daniel Cohen-Or. 2015. Computational Interlocking Furniture Assembly. *ACM Trans. on Graph. (SIGGRAPH)* 34, 4 (2015). Article No. 91.

Peng Song, Bailin Deng, Ziqi Wang, Zhichao Dong, Wei Li, Chi-Wing Fu, and Ligang Liu. 2016. CofiFab: Coarse-to-Fine Fabrication of Large 3D Objects. *ACM Trans. on Graph. (SIGGRAPH)* 35, 4 (2016). Article No. 45.

Peng Song, Chi-Wing Fu, and Daniel Cohen-Or. 2012. Recursive Interlocking Puzzles. *ACM Trans. on Graph. (SIGGRAPH Asia)* 31, 6 (2012). Article No. 128.

Robert Tarjan. 1972. Depth-first search and linear graph algorithms. *SIAM J. Comput.* 1, 2 (1972), 146–160.

Randall H. Wilson and Jean-Claude Latombe. 1994. Geometric Reasoning About Mechanical Assembly. *Artificial Intelligence* 71, 2 (1994), 371–396.

Shi-Qing Xin, Chi-Fu Lai, Chi-Wing Fu, Tien-Tsin Wong, Ying He, and Daniel Cohen-Or. 2011. Making Burr Puzzles from 3D Models. *ACM Trans. on Graph. (SIGGRAPH)* 30, 4 (2011). Article No. 97.

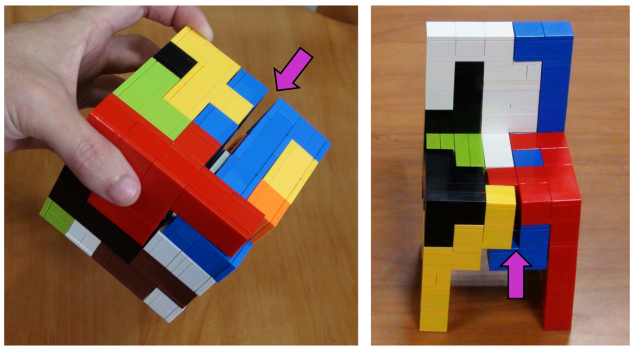


Fig. 6. Two interlocking puzzles designed by implementing [Song et al. 2012]: a 13-piece cube (left) and a 6-piece chair (right). Although being interlocking, both puzzles have the problematic gap issue (see arrows above) that loosens their structures. Note: LEGO bricks in fact have a tolerance of 0.20mm.