Symbolic OBDD Assembly Sequence Planning Algorithm Based on Unordered Partition with 2 Parts of a Positive Integer

Zhoubo Xu, Tianlong Gu, Rongsheng Dong

Guangxi Key Laboratory of Trusted Software, Guilin University of Electronic Technology, Guilin 541004, China
xzbli_11@guet.edu.cn,cctlgu@guet.edu.cn,ccrsdong@guet.edu.cn

Abstract. To improve solution efficiency and automation of assembly sequence planning, a symbolic ordered binary decision diagram (OBDD) technique for assembly sequence planning problem based on unordered partition with 2 parts of a positive integer is proposed. To convert the decomposition of assembly liaison graph into solving unordered partition with 2 parts of positive integer *N*, a transformation method from subassembly of the assembly to positive integer *N* is proposed, the judgment methods for the connectivity of a graph and geometrical feasibility of each decomposition combined with symbolic OBDD technique is proposed too, and all geometrically feasible assembly sequences are represented as OBDD-based AND/OR graph. Some applicable experiments show that the symbolic OBDD based algorithm can generate feasible assembly sequences correctly and completely.

Keywords: assembly sequence planning; automatic assembly planning; unordered partition; ordered binary decision diagram

1 Introduction

Assembly sequence planning is to find the feasible or optimal sequence that put the initially separated parts of an assembly together to form the assembled product, and always plays a key role in determining important characteristics of the tasks of assembly and of the finished assembly. Since 1984, a number of assembly sequence planning systems have been developed, such as precedence relations method [1], cut-set decomposition method [2] and soft computing technique [3], etc. Precedence relations method based on a set of questions whose answers lead to establishment conditions, from which one can reason the geometrically feasible assembly sequence. However, it will become a difficult and error-prone process for all but simplest assemblies. Cut-set decomposition method generates the assembly sequences by application of cut-set algorithm based on an assumption that the sequence of assembly is the reverse of disassembly sequence. This method suffers from the fact that there may be an exponential number of candidate cut-sets, and this may lead to the so-called combinatorial explosion, but it is easy to be programmed, so that it has been the most

commonly used method for generating assembly sequence. Soft computing technique can find good solutions quickly for complex products, but cannot generate feasible assembly sequences completely and cannot find optimal solutions.

Essentially, generating assembly sequences is one of typical combinatorial computing problem. It is well known that the number of feasible assembly sequences increases exponentially with the number of parts or components composing the whole products. In recent years, efficient symbolic algorithm based on ordered binary decision diagram (OBDD) or variant have been devised for hardware verification, model checking, testing and optimization of circuit [4,5], and have been demonstrated that the symbolic algorithms can deal with large scale problems which traditional algorithm can't solve. Recently, there has emerged a class of OBDD-based approaches in assembly sequence planning. Gu, Xu and Yang proposed symbolic OBDD representations for mechanical assembly sequences [6], the experimental results show that the storage space of OBDD based representation of all the feasible assembly sequences is less than that of AND/OR graph do. Gu and Liu developed an symbolic OBDD algorithm for assembly sequence planning which is limited to monotone linear assembly [7]. Gu and Xu presented a novel scheme to integrate constraint satisfaction problem model with OBDD for the assembly sequence planning, but the procedure consumes a lot of time with the problem of backtracking [8].

In this regard, this paper references the idea of cut-set decomposition method, and integrates unordered partition with 2 parts of a positive integer which is used to generate all feasible true cut-set with OBDD symbolic technique to generate assembly sequence. Based on symbolic OBDD assembly model of liaison graph and translation function, the one-to-one correspondence between the parts set of assembly $V=\{v_0,v_1,\ldots,v_{n-1}\}$and sets $W=\{2^0,2^1,\ldots,2^{n-1}\}$ is established firstly, so that any non-empty subset V' of V can be represented as a unique positive integer N ($1\le N\le 2^n-1$). After this, the decompositions of sub-graph of liaison graph with parts set V' can be enumerated by unordered partition with 2 parts of a positive integer, and the connectivity of each segment and geometrical feasibility corresponding to each of decompositions are checked by symbolic OBDD technique, and all geometrically feasible assembly sequences are represented by OBDD-based AND/OR graph. Finally, some applicable experiments show that the novel algorithm can generate feasible assembly sequences correctly and completely.

2 Formulating Assembly Knowledge via OBDD

Liaison graph is one of the role model for generating assembly sequences. Liaison graph is a undirected connected graph $G=<V,E>$, where V is a set of vertices that represent parts, and E is a set of edges that represent any of certain user-defined relations between parts called liaisons.

Given an assembly and its liaison graph $G=<V,E>$, we can convert the liaison graph to an OBDD by encoding the parts of the assembly with a length-l binary number, where $l=\lceil \log_2|V| \rceil$, each parts in V can be encoded as a vector of binary variables $\boldsymbol{x}=[x_0x_1\ldots x_{l-1}]$. Essentially, the liaisons are a relation on parts, so a liaison $(a,b)\in E$ can

be encoded as a vector of binary variables $[x_0x_1\ldots x_{l-1}y_0y_1\ldots y_{l-1}]$, where $[x_0x_1\ldots x_{l-1}]$ and $[y_0y_1\ldots y_{l-1}]$ are the encoded binary vectors corresponding to part *a* and *b* respectively. Hence, the liaison (*a*,*b*) can be represented as a Boolean characteristic function:

$$\xi(x_0, x_1, \ldots, x_{l-1}, y_0, y_1, \ldots, y_{l-1})=\alpha_0\cdot\alpha_1\cdot\ldots\cdot\alpha_i\cdot\ldots\cdot\alpha_{l-1}\cdot\beta_0\cdot\beta_1\cdot\ldots\cdot\beta_i\cdot\ldots\beta_{l-1}$$

where α_i is the appearance of variable x_i corresponding to i+1 bit code of part *a*, and β_i is the appearance of variable y_i corresponding to i+1 bit code of part *b*. If x_i=1, then $\alpha_i = x_i$, else $\alpha_i = x_i'$. If y_i=1, then $\beta_i = y_i$, else $\beta_i = y_i'$. Hence, the liaison graph *G* can be represented by the following characteristic function:

$$\phi_C(\boldsymbol{x}, \boldsymbol{y})=\sum \xi(x_0, x_1, \ldots, x_{l-1}, y_0, y_1, \ldots, y_{l-1})$$

where $\boldsymbol{x}=[x_0x_1\ldots x_{l-1}]$, $\boldsymbol{y}=[y_0y_1\ldots y_{l-1}]$.

The liaison graph provides only the necessary conditions but not sufficient to assembly two components. To be a feasible assembly operation, it is necessary that there is a collision-free path to assembly parts. Gottipolu and Ghosh [9] defined translational function to represent the relative motion between parts of assembly.

Translational function T: $P\times P\rightarrow\{0,1\}^6$, where P is a set of parts, $\{0,1\}^6$ is a 0-1 vector space with six dimension, each dimension correspond to the one of six directions of triorthogonal Cartesian coordinate system. Here, directions 1, 2, 3, 4, 5 and 6 indicate the six directions $+X$, $+Y$, $+Z$, $-X$, $-Y$, and $-Z$ of X, Y and Z axes respectively.

Let $(a, b)\in P\times P$, the correspond value of T function is 0-1 vector space with six dimension $(T_1(a,b), T_2(a,b), T_3(a,b), T_4(a,b), T_5(a,b), T_6(a,b))$, where $T_i(a,b)$=1 (i=1,2,…,6) if the part *b* has the freedom of translational motion with respect to the part *a* in the direction *i*, $T_i(a,b)$=0 if the part *b* has no freedom of translational motion with respect to the part *a* in the direction *i*.

For any part *a* and *b*, if $T_i(a,b)$=1, then it can also be represented by the Boolean function $\xi(x_0, x_1, \ldots, x_{l-1}, y_0, y_1, \ldots, y_{l-1})$. Hence, the *i*th component of translational function *T* can be represented by the following characteristic function:

$$\Phi_{T_i}(x_0, x_1, ..., x_{l-1}, y_0, y_1, ..., y_{l-1}) = \sum \xi(x_0, x_1, ..., x_{l-1}, y_0, y_1, ..., y_{l-1})$$

3 Formulating Subassembly via Integer

For an assembly with *n* parts, a subassembly of assembly can be characterized by its set of parts, and can be represented by an *n*-dimensional binary vector $[x_0x_1\ldots x_{n-1}]$ in which the *i*th component is 1 or 0, respectively, if the *i*th part is involved in the subassembly or not. So the assembly can be represented by full one binary vector.

Theorem 1 Let parts set $V=\{v_0,v_1,\ldots,v_{n-1}\}$, and a function $f(v_i)=2^i$ (i=0,1,…, n–1). If a function $g(x_0,x_1,\ldots,x_{n-1})=\sum_{x_i=1} f(v_i)$, where x_i=0 or 1, then any integer k ($1\le k\le 2^n-1$) is correspond to one and only one subassembly.

Proof. For any k, $1 \le k \le 2^n - 1$, there must exist an n-dimensional binary vector $[x_0 x_1 \ldots x_{n-1}]$ such that $\sum_{x_i=1} f(v_i) = k$. If $x_i=1$, it means that part v_i is involved in the subassembly, otherwise part v_i is not involved in the subassembly.

Assume such an n-dimensional binary vector $[x_0 x_1 \ldots x_{n-1}]$ is not unique, then there exist at least two n-dimensional binary vectors for the value k. Suppose that $[x_0 x_1 \ldots x_{n-1}]$ and $[y_0 y_1 \ldots y_{n-1}]$ are two n-dimensional binary vectors for the value k. if there are only $x_{i_1}, x_{i_2}, \ldots, x_{i_p}$ which have the value 1, where $0 \le i_1 \le i_2 \le \ldots \le i_p \le n-1$, then $g(x_0, x_1, \ldots x_{n-1}) = 2^{i_1} + 2^{i_2} + \ldots + 2^{i_p}$, namely, $k = 2^{i_1} + 2^{i_2} + \ldots + 2^{i_p}$. If there are only y_{j_1}, y_{j_2}, … , y_{j_q} which have the value 1, where $0 \le j_1 \le j_2 \le \ldots \le j_q \le n-1$, then $g(y_0, y_1, \ldots y_{n-1}) = 2^{j_1} + 2^{j_2} + \ldots + 2^{j_q}$, that is to say $k = 2^{j_1} + 2^{j_2} + \ldots + 2^{j_q}$. Thus $2^{i_1} + 2^{i_2} + \ldots + 2^{i_p} = 2^{j_1} + 2^{j_2} + \ldots + 2^{j_q}$.

① If k is odd, then it must have $i_1=0$, $j_1=0$, namely, $j_1=i_1$. Hence

$$2^{i_2} + 2^{i_3} + \ldots + 2^{i_p} = 2^{j_2} + 2^{j_3} + \ldots + 2^{j_q} \tag{1}$$

Let $l = \min\{i_2, i_3, \ldots, i_p, j_2, j_3, \ldots, j_q\}$, here might as well assume that $l=i_2$, extract lowest common multiple 2^i in both ends of equation (1), we have

$$2^{i_2}(1 + 2^{i_3 - i_2} + \cdots + 2^{i_p - i_2}) = 2^{i_2}(2^{j_2 - i_2} + 2^{j_3 - i_2} \cdots + 2^{j_q - i_2})$$

Hence

$$1 + 2^{i_3 - i_2} + \cdots + 2^{i_p - i_2} = 2^{j_2 - i_2} + 2^{j_3 - i_2} + \cdots + 2^{j_q - i_2} \tag{2}$$

It is evident that the left of the equation (2) is odd, so there must have $2^{j_2 - i_2} = 1$, that is $j_2 = i_2$. Therefore, the following equation holds:

$$2^{i_3 - i_2} + \cdots + 2^{i_p - i_2} = 2^{j_3 - i_2} + \cdots + 2^{j_q - i_2} \tag{3}$$

And it has $j_3 = i_3$ through the process of the proof similar to that of equation (1). For the same reason, there have $j_4 = i_4, \ldots, j_q = i_p$.

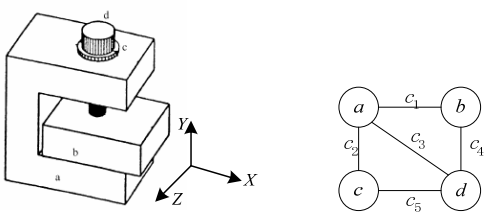
② If k is even, it can also proof that $j_1 = i_1, j_2 = i_2, \ldots, j_q = i_p$ through applying the similar process of the proof as equation (1). □

For example, the assembly shown in Fig.1 includes 4 parts. We assume that $f(a)=2^0$, $f(b)=2^1$, $f(c)=2^2$ and $f(d)=2^3$, then according to theorem 1, integer 7 can be represent the parts set $\{a, b, c\}$.

4 Generation of Assembly Sequences

Given an assembly and its liaison graph $G=<V, E>$, if a decomposition partition the graph G into two connected subgraph $G_1=<V_1, E_1>$ and $G_2=<V_2, E_2>$, where $V_1 \cap V_2 = \varnothing$, $V_1 \cup V_2 = V$, $E_1 \subseteq V_1 \times V_1$, $E_2 \subseteq V_2 \times V_2$, then this decomposition will be correspond to a

possible disassembly. In fact, the decomposition of a graph is to divide a set V with n parts into two non-intersect subset V_1 and V_2, such that $V_1 \cap V_2 = \varnothing$, $V_1 \cup V_2 = V$, and each of induced subgraph of subset V_1 and V_2 are connected.



(a) Assembly (b) Connection Graph

Fig. 1. A Simple Assembly

Unordered partition with 2 parts of positive integer N is corresponding to a possible disassembly operation. For the assembly shown in Fig.1, one of the unordered partition with 2 parts of integer 15 is 3 and 12, the corresponding disassemble operation is to disassemble assembly $\{a,b,c,d\}$ into subassembly $\{a,b\}$ and $\{c,d\}$. So, to find all possible disassembly operation is equal to find all unordered partition with 2 parts of integer N. But it need to check the feasibility of disassembly operation, which include two step, one step is check connectivity of each sub-graph after disassemble operation, the other step is check geometrical feasibility of two subassembly.

For an assembly has n parts, the assembly is correspond to integer 2^n-1. We assume unordered partition with 2 parts of positive integer 2^n-1 is p and q, p is correspond to the parts set V_1, and q is correspond to the parts set V_2. Judging the connectivity of induced sub-graph of pars set V_1 is as follows:

$$H_1(\boldsymbol{x},\boldsymbol{y})=V_1(\boldsymbol{x}) \wedge \phi_C(\boldsymbol{x},\boldsymbol{y}) \wedge V_1(\boldsymbol{y}) \tag{4}$$

$$New(\boldsymbol{x}) = \exists \boldsymbol{y}(New(\boldsymbol{y}) \wedge (H_1(\boldsymbol{x},\boldsymbol{y}) \vee H_1(\boldsymbol{y},\boldsymbol{x}))) \wedge \overline{Reached(\boldsymbol{x})} \tag{5}$$

$$Reached(\boldsymbol{x})=Reached(\boldsymbol{x}) \vee New(\boldsymbol{x}) \tag{6}$$

$New(\boldsymbol{x})$ and $Reached(\boldsymbol{x})$ are initialized as any part in $V_1(\boldsymbol{x})$. Equation (5) and equation (6) are iterative until $Reached(\boldsymbol{x})$ is not changed or equation $Reached(\boldsymbol{x})=V_1(\boldsymbol{x})$ is hold. If the equation $Reached(\boldsymbol{x})=V_1(\boldsymbol{x})$ is hold, then the induced sub-graph of pars set V_1 is connected, otherwise, it is not connected.

Judging the connectivity of induced sub-graph of parts set V_2 is same to V_1.

If both the induced sub-graph of parts set V_1 and V_2 are connected, checking the geometrical feasibility of subassembly V_1 and V_2 is to check whether the following equation is hold:

$$V_1(\boldsymbol{x}) \wedge V_2(\boldsymbol{y}) \wedge \Phi_{T_i}(\boldsymbol{x},\boldsymbol{y}) = V_1(\boldsymbol{x}) \wedge V_2(\boldsymbol{y}) \quad (i=1,2,...,6) \tag{7}$$

If for all i, equation (7) is not hold, then the assembly of subassembly V_1 to subassembly V_2 is not geometrically feasible, otherwise it is geometrically feasible, and insert the assemble operation into OBDD-based AND/OR graph $\Phi_S(\boldsymbol{x},\boldsymbol{y})$.

For example, the assembly shown in Fig.1 includes 4 parts. The pre-state and post-state of a disassembled operation can be characterized by $[x_0x_1x_2x_3]$ and $[y_0y_1y_2y_3]$ respectively, where x_0 and y_0, x_1 and y_1, x_2 and y_2, x_3 and y_3 are encoded binary number corresponding to part a, b, c and d, respectively. If x_i=1 then the corresponding part is involved in the subassembly, otherwise it is not involved. We still assume that $f(a)=2^0$, $f(b)=2^1$, $f(c)=2^2$ and $f(d)=2^3$, then the disassembled processes of the assembly are shown in table1.

Table 1. the disasssembled process of assembly shown in Fig.2

N	p	q	V_1	V_2	Disassembled Operation
15	3	12	$\{a, b\}$	$\{c, d\}$	$x_0'x_1'x_2x_3y_0y_1y_2y_3+$ $x_0x_1x_2'x_3'y_0y_1y_2y_3$
15	7	8	$\{a, b, c\}$	$\{d\}$	$x_0x_1x_2x_3'y_0y_1y_2y_3$
3	1	2	$\{a\}$	$\{b\}$	$x_0'x_1'x_2'x_3'$ $y_0y_1y_2'y_3'$
12	4	8	$\{c\}$	$\{d\}$	$x_0'x_1'x_2'x_3'y_0'y_1'y_2y_3$
7	2	5	$\{b\}$	$\{a, c\}$	$x_0x_1'x_2x_3'y_0y_1y_2y_3'$
7	3	4	$\{a, b\}$	$\{c\}$	$x_0x_1x_2'x_3'y_0y_1y_2y_3'$
5	1	4	$\{a\}$	$\{c\}$	$x_0'x_1'x_2'x_3'y_0y_1'y_2y_3'$

As shown in Table 1, the Boolean characteristic function of all feasible disassembled sequences of the assembly shown in Fig.1 can be formulated as:

$$\Phi_S(x,y)= x_0'x_1'x_2x_3y_0y_1y_2y_3+ x_0x_1x_2'x_3'y_0y_1y_2y_3+ x_0x_1x_2x_3'y_0y_1y_2y_3+ x_0'x_1'x_2'x_3'y_0y_1y_2'y_3'+ x_0'x_1'x_2'x_3'y_0'y_1'y_2y_3+ x_0x_1'x_2x_3'y_0y_1y_2y_3'+ x_0x_1x_2'x_3'y_0y_1y_2y_3'+ x_0'x_1'x_2'x_3'y_0y_1'y_2y_3'$$

Based on an assumption that the sequence of assembly is the reverse of that of disassembly, the OBDD representation of all feasible assembled sequences of the assembly are shown in Fig.2 (all the edges point to the terminal 0 which is omitted for clarity).

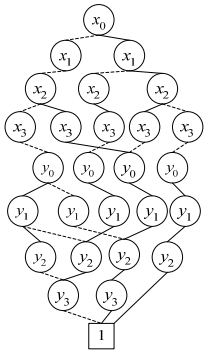


Fig. 2. OBDD Representation of Assembled Sequence of the Assembly Shown in Fig.1

5 Experiments

The symbolic algorithm proposed in this paper has been implemented in windows XP and the software package CUDD[10]. We have tested the performance of our algorithm on some practical assemblies, such as assembly shown in Fig.1, electrical controller(Fig.3) and Centrifugal Pump (Fig.4). Geometrically feasible assembly sequences of the assemblies were generated by the algorithm using Microsoft Visual C++. Experiments were performed on a P4 3GHZ with 512MB of memory. For the assembly shown in Fig.1, the OBDD for the product contains 28 OBDD node and 7 hyperarcs, and the CPU time is 0.015 seconds for generating all assembly sequences. In electrical controll case, the OBDD for the product contains 21442 OBDD node, and the CPU time is 996.437 seconds for generating all assembly sequences. In centrifugal Pump case, the OBDD for the product contains 3850 OBDD node, and the CPU time is 6280.437 seconds for generating all assembly sequences. For the centrifugal pump, it can not be solved in 10 hours by tranditional cut-set decomposition in the same enviorment, it shows that the symbolic algorithm is better than the tranditional algorithm which is attribute to the symbolic OBDD implicit operation.

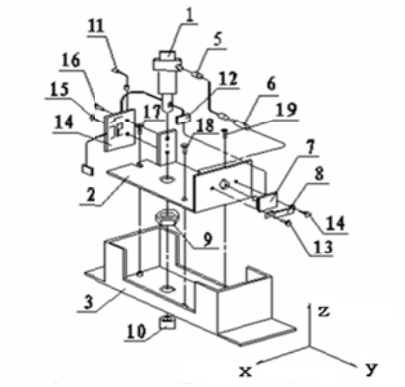


Fig. 3. An Electrical Controller

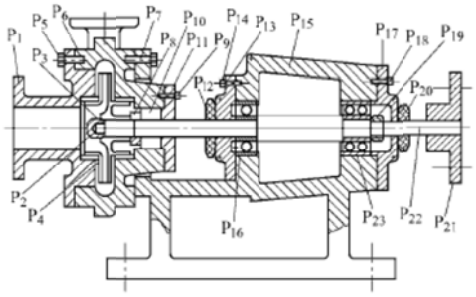


Fig. 4. A Centrifugal Pump

6 Conclusions

In this paper, the one-to-one correspondence between subassembly and integer is established, and a symbolic OBDD algorithm for generating mechanical assembly sequence is presented based on unordered partition with 2 parts of a positive integer method. Judging the connectivity of a graph and checking the geometrical feasibility of a subassembly are both implemented by implicitly symbolic OBDD manipulation. Some applicable experiments show that the novel algorithm can generate feasible assembly sequences correctly and completely.

Acknowledgements

This work hav been supported by National Natural Science Foundation of China (61100025,60963010,61063002) and Science Foundation of Guangxi Key Laboratory of Trusted Software (No. kx201119).

References

1. Fazio T L De, Whitney D E. Simplified generation of all mechanical assembly sequences. IEEE Journal of Robotics and Automation, 1987,3(6):640-658
2. Homen de Mello L S, Sanderson A C. A correct and complete algorithm for the generation of mechanical assembly sequences. IEEE Transactions on Robotics and Automaition, 1991, 7(2):228-240
3. Wang J F, Liu J H, Zhong Y F. A novel ant colony algorithm for assembly sequence planning. International Journal of Advanced Manufacturing Technology, 2005, 25(11): 1137-1143
4. Bryant R E. Symbolic boolean manipulation with ordered binary decision diagrams. ACM Computing Surverys,1992,24(3):293-318
5. Burch J, Clarke E, Mcmillan K, et al. Symbolic model checking:10^{20} states and beyond. Information and Computation, 1998, 98(2):142-170
6. Gu T L, Xu Z B, Yang Z F. Symbolic OBDD representations for mechanical assembly sequences . Computer-Aided Design, 2008, 40(4):411-421
7. Gu T L, Liu H D. The symoblic OBDD scheme for generating mechanical assembly sequences. Formal Methods in System Design, 2008, 33(1/3): 29-44
8. Xu Z B, Gu T L. A constraint satisfaction problem model and its symbolic OBDD solving for assembly sequence planning problem. Journal of Computer-Aided Design & Computer Graphics, 2010, 22(5): 803-810
9. Gottipolu R B, Ghosh K. A simplified and efficient representation for evaluation and selection of assembly sequences. Computer in Industry, 2003, 50(3):251-264
10. SOMENZI F. CUDD: CU decision diagram package release 2.3.1. http://vlsi. Colorado.edu/ fabio/ CUDD/cuddIntro.html