Survey on Automated LEGO Assembly Construction

Jae Woo Kim
ETRI
218 Gajeong-ro
Yuseong-gu
Daejeon 305-700, Korea
jae_kim@etri.re.kr

Kyung Kyu Kang
ETRI
218 Gajeong-ro
Yuseong-gu
Daejeon 305-700, Korea
kangk2@etri.re.kr

Ji Hyoung Lee
ETRI
218 Gajeong-ro
Yuseong-gu
Daejeon 305-700, Korea
ijihyung@etri.re.kr

ABSTRACT

LEGO has been very popular toy in the world because it is attractive and fun to play with and stimulates one's creativity by providing means to conveniently assemble a variety of interesting shapes using the limited types of given bricks. However, it is hard for the beginners to design and assemble complex models they desire to make without instructions. Building a LEGO assembly manually usually requires a significant amount of trial-and-error. LEGO company therefore presented the LEGO construction problem in 1998 and in 2001. The problem statement is "Given any 3D body, how can it be built from LEGO bricks?" In this paper we will investigate the current research efforts to address the LEGO construction problem. We will review the problem definition, formulation, and a variety of approaches to solve the problem. We will discuss the data representations for input 3D polygonal models and the LEGO assembly structures, cost functions that will guide the search for the optimal solution, and various solution methods.

Keywords

Engineering, Design, LEGO, Construction, Optimization, Evolutionary, Algorithms

1. INTRODUCTION

In the 1940s in Denmark, LEGO was designed, developed, and produced by the LEGO company, the most successful toy manufacturer [Sma08, Sil09]. LEGO has been very popular toy in the world because it is attractive and fun to play with and stimulates one's creativity by providing means to conveniently assemble a variety of interesting shapes using the limited types of given bricks(Fig. 1) [Tes13, Ono13].

However, it is hard for the beginners to design and assemble complex models without instructions [Ono13]. Building a LEGO assembly manually requires a significant amount of trials-and-errors [Tes13]. LEGO company therefore presented the LEGO construction problem in 1998 and in 2001. The problem statement is "Given any 3D body, how can it be built from LEGO bricks [Tim98]?"

Researchers tried to develop softwares that can automatically generate LEGO assemblies and assembly instructions from the geometric specifications of the desired object. In most research efforts, 3D polygonal model data were used as specifications of the objects.

LEGO construction problem is simple and easy to understand. It is however hard to solve using mathematical or algorithmic approaches on computer because there exist a number of different ways to construct a LEGO model for an object specified by users [Sma08].

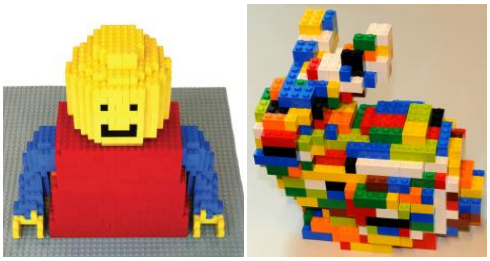

Figure 1. Computer generated LEGO representation(left) and real LEGO assembly(right) [Tes13]

Other than the applications for entertainment purposes described above, study on LEGO construction problem will have great practical value and contribute to other areas such as engineering design or engineering education because the process of LEGO assembly generation is similar to the generation of real engineering artifacts [Pey03]. Campbell et al. showed that how various physical and chemical principles related to nanoscale science and technology can be demonstrated using LEGO models [Cam12]; Wang et al. designed and developed a digital LEGO system that provide a generic representation of security protocols and used it in teaching students. The digital LEGO system helps the students conveniently understand these abstract concepts [Wan08]; Yip-Hoi and Newcomer used LEGO to teach CAD modeling techniques to engineers [Yip11].

Solution methods developed to solve the LEGO construction problem are related to other interesting

problems: Mitani et al. proposed a method to produce unfolded papercraft patterns of toy figures from 3D polygonal mesh data using strip-based approximation [Mit04]; Igarashi and Suzuki proposed a method to create close-fitting customized covers for three-dimensional objects [Iga11]; Xin et al. proposed a method to design and model burr puzzles from 3D geometric models [Xin11]; Lo et al. proposed a method to generate 3D Polyomino puzzle that constructs 3D surface model using Polyomino pieces [Lo09].

In this paper, we investigated a variety of approaches to solve the LEGO construction problem. In the next chapter, the problem definitions and formulations will be covered. In chapter 3, we will discuss the data representations for both of the input 3D polygonal models and the LEGO assembly structures. In chapter 4, we will introduce the cost functions that will guide the search for the optimal solution and in chapter 5 we will discuss various solution methods that have been used to solve the LEGO construction problem.

2. Automated LEGO Assembly Construction Problem

When a user has a design of an arbitrary object in mind, she will try to build an LEGO assembly with a significant amount of trials and errors. LEGO construction problem is to find the optimal way of converting 3D polygonal mesh data to LEGO representation given the number of LEGO bricks.

Smal defined the LEGO construction problem as the development of a software application that generates the LEGO building instructions for the given arbitrary real-world object [Sma08]. Here, the real world object is represented by 3D polygonal mesh models. The output of the LEGO construction software applications include LEGO model representations, 3D renderings of the LEGO model, and assembly instructions for building the LEGO models.

We need to simplify the problem by applying several restrictions that can reduce the search space for the solution [Gow980][Sma08][Tim98]. Real-world object that users desire to build must be firstly converted to an appropriate representation such as "legolised" representation. The legolised representation is a matrix whose elements can only have ones and zeros. The value one for an element denotes that the space is covered with a brick or a part of a brick, the value zero denotes an empty space [Sma08].

Bricks available in building a LEGO representation must be restricted to a limited number of types to save processing time for optimization by reducing the search space. Usually, "family" LEGO bricks and DUPLO bricks were used in previous research to address the problem [Pet01]. To save time and reduce the number of bricks, the inside of the sculpture must be kept hollow as far as the connectivity and stability are kept. Colors can be ignored to save processing time because incorporating color information into the problem can increase another dimension of search space resulting in drastic increase in search space [Sil09].

There are two major performance criteria that the solution of the LEGO construction problem must satisfy. The first criterion is that the created LEGO sculpture must be connected and stable. Another criterion is that the conversion from an object model to a corresponding LEGO representation must be complete in a reasonable time period [Sma08].

The automated LEGO construction problem can be formulated as an optimization problem to find the optimal LEGO structure that best represent the input real-world object. In general, optimization techniques can be divided into two categories: one is a deterministic technique and the other is a stochastic technique. Deterministic approaches are used to find the globally optimal solution and they are appropriate to the problems whose solution space is relatively small. Efficient state space search methods such as branch-and-bound methods, or algebraic methods are usually used to find the globally optimal solution. When the solution space is extremely large and it is therefore not feasible to find global optimal solution, Stochastic technique can be used. Stochastic technique finds good solutions in a reasonable time period by using heuristics and probability theories that guides the search [Sma08].

The automated LEGO construction problem cannot be solved using deterministic optimization techniques because the solution space is extremely large. We therefore discuss stochastic optimization techniques to solve the LEGO construction problem [Sma08].

There are a variety of solution methods to address the optimization problems and greedy methods, local search, beam search, cellular automata and evolutionary algorithms were used to solve the LEGO optimization problem. From next chapter, we will discuss the approaches used to solve the automated LEGO construction problems.

3. DATA REPRESENTATIONS

The real-world object that the users desire to create are usually given as a 3D polygonal mesh models. Users can create the 3D mesh models using modeling softwares or can easily download them from the internet [Lam06]. The first step in solving the LEGO construction problem therefore is to convert a given 3D polygonal mesh models to a data representation that is appropriate for the process of LEGO assembly generation. A typical data representation for the real-

world object is a "legolised" model proposed by [Gow98].

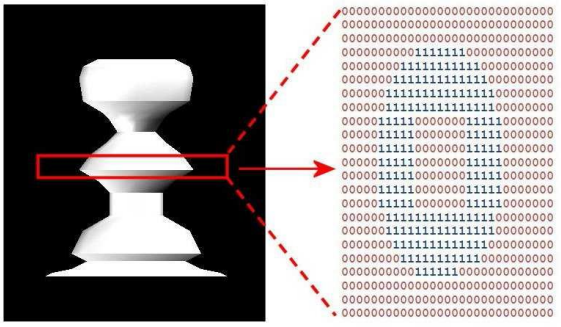

Figure 2. A 3D polygonal model and its legolised representation for a horizontal cut [Sam08]

Voxelization

Voxel representation is naturally employed by researchers to represent the real-world objects because basic LEGO bricks has the rectangular shapes that matches well with voxels. LamBrecht used ray casting technique in voxelizing the input 3D mesh models. Their approach casted a ray in a axis-aligned direction from the each column of voxels. The algorithm conducts voxelization by iterating through all the faces of the model in the cube and testing for intersection between the face and the ray [Lam06].

Silva et al. proposed a novel voxelization algorithm that uses point samples of the surface to determine which voxels each point belong to. Their algorithm assumes a uniform sampling of the surface. In general, it is however not guaranteed for most of the triangular meshes due to their shape irregularity with varying edge sizes. They therefore conducted a subdivision algorithm to transform the given mesh model into a uniformly sampled model where all lengths of all the edges are smaller than the user-specified resolution. They implemented their algorithms on the GPU to achieve the real-time performance [Sil09].

After or during the voxelization process current approaches hollowed the model to decrease the processing time by hollowing the model. This process is conducted by removing unnecessary bricks while stability is not damaged [Tes13][Lam06].

We have to consider the trade-off of the voxel resolution when we perform the voxelization process and specify appropriate resolution depending on the application purposes. If the resolution is high, we can represent the more detailed shapes of the model but it increases the processing time drastically. If the resolution is low, the voxelization algorithm runs fast but the quality of the model is not convincing. [Tes13]

LEGO Model Representation

Figure 3. Examples of basic LEGO bricks 1x1, 1x2, 1x3, 1x4, 1x6, 1x8, 2x2, 2x3, 2x4, 2x6, 2x8 [Ono13]

The simplest way of representing LEGO model is using the voxel representation. In this approach, each voxel can be identical to the unit LEGO brick of the size 1x1 or a part of a larger brick (Fig. 3) [Ono13]. In this approach, the voxel representation is converted to the LEGO representation within the voxel space.

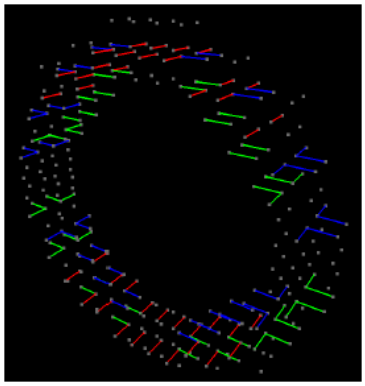

Figure 4. A legograph that represent the connectivity among the bricks in voxel space

In the beginning of the process, each voxel that the object covers is occupied by an unit brick and then replaced later by the larger bricks by merging the unit bricks considering connectivity among the bricks [Ono13].

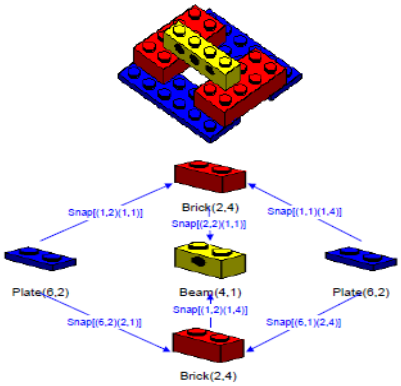

Figure 5. An example of a LEGO assembly structure and a corresponding assembly graph [Pey03]

Peysakhov and Regli used assembly graphs to represent feature-based connectivity of LEGO assemblies. An assembly graph is very expressive and can represent a variety of LEGO assembly

structures comparing other representations. In assembly graphs, the nodes represent LEGO elements and the edges represent connections among the elements. They also proposed a graph grammar that can be used to evaluate the validity of the LEGO assembly structure [Pey03].

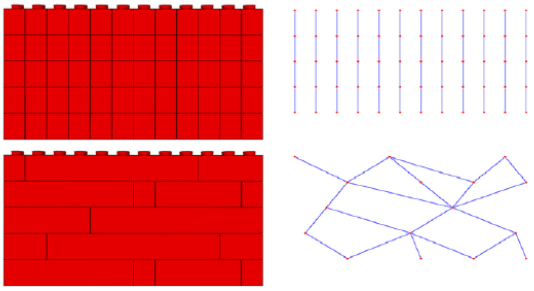

Figure 6. LEGO brick layouts (left) and corresponding graph representation [Tes13]

Testuz et al. proposed another graph representation for the LEGO structure. Fig. 6 shows the LEGO brick layouts and corresponding graph representations used to evaluate connectivity and stability of the construction. In their graph representation, a node denotes a LEGO brick and an edge denotes the connection between the bricks. The solidity optimization to improve the stability of the construction was conducted based on the assumption that the more LEGO bricks are connected, the stronger the connectivity will be [Tes13].

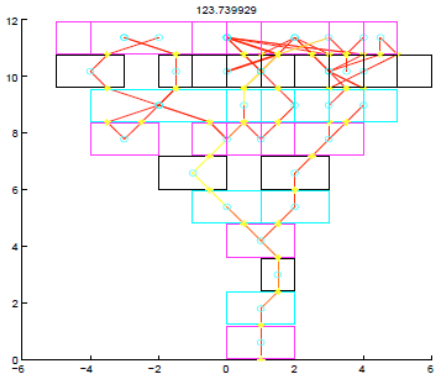

Figure 7 A brick layout and corresponding NTP representation in two--dimensional space [Fun98]

Funes and Pollack proposed a network of torque propagation structure to represent a LEGO assembly to evaluate the stability of the structure. A network of torque propagation (NTP) consists of: (1) a list of bodies, (2) a list of joints, (3) a list of forces, and a symbol G that denotes the "ground". Here a joint is defined as a the center position of the area of contact between a pair of bricks [Fun01].

When using evolutionary algorithms to solve the problem, the solution itself of the problem must be encoded as genotype representations. There are two approaches to represent genotype: one is direct and the other is indirect representation. Direct genotype representation is conceptually identical to the phenotype or the solution of the problem. In indirect representation phenotype can be constructed from the transformations of its genotype [Pey03].

The advantage of the indirect representation is that it can focus the search process through the feasible search space by significantly reducing the space. The disadvantage of the indirect representation however is that the standard genetic operators cannot be directly used [Pet01]

4. Cost Functions

The cost function for the optimization problem must be designed based on the performance criteria of the problem described in chapter 2. The most important factors to consider for cost function design are stability of the created LEGO assembly and the processing time to create it. Gower et al. introduced a set of heuristics that are useful in designing the cost function for the problem based on their rigorous research [Gow98][Sma08].

Gower et al. proposed six heuristics that are necessary to guarantee the stability of the created LEGO assembly. The first three heuristics are as follows: (1) A high percentage of the area of each brick should be covered by other bricks from above and below; (2) Larger bricks must be preferred over small bricks; (3) Bricks in consecutive layers should have alternating directionality [Gow98][Sma08].

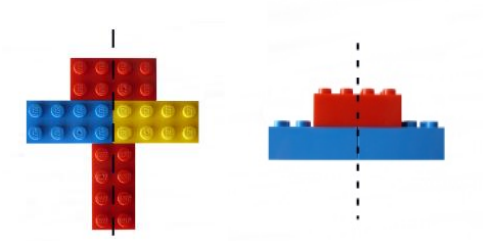

Figure 8 The boundary defined by the neighboring bricks (left) and the vertical boundary (right) [Sma08]

We need to be more careful at the boundaries of the whole model and vertical boundaries of each brick where connectivity is more vulnerable. The other three heuristics addressed the connectivity and stability problem at the boundaries: (1) A high percentage of the vertical boundaries of each brick should be covered by bricks in the consecutive layers; (2) A brick must be placed such that the middle of the side should be at the boundary defined by the neighboring bricks; (3) If a brick covers a vertical boundary in the previous layer, the middle of the brick must be aligned to the boundary [Gow98][Sma08].

Based on the heuristics Gower et al. defined a cost function as follows:

$$P = C_1P_1 + C_2P_2 + C_3P_3 + C_4P_4$$

where, P_1 relates to the alternating directionality, P_2 corresponds to coverage of the vertical boundaries, P_3 relates to the coverage of the boundary defined by the neighboring bricks, and P_4 encourages the use of larger bricks. C_i's represent the weights for each term.

Peysakhov and Regli 03 proposed a more advanced and flexible form of the cost function to evaluate the ability of an LEGO assembly relative to its performance and function. Their cost function use the attributes of the LEGO structure including weight, number of nodes, and size of the structure. Their cost function is as follows:

$$\frac{1 + \sum P_i(a_i)}{1 + \sum P_i(b_i) + \sum P_i(|c_i - t_i|)}$$

Here, P_i is the weight function that represents the importance of the parameter. They set the weight value as the equal value to the parameter itself for the most important parameters. They set the weight value to the square root of the parameter for less important ones. For the least important parameters, they used square root of square root of the parameter.

$$P_i = x_i, P_i = x^{1/2}, P_i = x^{1/4}$$

a_i denotes the properties that will be maximized such as reliability. b_i denotes the properties that will be minimized such as manufacturing cost, and c_i denotes the properties that will be as close as to the specific constant value t_i [Pey03].

5. Solution Methods

A variety of approaches have been proposed to solve the LEGO construction problem. In this section we will describe and discuss those solution methods including greedy algorithms, local search, beam search, cellular automata, and evolutionary algorithms

Greedy Algorithms

Ono and Alexis 13 proposed a method to convert a 3D mesh model into a corresponding LEGO model by using their replacement strategy. The input to the system is the 3D model and user-specified level-of-detail. The system. The system converts the input 3D model into a voxel model based on the level-of-detail, and then converts it to the LEGO model [Ono13].

The system places the unit LEGO brick of the size 1x1 to each voxel. It then merges the unit bricks to replace each voxel with larger bricks so that the resulting LEGO structure would be more stable. They represent the LEGO structure as a legograph shown in chapter 3 with three different types of links they defined. The replacement is conducted layer-by-layer, from bottom to top and the replacement is performed in each layer using a greedy method. In the replacement procedure, for each position the brick type with the highest score is chosen to be replaced. The strategy for the scoring is designed to guarantee the stability of the resulting LEGO structure and it is similar to the cost function described in chapter 4. When the LEGO structure is built, their system automatically generates the assembly instructions [Ono13].

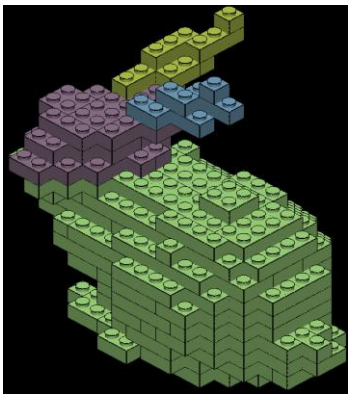

Figure 9. The result of greedy method by [Ono13]

Testuz et al. proposed a similar method to [Ono13] in that they fill the unit bricks into each voxel first and then merge and split the bricks sequentially to obtain the optimal layout using greedy method. Their merge algorithm randomly select a brick and find a legal set of neighbors. It then repeat choosing the neighbor with the lowest cost and select the neighbor with the lowest cost value until there are no more mergeable neighbors. This process repeats until there is no more brick to merge [Tes13].

In building the LEGO model Testuz et al. evaluate the stability of the model as other approaches do. To achieve this, they used the graph representation described in chapter 3. In the stability evaluation, they assumed that the stability will be stronger if the more bricks are connected to each other [Tes13].

Local search

In each step of the procedure, local search approach considers only a small subregion and attempt to find the best brick placement to fill the subregion [Gow98][Sma08]. Only a few bricks are permanently placed in each step considering the effect of the local placement for the global solution. Then the subregion slightly moves so that a new subregion overlaps the previous one as a sliding window [Sma08].

In this approach, the important issue is the size of the subregion. If the size of the subregion is too small, it is hard for the local placement contribute to the global optimization. On the other hand, if the size of the subregion is too large, the search space would be larger resulting in increase in processing time. The optimal size for the subregion therefore must be determined based on the size and characteristics of the input real-world object. [Sma08]

To apply simulated annealing to the LEGO construction problem, we can firstly divide each layer into subregions of smaller size. Then the subregions will be randomly filled with arbitrary placements of LEGO bricks resulting in the initial state. For each subregion a set of successor states are generated by replacing a small number of bricks with new bricks. New successor states are generated until we find a new state that decreases the energy. The search process will stop after the number of iterations specified by a user is complete or when an acceptable solution is found [Sma08].

Simulated Annealing

Simulated annealing is a variant of the hill-climbing technique that computes all possible successor states from the current state and then selects the best successor. The well-know limitation of the approach is that it can easily converges to the local minimum instead of the global optimum [Sma08].

At each iteration, simulated annealing algorithm considers a set of neighboring states from the current state. It probabilistically compares between moving to new states and staying in the current state and then decide the new state that minimizes the energy. This process repeats until it finds a satisfactory solution [Sma08].

Beam Search

A beam search is conceptually similar to the simulated annealing approach in which successor states are generated and evaluated to find a new state with better quality. A beam search approach a best-first search algorithm and it is different from the simulated annealing approach in that all the possible successor states are generated and evaluated using a cost function to find the new state with the best cost. The algorithm therefore searches for the best local solution at each step [Sma08].

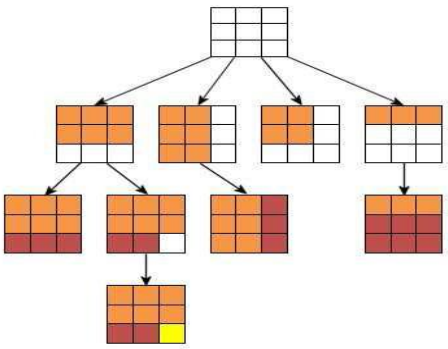

Figure 10. Beam search tree to a fill 3x3 layer using the standard LEGO bricks [Sma08]

At each step a beam search algorithm finds best *k* successors and they are added to their parent states. Then the search process continues while pruning the states that cannot generate any successor states of better quality from the search tree. The problem however is that it can focus on a too narrow search space resulting in not convincing solutions. An improvement for this problem is to select the *k* successors probabilistically with a higher probability of selecting the lower cost successors to create a broader search space [Sma08].

Cellular Automata

van Zijl and Smal proposed an approach using cellular automata based on the cost function proposed by [Gow98][Van08]. Their approach is conceptually similar to the merge/split approach using several heuristics that guides the search [Tes13]. The approach virtually cuts the given 3D object into horizontal two-dimensional layers. It finds the optimal 2D layout first and then join them to construct final 3D model. If we solve a 2D layout optimization problem separately, the stability of the resulting model cannot be guaranteed. They therefore used the Gower et al.'s heuristics during each step to solve the 2D problem to guarantee the solidity of the model [Gow98][Van08].

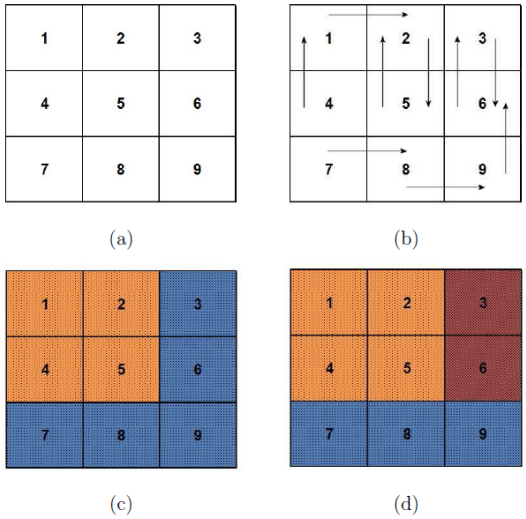

Figure 11. An example of cellular automata representation (a) the 2D grid, (b) potential merge neighbors, (c) potential new clusters, (d) the final three clusters [Sma08]

This approach used a legolised representation described in chapter 3. In initial stage, each cluster of unit size 1x1 that contains value one represent a unit LEGO brick. For each cluster the algorithm checks if it can be merged with any of its Von Neumann neighborhood. Two clusters can be merged if the merge can result in a new cluster that represent a larger valid LEGO brick. This merge operation is conducted for all the clusters in the layout. The order of merges can be random, front-to-back, or any other orders chosen by a user [Van08].

Evolutionary Algorithms

Evolutionary algorithms are very effective optimization technique for the problems whose optimal solution is hard to formalize. LEGO construction problem is a hard combinatorial optimization problems in which it is infeasible to find the optimal solution and the "good" solutions of reasonable quality are enough. Evolutionary algorithms could be a proper approach to solve the problems that have such features and nature [Pey03][Pet01].

To solve an optimization problem using evolutionary algorithms, we have to encode the solution of the problem as chromosomes, define the evaluation function, and develop mutation and recombination operators depending on the characteristics of the given problem. We have discussed about the genotype representation for LEGO construction problems in chapter 3 and we will therefore discuss about evaluation functions and operators developed to solve the problem using evolutionary algorithms [Pey03][Pet01].

There are two approaches to genotype representations: one is direct representation, and the other is indirect representation. In direct representation the genotype is conceptually identical to the corresponding phenotype. On the other hand, in indirect representation, the genotype is transformed to construct the corresponding phenotype. Indirect representation usually have more information about the phenotype and it therefore can focus the search space by reducing the space. The problem of indirect representation is that the standard operators such as mutation and crossover do not directly work. We therefore have to redefine the operators according to the structure of genotype [Pey03][Pet01].

0	Plate(6,2)
1	Plate(6,2)
2	Brick(2,4)
3	Brick(2,4)
4	Beam(4,1)

0>2	Snap[(1,2)(1,1)]
0>3	Snap[(6,2)(2,1)]
1>2	Snap[(1,1)(1,4)]
1>3	Snap[(6,1)(2,4)]
2>4	Snap[(2,2)(1,1)]
3>4	Snap[(1,2)(1,4)]

Figure 12. An example of genotype representation by [Pes03]

Peysakhov and Regli developed their chromosomes using a combination of two data structures: one is an array of all nodes, and the other is the adjacency hash table containing all edges as shown in fig 12. The key value of the hash table represents the position and direction of edges. For example, the key "1>3" means that the edge connects from the node 1 to the node 3. Hash table also describe how the LEGO elements are connected [Pey03].

The mutation operator of [Pey03] is applied with constant and low probability to provide the balance between the exploration and exploitation. When a mutation arises for a node, a LEGO brick is simply replaced by the same type brick with different size [Pey03].

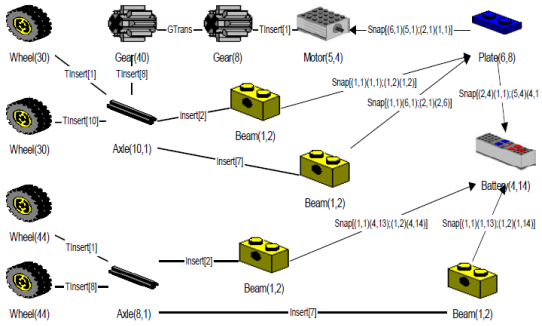

Figure 13. An example assembly graph for the LEGO car [Pey03]

Crossover is conducted by two operators: cut and splice. It selects two chromosomes for crossover and random points are selected respectively for the two chromosomes by cut operator. The tail parts of the parent chromosomes are then spliced with the head parts of them [Pey03].

Petrovic proposed more advanced and complicated operators as follows. His crossover operator firstly selects a rectangular region at random. Then a part of LEGO bricks are copied from one parent and other bricks that do not conflict with already placed bricks are copied from another parent [Pet01].

Petrovic suggested the following mutation operators because random mutation operator can generate overlaps.

- A brick is replaced by other random brick.
- A brick is added to an empty location randomly.
- A brick is shifted by one unit in one of the four possible directions.
- A brick is eliminated from the layout
- A brick is extended by one unit in one of the four possible directions.
- All bricks that are in a random rectangle are replaced by random bricks
- The whole layout is initialized again

In his mutation operators, larger bricks are always preferred to be replaced to increase stability of the structure [Pet01].

6. CONCLUSIONS

In this paper, we reviewed a variety of research efforts to address the automated LEGO construction problem. We investigated the problem definition and formulation, various data representations for 3D polygonal mesh models and LEGO assembly structures, cost functions to solve the optimization problem for LEGO construction and solution

methods a number of researchers proposed. To date, graph representations have been widely used to represent the LEGO structure and as solution methods, greedy algorithms, simulated annealing, beam search, cellular automata, and evolutionary algorithms have been used to automatically construct LEGO structure minimizing the number of bricks used and guaranteeing the stability of the built structure. Those approaches are useful to create a LEGO structure design for given 3D polygonal models for entertainment purposes and also can be useful for engineering education.

7. REFERENCES

[Cam12] Campbell D., Freidinger E., Querns M., Swanson S., Ellis A., Kuech T., Payne A., Socie B., Condren S. M., Lisensky G., Rasmussen R., Hollis T., Villarreal R., Campbell K., Exploring the Nanoworld with LEGO Bricks, A Free Book, Materials Research Science and Engineering Center, University of Wisconsin-Madison, 2012. http://education.mrsec.wisc.edu/LEGO/PDFfiles/nanobook.PDF

[Fun01] Funes P. J., Pollack J. B. Componential Structural Simulator. Technical Report, Brandeis University, 1998.

[Gow98] Gower R., Heydtmann A., Petersen H. LEGO: Automated Model Construction. Jens Gravesen and Poul Hjorth, pp. 81-94, 1998.

[Iga11] Igarashi Y., Suzuki H. Cover geometry design using convex hulls. Computer-Aided Design, 43, 9, pp. 1154-1162, 2011.

[Lam06] Lambrecht, B. Voxelization of boundary representations using oriented LEGO plates. University of California, Berkeley, 2006. http://lego.bldesign.org/LSculpt/lambrecht_legovoxels.pdf

[Lo09] Lo K. Y., Fu C. W., Li H. 3D polyomino puzzle. ACM Transactions on Graphics, 28, 5, Article No. 157, 2009.

[Mit04] Mitani J., Suzuki H. Making papercraft toys from meshes using strip-based approximate unfolding. ACM Transactions on Graphics, 23, 3, pp. 259-263, 2004.

[Ono13] Ono S., Alexis A. Automatic generation of LEGO from the polygonal data. International Workshop on Advanced Image Technology, pp. 262-267, 2013.

[Pet01] Petrovic P. Solving the LEGO brick layout problem using evolutionary algorithms. Technical Report, Norwegian University of Science and Technology, 2001.

[Pey03] Peysakhov M., Regli W. Using Assembly Representations to Enable Evolutionary Design of Lego Structures. Artificial Intelligence for Engineering Design, Analysis and Manufacturing, 17:155-68, 2003.

[Sil09] Silva L., Pamplona V., Comba J. Legolizer: A real time system for modeling and rendering LEGO representations of boundary models. XXII Brazilian Symposium on Computer Graphics and Image Processing (SIBGRAPI), pp. 17-23, 2009.

[Sma08] Smal E. Automated Brick Sculpture Construction. MS. Thesis, The University of Stellenbosch, 2008.

[Tes13] Testuz R., Schwartzburg Y., Pauly M. Automatic generation of constructable brick sculptures. Eurographics 2013 Short Papers, pp. 81-84, 2013.

[Tim98] Timcenko O. LEGO: How to build with LEGO. 32nd European Study Group with Industry Final Report, pp. 81-94, 1998. http://www.maths-in-industry.org/past/ESGI/32/Report/ESGI32.ps

[Van08] van Zijl L., Smal E. Cellular automata with cell clustering. Proceedings of Automata 2008 Workshop, Bristol, UK, pp. 425-440, 2008.

[Wan08] Wang W., Lu A., Yu Li., Li Z. A digital lego set and exercises for teaching security protocols. the 12th Colloquium for Information Systems Security Education, University of Texas, Dallas, TX, June 2-4, 2008.

[Win05] Winkler D. Automated brick layout. BrickFest, 2005. http://www.brickshelf.com/gallery/happyfrosh/BrickFest2005/automatedbricklayout.pdf

[Xin11] Xin S., Lai C. F., Fu C. W., Wong T.T., He Y., Cohen-Or D. Making Burr Puzzles from 3D models. ACM Transactions on Graphics, 30, 4, Article No. 97, 2011.

[Yip11] Yip-Hoi D. M., Newcomer J. L. Teaching CAD modeling using LEGO. American Society for Engineering Education, 2011. http://www.asee.org/public/conferences/1/papers/152/download