#176

EXTENDING SPACE SYNTAX WITH EFFICIENT ENUMERATION

Algorithms and Hypergraphs

ATSUSHI TAKIZAWA
Graduate School of Engineering, Osaka City University
takizawa@arch.eng.osaka-cu.ac.jp

ABSTRACT

In this research, we perform building-scale analysis, and the use of a convex map may be considered first. However, partitioning an architectural space is not easy, due to the ambiguity of the space. We believe that an analysis method to decompose the space with moderate granularity while considering its ambiguity is necessary. Hiller, for example, has been analysing space by convex polygonal coverings from before. However, these methods contain the NP-hard set cover problem. Based on this background, we propose a novel method that enumerates all patterns covering a relatively small space, such as a floor plan of a building with finite areas like convex polygons, and quickly extracts the optimum cover from them. The obtained cover is translated into a graph based on the concept of the hypergraph, and a method for calculating network features within it is proposed. We apply this method to the building-scale spatial data and examine the number of enumeration solutions, computational time, and network features of the obtained covering, etc. Finally, we show that the proposed method has excellent performance in these respects.

KEYWORDS

Set cover problem, convex cover, maximal clique, enumeration, hypergraph transversal computation, binary decision diagram

1. INTRODUCTION

Analysis methods of space syntax (Hiller and Hanson, 1984) include the axial map, convex map, isovist (Benedikt, 1979), and visibility graph analysis (VGA; Turner et al., 2001). They are used appropriately based on the scale of the analysis target. For example, the axial map is often used on a macro-scale, such as a street level, and other methods are often used on a smaller scale. In this research, we perform the building-scale analysis, and the use of a convex map may be first considered. However, partitioning an architectural space is not easy, due to the ambiguity of the space. In particular, in modern architecture, space partitioning tends to become ambiguous than before, and it is less likely that nodes can be clearly set like conventional convex maps.

Conversely, VGA analyses the space as a field of local points, rather than a space. Therefore, problems like the convex map do not arise. Conversely, because it is considered that designers and users recognize space as a region with a certain degree of unity, the granularity of VGA sometimes becomes too small. From the above discussion, we believe that an analysis method to decompose space with moderate granularity while considering its ambiguity is necessary. Hiller, for example, has been analyzing space by convex polygonal coverings for several years (Hiller and Hanson, 1984; Hiller, 2007). Covering elements allow partial overlap, but it is thought that the ambiguity of space can be considered to some extent. Moreover, because the axial map is a method of covering the space with a network of intersecting lines of sight, fundamentally similar ideas can be seen. Batty's method of covering with isovist (Batty and Rana, 2004) can

also be a method to structure the space with a covering. However, both methods contain the set cover problem. Because the set cover problem of a polygon is NP-hard (O'Rourke, 1983), it is not possible to find a solution in polynomial time. The heuristic algorithms, such as the Greedy algorithm is used in the algorithms of the axial map proposed so far (Turner et al., 2005; Poponis et al., 1998; Batty and Rana, 2004) and it has not been possible to obtain an exact solution.

Conversely, according to the idea of the maximum covering described later in the paper, the combination of coverings can be enormous. When considering such a property as ambiguity of space, it may become a new indicator that characterizes the quality of space. In addition, the development of recent discrete algorithm technology has been remarkable, even for NP-hard problems it is becoming possible to find optimal solutions and enumerate all solutions in real time.

Based on the above background, in this research, we propose a novel method that enumerates all patterns covering a relatively small space such as a floor plan of a building with finite areas like convex polygons, and quickly extracts the optimum cover from them. Currently, using the compressed data structure called a binary decision diagram (BDD; Bryant, 1986) and a zerosuppressed binary decision diagram (ZDD; Minao, 1993), we formulate the problem using some set operations of BDD and ZDD, focusing on the idea of hypergraph transversal computation. The obtained cover is translated into a graph, based on the concept of the hypergraph, and a method for calculating the network features within it is proposed. Finally, we apply this method to the building-scale spatial data and examine the number of enumeration solutions, computational time, and network features of the obtained covering, etc.

2. METHODS

In this section, we describe the proposed method.

2.1 TECHNICAL TERMS AND FUNDAMENTAL DATA STRUCTURES

A hypergraph H is a pair of (V, E) of a node set V and a hyperedge set E. Different from an edge of an ordinal graph, a hyperedge can hold more than three nodes in it. A transversal (or hitting set) for E is a set $T \subseteq V$, such that T hits every hyperedge in E, that is, $T \cap U \neq \emptyset$ for all $U \subseteq V$. A hitting set is minimal if no proper subsets are hitting sets. The transversal hypergraph of E is a hypergraph whose ground set is V and whose hyperedges are all minimally hitting sets for E. The hypergraph transversal computation, given a hypergraph, computes the transversal hypergraph by generating all minimal hitting sets.

A BDD is a graph representation of Boolean functions, which was introduced for an application of VLSI logic design and verification. Figure 1 illustrates an example of a BDD (and other decision trees). The node at the top is called the root. Each internal node has the three fields, namely V, Lo, and Hi. The V holds the index of a variable. The fields Lo and Hi point to other nodes, which are called Lo and Hi children, respectively. The arc to a Lo child is called a Lo arc and is illustrated by a dashed arrow. Similarly, the arc to a Hi child is called a Hi arc and is illustrated by a solid arrow. When a family of sparse sets is represented as a BDD, it is likely that there are many nodes whose Hi arcs point to F.

Minato introduced a variety of BDDs specialized for such set families, called a ZDD.

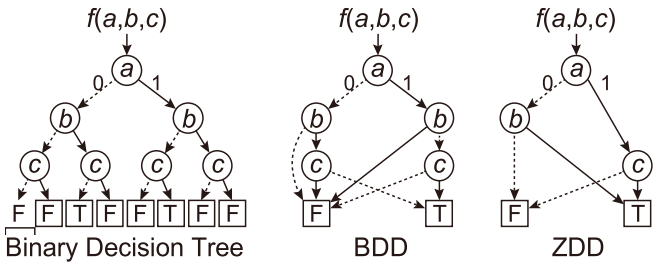


Figure 1 - Three kinds of decision trees expressing the same set of functions $f(1,0,1) = f(1,0,1) = T$

2.2 THE MINIMUM SET COVER PROBLEM FOR A SPATIAL POLYGON

Let P denote a polygon in Euclidean space R^2, allowing not only a simple polygon but also one with holes. We call a simple polygon $c \subseteq P$ a cover polygon and denote the maximum set of polygons as C . The maximum cover polygon is not completely covered by any of the other maximum cover polygons. The problem to be formulated is to enumerate all combinations of elements in C completely covering P. In addition, from among them, obtain the covering with the smallest number of the maximal covers and the largest area of them in dictionary order is also formulated.

In the above problem, it is necessary that one must judge whether P is completely covered. Considering this as a continuous geometric problem, formulation and implementation will become complicated. Conversely, in actual spatial analysis, it is often sufficient to generate discrete fine observation points in P as in VGA analysis, and check whether P covers all of them. Therefore, in this research as well, we formulate it as a problem covering all discrete points in P. Let S be the set of observation points in P. To enhance the accuracy of completely covering the area with discrete sampled points, we must generate points finely. However, it is not needed to use them all. Figure 2 illustrates an example to check whether the whole area of P is covered with two covering polygons $c^1 \cup c^2 = P$. While Figure 2a illustrates the whole set of S, we only need two points, e.g., $s^1, s^2 \in S$, as illustrated in Figure 2b. These points are included in c^1 and c^2, respectively. Meanwhile, if $s^3 \in S$ is selected, either c^1 or c^2 is selected because the point is included in both c^1 and c^2 and this is not the whole cover of P.

From the above discussion, we propose Algorithm 1. This algorithm checks the area containing each sampling point, and if the set of contained areas is the same as the other points already adopted, we discard the points and leave the necessary points (from line 2 to 9). Because the above part of the algorithm leaves points like s^3 , we eliminate them from line 10 to 13, where the $C'[0].\mathrm{pointset}$ denotes the point set covered by the oth cover of o^{th} and $C'.\mathrm{coverset}$ denotes the set of all covers for point s.

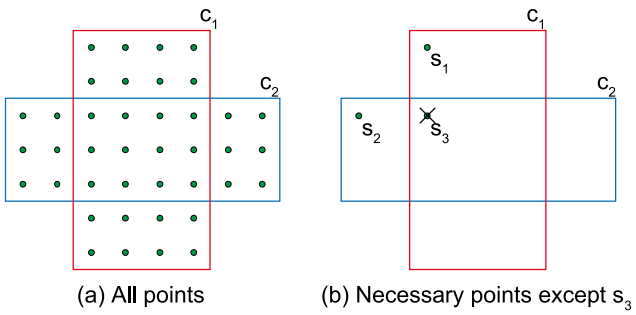


Figure 2 - Decimating sampling points.

Algorithm 1: Reduce to necessary points from all sampling points *S* for cover set *C*.

```
1:  function ReducePoints, (C,S)
2:      S′ ← ∅; CS ← ∅;
3:      for each point s ∈ S do
4:          C′ ← ∅
5:          for each point c ∈ C do
6:              if s is covered by c then C′ ← C′ ∪ {c};
7:          if C′ ∉ CS then
8:          S′ ← S′ ∪ {s};
9:          CS ← CS ∪ {C′} ;
10:     for each cover set C′ ∈ CS do
11:         if |C′|≥ 2 then continue;
12:         for each point s ∈ C′ [0]. pointset do
13:             if |s. coverset| ≥ 2 then S′ ← S′ \ {s} ;
14:     return S′;
```

Then, we can formulate this problem as a typical set cover problem that each element of S′ is covered with the elements of C. Although the set cover problem is NP-hard, as described above, we can get the optimal solution using the latest mathematical programing solver if the problem size is not so large. However, in this study, it is necessary to enumerate all possible covers to add a new criterion on the diversity of a space. We formulate this problem as a hypergraph transversal problem. Now, we have sampling points $s_1, \dots, s_{10} \in S$ and covers $c_1, \dots, c_5 \in C$, as illustrated in Figure 3. Table 1 lists the set of covers for each point in Figure 3.

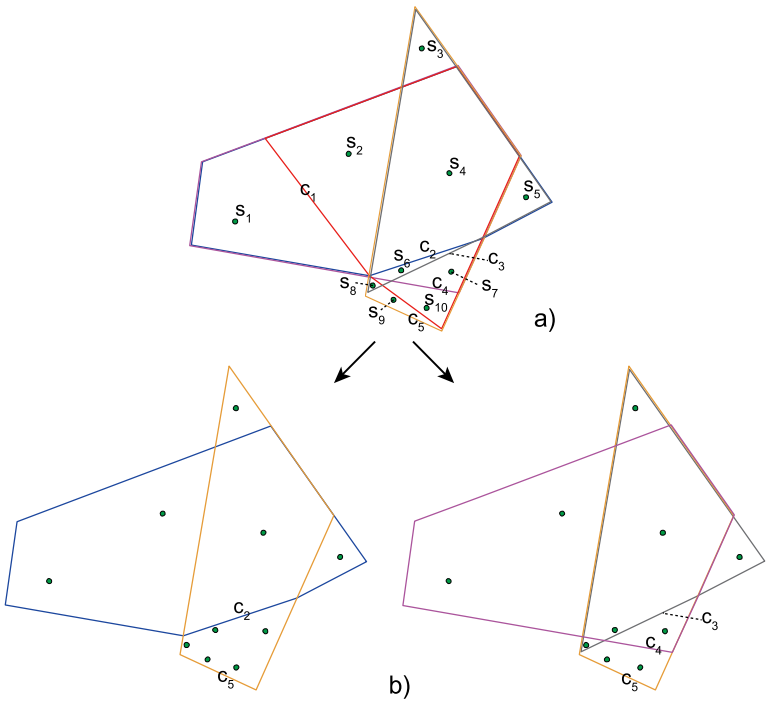


Figure 3 - An example of the minimum hitting sets: a) all cover elements and sampling points, b) all minimum hitting sets of a).

Let E denote the set of covers for each point. Then, we have a hypergraph $H=(C,E)$ where C is the ground set and E is the set of hyperedges. The subset of C that crosses all elements of E is a hitting set and its minimum is the minimum hitting set. In the example of Figure 3a, there exist two minimum hitting sets $\{c_2, c_5\}$ and$\{c_3, c_4, c_5\}$, as illustrated in Figure 3b. The problem with finding the minimum hitting set and the minimum covering is equivalent, and is similarly NP-hard. However, in recent years, some algorithms that can efficiently solve these kinds of problems of a certain scale have been proposed. In this research, we use the hypergraph traversal algorithm by Toda (2013), which is currently the most efficient algorithm. Because his algorithm uses BDDs and ZDDs as data structures, we describe how the problem is expressed with them. Each level of the node of BDDs/ZDDs corresponds to each cover in C. For example, the cover sets $\{c_1, c_2, c_4\}$ and $\{c_3, c_5\}$ that cover sampling points s_2 and c_3, respectively, in Table 1 are expressed by a ZDD, as illustrated in Figure 4. By adding the remaining cover sets in Table 1 to the ZDD, all covering elements for each sampling point are recorded in one ZDD and ZDD F, which is input data for Algorithm 2.

Sampling point	Set of covers for each point ($\in E$)
s_1	$\{c_2, c_4\}$
s_2	$\{c_1, c_2, c_4\}$
s_3	$\{c_3, c_5\}$
s_4	$\{c_1, c_2, c_3, c_4, c_5\}$
s_5	$\{c_2, c_3\}$
s_6	$\{c_1, c_3, c_4, c_5\}$
s_7	$\{c_1, c_4, c_5\}$
s_8	$\{c_3, c_5\}$
s_9	$\{c_5\}$
s_{10}	$\{c_1, c_5\}$

Table 1 - Sets of covers for each point in Figure 3.

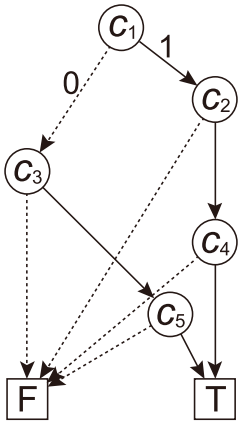


Figure 4 - ZDD expression of cover sets $\{c_1, c_2, c_4\}$ and $\{c_3, c_5\}$for sampling points s_2 and s_3, respectively.

Algorithm 2 enumerates all minimum hitting sets of F by using Toda's algorithm. This algorithm inputs a ZDD variable F, enumerates all hitting sets in line 2, reduces to the minimum hitting sets in line 3 and outputs them as a ZDD variable Z in line 4. The detail of functions Hit() and Min() is described in the appendix.

Algorithm 2: Enumerating the minimum hitting set of F.

```
1:  function EumMinHitSet (F)
2:      B ← Hit (F);
3:      Z ← Min (B);
4:      return Z
```

Then, we show Algorithm 3 that finds the cover having the smallest number of cover elements in Z with some efficient set functions that are implemented in the BDD/ZDD library we use. The class method of PermitSym(i) is a set function that extracts all combinatorial sets whose item size is less than or equal to Z in the ZDD variable Z, and returns them as a new ZDD variable. The class method Card () is also a set function that returns the number of items in the ZDD variable. This algorithm extracts the combinatorial set whose item size is just i in line 3 and outputs it if the size is greater than or equal to one in lines 4 and 5.

Algorithm 3: Find the set of covers with the smallest number of cover elements in Z

```
1:  function FindSmallestCovers (Z, C)
2:      for i ← 1 to i ≤|C| do
3:          Z′ ← z. PermitSym(i) — Z. PermitSym(i—1);
4:          if Z′ . Card () ≥ 1 then;
5:              return Z′
```

In general, Algorithm 3 returns a cover set having plural items. Among them, Algorithm 4 finds a cover with the largest total area of cover elements, where W[i] is the i -th element of a weight vector W, C[i] is the i-th element of a cover set C, and Area () returns the area of the cover. Chose_Best (W) is the class method that returns an item in the ZDD that maximizes the sum of the weight vector W given to each node of the ZDD. The area for each cover is given as a weight in line 3, the item that maximizes the total weight is extracted in line 4 and is returned in line 5.

Algorithm 4: Find a cover having the largest area in Z

```
1:  function FindLargestCovers (Z, C)
2:      for i ← 1 to i ≤|C| do
3:          W[i] ← C[i] . Area();
4:      z ← Z. Choose_Best (W);
5:      return z
```

With the above-described series of algorithms, it is possible to enumerate all the minimal traverses covering all the sampling points, and furthermore to extract the cover with the smallest number of covering elements and the largest area among them.

2.3 THE MAXIMAL CONVEX POLYGON AS A TYPE OF COVER ELEMENTS

In the previous section, we formulated the problem into a general framework without specifying the type of covering element, but to perform the calculation, we need to specify the type. In this study, we adopt the maximal convex polygon as a type of covering element. Covering with convex polygons is general but an important way to structure space, as Hiller showed in his books (Hiller and Hanson, 1984; Hiller, 2007). That is, because any two points in a convex area are visible, a convex area is likely to be a basic unit of space. However, there exists various types of convex polygons. For example, the convex polygon A in Figure 5 is the one formed by connecting the vertices of obstacles. The number of convex polygons of this type is finite. Conversely, convex polygons such as B and C are of a type in which the end points of the convex polygons of a single store of obstacles do not coincide. Several such convex polygons can be made. Conversely, convex polygons such as B and C are a type whose (some) vertices do not coincide with the vertices of obstacles. Such a convex polygon can be made infinite, but they are perceived as unstable areas that are difficult for residents to recognize as a unit of space. Convex polygon D is called a s-partition, proposed by Peponis et al. (1997). The s-partition is made by extending the line segment of each obstacle's face and dividing the space with the intersection of the line and other wall lines as vertices. The s-partition is thought to have the effect of dividing the space into spatial units that are cohesive, to some extent, for residents and architects.

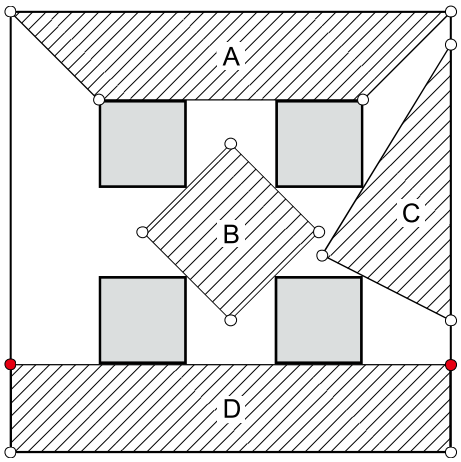


Figure 5 - Type of convex polygons in a floorplan.

From the above discussion, we consider both A and D types of convex polygons in this study. Algorithm 5 enumerates all maximal convex polygons as cliques whose vertex size is greater than or equal to three and outputs them as C. A clique of a visibility graph with greater than or equal to three nodes forms a convex polygon. For example, Figure 6 illustrates a visibility graph with a node set {a, b, c, d, e}. There exist five cliques whose node size is greater than or equal to 3; {a, b, c}, {a, b, d}, {b, c, d}, {c, d, e}, {a, b, c, d}. Among them, we have two maximal cliques {a, b, c, d} and {c, d, e} because other cliques {a, b, c}, {a, b, d}, {b, c, d} are subgraphs of {a, b, c, d}. Line 4 of Algorithm 5 enumerates all maximal cliques in G. Some efficient algorithms have been proposed to enumerate the cliques and we use Coudert's algorithm (1998), based on ZDDs.

Algorithm 5: Enumerate all maximal convex polygons as cliques from a floorplan polygon Z

1: **function** EnumerateMaximalConvexPolygons (P)
2: Divide the wall of P into vertices and edges based on s-partition and let V be the set of vertices;
3: Construct a visibility graph G connecting vertices visible to each other in P;
4: Enumerate all maximal cliques C whose vertex size is more than three from G;
5: **return** C

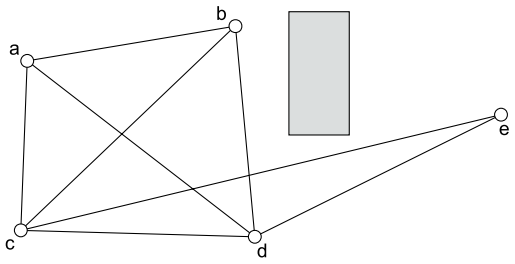


Figure 6 - Cliques in a visibility graph.

2.4 NETWORK CHARACTERISTICS OF A COVER AS A HYPERGRAPH

Next, we calculate some typical network characteristics for the cover. For that, we need to decide how to construct a graph from the cover. There can exist three methods to construct a graph. The first method is an axial map, where each cover element is regarded as a graph and each edge is drawn between each overlapped two covers. However, this method cannot consider the geometric characteristics of the overlapped area of the cover elements. However, although the value of the network characteristics of each node (cover) is generally different, this method does not determine which node's value should be adopted for the overlapped cover area.

Therefore, we partition the overlapped area. Figure 7 illustrates the partition of one of the maximal covers whose cover elements is $\{c_2, c_5\} = P$ in Figure 3. The polygon is partitioned into five nodes $\{v_1, v_2, v_3, v_4, v_5\} = V$. Here, we have two different approaches for constructing a graph. One is a general convex map where each two adjacent nodes are connected. The another is a hypergraph that (two or more) nodes in each cover element are regarded as elements of each hyperedge at the same time. In this study, we attempt the hypergraph modeling. We have the set of hyperedges $E = \{c_2 = \{v_2, v_3, v_4\}, c_5 = \{v_1, v_3, v_5\}\}$, whose node size is both three and the hypergraph $H=(V, E)$. Then, we calculate the mean depth and betweenness centrality in this hypergraph. Both measures require the shortest path computation. Because a hypergraph allows a hyperedge to have three or more nodes, the usual shortest path algorithm needs to be modified. Moreover, if the node size in a hyperedge is three or more, the network distance becomes shorter than that of a usual graph. For example, we now consider the shortest path from v_1 to v_5 in Figure 7. In a usual graph, it takes two paths because it passes through v_3. Conversely, in the hypergraph, it takes only one path because all those nodes are included in the same hyperedge v_5. We will discuss this effect and the meaning of hypergraph in the discussion.

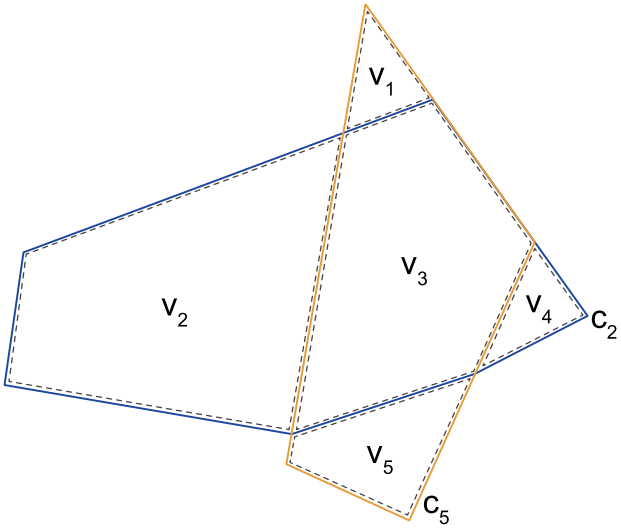


Figure 7 - Example of a hypergraph with cover partitioning.

3. VALIDATION AND RESULTS

In this section, we validate the method described in the previous section.

3.1 VALIDATION SETTINGS

We use the floorplan of the short-term treatment facility for emotionally handicapped children, that is designed by Sou Fujimoto. As illustrated in Figure 8a, the main characteristic of this architecture is that private rooms with squared shape are randomly distributed in space and various small and large open spaces are created in the gap. Those open spaces are not completely independent, but they are loosely connected with the surrounding other spaces, and their boundaries are ambiguous. The pitch of the sampling points is 10 cm. The specifications of the computer and compiler used for the validation are as follows: PC: VAIO Z (CPU: Intel Core i7-5557U, memory: 16 GB); operating system: Microsoft Windows 8.1 Professional x64; compiler: Microsoft Visual C++ 2015 (optimization option: Ox); BDD/ZDD library: Sapporo BDD in Graphillion (Inoue et al., 2016).

3.2 RESULTS

Figures 8b–g illustrate the process (steps b–g) and Figure 8h (step h) illustrates the validation result. In step b, polygon data is inputted. In step c, the polygon is s-partitioned and additional nodes are generated. In step d, a visibility graph is created by connecting vertices visible to each other, and the maximal cliques of the visibility graph are enumerated in step e. Then, we generate sampling points on the grid in step f and leave only necessary points in step g. Finally, by applying algorithms 2, 3, and 4, we find the optimal cover as illustrated in step h. As one can see, the relatively wide space located in the right-side of the floorplan is covered with a few large convex polygons. Meanwhile, many large convex polygons cover the wide space located in the left-side and it can be said that this space is more complex than the right space.

Because the main algorithm of the proposed method is one of enumeration, it is important to understand the scale of the problem to be solved and the computational time. Tables 2 and 3 list the size of the objects generated in each process and the computational time, respectively. The numbers that directly relate to the enumeration are the number of sampling points and the number of maximum convex polygons, which are 1,072 and 216, respectively. The number of enumerated minimal traverses is enormous. There is a minimum of 35 convex polygons to cover this space, but there are still 18,480 combinations of this. Conversely, the computational time is extremely short. It takes only 0.05 s for enumeration of maximal cliques, 1 s for enumeration of

minimal traversal, and 0.01 s for extraction of optimal solutions, which is the central algorithm. The memory used is also about 70 MB at maximum. This is an effect using BDD/ZDD which are compression data structures, and it may be impossible by the usual approach without compression.

Objects: polygon, graph or set	Size
Original floor plan polygon	93 (node), 93 (edge)
Additional nodes by s-partition	86
The visibility graph	179 (node), 1,728 (edge)
All maximal cliques	219 (all), 216 (except for type B and C polygons in Figure 6)
Sampling points	57,306 (all), 1,072 (necessary)
All minimum hitting sets	1,543,606,641,113,586,432
Convex polygons with minimum elements	35 (num. of elements), 18,480 (num. of combination)
Convex polygons with maximal elements	64 (num. of elements), 264,925,440 (num. of combination)

Table 2 - Size of objects generated in each process.

Step	Computational time (sec)
File input, s-partitioning and constructing a visibility graph	1.3
Enumerating all maximal cliques in the visibility graph	0.05
Generating sampling points	2
Enumerating all minimum hitting sets	1
Finding a cover with the smallest number of elements and largestarea	0.01

Table 3 - Computational time of each step of the example.

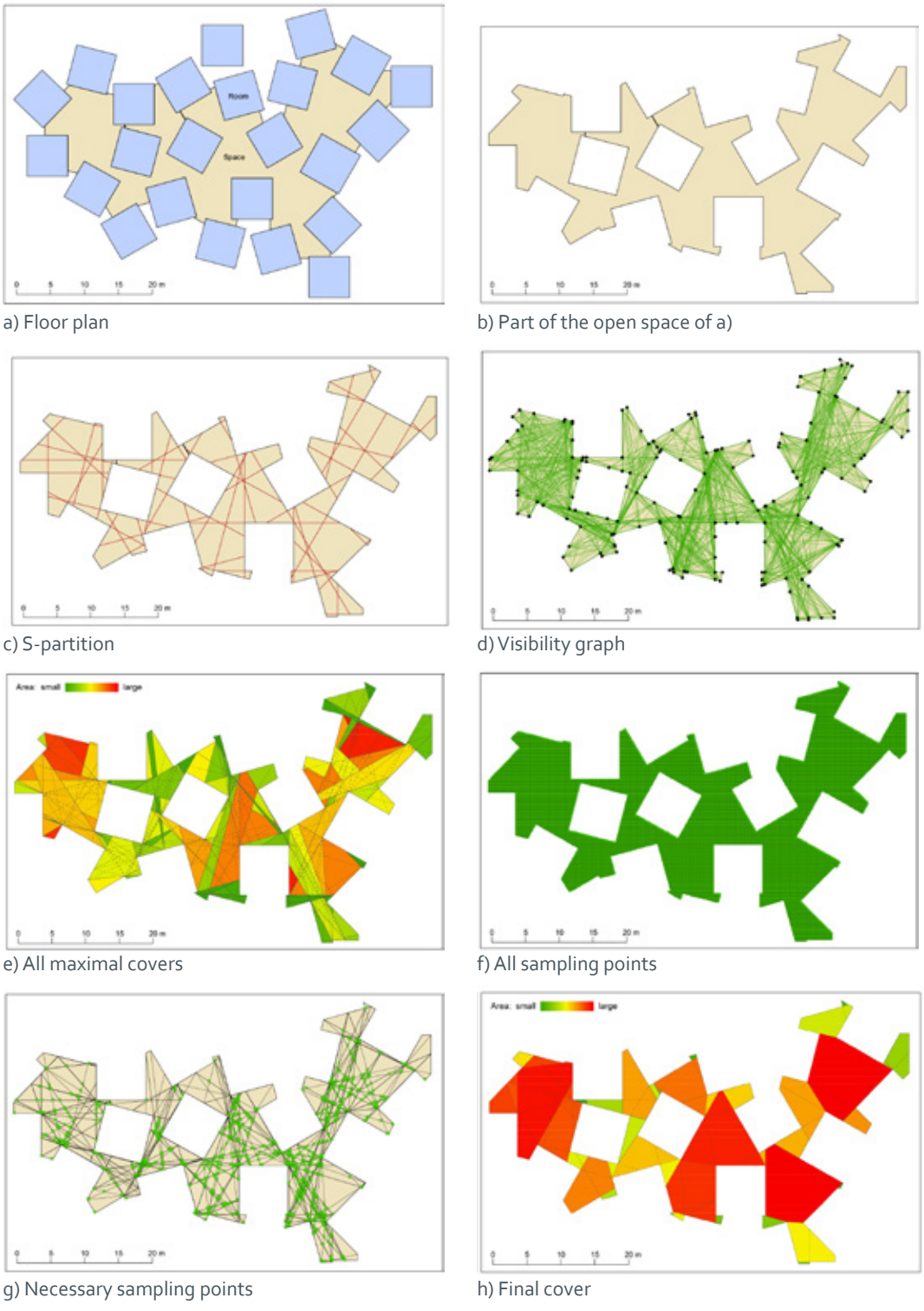
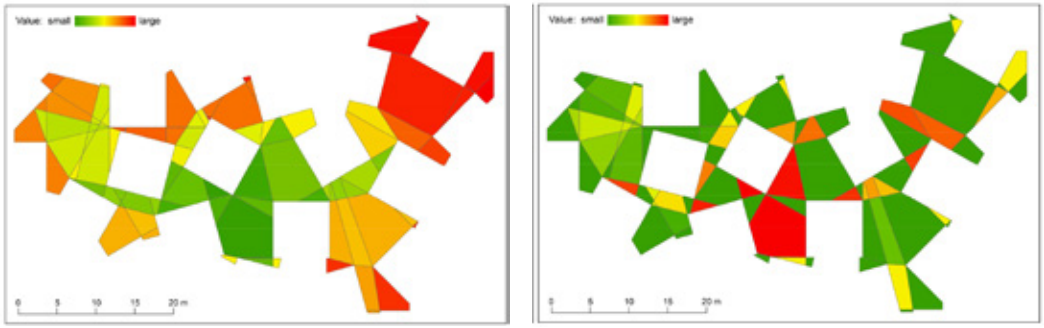


Figure 8 - Validation process of the proposed method with the target floor plan.

Figure 9 illustrates the graph characteristics proposed in 2.4 applied to the cover of Fig.8h. It is common for the average depth to be shallow in the central part of the space and deeper in the peripheral part, but when looking closely, there is a point where the monotonicity of the depth is reversed in the overlapping part of polygons. This is due to using hypergraph as described above. This tendency is more pronounced in betweenness centrality, and its value tends to be higher at the overlapped node than at the center of a large space.



a) Mean depth b) Betweenness centrality

Figure 9 - Distribution of the network measure in the final cover.

4. DISCUSSION

As explained earlier, since the network characteristics by the hypergraph is different from that of the usual graph, it is necessary to discuss how to evaluate it. Recent research on space syntax is increasingly distinguishing between indicators by visibility and movement. Because a hypergraph allows to include more than two nodes, it does not necessarily have to go through spatially adjacent nodes. Due to this nature, it can be said that hypergraph is a suitable method for modeling the network of visibility. In the first place, what does it mean to model the connection of space as a hypergraph? For example, an axial map models the line of sight as a node, regards the intersection with other lines as an edge of a binary relation, and segments the space as a normal graph. Therefore, the edge does not include information beyond the adjacency relation. On the other hand, the proposed method creates a hypergraph by partitioning covers into nodes and using the original cover as a hyperedge. Therefore, what corresponds to the line of sight of the axial map is modeled as a hyperedge rather than a node. Since hyperedges and nodes have geometric information, it can be said that the proposed method can graph the space preserving rich information, in a sense, than the axial map. In the future, it is necessary to verify with actual spatial cognition and behavior.

As another issue, since this research deals with enumeration problems, it is necessary to grasp to what extent the problem can be solved. The size of the polygon targeted in this study is about 100 nodes at most. However, for example, in the case of Gassin's data which is often used in the study of space syntax, the number of nodes exceeds 1000 points. When we enumerated it with this data, enumeration was impossible. In the case of problems of this scale, enumeration has to give up, but if we formalize as an aggregate covering problem and only find an optimal solution, using the current mathematical programming solver can solve it in a practical time sufficiently. In the future, it is necessary to investigate the calculation limit of the proposed method in detail.

5. CONCLUSIONS

In this research, we point out the problems of a kind of spatial analysis method that covers the space represented by Axial Map in the analysis methods of space syntax. As a new analytical method, we proposed algorithms of enumerating all covering patterns of space based on BDD / ZDD and algorithms of extracting optimal solutions. Then, we also proposed a method to obtain the network characteristics of the obtained covering based on the concept of hypergraph. As a result of verification of the proposed method with the floorplan of the short-term treatment facility where the number of nodes is about 100, it was possible to obtain enormous enumerated solutions and strict optimal solutions very fast with small memory. We found the network characteristics of the optimal solution and found that the value becomes relatively high mainly at the intersection of the cover. Future tasks include verification of the method in various spaces, understanding the spatial scale which is the calculation limit of the enumeration algorithm, development of new spatial features making full use of enumerated solutions, verification of spatial features by hypergraph etc.

ACKNOWLEDGEMENT

This study was supported by a Grant-in-Aid for Scientific Research (C) (No. 16K06652).

APPENDIX

The algorithms of Hit(p) and Min(q) proposed by Toda(2013) are listed below, where Lo(p) and Hi(p) returns Lo node and Hi node of node p respectively, V(p) returns the index of node p, and BBD_Unique(k, l, h) and ZDD_Unique(k, l, h) respectively return a unique BDD / ZDD node associated with a key(k, l, h) as illustrated in Figure 10, And(a, b)returns the product set of sets a and b, Dif f(a, b) returns the difference set obtained by subtracting b from a.

Algorithm 6: Given a ZDD p, compute the BDD for all hitting sets

```
1: function Hit(p)
2: if p=T_Z then return F_B;
3: if p=F_Z then return T_B;
4: hl ← Hit(Lo(p));
5: hh ← Hit(Hi(p));
6: t ← BBD_Unique (V(p), hh, T_B);
7: q ← And (hl, t);
8: return q;
```

Algorithm 7: Given a BDD q, compute the ZDD for all minimal sets

```
1: function Min(q)
2: if q=F_B then return F_Z;
3: if q=T_B then return T_Z;
4: mh ← Min(Hi(q));
5: ml ← Min(Lo(q));
6: t ← Dif f (mh, ml);
7: r ← ZDD_Unique (V(q), ml, t);
8: return r;
```

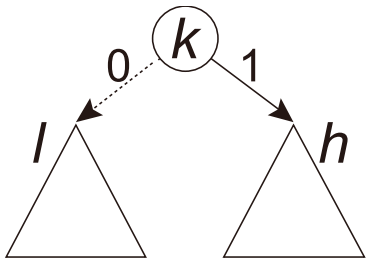


Figure 10 - Image of BDD_Unique (k, l, h) and ZDD_Unique (k, l, h).

REFERENCES

Batty, M., Rana, S. (2004), 'The automatic definition and generation of axial lines and axial maps', In *Environment and Planning B: Planning and Design,* 31, pp.615-640.

Benedikt, M. (1979), 'To take hold of space: isovists and isovist fields', *In Environment and Planning B*, Vol. 6, p.47-65.

Bryant, R. E. (1986), 'Graph-based algorithms for Boolean function manipulation', In *IEEE Transactions on Computers*, C-35(8), pp.677–691.

Coudert, O. (1997), 'Solving graph optimization problems with ZBDDs', In *Proceedings of European Design and Test Conference 1997*, pp.224-228.

Minato, S. (1993), 'Zero-Suppressed BDDs for Set Manipulation in Combinatorial Problems', In *Proc. of the 30th International Design Automation Conferenc*e, pp.272–277.

Hillier, B. and Hanson, J. (1984), *The Social Logic of Space*, London: Cambridge University Press.

Hillier, B. (2007), Space is the machine: a configurational theory of architecture, Space Syntax: London, UK. Green open access.

Inoue, T., Iwashita, H., Kawahara, J. and Minato, S. (2016), 'Graphillion: Software Library Designed for Very Large Sets of Labelled Graphs', In *International Journal on Software Tools for Technology Transfer*, 18(1), pp.57-66.

O'Rourke, J. and Supowit, K. (1983). 'Some NP-hard polygon decomposition problems', In *IEEE Transactions on Information Theory*, 29 (2), pp.181-190.

Peponis, J., Wineman, J., Rashid, M., Kim, S. H., Bafna, S. (1997), 'On the Description of Shape and Spatial Configuration inside Buildings: Convex Partitions and Their Local Properties', In *Environment and Planning B: Urban Analytics and City Science*, 24(5), pp.761-781.

Peponis, J., Wineman, J., Bafna, S., Rashid, M. and Kim, S. H. (1998), 'On the generation of linear representations of spatial configuration', *Environment and Planning B: Planning and Design*, 25, 559-576.

Toda, T. (2013), 'Hypergraph Transversal Computation with Binary Decision Diagrams', In *Proc. of 12th International Symposium on Experimental Algorithms* (SEA2013), pp.91-102.

Turner, A., Doxa, M., O'Sullivan, D. and Penn, A. (2001), 'From isovists to visibility graphs: a methodology for the analysis of architectural space, In *Environment and Planning B: Planning and Design*, 28, pp.103-121.

Turner, A., Penn, A. and Hillier, B. (2005), 'An algorithmic definition of the axial map', In *Environment and Planning B: Planning and Design*, 32, pp.425-444.