Int J Adv Manuf Technol (2007) 32: 1232–1244
DOI 10.1007/s00170-006-0438-1

ORIGINAL ARTICLE

Tianyang Dong · Ruofeng Tong ·
Ling Zhang · Jinxiang Dong

A knowledge-based approach to assembly sequence planning

Received: 22 July 2004 / Accepted: 31 October 2005 / Published online: 30 May 2006

Abstract Assembly planning plays a major role in the manufacturing industry. In order to reduce computational complexity, this paper presents a knowledge-based approach to the assembly sequence planning problem. The CSBAT (connection-semantics-based assembly tree) hierarchy proposed in this paper provides an appropriate way to consider both geometric information and non-geometric knowledge. In this research, the typical or standard CSBAT is applied to a given assembly problem. The structure of the KBASP (knowledge-based assembly sequence planning system) is proposed and there are different ways to construct plans for a CSBAT: by retrieving the typical base, by retrieving the standard base, and by geometric reasoning. The approach proposed in this paper can generate assembly sequences for each CSBAT directly, without the problem of merging plans for different child CSBATs. The application shows that the knowledge-based approach can reduce the computational complexity drastically and obtain more feasible and practical plans.

Keywords Assembly planning · Connection semantics · Graph matching · Knowledge base · Assembly model · Sequence planning

T. Dong (✉)
College of Software Engineering,
Zhejiang University of Technology,
Hangzhou, 310032, People's Republic of China
e-mail: dty@zjut.edu.cn
Tel.: +86-571-85290034

R. Tong · J. Dong
Institute of Artificial Intelligence,
Zhejiang University,
Hangzhou, 310027, People's Republic of China

L. Zhang
School of Economics and Management,
Zhejiang University of Science and Technology,
Hangzhou, 310012, People's Republic of China

1 Introduction

Assembly planning is a large scale and highly constrained combinatorial problem, encompassing assembly sequence planning and assembly task planning. Assembly planning plays a major role in the manufacturing industry. This automation constitutes one of the most important conditions to guarantee the future competitiveness of industrial companies. In the last decade, several approaches have been proposed to generate assembly sequences automatically. In summary, the existing approaches to the generation of assembly plans can be roughly classified into three main approaches: human interaction, geometry-based reasoning, and knowledge-based reasoning.

The method of human interaction mainly focuses on each user's query, either on the connection between a pair of parts or the feasibility of a single assembly operation [1, 2]. Clearly, this method is far from the goal of automation. Thereafter, a number of geometry-based reasoning approaches have been proposed. One general approach is the cut-set method by many researchers [3, 4]. The cut-set method follows the compute-and-test scheme, where all possible ways to partition an assembly into two connected subassemblies are generated, and each partition is tested for local freedom and global freedom using geometric reasoning. The other approach of geometry-based reasoning is the compute-and-generate strategy [5–7], where assembly motions are parameterized and block relations are derived to state which parts collide with other parts.

For the sake of generating good assembly plans, non-geometric assembly data, besides geometric assembly data, should also be used in assembly planning. There have been several approaches that generate assembly sequences by using high-level expert knowledge or experience. Chakrabarty and Wolter describe a planner that uses the structure both as a framework for structure-dependent definitions of good plans, as well as a tool for finding good plans more rapidly by reusing sub-plans for repeated substructures [8]. Swaminathan and Barber developed an experience-based assembly sequence planner for mechanical assemblies [9]. This approach utilizes case-based planning to store, retrieve, and modify existing cases or experience to generate assembly sequences. Yin et al. proposed a connector-based hierarchy approach that also seeks a plan-reuse-oriented solution to assembly planning based on the hierarchy description [10]. Fan [11] proposed a knowledge-based virtual assembly system which focused on the generation, selection, evaluation, and planning for the assembly. Marian et al. proposed some approaches based on genetic algorithms or neutral networks for assembly planning [12–14]. There are other approaches using assembly knowledge or artificial intelligence [15, 16].

Compared with the existing systems based on the reuse philosophy, the approach presented in this paper is to generate the feasible plans for assembly directly, and avoid the sophisticated merge of plans for substructures. The CSBAT (connection-semantics-based assembly tree) hierarchy proposed in this paper provides an appropriate way to consider both geometric information and non-geometric knowledge. By integrating geometry-based reasoning with knowledge-based reasoning, the computational complexity is reduced drastically, and the assembly sequences obtained are more feasible and practical.

The rest of this paper is organized as follows: Section 2 considers the representation problem of the assembly model. The strategy of assembly sequence planning is described in Section 3. Section 4 provides an example to illustrate the knowledge-based approach to assembly sequence planning. Conclusions and areas for future research are finally discussed in Section 5.

2 Assembly modeling

2.1 Connection-semantics-based assembly tree

A connector provides constraints on its jointed components to ensure that these components perform the required functions. Therefore, the connection is not only a thinking module to construct product design, but is also the core block to provide the components restriction. Connections provide some significant relationships among two or more assembled parts and can act as a foundation of assembly clustering.

In this research, connection semantics is used to represent each connection. Connection semantics is denoted as *Connect-Type* $(C)[A]$, where C is the set of connectors, A is the set of all parts constrained by connector C, and *Connect-Type* represents the connection type, including: *Bolt–Nut*, *Screw*, *Pin*, *Key*, *Roll-Fit*, *Mate*, *Insert*, etc. [17, 18].

It is noted that the connector parameter in some connection types is a set of connectors, namely, a group of several connectors. Two connections *Connect-Type*1 $(C_1)[A_1]$ and *Connect-Type*2 $(C_2)[A_2]$ satisfy the grouping relation if: (1) A_1=A_2; and (2) C_1 and C_2 can be assembled and removed only in the same direction, but in any order with respect to each other.

In order to access and interpret assembly easily, the assembly structure based on connection semantics can be represented as a CSBAT, denoted as T=(V, E). The CSBAT is one which has a distinguished node, called the root. The level of a node u in the CSBAT, denoted as $lev(u)$, is the length of the path connecting the root to u. If u is adjacent to v (denoted as u~v) and $lev(v)-lev(u)=1$, we say that u is the parent of v, and, conversely, v is the child of u. A CSBAT is a non-null rooted directed tree, in which each internal node represents the connection semantics, and each leaf node is a mechanical part. The connection corresponding to the root is called the root connection of a CSBAT. The CSBAT structure itself shows the assembly hierarchy, independent subassemblies, and sequences in the assembly. Figure 1 shows an example of a CSBAT. If the children of a CSBAT are leaf nodes, the CSBAT is called a primitive CSBAT. Therefore, the CSBAT shown in Fig. 1b is a primitive CSBAT.

2.2 Connection-semantics-based assembly relational model

The effectiveness of an assembly planner relies heavily on the input of the assembly representation [19]. To enable automated assembly sequence planning, all of the related information should be organized and represented as an assembly model. The assembly sequence planning problem is, essentially, a geometrical one, the assembly representation applied in this research will emphasize the geometric information, such as the shapes of the parts, their positions, and the contacts between the parts. A purely geometric description of the assembly cannot always generate a good assembly sequence. Some sequences may be feasible from a geometric point of view, but are impractical due to the special properties of some connections. Furthermore, the inclusion of non-geometric information helps to reduce the explosion of possible solutions. The assembly representation used in this research is the CSBARM (connection-semantics-based assembly relational model) that integrates both geometric and non-geometric assembly data [17, 18].

The CSBARM of an assembly is an undirected graph that includes two types of nodes: parts and connectors. Each node has its own attributes. The relationship between the nodes describes the connection of the parts. Essentially, the CSBARM is similar to the CBRM (connection-based relational model) used in the connector-based hierarchy approach [10], but some important changes have been

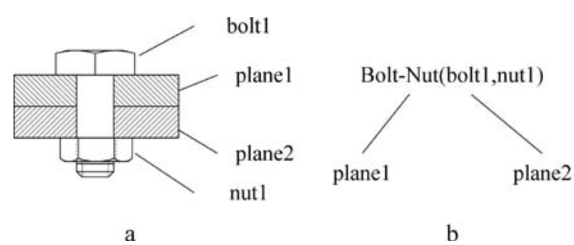


Fig. 1 **a** An example assembly. **b** Its CSBAT

made to support the planning strategy proposed in this research. The CSBARM for assembly can be denoted as <*P*, *C*, *M*>, where [17, 18]:

- *P* is a set of nodes, each of which corresponds to a part not belonging to the connectors in the assembly
- *C* is a set of nodes, each of which corresponds to a connector in the assembly
- *M* is a set of liaisons between two nodes, each of which corresponds to the connection between pairs of elements of $P \cup C$

The most important information obtained from all mating features is the degrees of freedom (DOF) of the mating entities. It is very important to match the CSBATs and CSBARMs, and retrieve the typical or standard plans from the plan base. To accomplish this, the mating feature is represented by a simple 3×4 matrix [7]. The elements represent the degree of freedom on the three major axes in 3D space. The configuration space for an assembly model typically has 3DOF in translation and 3DOF in rotation. The configuration space is usually subdivided into positive and negative directions, and is represented by individual elements in the matrix. This leads to the following mating matrix $\begin{bmatrix} x & -x & wx & -wx \\ y & -y & wy & -wy \\ z & -z & wz & -wz \end{bmatrix}$, where $\pm x$, $\pm y$, and $\pm z$ are linear translations and $\pm wx$, $\pm wy$, and $\pm wz$ are the rotations about the *X*, *Y*, and *Z* axes, respectively. The values of the elements in the mating matrices are either 0 or 1. Integer 1 indicates freedom of motion in the direction along the corresponding principal axis. Integer 0 indicates the motion is disallowed in the axial direction. If the degrees of freedom of the mating entities are constrained by two or more mating features, there will be a single mating matrix. In the mating matrix, the male part of the mating entities is the moving member and the female part is the fixed member [18].

2.3 Plan representation

The representation problem for assembly plans has received some attention due to the requirement of less storage and easy user understanding. The assembly sequence has been represented by AND/OR graphs [3], assembly trees [6], and assembly precedence graphs [9]. In this research, assembly precedence graphs (APG) are employed for plan representation. Figure 2 shows an example of an APG. It is noted that the APG may represent several feasible sequences to build an assembly. For example, the APG shown in Fig. 2 denotes two feasible sequences.

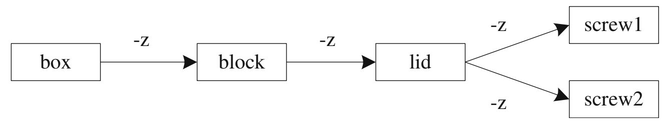


Fig. 2 Assembly precedence graphs (APG) of the container assembly

3 Knowledge-based approach to assembly sequence planning

The system structure of the KBASP (knowledge-based approach to assembly sequence planning) is shown in Fig. 3. The utilization of knowledge stored in the knowledge base is fundamental to the approach. Compared with other systems based on the reuse philosophy, the KBASP can generate the feasible plans for assembly directly, and does not need to merge plans for CSBATs. The KBASP is mainly composed of the following modules [17].

3.1 CSBAT generator

The reuse of stored plans in the knowledge base is fundamental to the proposed approach to assembly planning. If an assembly is represented as a CSBAT, then the plans can be generated by retrieving the knowledge base. There may be more than one CSBAT for an assembly. Therefore, it is necessary to select a preferred CSBAT out of multiple CSBATs that can be used to reuse stored plans in the knowledge base. In KBASP, a CSBAT hierarchy for an assembly is automatically derived from its CSBARM by geometric reasoning and knowledge-based reasoning according to some heuristic rules. The algorithms of generation for a CSBAT hierarchy will be discussed in Section 3.1 in detail.

3.2 Plan generator

This phase involves the searching of the knowledge base to find a match for the CSBAT provided to the *assembly sequence planning* module. If similar or the same CSBATs in the knowledge base do not give rise to useful plans for the provided CSBAT, the system will generate plans by geometric reasoning. In the KBASP, there are three ways to find plans for a CSBAT, which are attempted in the following order: (1) by retrieving the typical base; (2) by retrieving the standard base; and (3) by geometric reasoning. The plans obtained from the plan base are expressed in terms of part names that act as placeholders for actual parts. To make the plans useful, the dummy part names are converted to reflect the part names from the problem.

3.2.1 Typical plan retrieval and modification

It is generally desirable that similar or the same structure in all assemblies should be built in the same way during the assembly process, since this generally requires a smaller

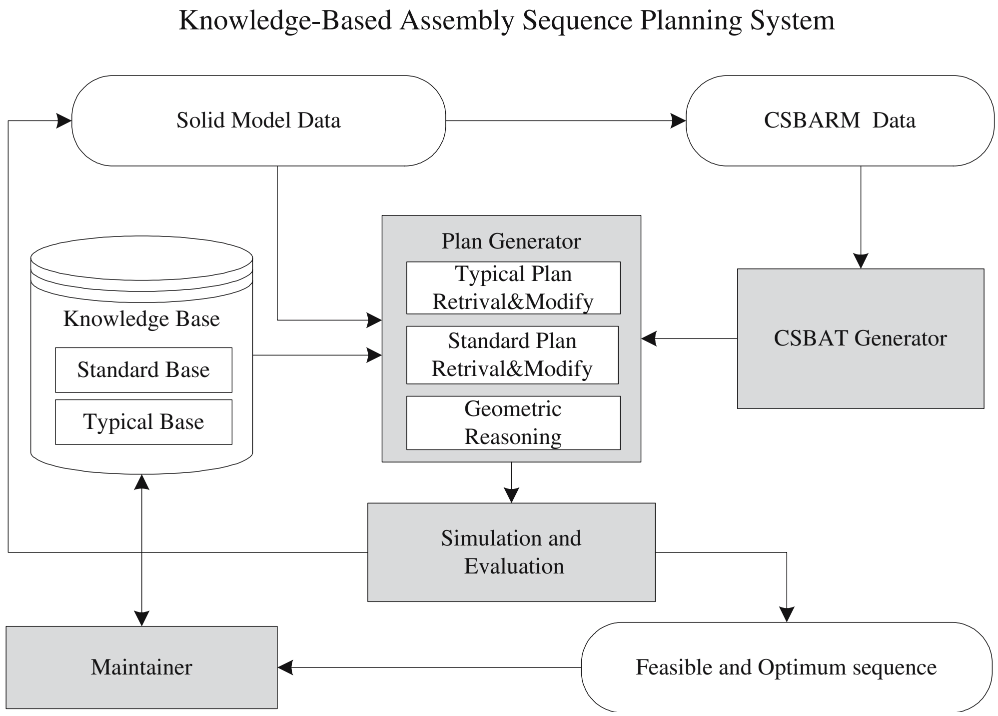


Fig. 3 System structure of KBASP (knowledge-based approach to assembly sequence planning)

variety of operations. Moreover, retrieving from the typical plan base can speed up the planning process, along with better plans being produced. Therefore, the plans of the typical assemblies are stored in the typical plan base for reuse. The typical plan base consists of the knowledge of CSBATs that are typical assemblies or subassemblies in the enterprise. For example, the motorcycle engine is a typical assembly in the motorcycle enterprise. Therefore, the assembly plans for the CSBAT of a motorcycle engine can be stored in the typical plan base for reuse. To support the reuse of plans for typical CSBATs, all kinds of knowledge is stored for each CSBAT in the typical plan base. There may be one or several assembly sequences corresponding to each typical CSBAT.

3.2.2 Standard plan retrieval and modification

If similar or the same CSBATs in the typical plan base do not give rise to useful plans for the provided CSBAT, the system next retrieves the stored plans from the standard plan base. For most primitive CSBATs, there are common and preferred procedures to assemble them. What's more, these assembly processes for primitive CSBATs are invariable. For instance, there are standardized procedures to install a ball bearing type of CSBAT. During assembly planning, plans for a primitive CSBAT are obtained by retrieving suggested plans from the standard plan base, instead of by reasoning about the low-level interactions among the parts of the CSBAT. By doing so, not only can a great deal of computation can be avoided, but also, better plans are obtained by integrating manufacturing experience with building CSBATs. The standard plan base consists of primitive CSBATs that are indexed by the types of their connections.

3.2.3 Geometric reasoning

If similar or the same CSBATs in the typical and standard bases do not give rise to useful plans for the provided CSBAT, the system next generates feasible plans for the CSBAT by geometric reasoning. Each child CSBAT of the provided CSBAT acting as a subassembly is called a super-part; therefore, the combinatorial explosion problem encountered in most geometric reasoning approaches is alleviated. As the parts of a CSBAT are assumed to come into contact with each other, the geometrically feasible plans are generated mainly by reasoning about the mating directions of each part in the CSBAT. In addition, directed-connector knowledge is exploited for the generation of the assembly precedence constraint graph. By integrating geometry-based reasoning with knowledge-based reasoning, the computation complexity is reduced drastically and the assembly sequences obtained are more feasible and practical.

3.3 Simulation and evaluation

After the *assembly sequence planning* stage, the feasible and practical solutions are found. At this stage, the user can simulate the assembly or disassembly process in the virtual environment. The main benefits of the *simulation and evaluation* stage are discussed as the following: (1) the *simulation and evaluation* module can provide tested and valuable information that might otherwise have required time-consuming and expensive physical experimentation; (2) it also assists in training assembly operators using virtual machines and virtual workpieces; (3) it actually controls and runs the operation of the real assembly processes through the manipulation of the virtual objects in the virtual environment. In addition, all feasible solutions are compared with each other at this stage. According to the proposed criteria, such as the assembling time or cost, the optimum assembly sequence is selected.

3.4 Maintainer

The *maintainer* unit decides whether the newly generated plans should be stored to the knowledge base for reuse. The user can tell the *maintainer* to store the typical CSBAT in the typical plan base if it is a typical subassembly. If the plans of a primitive connection type do not exist in the standard plan base, the *maintainer* module also stores the knowledge in the standard plan base. In the current implementation, the module is not fully featured. However, it is designed to be used in future developments.

3.5 Algorithm for the generation of a CSBAT

Each CSBAT is classified by the connection types stated in Section 2. As stated before, there may be more than one CSBAT for an assembly. Therefore, it is necessary to select a preferred CSBAT out of multiple CSBATs that can be used to reuse stored plans in the knowledge base. The preferred CSBAT can be selected by evaluating tentative CSBATs based on selection indices (*SI*s). The *SI* evaluates a cluster of parts in the CSBAT based on the following criteria [17, 18]:

1. Stability index (STI) The stability index, *STI* (*T*), of a subassembly *T* represents how stable all child CSBATs of *T* remain during the disassembly operation of the connectors from *T*. If the child CSBATs are not stably fastened, the parts may deviate from their correct position, and the operator has to take more time to reposition them. The index *STI* is defined as follows:

$$STI(T) = \sum_{P_m \subset subtree(T)} \sum_{P_j \in P_m} F_{st}\left(stb\left(P_j\right)\right) \tag{1}$$

where *subtree*(*T*) is a set of all child CSBATs of *T*. In this research, $F_{st}()$ maps the stability flag of a part into the time of basic motions: *reach*, *grasp*, *move*, *position*, *release*, and *reach*, as proposed by Kanai et al. [20]. The time of *grasp*, *position*, and *release* changes depending on the DOF of the part. The DOF of the part without considering the bilateral constraints is denoted as *df1* and the DOF considering the bilateral constraints is *df2*. The maximum of *df1* is 6 and that of *df2* is 12. So each basic motion *x* can be set a standard time T_x. $F_{st}()$ is defined by Eq. 2:

$$F_{st}\left(stb\left(P_j\right)\right) = \begin{cases} 0 & \left(stb\left(P_j\right)\right) = stable \\ W_{rep}C_{dev} & \left(stb\left(P_j\right)\right) = partly\_stable \\ W_{spt} & \left(stb\left(P_j\right)\right) = unstable \end{cases} \tag{2}$$

where

$W_{rep} = T_{reach} + T_{move} + W_{grasp} + W_{position} + W_{release}$, $W_{release} = \begin{cases} 0 & (df1 = 0) \\ T_{release} & (df1 \neq 0) \end{cases}$,

$W_{grasp} = \begin{cases} 0 & (df1 = 0) \\ T_{grasp} & (df1 \neq 0) \end{cases}$, $W_{position} = \begin{cases} 0 & (df1 = 0) \\ T_{position}/2 & (df1 = 1) \\ T_{position} & (df1 \geq 2) \end{cases}$, $C_{dev} = df2/12$,

and W_{spt} is the extra time required to support the unstable part by hand or using a jig.

2. Operation preference index (PRI) The operation preference index indicates the priority of the connection type. As we know, an assembly may have several CSBATs of different connection types. However, the operation complexity of each connection type is different. For instance, the CSBAT of a *Screw* type connection can be disassembled or assembled more easily than the CSBAT of the *Rivet* type. The index *PRI* is determined by the connection types as shown in Table 1.

3. Operation continuity index (CNI) The operation continuity index indicates how much extra motion the operator has to make for exchanging the connection types, mating directions, and tools. The index *CNI* is defined by Eq. 3:

$$CNI = k_1 \times TPI + k_2 \times DRI + k_3 \times TLI \tag{3}$$

where k_1, k_2, and k_3 are the coefficients and $k_1+k_2+k_3=1$. The *TPI* indicates the change of the connection type; if the CSBAT has a different root connection type from its parent CSBAT, the *TPI*=1, else, the *TPI*=0. The *DRI* indicates the change of the mating directions. If the mating direction of the CSBAT is different from that of its parent CSBAT connection, the value of *DRI*=(*the angle of direction change*)/90. That's to say, if the mating direction of the CSBAT is same as that of its parent CSBAT, the value of *DRI* is 0. The *TLI* depends on whether its own tools are same as that of its parent CSBAT. If the tools are different, the value of *TLI* is 1, else, the value is 0.

Table 1 Operation preference index (*PRI*)

Connection type	*PRI*
Mate, *Insert*	0.3
Bolt, *Bolt–Nut*, *Screw*, *Pin*	0.5
Key, *Roll-Fit*, *Gear*, *Belt-Mesh*, *Bearing*	0.6
Rivet, *Welding*	0.8

4. Parallelism index (PI) The parallelism of a CSBAT can be measured approximately by the number of the connectors which make up the connection and the number of parts in each child CSBAT. The index *PI* is defined by Eq. 4:

$$PI = k_1 \times CI - k_2 \times SPI = k_1 \times \frac{N_c}{N_{all}} \times \sum_{i=1}^{m-1} \sum_{j=i+1}^{m} \left| N_i - N_j \right| \tag{4}$$

where k_1 and k_2 are the coefficients and k_1+k_2=1, N_{all} is the number of all parts in the CSBAT, N_{c} is the number of connectors making the connection, and N_i is the number of parts in the child CSBAT *i*. The higher the value of *CI* implies the greater the number of the connectors which make up the connection. The lower the value of *SPI* implies the more operations can be done in parallel for different child CSBATs. However, the parallelism index should be carefully weighted in CSBAT generation, since the increase of parallelism to reduce the assembly time may result in excessive cost in part/subassembly transfer, manipulation, and assembly layout, etc.

5. Selection index (SI) The selection index is given as follows: $SI = e^{(-k_1 \cdot STI - k_2 \cdot PRI - k_3 \cdot CNI + k_4 \cdot PI)}$, where k_1, k_2, k_3, and k_4 are the assembly coefficients and k_1+k_2+k_3+k_4=1. The CSBAT that has the highest *SI* value is selected as the candidate CSBAT. The system prefers to select the CSBAT with smaller *STI*, *PRI*, and *CNI* values, but larger *PI* values. The coefficients can be assigned by the designer, based on the relative significance of each selection index on the overall assembly cost. For instance, if the system selects the CSBAT with the highest priority which requires more parallel operations, k_4 is set to the larger value compared with the other coefficients. In addition to the selection indices proposed in this paper, other indices can be incorporated into the *SI* determination for further improvement [6].

3.6 Typical plans retrieval and modification

After the generation of the CSBAT and CSBARM, the problem of typical plans retrieval from the typical plan base turns into the problem of matching the CSBAT and CSBARM. That's to say, the typical plans retrieval is a problem of graph matching. Plenty of algorithms for graph matching have been proposed with the specific aim of reducing the computational complexity of the matching

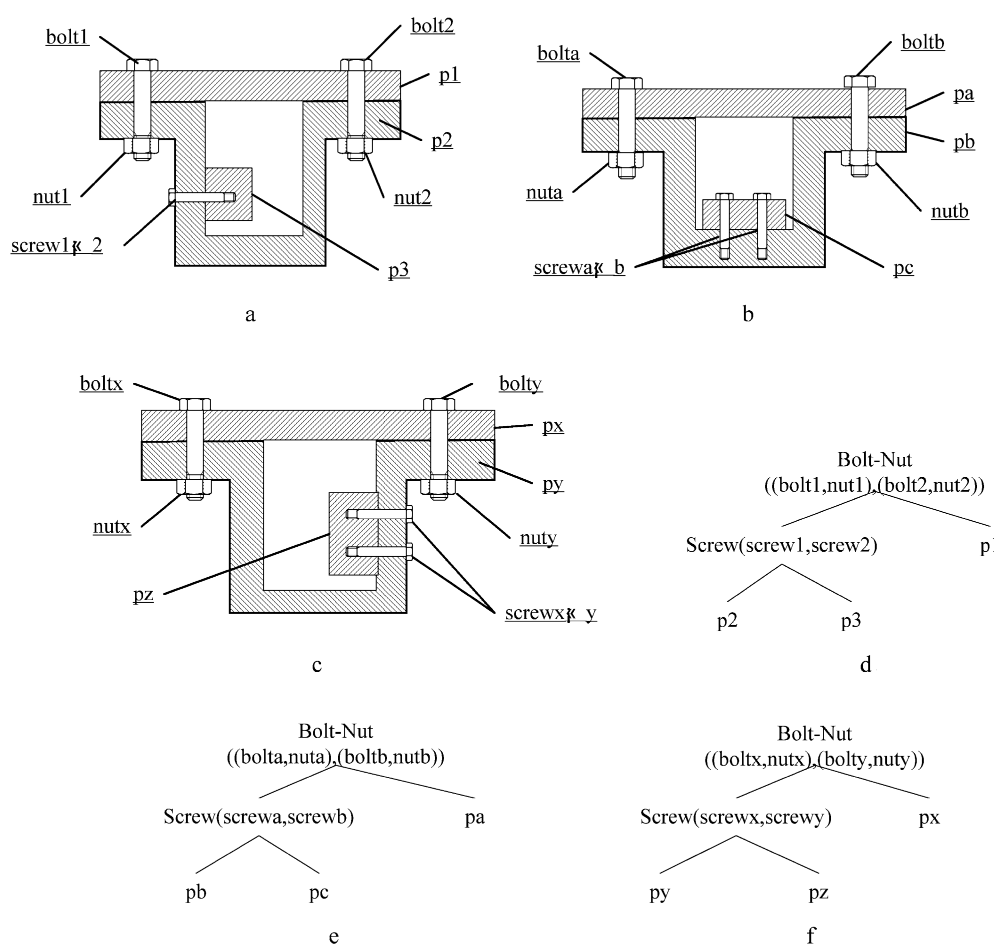


Fig. 4 **a** Provided assembly. **b**, **c** Two cases in the typical base. **d**, **e**, **f** Their CSBATs, respectively

algorithms [21–26]. These approaches are mainly applied in the areas of pattern recognition and machine vision. They include the recognition of graphic symbols, character recognition, shape analysis, video indexing, and object recognition, etc. Because of the complexity of the assembly relational model, graph matching in assembly planning is very difficult. In this paper, a novel approach to graph matching is proposed. We use a so-called partial assembly constrain satisfying strategy to dynamically prune improper typical CSBATs, in which two assemblies or subassemblies are determined to be unmatched, without the necessity to check their details. The partial assembly constrain satisfying strategy is implemented by the candidate CSBAT and the target CSBAT. Therefore, we introduce the notions of candidate CSBAT and target CSBAT.

Definition 1 Let $T_0=(V_0, E_0)$ and $T_1=(V_1, E_1)$ be two CSBATs. T_0 is the CSBAT of a typical assembly stored in the typical plan base and T_1 is one of the subassemblies of the provided assembly. Given $u, v \in V_0$ and any bijection ϕ: $V_0 \rightarrow V_1$, T_0 is called the candidate CSBAT of T_1, if ϕ satisfies the following conditions simultaneously:

(a) If $u \sim v$, then $\phi(u) \sim \phi(v)$, and if u is the parent of v, then $\phi(u)$ is the parent of $\phi(v)$
(b) If u is a leaf node, then u and $\phi(u)$ must have the same part type, else u and $\phi(u)$ have the same connection type

Definition 2 Let $T_0=(V_0, E_0)$ be the candidate CSBAT of $T_1=(V_1, E_1)$ and $A_0=(P_0, C_0, M_0)$, and $A_1=(P_1, C_1, M_1)$ be the CSBARMs of T_0 and T_1, respectively. Given $u, v \in (P_0 \cup C_0)$, and bijection ϕ: $(P_0 \cup C_0) \rightarrow (P_1 \cup C_1)$, any transformation (a series of translations and rotations) f: $M_0 \rightarrow M_1$, T_0 is called the target CSBAT of T_1 if ϕ, f satisfies the following conditions simultaneously:

(a) If there are contacts between u and v, then there must be the same mating type and connection type between $\phi(u)$ and $\phi(v)$ as those of u and v
(b) If $d(u, v)$ is the mating direction of u and v, then $d(\phi(u), \phi(v))=f(d(u, v))$
(c) If $m(u, v)$ is the mating feature of u and v, then $m(\phi(u), \phi(v))=f(m(u, v))$

Figure 4 shows an example of retrieving plans from the typical plan base. Figure 4a is the provided assembly and Fig. 4b,c are two cases stored in the base. According to definition 1 and definition 2, we know that the CSBATs shown in Fig. 4e,f are the candidate CSBATs of that in Fig. 4d, but only the CSBAT in Fig. 4f is the target CSBAT of the provided assembly.

In order to retrieve plans efficiently, we must organize the cases in the bases appropriately. Each case in the typical base is a solved assembly problem. Since the typical base may be big and non-linear, we must have a metric to choose the most possible case to be compared with the problem. A formula is used to calculate a priority number while comparing cases. The formula uses the information on whether the case matched, whether the matched case was successful in the past, and how often this case is selected in general. For a case A, we calculate its priority as follows: $priority(A) = \frac{NumofSuccess}{NumofSuccess+NumofFailure}$. The priority measures the number of successes over the total number of times that the case is selected. If the case is very successful, the chance of selecting it should be increased; if the case has had many failures, the chance of selecting it should be decreased. By considering the success and failure history of a case, the system selects not only the most similar, but also the most useful case for a problem. An example of a case stored in the typical base is shown in Table 2. Each case ID corresponds to the CSBAT and CSBARM of a solved assembly.

Now we introduce the algorithm of retrieving plans from the typical plan base. The first step of retrieval is to find all candidate CSBATs for the input CSBAT, then to determine which case is its target CSBAT, and finally to adjust the plans for the provided assembly. The algorithm is described as follows:

– Algorithm: Retrieve_Typical_Plans (T, M)
– Input: T is the CSBAT of the provided assembly, and M is its CSBARM
– Output: Partial assembly sequences

Step 1.
Push the provided T into *TestTreeStack*;
Step 2.
Check if *TestTreeStack* is null or not: if yes, the typical plans retrieval of the assembly has been finished successfully; else, pop a CSBAT as T_{cur} and let M_{cur} be the CSBARM of T_{cur};
Step 3.
Compute the nodes number and the maximum level of T_{cur}, then compare with each T_{case} that is the CSBAT of the case stored in the typical base, if there exists T_{case} that has the same number of nodes, root connection type, and maximum level to those of T_{cur}, then push T_{case} to *CandidateTreeStack* for further checking according to the case priority number, else, push the child CSBATs of T_{cur} into *TestTreeStack*, and go to step 2;
Step 4.
Check if *CandidateTreeStack* is null or not: if yes, there is no matching case in the typical base, go to step 2; else, pop a CSBAT as T_c;

Table 2 An example of case parameters

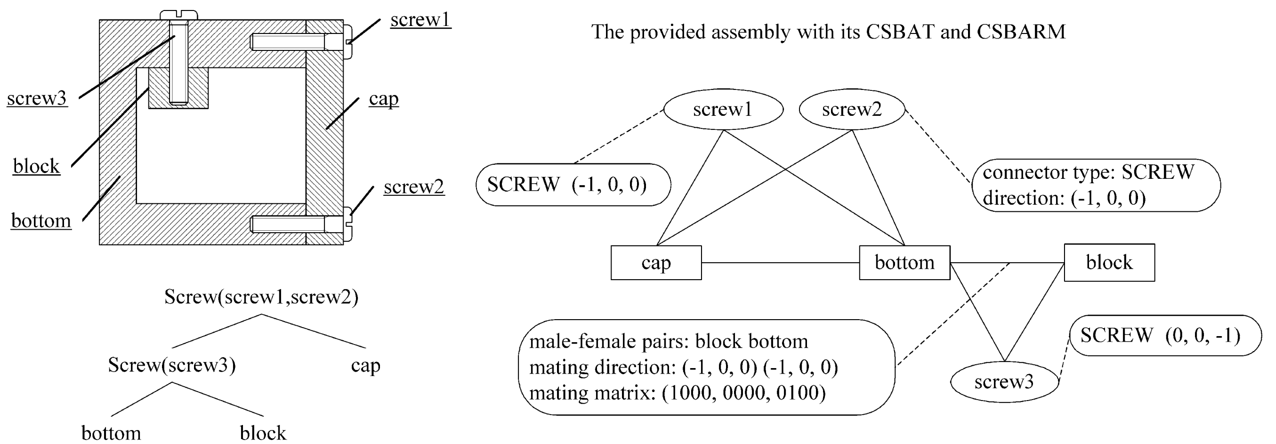
Case ID	No. of nodes	Root connection type	CSBAT max level	Priority	Success	Failure
...	...	...	...	...	...	...
1000	10	*Key*	5	0.900	18	2
1001	5	*Bolt–Nut*	3	0.910	182	18
1002	5	*Bolt–Nut*	3	0.610	122	78
...	...	...	...	...	...	...

Step 5.

Check whether bijection ϕ exists, for ϕ: $T_{cur} \rightarrow T_c$, $\forall u$, $v \in T_{cur}$, and $u \sim v$, satisfying the conditions: (a) $\phi(u) \sim \phi(v)$, and (b) u and $\phi(u)$ have the same part type or connection type, if yes, T_c is a candidate CSBAT of T_{cur} and let M_c=M_c; else, go to step 4;

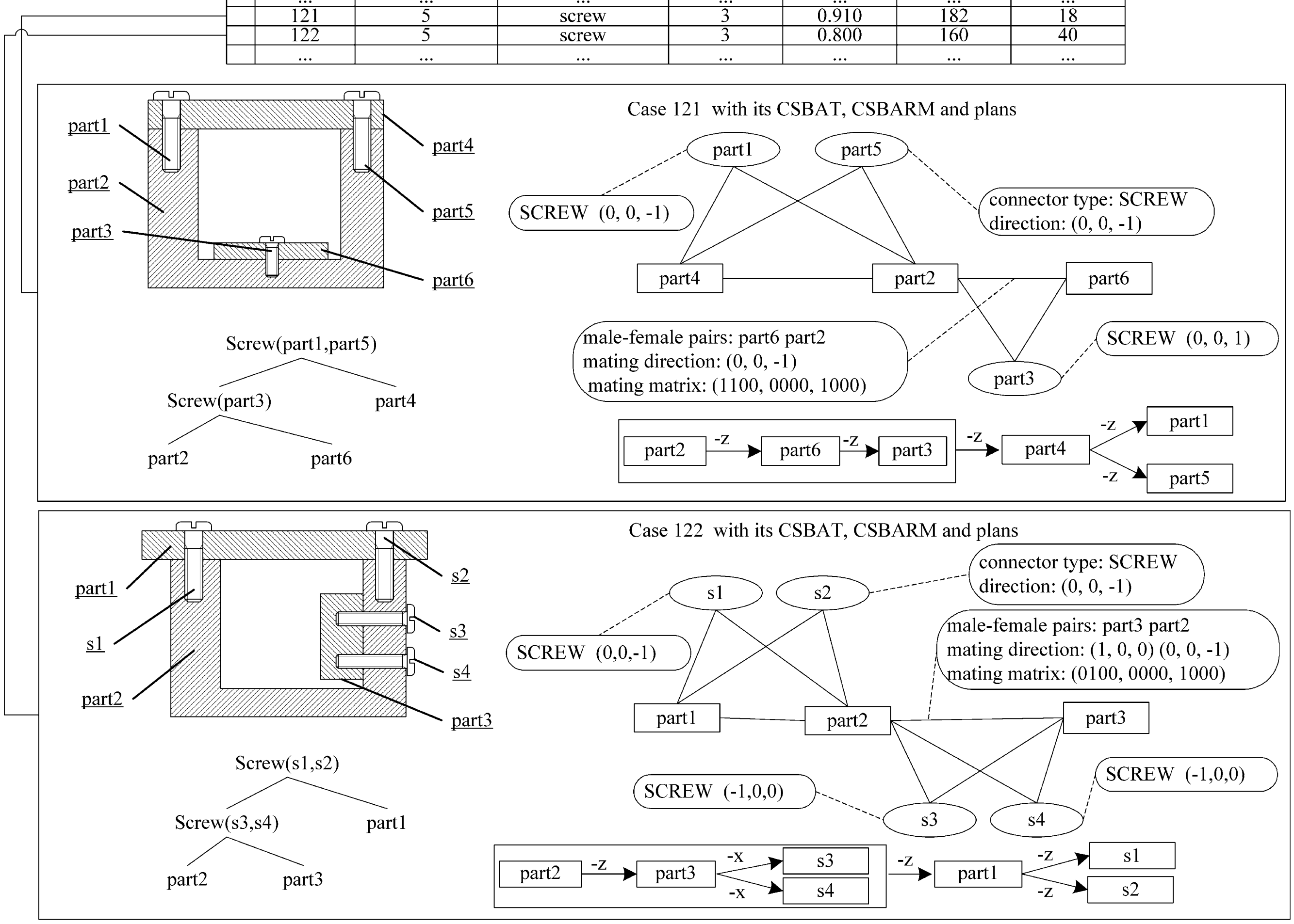
Step 6.

Check whether bijection ϕ and f exist, for ϕ: $(P_{cur} \cup C_{cur}) \rightarrow (P_c \cup C_c)$, f: $M_{cur} \rightarrow M_c$, u, $v \in (P_{cur} \cup C_{cur})$, satisfy-



a

	caseid	Num of nodes	Root connection	Max level	priority	success	failure
	...	...	...	...	...	...	...
	121	5	screw	3	0.910	182	18
	122	5	screw	3	0.800	160	40
	...	...	...	...	...	...	...



b

Fig. 5 **a** The provided assembly for retrieval. **b** All cases stored in the typical plan base ing the conditions: (a) mating type and connection type between $\phi(u)$ and $\phi(v)$ is the same as those of u and v, (b) $d\ (\phi(u), \phi(v))=f(d(u, v))$, and (c) $m(\phi(u), \phi(v))=f(m\ (u, v))$; if yes, it is a target CSBAT of T_{cur}; else, go to step 4;

Step 7.
Simplify the assembly model M and make the subassembly corresponding to T_{cur} be a super-part, adjust plans of T_{c} that are stored in the base to T_{cur}, get the partial plans of super-part T_{cur}, go to step 2;

Step 8.
Output partial assembly sequences with their super-part ID.

An example is provided to illustrate the procedure of retrieval from the typical base. The provided assembly with its CSBAT and CSBARM is shown in Fig. 5a, and the cases stored in the typical base are shown in Fig. 5b. The two cases shown in Fig. 5b are the candidate CSBATs of the provided assembly. In order to simplify the graphs, not all labels with the edges or nodes are specified in Fig. 5. Figure 6 shows how to determine which candidate CSBAT is a target CSBAT.

3.7 Standard plans retrieval and modification

The problem of standard plans retrieval and modification is also a problem of matching the CSBAT and CSBARM. Therefore, the approach to matching is similar to that of typical plans retrieval. However, it is different in some aspects. The CSBAT to retrieve plans from the standard base is transformed into a primitive CSBAT first. That means each child CSBAT is looked at as a whole, and the

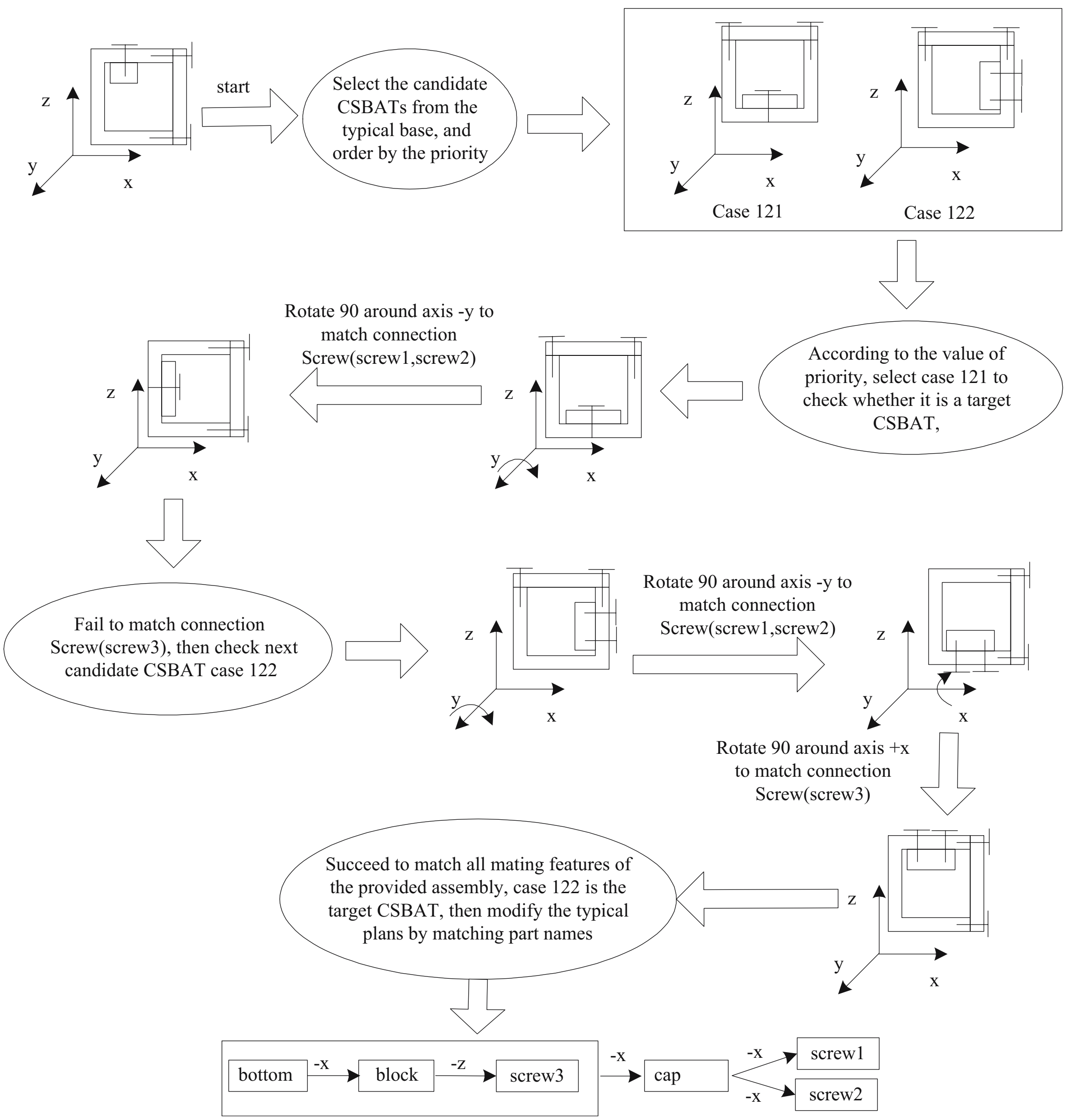


Fig. 6 An example of retrieval from the typical plan base

assembly constraints between parts in different child CSBATs are transformed into assembly constraints between child CSBATs. Figure 7 shows an example of transforming a non-primitive CSBAT to a primitive CSBAT.

In KBASP, the stored plans for a primitive CSBAT are returned by searching the standard plan base according to its connection type and mating features between the composed subassemblies or parts. For instance, for the assembly shown in Fig. 7a, its primitive CSBAT *SCREW* (*screw*1, *screw*2) [*subasm*1, *cap*] shown in Fig. 7b matches the connection *SCREW*(*s*1) [*p*1, *p*2] in the standard plan base. It is noted that the provided CSBAT has two connectors: *screw*1 and *screw*2. These two connectors are grouped, but they do not affect the matching between these two CSBATs. The *subasm*1 *SCREW*(*screw*3) [*bottom*, *block*] can also obtain the plans from the standard base.

4 Application examples

In order to illustrate the efficiency of the planner, we present a more complicated example *Gear Case*, which consists of the 32 parts shown in Fig. 8a,b. In some ways, it is a bad case for any purely geometric assembly, such as GRASP, that generates a large amount of AND/OR graphs, since the eight bolt–nuts assemblies can be placed in any order [8]. It is also difficult for any planner based on the cut-set approach, since there are so many candidate subassemblies existing in the first decomposition. The CSBARM of *Gear Case* is shown in Fig. 8c. In order to simplify the graph, the connectors constraining the same parts, such as *bolt–nut*1~8, are combined in a single node. The label for each edge is omitted for the same reason. To build the plans for the assembly, the CSBARM and the geometric information of the solid model are input into the KBASP system.

In the KBASP system, the first step to assembly planning for the *Gear Case* is to generate the CSBAT hierarchy by decomposing its CSBARM with respect to grouped connectors or a single connector. There exist more than one CSBAT for this assembly. Therefore, we use the proposed selection indices in Section 3.1 to select the preferred CSBAT out of multiple CSBATs. By using the algorithm for the generation of the CSBAT, the first selected connector group that can be disassembled is the bolt–nut set: {*bolt-nut*1~8}. {*cap*}, which is one subassembly with respect to {*bolt-nut*1~8}, consists of only one part, so it does not need to be subdivided further.

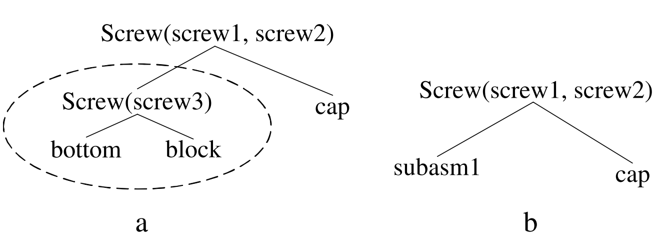


Fig. 7 **a** Non-primitive CSBAT. **b** Primitive CSBAT

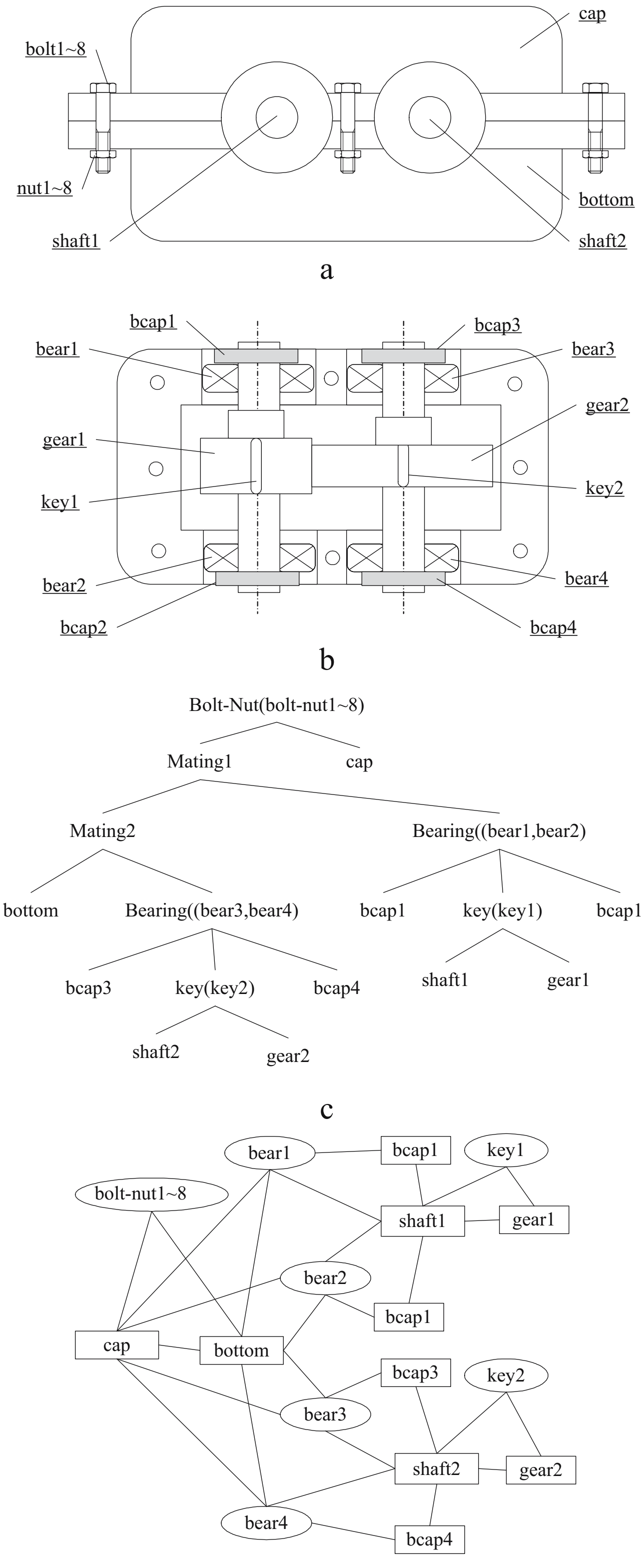
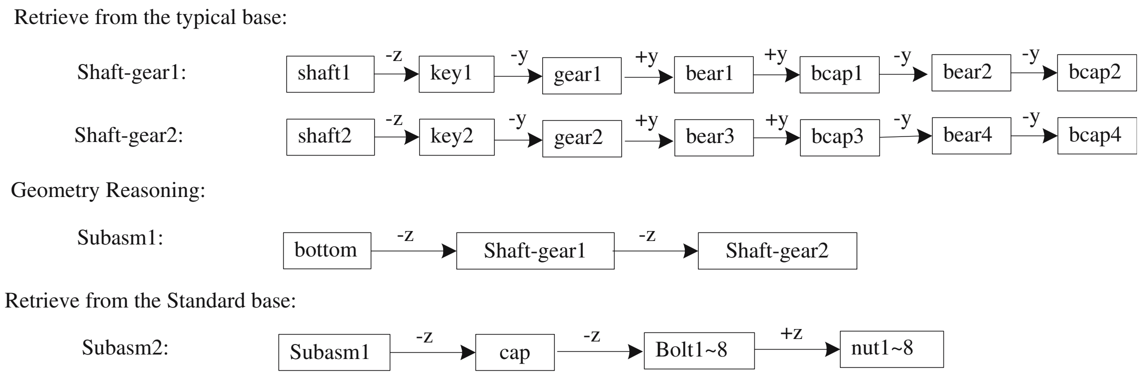


Fig. 8 **a** Solid model of *Gear Case*. **b** Its cut-open view. **c** Its CSBARM. **d** Its CSBAT. **e** Constructed plans

However, the other subassembly composed of 15 parts can be subdivided with the *Mating* type connection. With respect to *Mating*1, there are two subassemblies {*bottom*, *bear*3, *bear*4, *key*2, *shaft*2, *gear*2, *bcap*3, *bcap*4} and

Fig. 8 (continued)



e

{*bear*1, *bear*2, *key*1, *shaft*1, *gear*1, *bcap*1, *bcap*2}. The latter subassembly can be subdivided into {*bcap*1}, {*bcap*2}, {*key*1, *shaft*1, *gear*1} with respect to {*bear*1, *bear*2}, and {*key*1, *shaft*1, *gear*1} is subdivided into {*shaft*1} and {*gear*1} with respect to {*key*1} furthermore. The former subassembly {*bottom*, *bear*3, *bear*4, *key*2, *shaft*2, *gear*2, *bcap*3, *bcap*4} is subdivided into {*bottom*} and {*bear*3, *bear*4, *key*2, *shaft*2, *gear*2, *bcap*3, *bcap*4} with *Mating* type connection. Then, the CSBAT of {*bear*3, *bear*4, *key*2, *shaft*2, *gear*2, *bcap*3, *bcap*4} can be generated as that of {*bear*1, *bear*2, *key*1, *shaft*1, *gear*1, *bcap*1, *bcap*2}. The CSBAT is shown in Fig. 8d.

Subsequently, the KBASP system constructs the assembly plans for the *Gear Case* in three ways. Because the *Shaft-Gear* is a typical assembly stored in the typical base, the plans of subassemblies {*bear*1, *bear*2, *key*1, *shaft*1, *gear*1, *bcap*1, *bcap*2} and {*bear*3, *bear*4, *key*2, *shaft*2, *gear*2, *bcap*3, *bcap*4}, which are denoted as *Shaft-Gear*1 and *Shaft-Gear*2, respectively, can be retrieved from the typical base. For the mating connections *Mating*1 and *Mating*2, they do not find matched cases both in the typical nor the standard plan base. So the plans for *Subasm*1 are constructed by geometric reasoning. As to the primitive CSBAT *Bolt–Nut* [*bolt-nut*1~8] {*Subasm*1, *cap*} is retrieved and adapted from the case in the standard plan base. The plans of *Gear Case* are shown in Fig. 8e.

In order to analyze the efficiency of the knowledge-based approach, a set of experiments was contrived using an assembly as shown in Fig. 9, which simply consists of *N*+1 plates in a horizontal line. Each plate has a *Bolt–Nut* connection with its neighbors, so there are *N* primitive *Bolt–Nut* connections. Each plate may be simply removed by the disassembly of the bolt and nut that constrains the plate. The simple assembly was chosen because it is easy to increase the number of parts.

As in the analysis, the knowledge base in the planner was disabled and separate runs were performed. In real assemblies, CSBATs with very large numbers of child CSBATs usually contains many repetitions of the same type CSBATs, rather than many different CSBATs. In this section, we study the behavior of the planner on a CSBAT with many repeated child CSBATs, and demonstrate that the use of the knowledge base can drastically improve performance in that case. The KBASP has been implemented partially with programming in C++ based on the Dened ENVISION system on a Pentium-IV_compatible PC. The graph in Fig. 10 compares the run-times with and without the use of the knowledge base. In the graph, we see exponential growth in the run-times without the use of the knowledge base. When the knowledge base is turned on, the performance of the planner improves significantly. In this case, plans do not have to be generated separately for each *Bolt–Nut* connection. After the first run, the set of plans is simply retrieved from the knowledge base. This produces a relatively small speed-up in this example, because the similar structures are small. It is noted that there is still a possibility of doing exponential work, as different combinations of child CSBATs must be considered. The speed-up gained by the use of the knowledge base may be less dramatic than in this case. But the influence of the generations of plans for complicated assemblies is great. The KBASP system has planned for the assemblies of a variety products, including several industrial examples, such as *Wave-Hand* (Fig. 11) and *Toy Motor Grab* (Fig. 12). The application shows that the knowledge-based approach can reduce the computation complexity drastically and obtain more feasible and practical plans.

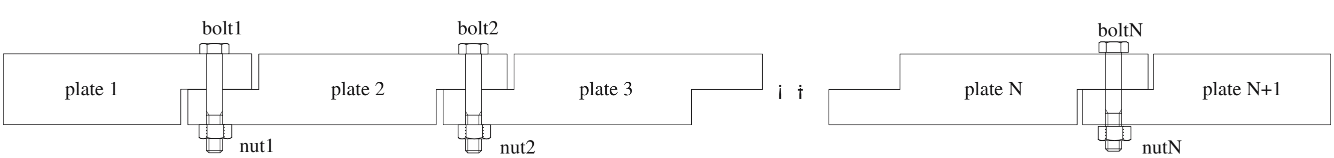


Fig. 9 Assembly consisting of *N*+1 plates

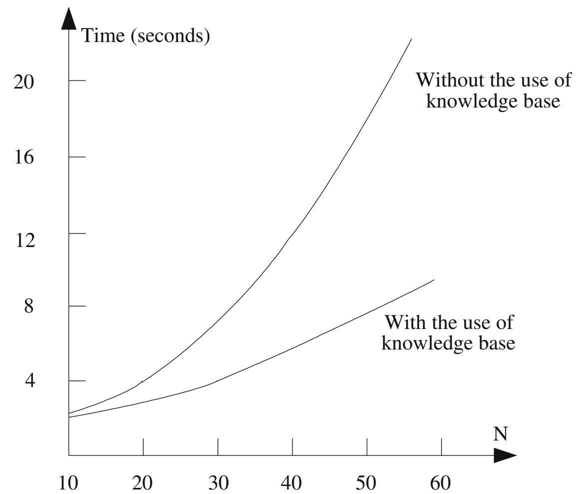


Fig. 10 The run-times of planning for the assembly

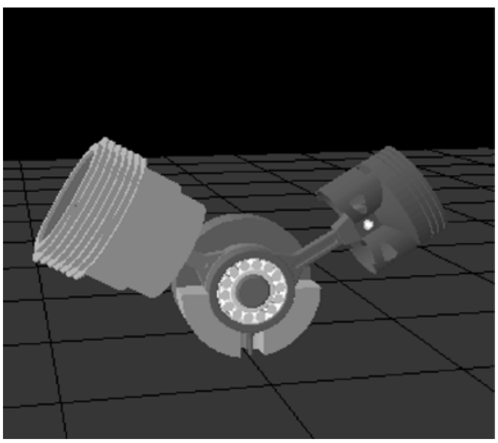


Fig. 11 *Wave-Hand*

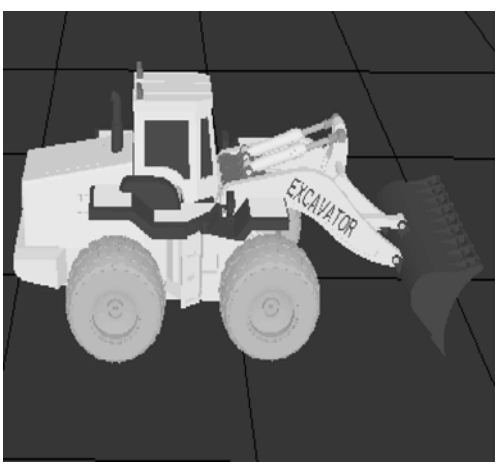


Fig. 12 *Toy Motor Grab*

5 Conclusions and future work

The knowledge-based approach proposed in this paper solves the problem of assembly sequence planning by integrating geometry-based reasoning with knowledge-based reasoning. This realization assists in significantly reducing the complexity and amount of planning required to determine more feasible and practical sequences for the assembly of production components. To verify the validity and efficiency of the approach, a variety of assemblies, including some complicated products from the industry, are tested in our KBASP (knowledge-based assembly sequence planning system). With the KBASP that we have developed, we reach three goals:

– Proposed the conception of CSBATs (connection-semantics-based assembly trees) and gave a uniform representation of assemblies that containing all of the information required for assembly sequences planning
– The design of an automated, systematic, and user-friendly method for generating assembly sequences
– The easy selection by the user of the most efficient and practical assembly sequences

Although KBASP generates a general-purpose geometric reasoning with the knowledge about how to build specific structure, there remains much to do. Future work should involve two main aspects:

– More non-geometric information, such as the assembly intents, should be utilized in planning
– Find more robust assembly coefficients by applying the approach to more various assembly environments, and find better CSBATs by lifting or extending the selection criteria.

Acknowledgements We gratefully acknowledge the support of the Special Funds for Major State Basic Research Program of China (program 973) under grant no. 2002CB312106 and the Zhejiang Natural Science Foundation of China under grant no. 602092. We would also like to thank the editors and the anonymous referees for their insightful comments.

References

1. De Fazio TL, Whitney DE (1987) Simplified generation of all mechanical assembly sequences. IEEE Trans Robot Autom 3 (6):640–658
2. Wilson RH (1995) Minimizing user queries in interactive assembly planning. IEEE Trans Robot Autom 11(2):308–312
3. Homem de Mello LS, Sanderson AC (1991) A correct and complete algorithm for the generation of mechanical assembly sequences. IEEE Trans Robot Autom 7(2):228–240
4. Baldwin DF, Abell TE, Lui M-CM, De Fazio TL, Whitney DE (1991) An integrated computer aid for generating and evaluating assembly planning. IEEE Trans Robot Autom 7(1):78–94
5. Zhang B, Zhang L (1991) A new algorithm for planning mechanical assembly sequences. Chinese J Comput 14(8):561–569 (in Chinese)
6. Lee (1994) Subassembly identification and evaluation for assembly planning. IEEE Trans Syst Man Cybern 24 (3):493–503
7. Ataie-Ashtiani B, Lockington DA, Volker RE, Eng T-H, Ling Z-K, Olson W, McLean C (1997) Feature-based assembly modeling and sequence generation. Comput Ind Eng 36 (1):17–33

8. Chakrabarty S, Wolter J (1997) A structure-oriented approach to assembly sequence planning. IEEE Trans Robot Autom 13 (1):14–29
9. Swaminathan A, Barber KS (1996) An experience-based assembly sequence planner for mechanical assemblies. IEEE Trans Robot Autom 12(2):252–266
10. Yin ZP, Ding H, Li HX, Xiong YL (2003) A connector-based hierarchical approach to assembly sequence planning for mechanical assemblies. Comput Aided Des 35(1):37–56
11. Fan J, Dong (2001) KVAS: a knowledge-based virtual assembly system. In: Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics, Taipei, Taiwan, October 2001, pp 1041–1046
12. Marian RM, Luong LHS, Abhary K (2003) Assembly sequence planning and optimisation using genetic algorithms: part I. Automatic generation of feasible assembly sequences. Applied Soft Comput J 2(3):223–253
13. Chen R-S, Lu K-Y, Yu S-C (2002) A hybrid genetic algorithm approach on multi-objective of assembly planning problem. Eng Appl Artif Intell 15(5):447–457
14. Chen CLP, Pao YH (1993) An integration of neural network and rule-based systems for design and planning of mechanical assemblies. IEEE Trans Syst Man Cybern 23(5):1359–1371
15. Chen Z, Zhang B (1991) The generation of assembly precedence constraint based on knowledge. J Softw 9:487–495 (in Chinese)
16. Zha XF, Du H (2002) A PDES/STEP-based model and system for concurrent integrated design and assembly planning. Comput Aided Des 34(14):1087–1110
17. Dong T, Tong R, Dong J, Zhang L (2004) Knowledge-based assembly sequence planning system. In: Proceedings of the 8th International Conference on Computer Supported Cooperative Work in Design (CSCWD 2004), Xiamen, China, May 2004, pp 516–521
18. Dong T, Tong R, Zhang L, Dong J (2005) A collaborative approach to assembly sequence planning. Adv Eng Inform 19(2):155–168
19. Gui J-K, Mäntylä M (1994) Functional understanding of assembly modeling. Comput Aided Des 26(6):435–451
20. Kanai S, Takahashi H, Makino H (1996) ASPEN: computer-aided assembly sequence planning and evaluation system based on predetermined time standard. Annals CIRP 45(1):35–39
21. Pelillo M., Siddiqi K, Zukker S (1999) Matching hierarchical structures using associated graphs. IEEE Trans Pattern Anal Mach Intell 21(11):1105–1120
22. Xu X, Liu W, Jin X, Sun Z (2002) Sketch-based user interface for creative tasks. In: Proceedings of the 5th Asia-Pacific Conference on Computer Human Interaction (APCHI 2002), Beijing, China, November 2002, pp 560–570
23. Chiu HP, Tseng DC (1997) Invariant handwritten Chinese character recognition using fuzzy min-max neural networks. Pattern Recog Lett 18(5):481–491
24. Suganthan PN, Yan H (1998) Recognition of handprinted Chinese characters by constrained graph matching. Image Vis Comput 16(3):191–201
25. Yeung M, Yeo BL, Liu B (1998) Segmentation of video by clustering and graph analysis. Comput Vis Image Underst 71(1):94–109
26. Ahmadyfard AR, Kittler J (2002) Using relaxation technique for region-based object recognition. Image Vis Comput 20(11):769–781