Inferring Feasible Assemblies from Spatial Constraints

Federico Thomas and Carme Torras

Abstract— **This paper treats two different problems in the analysis of assemblies, and algorithms are described for each. The first problem is the selection of consistent sets of part feature relationships, and it corresponds to a search over possible configurations of parts that are consistent with feature set mappings. The second problem is the evaluation of the kinematic consistency of an assembly that has been defined by consistent feature sets. These two problems are linked together as two of the steps required in a search for all correct assembly configurations of a given set of parts. Several of the other necessary steps related to part interference, path feasibility, and workcell device kinematics are referred to but not analyzed here. The proposed search algorithm is based on a constraint posting strategy, i.e., rather than generating and testing all specific alternatives, chunks of the search space are progressively removed from consideration by constraints that rule them out until finally one satisfactory alternative is found.**

I. Introduction

FULL automatic planning of robotic assembly tasks is still an open problem that has been addressed mainly during the last decade. Relevant work on the area ranges from artificial intelligence (AI) planning to geometric modeling and computational geometry, including fine motion planning, grasp planning, etc.

Planning has been an important research topic in artificial intelligence, and the AI approach has dominated much of the research in robot task planning. AI planners take as inputs a description of an initial state, a final state, and a set of legal operations. Their output is a sequence of legal operations to be performed to achieve the final state. Extending an existing AI planner to generate assembly plans is not an easy task since such an *extension* must be able to reason explicitly not only about space occupancy of workpieces described in terms of their 3-D geometric models, but also about kinematic constraints and physics (e.g., stability of subassemblies). Toward this goal some specific planners have been proposed, but the scope of the problems they are able to tackle is very limited. A classical example is BUILD [7].

Because of the influence of AI planners, the input to most assembly planners includes a relational model of the final assembly and the task consists of obtaining a sequence of actions based on one or more criteria such as time required, resources needed, etc. (see [6] and [25] for recent surveys). Nevertheless, there is no previous work on the inference of final assemblies given a general description of them. For a human operator, to give the description of a final assembly can often be more cumbersome than giving a sequence of assembly operations at the same level many assembly planners actually do. Manual positioning of each workpiece in a final assembly or in a mechanism by the user of a solid modeler is very tedious and error prone, as pointed out in [24]. These considerations do not mean that the problem of obtaining the final configuration is harder to solve than obtaining an implicit representation of all possible assembly sequences. Actually, as will become clear later, the underlying problems are the same in both cases: a final assembly is feasible if a sequence of operations exists that allows it to be achieved.

This paper deals with the problem of obtaining the spatial relationships between workpieces in their final assembly, from an initial description consisting of the models of the workpieces and, if desired, instructions that specify constraints on the degrees of freedom between parts of different workpieces.

We have adopted a constraint-based approach to solve the problem in which the process of synthesizing assembly configurations is conceived as a progressive refinement of an initial hypothesis by the application of successive constraints [21], [22]. These constraints, which we refer to as *spatial constraints*, are of two types: *constraints on the degrees of freedom* (DOF) between the parts of the workpieces and *constraints of nonintersection* between workpieces. The former type reduces the dimensionality of the space of the workpieces' DOF (that is, the space of assembly configurations or *configuration space*), while the latter type eliminates from configuration space the zones that lead to interferences.

Actually, spatial constraints are those constraints that are independent of the environment. Other constraints such as those imposed by stability and support criteria between workpieces, and accessibility or grasping requirements are, obviously, dependent on the working cell where the assembly task is to be carried out.

This paper is structured as follows. In Section II, an example that illustrates the main problems associated with the inference of final assemblies is given, and the approach proposed to deal with such problems is sketched. Section III describes the way assemblies, workpieces, and subparts are represented. Section IV explains how a hypothesis of the best assembly is obtained, then the detection of new spatial constraints and the way they are satisfied to obtain a better hypothesis. Section V is devoted to describing the method to deal with constraints on the DOF,

Manuscript received February 14, 1990; revised September 13, 1991. This work was partially supported by the Comisión Interministerial de Ciencia y Tecnología under the project "Automatic spatial reasoning based on constraints" (TIC 88-0197), and by the Fundación Areces, under the project SEPETER.

The authors are with the Institut de Cibernètica (UPC–CSIC) Diagonal 647, 2 planta 08028 Barcelona, Spain.

IEEE Log Number 9105869.

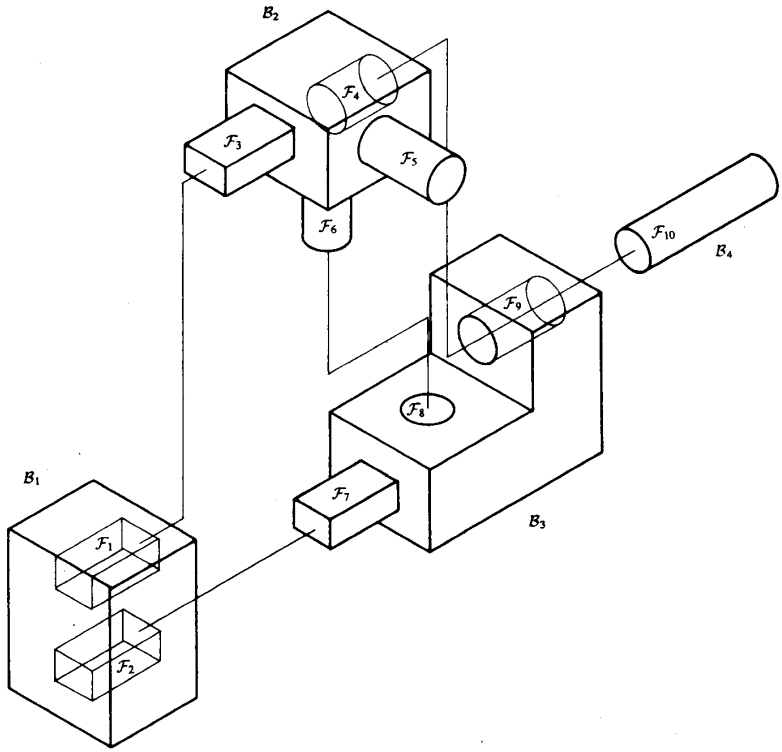


Fig. 1. Example used throughout this work to illustrate the main problems associated with the inference of final assemblies.

and some suggestions of how the described algorithm could be extended to encompass both nonintersection constraints and the generation of assembly sequences are also provided. Finally, in Section VI, conclusions and prospects for future research are stated.

II. Problem and Approach—An Example

Let us suppose that the workpieces $\mathcal{B}_1$, $\mathcal{B}_2$, $\mathcal{B}_3$, and $\mathcal{B}_4$ in Fig. 1 must be assembled. These workpieces have complementary subparts $\mathcal{F}_i$ that can be mated together. For the moment, we assume that compatibilities between complementary subparts are limited to pairs of subparts. However, this simplification will be removed in Section IV-B since some fastenings actually involve three or more subparts of different workpieces ($\mathcal{F}_4$, $\mathcal{F}_9$, and $\mathcal{F}_{10}$ in the example).

Each compatibility between complementary subparts defines one or more constrained movements. For example, a prismatic rod of square base and a pocket define four different translational movements, and a shaft and a hole through a wall ($\mathcal{F}_9$ and $\mathcal{F}_{10}$ in our example) define two cylindrical movements. In this section, let us assume for the sake of simplicity that each compatibility defines only one constrained movement (from the following section on, this assumption will be withdrawn). In this simplified context, an assembly can be unambiguously described by means of a set of pairs of complementary subparts of different workpieces. However, some of these sets represent unfeasible assemblies. The set $\{(\mathcal{F}_5, \mathcal{F}_8), (\mathcal{F}_6, \mathcal{F}_9)\}$ is an example, i.e., the pair $(\mathcal{F}_5, \mathcal{F}_8)$ is inconsistent with $(\mathcal{F}_6, \mathcal{F}_9)$ because these pairings lead to interference between workpieces $\mathcal{B}_2$ and $\mathcal{B}_3$. Note that inconsistencies are not always binary. For instance, any pair of elements of the set $\{(\mathcal{F}_5, \mathcal{F}_9), (\mathcal{F}_3, \mathcal{F}_1), (\mathcal{F}_7, \mathcal{F}_2)\}$ are consistent, but altogether they are inconsistent. This last situation can be detected by translating complementarity between subparts into kinematic constraints and finding whether there exists at least one configuration that satisfies all of them.

There are other important validity conditions besides the interference-free and kinematic ones mentioned above. For example, it could happen that workpieces could not be assembled because there were no collision-free path to bring them into contact from a situation in which they were sufficiently far apart. In our example, the set $\{(\mathcal{F}_5, \mathcal{F}_9)(\mathcal{F}_6, \mathcal{F}_8)\}$ would give rise to a kinematically valid subassembly of $\mathcal{B}_2$ and $\mathcal{B}_3$ without interference, but it would be impossible to assemble. Obviously, validity conditions of this kind can be checked using path-planning algorithms and, in some cases, kinematic considerations.

The naive way of solving the problem would be to enumerate all possible sets of matchings between complementary subparts and finding those that correspond to a feasible assembly. Obviously, this is not the best way to solve the problem: the same tests would be repeated many times since the algorithm would not keep track of partial inconsistencies.

The constraint-based approach taken here assumes that, at the beginning of the search, the problem is weakly constrained since it is only constrained by compatibilities between sub-

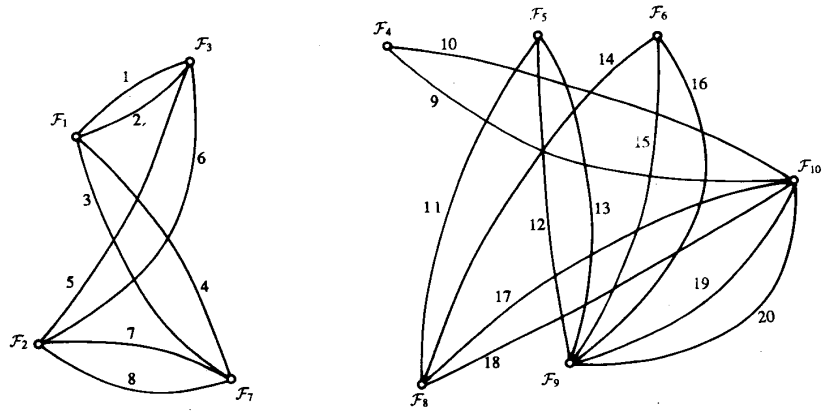


Fig. 2. GF graph associated with the assembly problem shown in Fig. 1.

parts. Then it is possible to generate a hypothetical solution that is not likely to correspond to a feasible assembly. In the process of checking the feasibility of the proposed hypothesis, new constraints that define the problem in more detail will be discovered. These constraints are used to obtain a better hypothesis. The process is repeated until a feasible assembly is found or the algorithm fails to find a hypothesis that satisfies all posted constraints. This approach is an instance of the classic "generate and test" paradigm [1] and the organization of the paper into two main sections—"generating the most promising hypothesis" and "testing the feasibility of a hypothesis"—tries to make this point clear.

III. Graph of Compatibilities Between Features

There are two levels of representation of a solid: 1) geometric and topological, and 2) semantic. The latter can be extracted from the former, which corresponds to the information stored in the data bases generated by most solid modelers.

The extraction of semantic information plays an important role in the link between CAD and CAM. This extraction has been addressed in the context of process planner development, where the goal is to find a plan to obtain a workpiece from a given stock by a sequence of drilling and milling operations carried out by a numerically controlled machine. The goal is to develop a system able to classify patterns of geometry (concave, convex, perpendicular, colinear, etc.) and topology (adjacent, number of sides, number of edge loops, etc.), according to predefined machinable features [27].

Different techniques have been applied in order to extract semantic information from a CAD data base. They range from recursive volume decomposition to syntactic pattern recognition and rule-based systems. For a brief description of these methods, see [9].

Semantic information is usually stored as a *graph of features*, where a *feature* is defined as a set of connected faces, or a subpart of a workpiece, related to a specific manufacturing process (examples are holes, slots, pockets, threads, etc.). A node of this graph is similar to a frame as used in knowledge-based system implementations. Slots exist for the boundary representation (BREP) of the feature and for all the necessary local manufacturing data, including links to other feature nodes.

The algorithm described below assumes that the extraction of all features of the involved workpieces, according to a dictionary of possible features [9], has already been carried out. Afterward, *compatibilities* between features are established, leading to a graph of compatibilities between features (GF graph). Nodes in this graph stand for features and arcs for compatibilities between them. The number of arcs between two nodes equals all the possible constrained movements between the corresponding features, so that each compatibility has a unique translation into constraints on the DOF. The GF graph for the workpieces in Fig. 1 appears in Fig. 2.

Compatibilities can be weighted. These weights measure the complementarity of shapes and the similarity of parameters, allowing us to deal with complementary subparts that do not exactly match each other.

IV. Generating the Most Promising Hypothesis

Once the GF graph has been created, one can look for an optimal match between workpieces. Assuming that fastenings involve only pairs of features, this problem can be conceptualized as a graph-matching problem [3, p. 339]. This simplification will be removed in Section IV-B. A matching of a graph is defined as a set of arcs, no two of which share a node. Given a graph, it is a well known problem to find a match that has as many arcs as possible. This problem would correspond, in our context, to the problem of finding an assembly involving as many complementary subparts as possible. Since we have also given weights to the arcs, the challenge is to find the match that has the largest total weight and also satisfies all detected spatial constraints.

A. Obtaining an Optimal Matching

The *incidence matrix* A of a GF graph has its entries defined as follows:

$$a_{ij} = \begin{cases} 1, & \text{if compatibility } j \text{ involves feature } i, \text{ and} \\ 0, & \text{otherwise} \end{cases} \quad (1)$$
$$\forall i = 1, \ldots, m$$

$$\forall j = 1, \ldots, n \tag{2}$$

m and n being the number of features and the number of compatibilities, respectively.

Let w_j be the weight for compatibility j, which measures the complementarity of shapes and the similarity of parameters of the corresponding features. These weights are heuristic and determine the order in which the hypotheses are obtained.

To obtain the first hypothesis, the following function is maximized:

$$\sum_j w_j x_j \tag{3}$$

subject to the constraints

$$\sum_{j=1}^{n} a_{ij} x_j \leq 1, \quad i = 1, \ldots, m \tag{4}$$

$$x_j \geq 0, \quad j = 1, \ldots, n. \tag{5}$$

The interpretation for the variables x_j is that

$$x_j = \begin{cases} 1, & \text{if compatibility } j \text{ is in the matching, and} \\ 0, & \text{otherwise.} \end{cases} \tag{6}$$

What is very important here is that it is guaranteed that $x_j \in \{0, \frac{1}{2}, 1\}$ in an optimal solution to the linear problem above. This is because all vertices of the convex polytope defined by the semihyperspaces (4) and (5), where a_{ij} are the elements of the incidence matrix of a nondirected graph with m nodes and n arcs, have coordinates 0, $\frac{1}{2}$, or 1, and the term $\frac{1}{2}$ only appears when there exist cycles of odd cardinality (Edmonds' theorem) [19, p. 256].

Let us assume, for the moment, that there are no cycles of odd cardinality in our graphs of compatibilities. It is then easy to solve the matching problem by finding a solution to the linear problem above using the simplex algorithm. Obviously, an optimal solution to the matching problem does not always correspond to a feasible assembly. Thus, it is necessary to verify this feasibility. If the current optimal matching corresponds to an unfeasible assembly, then unsatisfied spatial constraints will be detected during this verification and a new optimal solution must be found.

The simplex algorithm starts at one solution that satisfies a set of given constraints and proceeds toward an optimum. Nevertheless, not all constraints are known at this point so that when a new constraint is added, the optimal solution becomes impossible. However, it is well kown that it is possible to proceed toward an optimum from an impossible primal solution by applying the dual simplex [19, p. 50].

Assemblies cannot always be described in terms of pairings of features. The linear programming methodology described above allows us to easily represent the previously discarded situations, thus removing the simplification adopted up to here. If feature i can be simultaneously mated to n different features in the set $\mathcal{Q}$, then we set

$$a_{ij} = \frac{1}{n}, \quad \forall j \in \mathcal{Q}. \tag{7}$$

Actually, it is easy to express other complex constraints. For example:

1) Features i and j cannot appear together in any matching

$$\sum_k a_{ik} x_k + \sum_l a_{jl} x_l \leq 1. \tag{8}$$

2) Either feature i or feature j must appear in all matchings

$$\sum_k a_{ik} x_k + \sum_l a_{jl} x_l = 1. \tag{9}$$

3) If feature i appears in a matching, then feature j must also appear

$$\sum_k a_{ik} x_k - \sum_l a_{jl} x_l \leq 0. \tag{10}$$

These constraints will be called *matching constraints* to distinguish them from spatial constraints.

If there exist features that can be mated simultaneously to more than one other feature or the GF graph has cycles of odd cardinality, the solution to the linear programming problem is not guaranteed to be an integer, according to Edmonds' theorem. Constraints that remove noninteger solutions from the search space are referred to as *cuts*. Cuts are automatically added to ensure integrality of solutions. These cuts are generated according to the simple algebraic method proposed by Gomory (see [19, p. 326] for details).

The problem tackled can be described then as follows:

Input: A GF graph and (optionally) a set of matching constraints.

Output: A set of arcs that correspond to a feasible assembly.

The problem above is NP hard, as is proved in the appendix. The algorithm we have proposed to solve it consists of repeating the three following steps, until no unsatisfied constraints are detected or added by the user:

Step 1: Find a vertex of the polytope defined by the matching constraints that maximize a given objective function.

Step 2: If the vertex is not an integer, introduce a cut that strictly eliminates this vertex.

Step 3: If the vertex is an integer, verify the feasibility of the solution. If the solution is not feasible, derive as many matching constraints as possible which eliminate this and possibly other vertices of the polytope.

Next, we will see how this algorithm works through an example.

B. Example

Let us look at how a final assembly can be obtained for the workpieces in the example of Fig. 1, using the spatial constraints derived during the search process. The GF graph for this problem appears in Fig. 2. It contains 10 nodes and 20 arcs.

TABLE I
OBTAINING FEASIBLE ASSEMBLIES FROM SPATIAL CONSTRAINTS FOR THE WORKPIECES IN FIG. 1

Step	Hypothesis	Constraints		Total Iterations	Total Gomory Cuts	Remaining Solutions
		DOF	User			
1	$\{x_3, x_5, x_9, x_{11}, x_{15}\}$	$x_3 + x_5 + x_{11} \leq 2$ $x_{11} + x_{15} \leq 1$		9	0	256
2	$\{x_4, x_5, x_9, x_{11}, x_{16}\}$	$x_4 + x_5 + x_{11} \leq 2$ $x_{11} + x_{16} \leq 1$		16	1	234
3	$\{x_4, x_6, x_9, x_{11}, x_{20}\}$	$x_4 + x_6 + x_{11} \leq 2$		29	4	216
4	$\{x_3, x_6, x_9, x_{11}, x_{20}\}$	$x_3 + x_6 + x_{11} \leq 2$		34	5	212
5	$\{x_3, x_6, x_9, x_{16}, x_{17}\}$	$x_3 + x_6 + x_{16} \leq 2$		52	7	208
6	$\{x_3, x_6, x_9, x_{12}, x_{14}\}$	$x_3 + x_6 + x_{12} \leq 2$ $x_{12} + x_{14} \leq 1$		54	7	204
7	$\{x_3, x_6, x_9, x_{13}, x_{14}\}$	$x_3 + x_6 + x_{13} \leq 2$ $x_{13} + x_{14} \leq 1$		56	7	184
8	$\{x_3, x_6, x_9, x_{15}, x_{17}\}$	$x_3 + x_6 + x_{15} \leq 2$		58	7	164
*9	$\{x_3, x_6, x_9, x_{14}, x_{20}\}$		$x_3 + x_6 + x_9 + x_{14} + x_{20} \leq 4$	60	7	160
10	$\{x_3, x_6, x_{10}, x_{14}, x_{20}\}$	$x_3 + x_6 + x_{10} + x_{20} \leq 3$		62	7	159
*11	$\{x_3, x_6, x_{10}, x_{14}, x_{19}\}$			87	11	158

Its incidence matrix is obtained according to (1), with the exception of the following elements:

$$a_{10,9} = a_{10,10} = a_{10,17} = a_{10,18} = a_{10,19} = a_{10,20} = \frac{1}{2} \tag{11}$$

which are obtained according to (1). Thus, we get (12), given at the bottom of this page.

If all complementary features fit exactly, all arcs will have the same weight, and the 256 solutions involving 9 features will have the same maximum confidence. Only four of these solutions, which are totally equivalent because of the symmetries of workpieces $\mathcal{B}_1$ and $\mathcal{B}_4$, are feasible. These feasible matchings, with maximum confidence, are:

$$\begin{aligned} &\{x_3, x_6, x_{10}, x_{14}, x_{19}\} \\ &\{x_3, x_6, x_9, x_{14}, x_{20}\} \\ &\{x_1, x_7, x_{10}, x_{14}, x_{19}\} \\ &\{x_1, x_7, x_9, x_{14}, x_{20}\} \end{aligned} \tag{13}$$

where the subindices of x refer to the compatibility numbers shown in Fig. 2 (e.g., x_{14} refers to the unique matching of features $\mathcal{F}_6$ and $\mathcal{F}_8$, while x_1 and x_6 refer to the two alternative matchings of feature $\mathcal{F}_3$ derived from the symmetric placement of features $\mathcal{F}_1$ and $\mathcal{F}_2$ in workpiece $\mathcal{B}_1$).

A solution to the problem is reached after 9 iterations of the 3-step algorithm described in the preceding section, which entail 60 iterations of the simplex algorithm. Only 8 hypotheses are generated, which lead to 12 inconsistencies due to kinematic problems. Note that the lack of at least one remaining degree of freedom in each subassembly—to permit effectively assemblying it—is considered a kinematic problem (e.g., $x_{11} + x_{15} \leq 1$). Moreover, seven cuts are also generated in order to ensure integrality of hypotheses.

Table I shows the list of generated hypotheses, the constraints added after checking their feasibility, the number of iterations of the simplex algorithm, the number of Gomory cuts, and the number of solutions with the same confidence remaining in the search space.

In step 9 a feasible assembly is found. If this assembly is not the one the user wanted, a new constraint that strictly eliminates this solution is introduced, leading to another feasible solution after generating two new hypotheses.

V. TESTING THE FEASIBILITY OF A HYPOTHESIS

Any optimal matching is first translated into constraints on the DOF between the involved workpieces. For example, a match involving only shafts and holes would be translated into

$$A = \begin{bmatrix} 1. & 1. & 1. & 1. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. \\ 0. & 0. & 0. & 0. & 1. & 1. & 1. & 1. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. \\ 1. & 1. & 0. & 0. & 1. & 1. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. \\ 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 1. & 1. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. \\ 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 1. & 1. & 1. & 0. & 0. & 0. & 0. & 0. & 0. & 0. \\ 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 1. & 1. & 1. & 0. & 0. & 0. & 0. \\ 0. & 0. & 1. & 1. & 0. & 0. & 1. & 1. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. \\ 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 1. & 0. & 0. & 1. & 0. & 0. & 1. & 1. & 0. & 0. \\ 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 1. & 1. & 0. & 1. & 1. & 0. & 0. & 1. & 1. \\ 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & .5 & .5 & 0. & 0. & 0. & 0. & 0. & 0. & .5 & .5 & .5 & .5 \end{bmatrix}. \tag{12}$$

the alignment of the axes of the shafts and their corresponding holes. In other words, the depths and angles of the shafts in the holes would remain as variables, giving rise to constrained movements with two DOF each. Two conditions now must be satisfied for the current optimal matching to be feasible: 1) it must be possible to simultaneously satisfy all constraints on the DOF generated so far and 2) there must be at least one remaining DOF between the workpieces in each subassembly, allowing their assembly. An efficient algorithm for propagating constraints on the DOF is needed to check these conditions. This algorithm not only must discern whether it is possible to simultaneously satisfy all established constraints but also must assign values to the constrained DOF's and obtain relationships between them. If the analyzed matching is kinematically feasible (i.e., the two conditions above are satisfied), then two additional conditions must be satisfied: 3) there must be no interference between workpieces in the final assembly and 4) there must exist a valid assembly sequence. These two additional conditions are not addressed in this paper.

A. Constraints on the Degrees of Freedom

Matings of features have a direct translation into constraints on the DOF. This translation requires that the legal motion for each compatible pair of predefined features is known. Thus, it has to be either provided by the user or inferred automatically by the system itself. The main underlying problem to infer legal motions directly from geometric models of predefined features is, in most cases, finding local symmetries or finding cycles of edges when working with polyhedral workpieces [26].

For each hypothesis obtained as described in the preceding section, a *graph of spatial relationships*—or GR graph—is generated, whose nodes correspond to workpieces and whose arcs are labeled with sets of legal relative spatial transformations. These sets of legal transformations are, in most cases, easily represented through a matrix equation that links the coordinate reference frames of the corresponding workpieces. These equations contain variables or DOF's. When values are assigned to these variables, an element of the corresponding transformation set is obtained. Each of these sets is defined as a *constraint on the DOF* and will be denoted by $\boldsymbol{R}_i$, where i is the same subindex as that of the compatibility it corresponds to, in this case x_i.

If $\boldsymbol{R}_i$ is the set of legal transformations from the reference frame of $\mathcal{B}_1$ to the reference frame of $\mathcal{B}_2$, $\boldsymbol{R}_i^{-1}$ denotes the set of legal transformations in the other way around, i.e., from $\mathcal{B}_2$ to $\mathcal{B}_1$.

The automatic manipulation of constraints on the DOF has attracted a lot of attention not only in kinematics but also more recently in the design of object-level robot programming languages such as RAPT [20]. Several algebraic symbolic approaches have emerged, among which we will mention a system of rewriting rules [20] and a table look-up procedure [15].

Algebraic symbolic (as opposed to numerical) methods for dealing with constraints on the DOF can shed light on basic aspects of linkage behavior. As will be shown later, the way in which constraints on the DOF propagate provides useful information on the sequence of assembly.

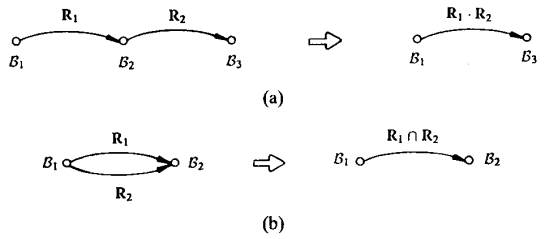


Fig. 3. The two basic operation with constraints on the DOF's (a) Composition of constraints. (b) Intersection of constraints.

The algebraic symbolic method used by the RAPT interpreter consists of two stages: Finding a solution for the rotation component that will fix some angles, followed by the formation of real equations involving variables that represent linear displacements and possibly sines and cosines of remaining angle variables, which, in general, are difficult to solve.

A more recent approach takes advantage of the fact that the legal relative motions resulting from mating two complementary features, such as cylinders, prisms, or grooves, constitute subgroups of the Euclidean group, leading to a simple procedure for constraint manipulation [23].

The two basic operations with constraints on the DOF are constraint composition and constraint intersection. The former (see Fig. 3(a)) involves finding the constraint between workpieces $\mathcal{B}_1$ and $\mathcal{B}_3$ that results from composing the constraint between $\mathcal{B}_1$ and $\mathcal{B}_2$ (say $\boldsymbol{R}_1$) with that between $\mathcal{B}_2$ and $\mathcal{B}_3$ (say $\boldsymbol{R}_2$), which will be denoted by $\boldsymbol{R}_1 \cdot \boldsymbol{R}_2$. The latter operation (see Fig. 3(b)) permits combining two given constraints, $\boldsymbol{R}_1$ and $\boldsymbol{R}_2$, between the same two workpieces into a single resulting constraint, which will be denoted by $\boldsymbol{R}_1 \cap \boldsymbol{R}_2$. These operations admit formal definitions that can be found elsewhere [10].

If, as the result of intersecting two constraints between the same two workpieces, the empty set is obtained, the constraints are said to be *inconsistent*. The goal now is to verify the consistency of entire GR graphs. This can be stated as a problem of consistency in networks of relations. As pointed out in [17], any representation of the constraints that allow composition and intersection is sufficient for this purpose.

Informally, a GR graph is consistent if there exist configurations of the workpieces whose relative coordinate transformations satisfy the corresponding constraints. Obviously, a GR graph without cycles is always consistent; thus, it is easy to realize the important role of cycles in GR graphs.

Let C_i be a cycle whose arc set is labeled with the constraints

$$\{\boldsymbol{R}_1, \boldsymbol{R}_2, \ldots, \boldsymbol{R}_j, \ldots, \boldsymbol{R}_n\} \tag{14}$$

according to Fig. 4(a). Then the constraint $\boldsymbol{R}_j$ can be substituted by

$$\boldsymbol{R}_j^i = \boldsymbol{R}_j \cap (\boldsymbol{R}_{j-1}^{-1} \cdots \boldsymbol{R}_1^{-1} \cdot \boldsymbol{R}_n^{-1} \cdots \boldsymbol{R}_{j+1}^{-1}) \tag{15}$$

without modifying the consistency of the corresponding GR graph (see Fig. 4(b)). In order to simplify the notation, we will write $\mathcal{S}(\boldsymbol{R}_j, C_i)$ for the intersection in (15) and we will

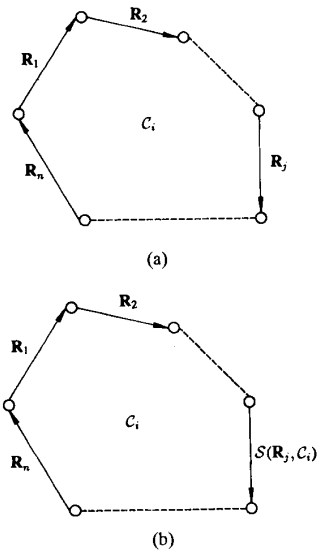


Fig. 4. The basic mechanism for constraint propagation. A constraint R_j, labeling an arc in a cycle C_i (a), can be substituted by $\mathcal{S}(R_j, C_i)$ (b).

say that, to obtain R_j^i, the cycle C_i is solved for R_j. This is the basic mechanism for constraint propagation as it is shown in Section V-A-3.

Next, before introducing a general algorithm for verifying the consistency of GR graphs, a few concepts on cycles in graphs are reviewed.

1) Preliminaries on Cycles: We denote a cycle by the set of its arcs. For cycles C_1 and C_2 in a graph $G = (V, E)$ such that $C_1 \cap C_2 \neq \emptyset$, we define two basic operations that will be useful later, namely the *sum* $C_1 + C_2$ and the *ring sum* $C_1 \bigoplus C_2$ as the usual union and symmetric difference set operations, respectively:

$$C_1 + C_2 = \{e \in E \mid e \in C_1 \text{ or } e \in C_2\} \tag{16}$$

$$C_1 \bigoplus C_2 = \{e \in E \mid (e \in C_1 \text{ and } e \notin C_2) \text{ or } (e \notin C_1 \text{ and } e \in C_2)\}. \tag{17}$$

Both operations are commutative and associative, the outcome of the ring sum being a cycle or a set of cycles.

A set of cycles $\mathcal{H}$ in a graph $G = (V, E)$ is said to be *a complete set of basic cycles* if 1) every cycle in the graph can be expressed as a ring sum of some cycles in $\mathcal{H}$ and 2) no cycle in $\mathcal{H}$ can be expressed as a ring sum of other cycles in $\mathcal{H}$. The cardinality of a complete set of basic cycles is $\mu = \mid E \mid - \mid V \mid + 1$, which is called the cyclomatic number. Hence, the maximum number of cycles is $2^\mu - 1$.

Let X be a complete set of basic cycles in a graph G. The *cycle graph* $\mathcal{D}_X(G)$ of X is the graph with vertex set X and arcs joining two distinct nodes if and only if the corresponding cycles have an arc in common. Constraints labeling a shared arc are called *shared constraints*.

In a plane representation of a planar connected graph, the set of cycles forming the interior regions—called region cycles—constitutes a complete set of basic cycles. There are $\binom{\mu+1}{\mu}$ different complete sets of region cycles. This can be easily seen by noting that a planar graph can be embedded in the surface of a sphere. The number of region cycles in the surface of a sphere would be $\mu+1$, which are also the shortest cycles for a planar graph. Thus, if G is a planar graph, $\mathcal{D}_X(G)$ is a subgraph of the dual graph of G (see [4, p. 106]). Note also that, in planar graphs, an arc can only be shared by two basic cycles.

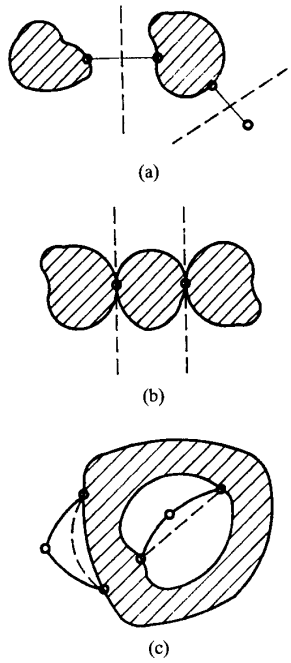


Fig. 5. Operations applied for the isolation of blocks: (a) Elimination of cutlines. (b) Splitting cut points. (c) Composition of constraints in series.

The distance $\text{dist}_X(C_1, C_2)$ between two basic cycles $C_1, C_2 \in X$ is the minimum number of arcs in a path from C_1 to C_2 in $\mathcal{D}_X(G)$.

2) Isolation of Blocks in a GR Graph: When a constraint on the DOF is posted, it can affect other workpieces different from those upon which it is incident; however, in general, a constraint is limited in its *scope*. In order to isolate the subgraphs that constitute the scopes of the different constraints, the following operations are automatically applied:

1) Elimination of cutlines or bridges. This includes the elimination of pendant constraints (see Fig. 5(a)).
2) Split cutpoints or articulation nodes into two nodes to produce two disjoint subgraphs (see Fig. 5(b)).
3) Elimination of nodes of degree two by composing constraints in series. This eliminates all nodes of degree two (see Fig. 5(c)). Obviously, this third operation is not applicable to cycles consisting of only two arcs.

As a result of these operations, a set of subgraphs, or simply *blocks*, are obtained. A GR graph is consistent if and only if each of its blocks is consistent.

3) Propagation of Constraints in a Block: A general procedure to *propagate* the effect of constraints in each of the blocks

of a GR graph has been devised, either to characterize the set of configurations that satisfy all the constraints or to find that there exist no such configurations, which will allow us to discover matching constraints to generate a better hypothesis.

The propagation process consists in *filtering* all constraints, i.e., eliminating from the constraints those transformations that cannot appear in any solution. Eventually, if all constraints are reduced to only one element, a single solution is obtained.

Global consistency in a block G is checked by removing local inconsistencies, i.e., by eliminating inconsistencies in basic cycles, which is equivalent to ensuring *node consistency* in $\mathcal{D}_X(G)$, and by eliminating inconsistencies between adjacent basic cycles, which is equivalent to ensuring *arc consistency* in $\mathcal{D}_X(G)$. The following procedure implements this idea:

```
procedure filter_constraints;
input: G; /* a block */
output: incon_set, p;
incon_set:= ∅;
p:= -1;
repeat
   stop:= true;
/* check node consistency */

   p := p + 1;
   forall basic cycles C_i do
     forall constraints R_j do
         R_j^i := S(R_j, C_i);
         if R_j^i == ∅ then
           incon_set:= {C_i};
           return();
         endif;
     enddo;
   enddo;
/* check arc consistency */

   p := p + 1;
   forall shared constraints R_j do
     R_T := ∩_k R_j^k;
     if R_T == ∅ then
       incon_set:= {{C_k}_k};
       return();
     endif;
     if R_T ≠ R_j then
       stop:= false;
       R_j := R_T;
     endif;
   enddo;
until stop;
end.
```

Inconsistencies can be found when obtaining either $\mathcal{S}(\boldsymbol{R}_j, C_i)$ or $\cap_k \boldsymbol{R}_j^k$. In the first case, the cycle C_i is included in `incon_set`; in the second one, the set of cycles sharing the constraint $\boldsymbol{R}_j$ are included in `incon_set`.

Computing $\mathcal{S}(\boldsymbol{R}_j, C_i)$ and $\cap_k \boldsymbol{R}_j^k$ requires solving kinematic equations whose complexity depends on the kind of representation used (homogeneous transformations, quaternions, etc.). These equations are easier to solve if all the involved constraints have entire DOF (not just ranges of them) that are independent from each other (not related by equations) [23]. Roughly, this entails that all planes and axes of symmetry of the involved features are either parallel or orthogonal in the final assembly. In this particular case, since an integer number of DOF is removed at each iteration, it is easy to justify that the procedure given above halts and that its complexity is linear in the number of shared constraints [18]. If the block G is planar, then the complexity is also linear in the number of cycles μ [18].

The significance of the described procedure is that it only involves repeatedly solving a set of basic cycles, which, in the case above, are becoming simpler to solve at each step. Further savings can be obtained by selecting the complete set of *shortest* basic cycles, which, in the case of a planar graph, consists of region cycles.

Only in the particular case where all constraints are of the type above does the fact that the procedure halts with `incon_set` $= \emptyset$ imply that block G is consistent. In the other cases, the existence of an actual solution has to be checked through a backtrack search [18].

If, on the contrary, the procedure halts with `incon_set` $\neq \emptyset$, then G is guaranteed to be inconsistent, and we need to derive a set of matching constraints that permits obtaining a more promising hypothesis at the next iteration of Steps 1 and 2 of the algorithm in Section IV-A. This is the subject of the following section.

4) Obtaining Tightest Matching Constraints: After the procedure in the preceding section has ended, each cycle (or set of cycles) in `incon_set` has to be postprocessed in order to obtain one or more matching constraints. We would like these constraints to reduce the search space as much as possible. Observe that all we know at the end of the procedure is that

$$\sum_{\text{dist}_X(C_i,C_j)\leq p/2} C_j \tag{18}$$

is inconsistent, $\forall C_i \in$ `incon_set`, if p is even and

$$\sum_{\exists C_i \in \mathcal{C} |\ \text{dist}_X(C_i,C_j)\leq p/2} C_j \tag{19}$$

is inconsistent, $\forall \mathcal{C} \in$ `incon_set`, if p is odd.

Let us suppose that $\mathcal{T}$ is the set of the indices of the arcs in an inconsistent set $C_1 + C_2 + \ldots C_n$ (i.e., $C_1, C_2, \ldots C_n$ are *jointly inconsistent cycles*); then the corresponding arcs in the GF graph cannot appear together in another hypothesis, and thus, the detected matching constraint would be

$$\sum_{j\in\mathcal{T}} x_j \leq |\ \mathcal{T}\ | - 1. \tag{20}$$

However, it could be that there exists a subset of $C_1 + C_2 + \ldots + C_n$ that is also inconsistent, and thus it is possible to generate a matching constraint involving fewer variables than those in (20), which, therefore, would introduce a more drastic cut in the search space. Thus, our aim is to find minimal sets of jointly inconsistent arcs.

In order to describe how to obtain the tightest constraints, we need to introduce some concepts about the consistency of cycles.

Let $\mathcal{N}$ be the set of all possible sums of cycles in a block. The set inclusion relation induces a partial order on $\mathcal{N}$. This partial order can be interpreted in terms of consistency relations as follows. Let $N_1 \subset N_2$, with $N_1, N_2 \in \mathcal{N}$; we have

- if N_1 is inconsistent, then N_2 is inconsistent, and
- if N_2 is consistent, then N_1 is consistent.

In particular, if C_1 and C_2 are two cycles such that $C_1 \cap C_2 \neq \emptyset$, we have

- If C_1 is inconsistent, then $C_1 + C_2$ is inconsistent.
- If $C_1 + C_2$ is consistent, then $C_1 \bigoplus C_2$ is consistent.

If $C_1 + C_2$ is inconsistent, the cycle $C_1 \bigoplus C_2$ may, or may not, be inconsistent. If it is inconsistent, it will provide a tighter matching constraint than $C_1 + C_2$.

Now, $N^* \in \mathcal{N}$ will originate a *tightest matching constraint* if and only if

- N^* is inconsistent, and
- $\forall N \in \mathcal{N}$ such that $N \subset N^*, N$ is consistent.

Next, we will describe how to obtain tightest constraints in the case that the procedure `filter_constraints` outputs a p that is even. The case in which p is odd can be treated analogously.

Once the set of jointly inconsistent cycles $\{C_j\}_{j\in\mathcal{T}}$ around a cycle $C_i \in$ `incon_set` has been found according to (18), one must test the consistency of all the sets $N \in \mathcal{N}$ such that

$$N \subseteq C_i + \sum_{j\in\mathcal{J}} C_j$$

$\forall \mathcal{J} \subseteq \mathcal{T}$ such that $\{C_i\} \cup \{C_j\}_{j\in\mathcal{J}}$ conform a connected subgraph of $\mathcal{D}_X(G)$.

These tests can be performed using again the procedure `filter_constraints`. If such tests are carried out following the partial ordering in $\mathcal{N}$, then it is guaranteed that the tightest constraints will be obtained.

Observe that many different search strategies could be implemented to find the tightest constraints. Each would involve a different way of traversing the graph representing the partial order on $\mathcal{N}$. The one we have outlined here consists of a kind of depth-first search to find an initial set of constraints, followed by a breadth-first process to refine these constraints. In some cases, this second search stage can be very time consuming, and it may be preferable to generate a new hypothesis on the basis of the constraints obtained in the first search stage. In sum, there is always a tradeoff between time spent in tightening the constraints and time spent by the successive applications of the simplex algorithm to find hypotheses that satisfy the constraints. Exploring how much constraint refinement would lead to an "optimal" procedure is a topic of further research.

5) Example: Returning to our example in Fig. 1, we will describe in detail how the algorithm works on a particular hypothesis. After the seventh hypothesis has been obtained (see Table I), the GR graph shown in Fig. 6(a) is generated, which contains only one planar block (see Fig. 6(b)) whose graph of region cycles appears in Fig. 6(c). There are two region cycles in the block G obtained, leading to a $\mathcal{D}_X(G)$ graph with only two nodes. Then the first step of the filtering

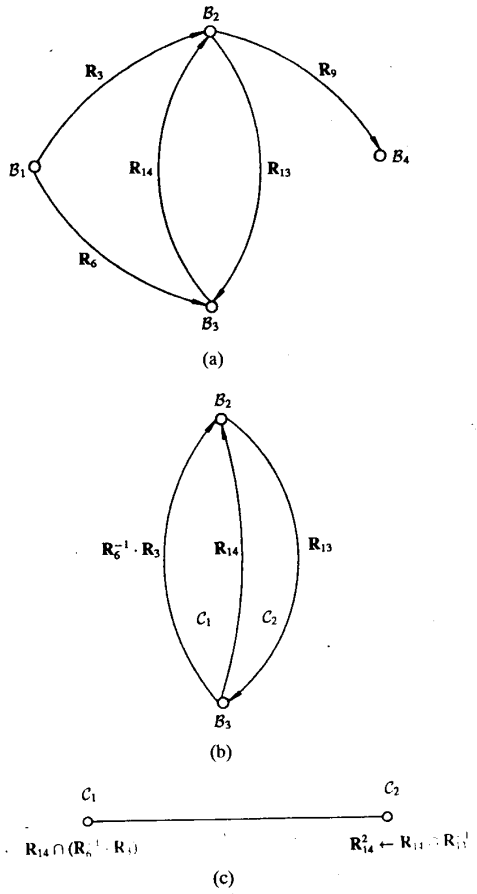


Fig. 6. (a) GR graph obtained for the seventh hypothesis in *Table I*, (b) the only block contained in this GR graph, and (c) the corresponding graph of basic cycles.

procedure (node consistency check) consists of obtaining

$$\begin{aligned} \boldsymbol{R}_{14}^1 &= \mathcal{S}(\boldsymbol{R}_{14}, C_1) = \boldsymbol{R}_{14} \cap (\boldsymbol{R}_6^{-1} \cdot \boldsymbol{R}_3) \\ \boldsymbol{R}_{14}^2 &= \mathcal{S}(\boldsymbol{R}_{14}, C_2) = \boldsymbol{R}_{14} \cap \boldsymbol{R}_{13}^{-1}. \end{aligned} \tag{21}$$

Note that both $\boldsymbol{R}_{14}^1$ and $\boldsymbol{R}_{14}^2$ contain only one legal transformation between $\mathcal{B}_2$ and $\mathcal{B}_3$ each, since the position of the workpiece $\mathcal{B}_2$ with reference to $\mathcal{B}_3$ is fixed in both cases. Each of the two cycles is thus consistent.

The second step of the filtering procedure (arc consistency check) consists of intersecting the solutions obtained for the shared constraint to actualize them, i.e., finding $\boldsymbol{R}_{14}^1 \cap \boldsymbol{R}_{14}^2$. Since the resulting position of $\mathcal{B}_2$ with reference to $\mathcal{B}_3$ is different in both cases, $\boldsymbol{R}_{14}^1 \cap \boldsymbol{R}_{14}^2 = \emptyset$. Therefore, the cycles C_1 and C_2 are jointly inconsistent. Thus, this allows us to post the following matching constraint:

$$x_3 + x_6 + x_{13} + x_{14} \leq 3 \tag{22}$$

according to (20).

We note that, since the cycle C_2 involves only two workpieces and when it is solved the position between both workpieces is fixed, no DOF remains to permit assemblying

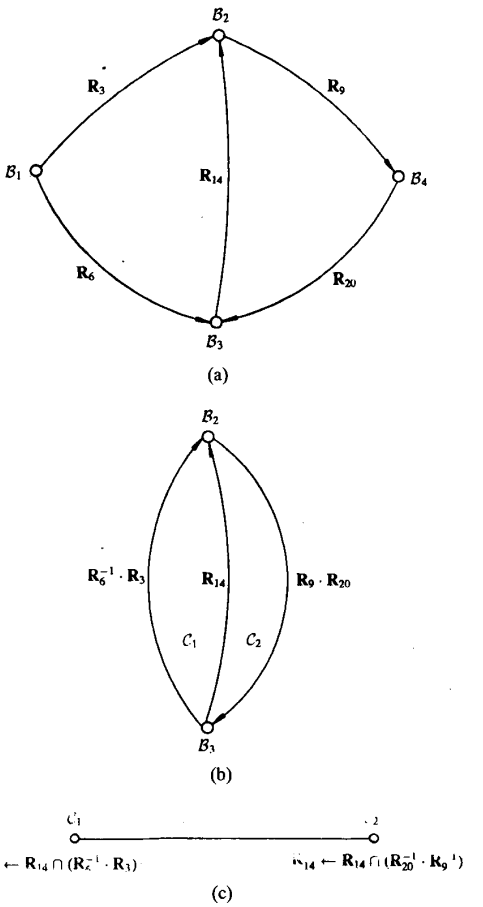


Fig. 7. (a) GR graph obtained for the ninth hypothesis in *Table I*, (b) the only block contained in this GR graph, and (c) the corresponding graph of basic cycles.

them. Thus, the following matching constraint is also detected:

$$x_{13} + x_{14} \leq 1. \tag{23}$$

In order to find a tighter constraint than (22), we proceed as explained in Section V-A-4. Since the empty set is obtained when trying to solve the cycle $C_1 \bigoplus C_2$ for any of its constraints, what follows is a tighter constraint than (22):

$$x_3 + x_6 + x_{13} \leq 2. \tag{24}$$

Note that there are two other sets of region cycles for this block that would provide this constraint in a more straightforward way.

A feasible matching is found for workpieces in Fig. 1 in step 9 of the search process (see Table I). The associated GR graph appears in Fig. 7(a), the block it contains appears in Fig. 7(b), and the corresponding graph of basic cycles is shown in Fig. 7(c). In this case, the hypothesized assembly is found to be feasible; the three remaining DOF (two translational and one rotational) permit assemblying it.

B. Constraints of Nonintersection and Assembly Sequences

The assembly configuration problem tackled here, despite its static nature, has to be viewed as a combination of the following three different problems:

1) The *check interference problem*: It must be possible to detect intersections between objects in fixed positions. For instance, in the example of Fig. 1, the matching $\{x_9, x_{12}, x_{17}\}$ permits fixing the position of workpiece $\mathcal{B}_2$ with reference to $\mathcal{B}_3$. Thus, $\mathcal{B}_2$ and $\mathcal{B}_3$ must be checked to be interference free in this fixed position.
2) The *find space problem*: After complete constraint propagation on a GR graph without finding any inconsistency, it is possible that there remain DOF between the workpieces. Thus, it is necessary to find interference-free locations for workpieces. For instance, in the example of Fig. 1, the matching $\{x_{14}\}$ leads to a GR graph with two DOF. In this case, there exist only two discrete interference-free locations between $\mathcal{B}_2$ and $\mathcal{B}_3$.
3) The *find path problem*: It must be possible to ensure that there is a collision-free path to bring workpieces into contact from a situation in which they are sufficiently far apart.

The easiest way to deal with nonintersection constraints is by using a configuration space (c-space) representation [16]. The main problem concerning c-spaces is their construction because of the computational effort involved. Thus, it is very important to avoid their explicit representation or, at least, to use a lazy computation strategy.

Since clearances between workpieces in the assembly domain can be arbitrarily small, an exact representation of the boundaries of obstacles in c-space is required. Thus, the typification of nonintersection constraints followed in [5], framed within the predicate-based approach described in [2], seems to be the most suitable to this end.

The obvious extension of the method for dealing with nonintersection constraints is the ability to generate assembly sequences [11]–[13]. This would allow the system to distinguish those configurations for which there is a feasible assembly sequence from those for which, despite satisfying all previously generated constraints, there is no such a sequence.

Previous work on finding assembly sequences, given a final assembly, assume the three following hypotheses: 1) exactly two workpieces or subassemblies are joined at each time; 2) whenever workpieces are joined forming a subassembly, no later relative motion between them is carried out; and 3) the models of the assemblies represent explicitly the fastenings that bind one part to another. The last hypothesis is equivalent to the explicit representation of the features and the legal motions for any compatible pair we have taken.

Some precedence relations between assembly operations can be obtained directly from the way in which constraints on the DOF have been propagated. Each subassembly in an assembly sequence must have at least one DOF that permits disassemblying it. In the example of Fig. 7(b), solving both basic cycles for the shared constraint fixes the relative position of $\mathcal{B}_2$ with respect to $\mathcal{B}_3$; this implies that the aforementioned two workpieces must be mated before either $\mathcal{B}_1$ or $\mathcal{B}_4$ are inserted.

VI. Conclusions

We have presented a constraint-based algorithm to infer assembly configurations, as well as specialized techniques to deal with constraints on the DOF.

The algorithm described has been devised as an aid for assembly specification and automatic placing of workpieces in assemblies. It is well suited to deal with workpieces that can be described in terms of predefined three-dimensional features.

The correctness and completeness of the described algorithm relies on the correctness and completeness of the algorithms for dealing with spatial constraints. While correct, there are no algorithms for dealing with spatial constraints on the DOF that have been proved to be complete. Since it is always possible to decide whether two workpieces can be joined based on geometrical criteria, further research must focus on finding such algorithms.

The solutions to the problem can be seen as the configurations remaining after carrying out all the eliminations implied by the spatial constraints. The algorithm presented here produces the solutions one at a time, according to an optimality criterion.

The method has been devised so that it can be easily extended to deal with new constraint types. Thus, the obvious extension is the addition of all those constraints dependent on the working cell where the assembly task is to be carried out. These constraints would include those imposed by stability and support criteria between workpieces, and accessibility or grasping requirements. However, adding other constraints different from spatial constraints requires modifying the optimality criterion.

Appendix
Complexity Considerations

Proposition: The problem stated in Section IV-A is NP hard.

Proof: We find a reduction from the satisfiability problem to the problem of inferring feasible assemblies. The satisfiability problem can be stated as follows: Given a collection of clauses $C = \{c_1, c_2, \ldots c_n\}$ on a finite set $X = \{x_1, x_2, \ldots x_m\}$ of variables, find whether there exists a truth assignment for X that satisfies all the clauses in C.

We shall construct a set of workpieces (see Fig. 8) such that C is satisfiable if and only if there exists an optimum matching of their features that constitutes a feasible assembly. First a rectangular workpiece having n rows and $2m$ columns of potential holes is designed (see Fig. 8(a)), the columns corresponding to the variables and their logic negations in alternation. A hole is effectively drilled in the matrix position ij if and only if

$$x_{(j+1)/2} \in c_i, \text{ when } j \text{ is odd}$$

or

$$\overline{x}_{j/2} \in c_i, \text{ when } j \text{ is even.}$$

Next, m workpieces—one for each variable—are designed, having n holes each, as shown in Fig. 8(b). The two symmetric placements of each of these workpieces correspond to a variable and its negation.

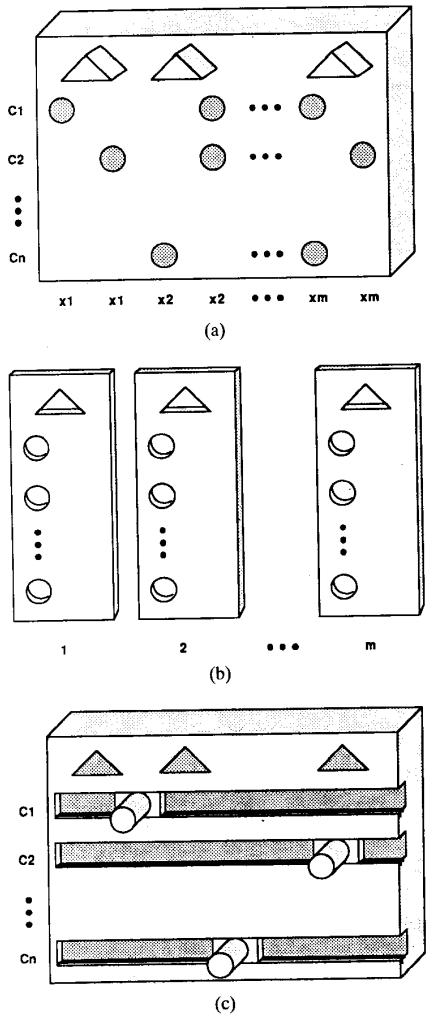


Fig. 8. Workpieces used in the reduction of SAT to the problem of inferring final assemblies: (a) Workpiece that codifies the clauses. (b) Workpieces that correspond to one variable each. (c) Workpieces corresponding to one clause each which ensure that, in a maximum matching, all clauses are satisfied.

Finally, a board and n sliding workpieces with one shaft each are designed (see Fig. 8(c)), which are aimed at ensuring that at least one hole for each clause is made to correspond with a hole for a given variable or its negation.

Note that shafts and triangular prisms are defined in such a way that they can be simultaneously mated to two other features, all the remaining matings being required to be one to one.

Thus, if a matching involving $4n + 2m$ arcs of the GF graph for this problem is found ($2n$ shaft-hole arcs, $2n$ groove-rectangular prism arcs, and $2m$ triangular prism-triangular hole arcs), then it is easy to derive a truth assignment for X that satisfies all the clauses in C. Conversely, if no such optimal matching leads to a feasible assembly (i.e., the maximum matching that corresponds to a feasible assembly has a lower

number of arcs), then there is no truth assignment for X that satisfies all the clauses in C. ■

Since all possible solutions already appear as vertices of the polytope defined by (4) and (5) at the very beginning of the optimization, the problem of obtaining the most promising hypothesis has lower complexity than the general integer linear programming problem. Thus, other algorithms different from the simplex would provide a solution at less computational effort [14]. Actually, many optimization problems on graphs can be formulated as integer programs. There exists a polyhedron naturally associated with any integer program, namely, the convex hull of all integer solutions. The existence of a polynomial algorithm for a combinatorial optimization problem is often related to finding a system of linear inequalities that define this convex hull [8].

Because of the NP hardness result and even if the most promising hypothesis could be found in polynomial time, the iteration to find a feasible assembly cannot presumably be carried out in polynomial time. All these considerations can be made independently of the complexity of testing the feasibility of a hypothesis. If this test could be carried out in polynomial time, then the problem would be NP complete.

ACKNOWLEDGMENT

We thank J. Canny from the University of California at Berkeley for his suggestions as well as some useful references.

REFERENCES

[1] B. G. Buchanan, G. L. Sutherland, and E. A. Feigenbaum, "Heuristic DENDRAL: A program for generating explanatory hypotheses in organic chemistry," in *Machine Intelligence 4*, B. Meltzer and D. Michie, Eds. Edinburgh, U.K.: Edinburgh Univ. Press, 1969, pp. 209–280.

[2] J. Canny, *The Complexity of Robot Motion Planning*. Cambridge, MA: MIT Press, 1988.

[3] N. Christofides, *Graph Theory. An Algorithmic Approach*. New York: Academic, 1975.

[4] N. Deo, *Graph Theory with Applications to Engineering and Computer Science*. Englewood Cliffs, NJ: Prentice-Hall, 1974.

[5] B. R. Donald, "Local and global techniques for motion planning," M.Sc. thesis, Massachusetts Institute of Technol., Cambridge, May 1984.

[6] R. S. Doshi, R. S. Desai, R. Lam, and J. E. White, "Integration of artificial intelligence planning and robotic systems with Airobic," presented at IMACS Conf. Expert Syst., Purdue Univ., West Lafayette, IN, Dec. 1988.

[7] S. E. Fahlman, "A planning system for robot construction tasks," *Artificial Intell.*, vol. 5, pp. 1–49, 1974.

[8] M. Grötschel, L. Lovász, and A. Schrijver, "The elipsoid method and its consequences in combinatorial optimization," *Combinatorica*, vol. 1, pp. 169–197, 1981.

[9] M. R. Henderson, "Extraction of feature information from three dimensional CAD data, Ph. D. dissertation," Purdue Univ., West Lafayette, IN, 1984.

[10] J. M. Hervé, "Analyse structurelle des mécanismes par groupe des déplacements," *Mechanism Machine Theory*, vol. 13, pp. 437–450, 1978.

[11] L. S. Homem de Mello and A. C. Sanderson, "A correct and complete algorithm for the generation of mechanical assembly sequences," *IEEE Trans. Robotics Automat.*, vol. 7, no. 2, pp. 211–227, 1991.

[12] ——, "Representations of mechanical assembly sequences," *IEEE Trans. Robotics Automat.*, vol. 7, no. 2, pp. 228–240, 1991.

[13] W. Jentsch and F. Kaden, "Automatic generation of assembly sequences," in *Artificial Intelligence and Information-Control Systems of Robots*, I. Plander, Ed. Amsterdam: North-Holland, 1984.

[14] N. Karmarkar, "A new polynomial-time algorithm for linear programming," *Combinatorica*, vol. 4, pp. 373–395, 1984.

[15] N. Koutsou, "Planning motion in contact to achieve parts mating," Ph. D. dissertation, Univ. of Edinburgh, Edinburgh, U.K., 1986.

[16] T. Lozano-Pérez, "Spatial planning: A configuration space approach," *IEEE Trans. Comput.*, vol. C-32, pp. 108–120, Feb. 1983.

[17] A. K. Mackworth, "Consistency in networks of relations," *Artificial Intell.*, vol. 8, pp. 99–118, 1977.

[18] A. K. Mackworth and E. C. Freuder, "The complexity of some polynomial network consistency algorithms for constraint satisfaction problems," *Artificial Intell.*, vol. 25, pp. 65–74, 1985.

[19] C. H. Papadimitriou and K. Steiglitz, *Combinatorial Optimization: Algorithms and Complexity*. Englewood Cliffs, NJ: Prentice-Hall, 1982.

[20] R. J. Popplestone, A. P. Ambler, and I. M. Bellos, "An interpreter for a language for describing assemblies," *Artificial Intell.*, vol. 14, pp. 79–107, 1980.

[21] F. Thomas, "Planificación de tareas robotizadas de ensamblaje basada en análisis de restricciones," Ph.D. dissertation , Polytechnic Univ. of Catalonia, Spain, June 1988 (in Spanish).

[22] F. Thomas and C. Torras, "Constraint-based inference of assembly configurations," in *Proc. IEEE Conf. Robotics Automat.* (Philadelphia, PA), Apr. 1988.

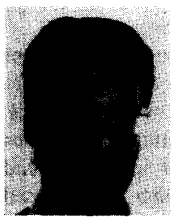
[23] ——, "A group theoretic approach to the computation of symbolic part relations," *IEEE J. Robotics Automat.*, vol. 4, pp. 622–634, Dec. 1988.

[24] R. B. Tilove, "Extending solid modeling systems for mechanism design and kinematic simulation," *IEEE Comput. Graphics Applicat. Mag.*, vol. 3, pp. 9–19, May-June 1983.

[25] C. Torras, "Planning for problem solving: A survey," in *AI and Expert Systems in Scientific Computing*, R. M. Huber, C. Kulikowski, J. M. Davis, and J. P. Krivine, Eds., *IMACS Transactions Series*. Basel, Switzerland: Baltzer, 1989.

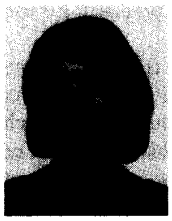
[26] J. M. Valade, "Geometric reasoning and synthesis of assembly trajectories," *Int. J. Robotics Res.*, to be published.

[27] J. R. Woodwark, "Some speculations on feature recognition," in *Geometric Reasoning*, D. Kapur and L. Mundy, Eds. Cambridge, MA: MIT Press, 1989.



Federico Thomas was born in Barcelona in 1961. He received the B.Sc. degree in telecommunications engineering from the Escuela Técnica Superior de Ingenieros de Telecomunicación de Barcelona at the Polytechnic University of Catalonia in 1984 and the Ph.D. degree in computer science from the Polytechnic University of Catalonia, Barcelona, in 1988.

Since 1985, he has been with the Institute of Cybernetics. At present he holds the position of Associate Professor in the Spanish Council for Scientific Research.



Carme Torras was born in Barcelona in 1956. She received the M.Sc. degree in mathematics from the Universitat de Barcelona in 1978, the M.Sc. degree in computer science from the University of Massachusetts, Amherst, in 1981, and the Ph. D. degree in computer science from the Universitat Politècnica de Catalunya in 1984.

Since 1981, she has been with the Institut de Cibernètica, Barcelona, conducting research on robot motion planning and neurocomputing. She is author of the monograph "Temporal-Pattern Learning in Neural Models" (*Lecture Notes in Biomathematics*, Springer-Verlag, 1985) and a co-author of the book *Robótica Industrial* (Marcombo, 1986). At present, she holds the position of Professor of Research in the Spanish Council for Scientific Research and she teaches doctoral courses in the fields of robotics and artificial intelligence at the Universitat Politècnica de Catalunya.