A Multi-Phase Search Approach to the LEGO Construction Problem

Ben Stephenson
Department of Computer Science
University of Calgary
2500 University Drive NW
Calgary, Alberta T2N1N4

Abstract

The task of determining which LEGO bricks to use to construct a volume is known as the LEGO Construction Problem. This is a challenging problem because even small volumes can be constructed in a tremendously large number of ways. As a result, an exhaustive search is impractical, and more nuanced search strategies must be employed to find a good, though not necessarily optimal, solution.

This paper describes a multi-phase search approach to the LEGO Construction Problem. Our first search phase uses heuristics to identify a moderate number of candidates for each layer in the model. This is followed by two different search strategies which identify alternative brick arrangements that reduce the number of connected components, undesirable edges, and bricks in the model. A final highly localized search is applied to bricks at the boundaries between the model's connected components if the previous search processes fail to reduce the model to a single connected component. Applying this four-phase search strategy to a diverse selection of models has demonstrated that it normally finds a result that consists of a single connected component when such a solution exists, and that the models are structurally sound when built.

Introduction

LEGO bricks are a popular children's toy. The colorful plastic blocks, which snap together, can be assembled in a seemingly endless number of ways, allowing builders to construct a wide variety of different objects. The bricks are easy to separate, allowing them to be reused in different models over time.

While LEGO bricks are primarily used as a children's toy, some adults use them to construct models as a hobby or as a profession. These models can be huge, consisting of thousands, tens of thousands, or even hundreds of thousand of bricks. An example of such a work, constructed by the author of this paper, is shown in Figure 1. It is 29 inches long, 13 inches wide, 20 inches tall and weighs 22 pounds. Approximately 6,000 bricks were used during its construction. Such models take a significant amount of time and skill to construct. Estimating which bricks are needed to build a large model is difficult, and a significant amount of trial and error may be necessary during construction in order to reach a result that consists of a single connected component[1].

The task of determining which parts to use to construct a volume using LEGO bricks is known as the LEGO Construction Problem (van Zijl and Smal 2008). This is a challenging problem because the number of different ways that an object can be constructed from LEGO bricks is tremendously large, even for relatively small models. As a result, an exhaustive search is not practical. Instead, techniques must be used that focus search efforts on portions of the model where they will be most beneficial, and each search must be constrained to a small subset of the entire model.

This paper describes our effort to solve the LEGO Construction Problem using a multi-phase search approach. We describe four different search processes and the benefit that each provides toward reaching a good solution. Our experimental results show that this approach generates good results for a wide variety of models.

[1] A connected component is a non-empty collection of LEGO bricks that are snapped together. The adjective, *connected*, is used to describe a model (or other collection of bricks) that consists of a single connected component.

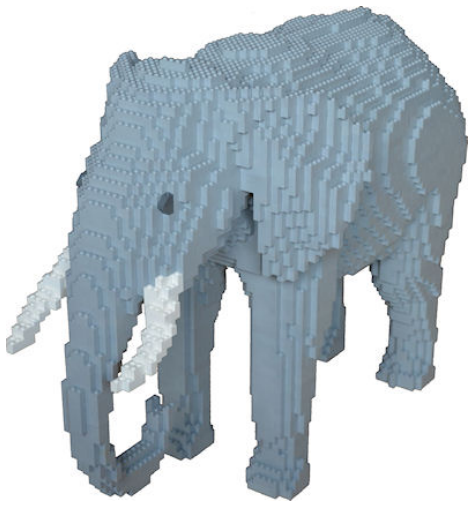

Figure 1: An Elephant Constructed Using Approximately 6000 LEGO Bricks

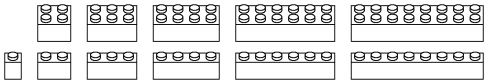

Figure 2: Shapes of Bricks used to Construct a Model

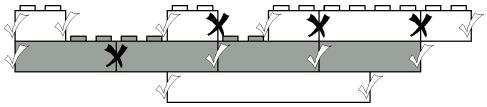

Figure 3: Desirable and Undesirable Edges

Related Work

A great deal of research has been conducted on techniques for realizing physical representations of 3D models. The materials used to construct the physical representation vary, and have included paper crafts (Mitani and Suzuki 2004), plush toys (Mori and Igarashi 2007), interlocking puzzles (Song, Fu, and Cohen-Or 2012), wire mesh sculptures (Garg et al. 2014) and inflatable objects (Skouras et al. 2014), in addition to the materials commonly used by fabrication techniques like 3D printing and CNC milling.

We choose to realize a physical representation of a 3D model using LEGO bricks. This choice is not unique. LEGO Bricks have previously been combined with 3D printing to speed up prototyping (Mueller et al. 2014), and a variety of techniques have previously been applied to the LEGO Construction Problem that we address in this paper.

Early work in this area made several recommendations about how to construct a connected model, including that larger bricks should be preferred and bricks in adjacent layers should have differing directionality (Gower, Heydtmann, and Petersen 1998). They went on to recommend a simulated annealing approach, but they do not report any experimental results to demonstrate its effectiveness. Subsequent work has reported success with simulated annealing for a collection of 3 models (Kozaki, Tedenuma, and Maekawa 2016).

The result of using a genetic algorithm to solve the LEGO Construction Problem has been reported on two occasions (Petrovic 2001; Lee et al. 2015). On the first occasion, the algorithm was tested using random data, making it difficult to determine whether or not the technique is effective for models that one would be likely to construct. In the second case, the algorithm was tested on 23 models. Those experiments found solutions with the minimum number of connected components for at least one run (out of 100 attempts) for 19 of the 23 models.

The application of cellular automata to the LEGO Construction Problem has also been considered (van Zijl and Smal 2008). That work reports success for 3 models that differ significantly in their size. A similar split/merge approach was subsequently reported that also constructs and analyzes a connectivity graph for the model (Testuz, Schwartzburg, and Pauly 2013). They report results for a dozen models, each at four different scales, and achieve a single connected component for 40 out of those 48 test cases.

Finally, the LEGO Construction Problem has also been approached from a force-stability point of view (Luo et al. 2015). This work considers the forces between the bricks, using more than just simple connectedness to predict whether or not a model will be able to support its own weight when assembled. They report that their technique was successful for the 6 models that they tested.

We have extended the work in this area by implementing a novel multi-phase search approach to the LEGO construction problem. Our approach has been applied to 30 distinct models, which is a larger collection than most other studies in this area. Experimental results show that we achieve a result containing the minimum possible number of connected components with greater frequency than either the genetic algorithm described by Lee et al. or the connectivity graph approach used by Testuz, Schwartzburg, and Pauly, which are the only other works that report results for 10 or more models.

Search Algorithm

We have used a multi-phase search approach to tackle the LEGO Construction Problem. The input to our approach is a collection of voxels, or small units of volume, that must be filled with Lego bricks. A collection of voxels can be identified in several different ways, including using the ray tracing approach described at the beginning of our Experimental Results section, or it can be hand-generated by an artist. Each voxel in the input set is the same size as a 1x1 LEGO brick. Our algorithm does not require that any particular technique be used to generate the input voxel collection.

We construct the model from bottom to top, paralleling the process typically taken by human builders (Gower, Heydtmann, and Petersen 1998) and additive manufacturing processes (Kozaki, Tedenuma, and Maekawa 2016). Once a layer is complete it is generally left unmodified, again, following the process typically used by a person. During the search, the algorithm is free to use any number of bricks of each of the sizes shown in Figure 2. The goal of our search algorithm is to minimize, in order of importance:

1. The number of connected components in the model
2. The number of undesirable edges in the model
3. The number of bricks in the model

While the first and third goals are self-explanatory, some elaboration is needed on the second goal. A non-empty arrangement of LEGO bricks will always have edges. An edge is said to be *external* if it occurs at the boundary of the model, meaning that there is a LEGO brick on one side of the edge and an open space on the other side of the edge. An edge is said to be *internal* if it occurs between two LEGO bricks.

Some edges are desirable while others are undesirable. Each brick in the model often has a mixture of both types of edges. Both types of edges may occur along the same face of a brick because the face may overlap edges in the previous layer in some areas, but not others. Several desirable and undesirable edges are presented visually in Figure 3. The desirable edges are marked with a white check mark while the

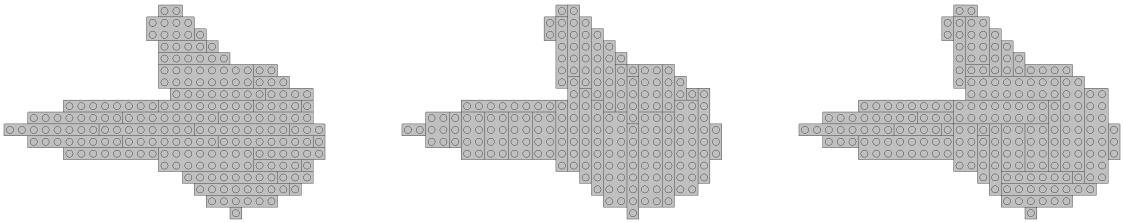

Figure 4: Layers with Different Orientations

undesirable edges are marked with a black X. The following edges are undesirable because they do not promote strength or connectivity within the model:

- An internal edge that is over an open space in the previous layer, such as the left-most undesirable edge in Figure 3.
- An internal edge that is over an external edge in the previous layer, such as the right-most undesirable edge in Figure 3.
- Any edge that is over an internal edge in the previous layer, such as the middle two undesirable edges in Figure 3.

The remaining edges are desirable because they are solidly supported by bricks in the previous layer.

We use up to four different search algorithms to construct each layer in the model. The first search generates 36 candidate layers using heuristics, and selects the best candidate as the starting configuration for the subsequent search processes. Then two additional search processes are performed. The first is focused on removing undesirable edges and is only applied to bricks adjacent to them. The second search is applied to the entire layer, allowing it to perform further reductions in the number of components, undesirable edges and bricks in the model. Each of these search strategies is described in detail in the following subsections. A fourth and final search strategy is employed if the result of the prior strategies is disconnected. The final search process is described in a later section.

Initial Heuristics

We begin the process of locating a suitable arrangement of bricks for each layer in the model by applying a collection of heuristics. These heuristics were developed from previously published recommendations (Gower, Heydtmann, and Petersen 1998) and our own experience building large models. All of the heuristics begin from a state where every voxel in the level is represented by a 1x1 brick. New states are reached by merging adjacent 1x1 bricks (voxels) to form larger bricks. Different arrangements of bricks are generated by merging the voxels in different orders and with different restrictions on which merges can be performed.

After each arrangement is generated, it is checked to see if it contains any bricks that do not overlap with any voxels in either the layer above or the layer below the brick. We use the term *unconnected* to describe such bricks. Arrangements containing unconnected bricks are normally discarded because a result containing an unconnected brick will have multiple connected components, which is contrary to our first objective. In the rare situations when all of the arrangements generated by our heuristics contain unconnected bricks, the arrangement with the fewest such bricks is retained. Then the subsequent search phases are given the opportunity to try and replace the unconnected bricks and generate a connected model.

The arrangements that we generate with our heuristics can be broadly grouped into three categories: arrangements where most of the bricks have left-right directionality, arrangements where most of the bricks have front-back directionality, and arrangements where the bricks have a mixture of both directionalities. Figure 4 shows examples of each of these directionalities for a layer in the Stanford Bunny model. The left arrangement has left-right directionality; the middle arrangement has front-back directionality; the right arrangement contains a mixture of both directionalities.

Arrangements with left-right directionality are created by repeatedly merging adjacent bricks with a restriction that the newly created brick must be at least as large in the left-right direction as it is in the front-back direction. Four arrangements are created by starting the process from different points in the layer. After every possible left-right merge has been performed, the directionality restriction is removed, and further merges are performed where possible.

When this process is performed, it is common for 1x1 bricks to appear at the ends of rows. For example, if a row contains 7 voxels, then the first 6 voxels will be merged into a 1x6 brick, and the final voxel will remain as a 1x1 brick. We generate four additional arrangements by following a variation of the previous process that ends odd-length rows with a 1x3 brick instead of a 1x1 brick. Under this revised process, a row containing 7 voxels would be represented as a 1x4 brick followed by a 1x3 brick. In some circumstances, this is a preferable arrangement because the 1x3 brick may be possible to connect to the remainder of the model when the 1x1 brick would be unconnected.

Our final strategy for generating arrangements with left-right directionality attempts to locate smaller bricks toward the middle of the model by walking the perimeter of the collection of voxels and repeatedly choosing the largest possible brick with left-right orientation. Four different arrangements are generated by starting from different points along the perimeter of the layer. Voxels that still need to be filled after the perimeter walk completes are merged into the largest possible bricks with left-right orientation using a greedy algorithm.

A total of 12 arrangements are generated that have left-right directionality. The same number of arrangements are also constructed with front-back directionality by following the same general process with a front-back restriction on the directionality of the merged bricks.

Twelve mixed directionality arrangements are created using a greedy approach. Four arrangements are generated by repeatedly identifying the largest collection of voxels that can be merged into a larger brick, starting from each corner of the model. An additional 8 mixed directionality candidates are created by walking around the perimeter of the voxels in the layer. The walks begin from four different locations, with a clockwise walk and a counter-clockwise walk beginning from each location. Any voxels that remain in the middle of the layer after the walk completes are also merged in a greedy manner.

Each of the 36 arrangements described previously is compared with the previous layers to determine the number of connected components and the number of undesirable edges that would occur if the arrangement was selected. The arrangement that results in the smallest number of connected components is retained as the initial state for the next search phase, with a tie being resolved by selecting the arrangement with the smallest number of undesirable edges.

Edge Reduction

The Edge Reduction phase is a focused search effort that is applied to bricks adjacent to undesirable edges. Each stud around the perimeter of the brick is checked and marked as either being a desirable or undesirable edge. Then each undesirable edge segment is processed to see if it can be removed. Edges that occur over an empty space, such as the left-most undesirable edge in Figure 3 are tackled first. Such edges are the highest priority because they generally divide the current state into multiple connected components. These are followed by undesirable edges where one edge is internal and one edge is external, such as the right-most undesirable edge in Figure 3. Cases where both edges are internal are tackled last because such edges tend to be toward the interior of the model, and as such, are less likely to cause the model to be disconnected. Each edge is tackled in the following manner:

1. The six voxels closest to the edge are identified. The bricks that overlap with those six voxels are the local region for the edge. All other bricks in the layer are the non-local region.
2. A search is performed to identify legal combinations of bricks that have the same shape as the local region. We refer to each combination as a local candidate, and each local candidate represents a potential transition to a new state for the current layer. Our search for local candidates is limited to combinations that are 4 bricks or less. This limit is imposed for two reasons. First, it reduces the search space which reduces the running time. Second, local candidates with larger numbers of bricks have more edges, making them less likely to be successful at reducing the number of undesirable edges in the current layer of the model. Furthermore, using more bricks in the current layer will make it more difficult to find a good solution for the next layer.
3. Each local candidate is joined with the non-local region to generate a new version of the layer. During the joining process bricks in the candidate local region are merged with adjacent bricks in the non-local region, ensuring that the layer does not contain any pairs of bricks that can be reduced to a single brick.
4. The version of the layer that has the fewest connected components, followed by the smallest number of undesirable edges when overlaid with the previous layer is used to replace the current layer in a hill climbing fashion. If none of the local candidates is an improvement then the original layer is retained.

The edge reduction process continues until no further edges can be removed using this algorithm.

Adjacent Brick Recombination

After the edge reduction process completes, an additional search is applied to all pairs of adjacent bricks in the layer. This search is able to provide additional improvements to the layer because it is not localized to the bricks adjacent to undesirable edges. In some cases, it is also able to remove additional undesirable edges because of the difference in the approach that is taken by this search phase compared to the edge reduction phase. The following steps are taken to process each pair of adjacent bricks:

1. The pair of bricks under consideration are defined to be the local region. All of the remaining bricks in the layer are the non-local region.
2. A search is performed to identify all combinations of up to 4 bricks that have the same shape as the local region, resulting in a collection of local candidates that represent possible transitions to new states.
3. Each local candidate is joined with the non-local region to form a new version of the layer. Adjacent bricks are merged where possible, ensuring that the layer does not contain any pairs of bricks that can be reduced to a single brick.
4. The version of the layer that has the fewest connected components, followed by the smallest number of undesirable edges, followed by the smallest number of bricks, when overlaid on the previous layer is used to replace the current layer in a hill climbing fashion. If none of the local candidates results in an improved version of the layer then the original layer is retained.

While the process followed by this search is similar to the edge reduction process, it achieves additional gains for many layers. The additional gains occur because it is applied throughout the layer, not just to bricks adjacent to undesirable edges. In addition, each local region is generally smaller than the local regions used during the edge reduction search. As a result, this search considers some combinations of bricks that were not considered previously. The search continues until a fixed point is reached that does not allow any further reductions in the number of connected components, undesirable edges, or bricks using this technique.

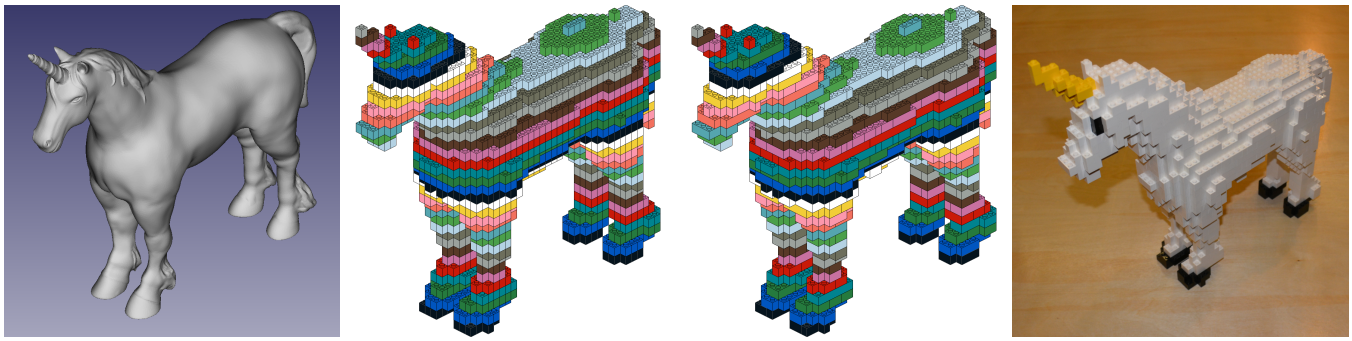

Figure 5: Build Process from Start to Finish

Final Adjustments

Once the previous phases have been completed, the total number of connected components in the model, rather than just the connected components between adjacent layers, is determined. If the model has more than one connected component then a final search process is launched that attempts to connect the components. Otherwise the final adjustment phase is omitted and the model that resulted from adjacent brick recombination is presented as the result.

When needed, the final adjustment phase begins by identifying all bricks in the model that are adjacent to a brick in a different connected component. Each of these bricks is processed in the following manner:

1. A local region is constructed which consists of the brick under consideration and all of the bricks adjacent to it that are in a different connected component. This generally produces larger local regions than those used by the edge reduction and adjacent part recombination phases.
2. Combinations of bricks that can be used to construct the same shape as the local region are considered as candidates. The number of bricks used in each combination is bound by the lesser of 7 and the number of bricks in the local region, plus two.
3. If a local candidate reduces the number of connected components in the model then it is used instead of the original local region. Ties are resolved by selecting the candidate that contains the smallest number of bricks.

A deeper search is used for the final adjustment phase because this search process is generally performed on a very small number of bricks. As a result, the running time is less of a concern, and the introduction of additional bricks will not have as much negative impact on the rest of the model because the remaining layers in the model have already been computed.

Like the other search processes, this one is applied until a fixed point is reached. The fixed point is normally reached after very few iterations because the number of disconnected components is small, and correcting one disconnected component rarely has any impact on the other components.

The result of applying the final adjustments described in this section is presented to the builder. If the model was not reduced to a single component then the output includes a version of the model with the disconnected components highlighted. This allows the builder to easily identify the disconnected regions and address them. Figure 5 shows the result of the entire model building process, going from a 3D model to its voxel representation, followed by a version that uses the bricks identified by our multi-phase search process and the version constructed using physical LEGO bricks.

Experimental Results

We have applied our multi-phase search approach to the LEGO Construction Problem to 30 distinct models. One of these models is the widely used Stanford Bunny, available from The Stanford 3D Scanning Repository[2]. The remaining 29 models are from the Yahoo! JAPAN collection at Thingiverse.com[3]. In all cases, the perimeter of the 3D model was converted to 1x1 brick-sized voxels using axis aligned ray casts through the x-y, y-z and x-z planes. Each intersection between a ray and the surface of the 3D model was rounded to the center of the closest voxel, and then a 1x1 brick was placed at that location. Once the voxelized surface was constructed, the model was filled and hollowed, resulting in a shell with a minimum thickness of 3 voxels.

After the collection of voxels was constructed our complete search process was applied to each of the models. Information about the models, as well as the result of applying our search process to them, can be found in Table 1. The input sets that we used vary in size by an order of magnitude, occupying between 1,764 and 20,293 voxels. The shapes of the models also vary significantly. Some models, like Swings and Scorpion, have numerous thin sections while other models, like Clam and Arc de Triomphe, lack them entirely. Some models, such as Ray, contain a small number of large layers, while other models, such as Eiffel tower, have many small layers. Many of the models contain a combination of both. While we don't claim that this collection of models is representative of every model that one might like to construct using LEGO bricks, it is a collection that is sufficiently diverse to suggest that our results are applicable to a wide variety of models.

The Edge Reduction columns show the changes made to the model as a result of completing this search process.

[2] http://graphics.stanford.edu/data/3Dscanrep/

[3] https://www.thingiverse.com/YahooJAPAN/designs

Input Model		Edge Reduction			Adjacency			Final Adjustment			Result		
Model Name	Voxels	CC	Bad Edges	Bks	CC	Bad Edges	Bks	CC	Bad Edges	Bks	CC	Bad Edges	Bks
Arc de Triomphe	20,293	-139	-1,985	360	0	-54	-309	-2	18	2	5^*	2,668	2,648
Butterfly	5,213	-104	-595	-64	-2	-7	-43	-10	15	3	5^+	2,215	831
Clam	9,214	-175	-1,429	108	-3	-53	-72	-	-	-	1	4,070	1,588
Deer	4,204	-84	-601	72	0	-8	-30	-4	8	0	6^*	953	896
Dragonfly	6,292	-137	-708	140	-2	-15	-48	-4	0	2	1	2,673	1,081
Eiffel Tower	3,245	-64	-236	44	0	-5	-25	0	0	0	11^*	676	662
F1 Race car	6,231	-73	-997	93	0	-25	-75	-	-	-	1	1,716	871
Flamingo	9,695	-188	-1,363	179	-1	-30	-83	-2	8	2	1	2,373	1,969
Giraffe	3,654	-48	-521	64	0	-4	-20	-2	8	2	1	668	839
Mammoth	5,884	-128	-924	135	-2	-10	-43	-1	3	1	3^*	1,299	1,277
Orangutan	3,673	-69	-479	78	-1	0	-32	-	-	-	1	940	806
Peacock	4,470	-161	-522	67	-2	2	-67	0	0	0	8^*	923	1,060
Pelican	3,604	-66	-479	85	-2	-9	-21	-	-	-	1	1,008	793
Piranha	5,521	-69	-799	128	-1	-16	-100	-3	8	2	3^*	698	890
Poodle	6,694	-112	-987	131	-1	-11	-50	-1	1	0	1	1,535	1,403
Ray	7,572	-40	-765	37	-1	-65	-49	-3	15	3	1	2,500	1,036
Scorpion	5,056	-180	-683	96	-3	-12	-40	-5	7	2	11^+	1,813	1,163
Seahorse	4,034	-79	-488	99	-3	-2	-47	-2	2	0	1	757	842
Skull	5,706	-116	-893	64	-3	-10	-45	-	-	-	1	1,465	1,140
Sphinx	18,871	-301	-2,842	389	-5	-28	-215	-1	2	0	1	4,362	3,386
Spiral Shell	4,542	-94	-859	57	0	-17	-49	-	-	-	1	1,341	934
Squirrel	5,809	-115	-844	114	-4	-7	-50	-1	5	1	1	1,514	1,105
Stanford Bunny	11,428	-171	-1,721	154	-3	-41	-126	-1	4	1	1	2,878	2,133
Starfish	3,801	-41	-641	30	0	-6	-42	-	-	-	1	876	644
Sun Fish	4,387	-36	-725	109	-1	-11	-54	-	-	-	1	709	827
Swings	1,764	-10	-12	-2	0	0	0	-2	-3	0	3^*	567	594
Torii	5,500	-23	-472	137	0	-7	-74	-	-	-	1	446	896
Triceratops	9,764	-230	-1,814	170	-1	-16	-89	-1	6	1	1	2,578	1,982
Tuna	7,564	-115	-1,228	121	-2	-15	-89	-1	4	1	4^+	1,909	1,288
Unicorn	5,152	-79	-756	103	0	-11	-63	-1	4	0	2^*	1,011	1,053

Table 1: Model Characteristics and Experimental Results

Specifically, they show the change in the number of connected components (CC), undesirable edges (Bad Edges), and bricks (Bks) in the model. A negative number is used to denote a reduction, which is a desirable result for all of these measures. A positive number denotes an increase in the value.

Edge reduction removes a large number of undesirable edges from most models. In the best case, which occurs for the Sun Fish model, more than half of the undesirable edges that were present in the heuristically generated layers that were used as the starting point for subsequent search phases were removed. This is particularly noteworthy because the heuristically generated layers were selected, in part, to minimize the number of undesirable edges.

On average, performing this search reduced the number of undesirable edges by over 35 percent. The worst performance was observed for the Swings model. It performed particularly poorly, with a reduction of only 2.1 percent, because much of the model consists of thin columns for the frame and chains that hold the swings. These thin sections have a much more limited search space than the larger areas present in the other models. This restricted the number of alternatives that could be considered to reduce the number of undesirable edges.

The reductions in the number of connected components and undesirable edges achieved by the Edge Reduction search come at the expense of increasing the number of bricks in the model. This is undesirable because it typically increases both the cost to construct the model and the time needed to do so. On average, the number of bad edges removed from a model was 912 while the number of bricks increased by 110. As a result, in the average case, using one additional piece to construct the model removed more than 8 undesirable edges. In our opinion, the resulting increase in the strength of the finished model fully justifies the increase in the number of bricks, particularly when many of them are successfully removed by the next search process.

The Recombination columns show the impact the Adjacent Part Recombination phase had on the model. These results are notably different from the results achieved by the Edge Reduction phase. In particular, while the Edge Reduction phase resulted in large reductions in the number of connected components and the number of undesirable edges while increasing the number of bricks needed to construct

the model, the Adjacent Part Recombination search achieves modest decreases in the number of connected components and undesirable edges while achieving substantial reductions in the total number of bricks used for most models. For example, Adjacent Part Recombination reduced the number of bricks in the model by more than 10 percent for both the Arc de Triomphe and Piranha models, again while simultaneously realizing modest improvements in the number of connected components and undesirable edges. On average, the reduction in the number of bricks was just less than 5 percent. As was the case with Edge Reduction, the Swings model showed the worst performance, with no change in the number of connected components, undesirable edges or bricks, because of the limited search space.

The Final Adjustment columns summarize the impact of performing this highly localized transformation. Dashes are used to indicate that the Final Adjustment phase was omitted because the model had already been reduced to a single connected component by earlier searches. The results for the remaining models show that performing this search process successfully reduced the number of connected components in 19 of the 21 models that it was applied to. No reduction was achieved in the final two cases because the models contained combinations of voxels that cannot be constructed as a single connected component using the bricks shown in Figure 2. Such structures are discussed in detail in the next section.

In 10 of the 19 cases where a reduction occurred, the changes made by the Final Adjustment phase reduced the model to a single connected component. In an additional 6 cases, the final adjustment phase reduced the model to the minimum number of components that can be physically constructed. Entries greater than 1 that reached the minimum number of connected components are marked with a *. The final reduction failed to achieve the minimum number of connected components for 3 models: Butterfly, Scorpion and Tuna. These entries are marked with a $^+$.

In the Butterfly model there were two additional connected components that could be removed, but were not successfully replaced by this search process. These connected components consisted of a total of 4 voxels, or 0.077 percent of the voxels in the model. In the Scorpion model, one additional connected component consisting of 2 voxels could have been removed. Those two voxels represent less than 0.040 percent of the voxels in the model. In both cases, the disconnected bricks can simply be omitted from the model as they have little, if any, effect on its overall visual impact. Even with the two models side by side, it is unlikely that a casual viewer would notice the difference.

The connected component that could have been removed from the Tuna model consisted of 3 bricks: two 1x1 bricks and a 2x3 brick, for a total of 8 voxels. While these 8 voxels only account for approximately 0.11 percent of the voxels in the model, they form its bottom lip, and simply omitting them meaningfully detracts from the look of the model. As such, the builder would need to either identify a way to rearrange the bricks that the search processes failed to identify, change the model, or use bricks that are not included in Figure 2 to construct it in a connected manner.

Investigating these three connectedness failures revealed that they occurred for two different reasons. The failures in the Butterfly and Scorpion models occurred because all 36 candidates constructed by our initial heuristics contained at least one unconnected brick for the layers where the disconnected components were present. This forced the subsequent search processes to start from particularly poor configurations in both of these cases, and unfortunately, they failed to find their way to a fully connected result. As a result, we believe that further effort expended on the initial heuristics would likely result in an outcome that contained the minimum number of connected components. Future versions of these initial heuristics should perform additional work to locate a starting configuration that does not contain any unconnected bricks when every candidate returned by the current heuristic contains one or more unconnected bricks.

The failure to reach a connected result in the Tuna model occurred for the opposite reason. Our analysis of the initial heuristics failed to recognize that the 2x3 brick was unconnected; the analysis detected that the 2x3 brick overlapped with the 1x1 bricks in the adjacent layer, and incorrectly concluded that it was a reasonable choice rather than investigating further to realize that the 1x1 bricks were otherwise disconnected, and as such, did not actually provide any support. As a result, layers containing this 2x3 brick were not discarded from the initial set of heuristically generated layers. If they had been, a different heuristically generated layer would have been selected, and this portion of the model very likely would have been connected. Performing a more detailed analysis of connectedness on the heuristically generated layers is possible, but at some point the time required to perform the analysis will likely become impractical. Additional investigation in this area is needed to develop a more thorough analysis that runs in a reasonable amount of time.

Non-Buildable Configurations

Some of the models in the set tested during this study cannot be reduced to a single connected component because they include an arrangement of voxels that is physically impossible to construct using the parts shown in Figure 2. Two examples of such constructs are examined in this section, though other impossible-to-construct configurations also exist and occur in our collection of 30 models.

The images in Figure 6 show two different ways of constructing one of the tusks in the Mammoth model (the rest of the mammoth has been omitted for clarity). In each case, the

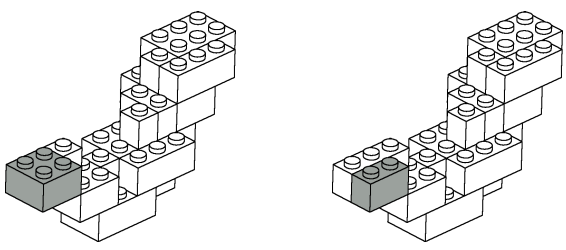

Figure 6: Possible Configurations for the Tip of the Mammoth's Tusk

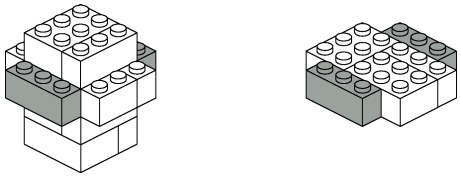

Figure 7: A Small Part of the Eiffel Tower

white portion of the tusk is one connected component, while the gray portion of the tusk is a separate connected component. These are the only two minimal configurations for constructing this shape; any other configuration can be reduced to one of these two configurations by repeatedly merging adjacent bricks, and as such, cannot be an improvement on either of these cases. As a result, there is no combination of the bricks in Figure 2 that will allow the tip of the tusk to be constructed as a single connected component. More generally, if a single layer, without any bricks immediately above or below it, has a shape that is wider at an unsupported point than at its supported point then there is no way to construct the shape in a connected manner.

Some types of corners are impossible to construct as a single connected component. In this case, a single point of connection is needed to support bricks on both sides of the corner. Since two bricks cannot be connected to the same stud, such a configuration cannot be constructed. This configuration occurs in several of the models, including a dozen occurrences in the Eiffel Tower model resulting in 10 components that cannot be removed. Four occurrences of the problematic corner are shown in Figure 7, along with the layer that contains them. The largest component of the model is shown in white. The two disconnected components are each shown in gray. No matter how the bricks in that layer, or any other layer in the model, are rearranged, there will always be at least 3 connected components.

We elect to provide a result that contains multiple connected components to the builder. This allows the builder to decide how to address the problem. Options include adding or removing voxels to eliminate the non-buildable configuration, using bricks that are not included in Figure 2 to build the desired shape in a connected manner, or using 3 layers of LEGO plates[4] to construct the shape in a connected manner.

Future Work

While the search process that we described previously generates a result with the minimum possible number of connected components for 90 percent of the models that we considered, changes to the initial heuristic phase would likely allow our algorithm to generate a result containing the minimum number of connected components for the remaining three models. We look forward to investigating these cases further, implementing improvements and evaluating the impact of those changes on the three problematic models (and the other 27 models considered in this study). In addition, we would like to evaluate the effectiveness of our strategy on a wider collection of models, including all 142 models in the Yahoo! JAPAN collection. Doing so will provide additional insight into the problem and will allow us to further refine our process so that it generates even better results.

Another area of future work that we would like to explore is improved performance for our algorithm. Right now the time required to generate the result on a Core i7 930 running at 2.80 GHz varies from as little as 11 minutes to almost 28 hours, with an average run time of 4 hours, 30 minutes. We consider these run times acceptable because, on average, they are similar to the amount of time that would be required to assemble the model from physical bricks. However, shorter run times that achieve the same result are, of course, desirable. Part of the long run times is due to our choice to implement our algorithm with Python. While using Python allowed for rapid experimentation and easier debugging, re-implementing our solution in C++ would likely reduce the run times by at least an order of magnitude. There are also a number of ways that the process could be parallelized. This would further reduce the running time by allowing our implementation to make effective use of additional cores.

Conclusion

This paper has described a multi-phase search approach to the LEGO Construction Problem. Our novel approach uses up to four different search phases to arrive at its result. Each search phase has its own strengths and makes different contributions to the quality of the results that we have achieved.

The initial heuristics generated 36 candidates for each layer so that a reasonable starting point could be selected for the subsequent search processes. Then the Edge Reduction phase removed a substantial portion of the undesirable edges from the model, while also reducing the number of connected components. This was followed by the Adjacent Part Recombination search. It had a modest positive impact on the number of connected components and undesirable edges in the model while also reducing the number of bricks needed to construct it. A final adjustment was applied to models that were not fully connected after the previous search processes completed. This highly targeted search was instrumental in reaching the minimum number of connected components for many models.

Testing our algorithm revealed that it identified a solution with the minimum possible number of connected components for 90 percent of the models considered in this study. The three models that were not successfully reduced to the minimum number of components each had one or two additional components totaling a maximum of 8 voxels. We have identified the factors that caused a disconnected result for each of these three models, and we look forward to implementing changes to our initial heuristics that will address these deficiencies in the future.

[4]LEGO plates are one third of the height of LEGO bricks. Plates can be used to construct volumes as a single connected component that cannot be constructed as a single connected component using bricks because the larger number of layers when using plates provides additional opportunities for overlap.

References

Garg, A.; Sageman-Furnas, A. O.; Deng, B.; Yue, Y.; Grinspun, E.; Pauly, M.; and Wardetzky, M. 2014. Wire mesh design. *ACM Trans. Graph.* 33(4):66:1–66:12.

Gower, R. A. H.; Heydtmann, A. E.; and Petersen, H. G. 1998. LEGO: Automated model construction. *32nd European Study Group with Industry – Final Report* 81–94.

Kozaki, T.; Tedenuma, H.; and Maekawa, T. 2016. Automatic generation of lego building instructions from multiple photographic images of real objects. *Comput. Aided Des.* 70(C):13–22.

Lee, S.; Kim, J.; Kim, J. W.; and Moon, B.-R. 2015. Finding an optimal LEGO® brick layout of voxelized 3D object using a genetic algorithm. In *Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation*, GECCO '15, 1215–1222. New York, NY, USA: ACM.

Luo, S.-J.; Yue, Y.; Huang, C.-K.; Chung, Y.-H.; Imai, S.; Nishita, T.; and Chen, B.-Y. 2015. Legolization: Optimizing LEGO designs. *ACM Trans. Graph.* 34(6):222:1–222:12.

Mitani, J., and Suzuki, H. 2004. Making papercraft toys from meshes using strip-based approximate unfolding. In *ACM SIGGRAPH 2004 Papers*, SIGGRAPH '04, 259–263. New York, NY, USA: ACM.

Mori, Y., and Igarashi, T. 2007. Plushie: An interactive design system for plush toys. In *ACM SIGGRAPH 2007 Papers*, SIGGRAPH '07. New York, NY, USA: ACM.

Mueller, S.; Mohr, T.; Guenther, K.; Frohnhofen, J.; and Baudisch, P. 2014. fabrickation: Fast 3d printing of functional objects by integrating construction kit building blocks. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '14, 3827–3834. New York, NY, USA: ACM.

Petrovic, P. 2001. Solving LEGO brick layout problem using evolutionary algorithms. Technical report, Norwegian University of Science and Technology.

Skouras, M.; Thomaszewski, B.; Kaufmann, P.; Garg, A.; Bickel, B.; Grinspun, E.; and Gross, M. 2014. Designing inflatable structures. *ACM Trans. Graph.* 33(4):63:1–63:10.

Song, P.; Fu, C.-W.; and Cohen-Or, D. 2012. Recursive interlocking puzzles. *ACM Trans. Graph.* 31(6):128:1–128:10.

Testuz, R.; Schwartzburg, Y.; and Pauly, M. 2013. Automatic Generation of Constructable Brick Sculptures. In Otaduy, M.-A., and Sorkine, O., eds., *Eurographics 2013 - Short Papers*. The Eurographics Association.

van Zijl, L., and Smal, E. 2008. Cellular automata with cell clustering. In Adamatzky, A.; Alonso-Sanz, R.; Lawniczak, A. T.; Martnez, G. J.; Morita, K.; and Worsch, T., eds., *Automata*, 425–441. Luniver Press, Frome, UK.