Journal of Manufacturing Systems 54 (2020) 196–203

Contents lists available at ScienceDirect

Journal of Manufacturing Systems

journal homepage: www.elsevier.com/locate/jmansys

A constraint model for assembly planning

Csaba Kardos[a,b,*], András Kovács[a], József Váncza[a,b]

[a] EPIC Center of Excellence in Production Informatics and Control, Institute for Computer Science and Control, Budapest, Hungary
[b] Department of Manufacturing Science and Technology, Budapest University of Technology and Economics

ARTICLE INFO

Keywords:
Assembly planning
Optimization
assembly features
constraint programming

ABSTRACT

The importance of computer-aided process planning (CAPP) for assembly is widely recognized, as it holds the promise of efficient and automated construction of solutions for a complex, geometrically, technologically, and economically constrained planning problem. This complexity led to the introduction of decomposition approaches, separating the macro-level planning problem that oversees the complete assembly process from the various micro-level problems that look into the details of individual assembly operations. The paper introduces a constraint model for solving the macro-level assembly planning problem based on a generic feature-based representation of the product and the assembly operations involved. Special attention is given to capturing the feedback from micro-level planners expressed in the form of feasibility cuts, and hence, to the integration of the approach into a complete CAPP workflow. Results on three case studies from different industries are also presented to illustrate the practical applicability of the approach.

1. Introduction

Assembly is the ultimate step from the idea of products conceived by design to their form, structure and functions realized by production. Planning all the details of this transition requires the efficient use of information and knowledge from a number of different sources related to product design, parts manufactured, assembly technologies and processes as well as resources such as tools, fixtures, grasping and handling devices, human and robotic operators. The generated assembly plans should meet a rich set of requirements and comply with criteria like minimal cycle time, resource and energy efficiency, ergonomics to name only the most important ones. Primarily, the constraints are of geometric nature, as parts and subassemblies have to be moved and fit, fixtures and tools have to be applied as the assembly process advances in a more and more densely populated space.

No wonder assembly planning poses a number of intriguing questions for production engineering, from the very inception of the field: How to interpret a product model with the backdrop of the actually available assembly technology and resources? How to identify tasks which are potentially executable and how to define their appropriate ordering and resource assignment? How can one maintain, in the course of the assembly process, geometrical feasibility among any objects involved, let they be parts, subassemblies, tools, fixtures or any other elements of the production environment? In general, how to solve a problem when there is no monopoly of assembly planning knowledge and requirements may easily be proven irreconcilable?

The importance of automated computer-aided process planning (CAPP)–also in assembly–was recognized early [1]. It became also clear and confirmed time and again since, that CAPP in general can be solved only by the application of well-proven decomposition principles [2]. While exploiting locality led to various feature-based models, hierarchical decomposition resulted in macro-level planning, which concentrates on combinatorial decisions as for ordering and resources of tasks, and micro-level planning activities, which are responsible for path planning, fine-tuning of technological process parameters, generation of work instructions or robot codes.

The goal of this paper is to present a *constraint-based model* for macro-level assembly planning that provides a resolution to the above issues. As constraint models in general, it has a strong representation power to capture all important aspects of assembly planning. The model directly supports making decisions over the statement of the actual planning problem including assembly features and tasks, subassemblies, the ordering and resource assignment of tasks in a least-commitment manner. It is open to incorporate new constraints on the fly, as demanded by the micro-level analysis of partial solutions. Specifically, by means of generalized precedence constraints the model facilitates making new statements over the ordering relations of tasks and the resources assigned to them. With the progress of the assembly planning process a product model containing basically geometric

* corresponding author.
E-mail addresses: csaba.kardos@sztaki.hu (C. Kardos), andras.kovacs@sztaki.hu (A. Kovács), vancza@sztaki.hu (J. Váncza).

https://doi.org/10.1016/j.jmsy.2019.11.007
Received 28 July 2019; Received in revised form 30 September 2019; Accepted 15 November 2019
Available online 28 December 2019

information of parts and their relations is interpreted and enriched step by step with pieces of technological knowledge. The model assists a cautious approach to planning: if it gets over-constrained, the actual problem has indeed no solution, whereas one can be sure that a final plan, if it passes all micro-level evaluations, completely complies with the relevant domain knowledge and is optimal according to the given criterion. We focus here on the aspect of modelling and leave the solution of assembly planning problems to powerful constraint programming engines.

In what follows, Section 2 discusses the related literature, then Section 3 presents the generic planning workflow. Section 4 focuses on the statement of the constraint-based assembly planning problem, while the formal model of macro-level planning is presented in Section 5. According to the workflow, this model is extended with new constraints generated by micro-level evaluation as explained in Section 6. The results of computational experiments are summarized in Section 7, while Section 8 concludes the paper.

2. Related work

Traditionally, three main subproblems of assembly planning are distinguished in the literature, namely: Assembly Line Balancing (ALB), Assembly Path Planning (APP) and Assembly Sequence Planning (ASP). In addition to their different objectives, the different subproblems are usually separated by the applied representation of the problem as well [3]. There are a number of papers which focus on delivering solutions to one of the subfields, however, only a few integrated approaches exist [3–5].

In the environment of a single workcell, ALB can be assumed to be out of the scope and thus ASP and APP are subjects of solution efforts. ASP is typically formulated as a combinatorial optimization problem, while for APP solutions are usually produced by reasoning on a detailed geometrical model [6]. ASP is generally considered to be an NP-hard problem and therefore numerous heuristics and soft computing methods have been suggested to solving it, but classic optimization tools are also often applied [5,7–9]. Minimizing the changeovers, the number of assembly directions or the required time for the assembly are very common objective functions of ASP [5,10].

There are works on ASP which also consider the geometric feasibility of the resulting sequence. A general approach for combining ASP with geometric reasoning is to construct a search space which is assumed to contain all the relevant geometrical and technological constraints. A key aspect here is the definition of the search space. A seminal paper tackles this problem [11], aimed at generating all feasible assembly sequences for a product. In [12] and [13] the concept of directional and non-directional blocking graphs were used in order to represent the geometrical constraints. In [14] the stereographical projections of parts are used to include the collision space into the assembly sequencing. The work presented in [15] generates assembly sequences directly from the CAD models, which can be later used in optimization. In [16], part interaction clusters are defined to generate the precedence constraints from geometrical data. The sequencing and planning is carried out in this search space. In [17], the geometric models of assemblies are translated into an attributed part layout graph in order to identify functional subassemblies, which provides the basis for an ASP model using particle swarm optimization. Another approach that uses subassembly identification from 3D models is presented in [18], which proposes the concept of disassembly interference graph and uses this information to generate feasible assembly sequences. The work introduced in [19] discusses generating part precedence diagrams automatically for assembly planning considering tool assignment and changeovers. By extracting contact relations between parts, [20] proposes an approach to find feasible disassembly paths for parts, thus obtaining a feasible assembly sequence. In [21], by identifying and removing the standard elements, a simplified assembly model is analysed by collision detection to generate all feasible assembly sequences.

Features are, also in assembly, the most traditional and broadly used concepts for matching means and ends of production [8]. Assembly features offer a representation which can contain the information required for defining and solving the macro-level optimization problem of ASP, but at the same time they can also represent the geometric data required for building up a detailed micro-level world necessary for APP [22,23]. Domain knowledge required for the correct realization of assembly features can be captured in terms of function blocks [24], and this approach can be extended to the adaptive and distributed control of assembly operations, too [25].

A hierarchical approach is presented in [26] for finding the optimal assembly sequence in 2D, which maximizes the assembly angles during the assembly. The set of feasible solutions is determined by analysing geometrical accessibility. In [27] a genetic algorithm is applied to minimize the number of assembly direction changes and the total stress for flexible parts. An ASP solution is presented in [28], which first extracts precedence constraints from the geometrical models and then a search loop is executed.

In [10] it is stated that a combination of a feature based model extended with expert rules can deliver more comprehensive results for CAPP. In [7], based on the lessons of a literature review, a general optimization scheme for ASP is recommended as an integral loop in any assembly planning process. Having a loop without external feedback, however, assumes that every information is available for conducting a successful search and leaves no room for injecting constraints back from a more detailed evaluation, which due to its complexity can not be part of the original search. In [29] a constraint programming based approach is introduced for solving CAPP in sheet metal bending, where the solver communicates with external experts through rule-based feedbacks, thus reducing the domain of decision variables in each iterative solution cycle. The approach of using external experts also appears in [30], where a hierarchical decomposition for task and motion planning is applied in order to reduce and postpone the execution of costly calculations. A similar approach also appears in [31], where a “logical layer” is applied for ASP and a “physical layer” is used for APP.

In [8,32] the authors already presented a feature-based hierarchical workflow for solving the assembly planning problem by the integration of sequencing, resource assignment and geometric validation, by applying a Benders decomposition scheme [33]. The macro-level solution was built around a mixed integer programming (MIP) solver for which disjunctive rules provided the feedback in each solution loop. A strong assumption of these works was that all assembly features to be realized were given in the input, and they had to form a tree-structured liaison graph. In the approach presented below, this assumption is lifted.

3. Feature-based assembly planning workflow

The assembly planner presented in the paper operates in an iterative, hierarchical, mixed-initiative CAPP workflow, which starts from the models of the assembled product and the applicable resources, and ends with the generation and post-processing of work instructions (Fig. 1). Here, the paper's main focus is set on the macro-level planning model. Still, in order to better position the approach and to highlight its relationship to other planning steps, the complete workflow and its key concepts are introduced briefly below.

The workflow starts from the geometric models of the parts composing the assembly, which are rigid, tolerance-free 3D objects characterized by their geometrical model. They are either individual parts, or *composites* merged from multiple related parts that must be assembled at the same time, using identical resources (e.g., multiple identical, parallel-axis screws joining the same parts). The application of composites helps reducing the size of the planning problem.

With parts positioned in their assembled state, *connectivity analysis* between them can be executed. The physical connections between the parts define the *connectivity graph*, whose nodes are the parts and an edge between two nodes denotes a contact between the two geometries

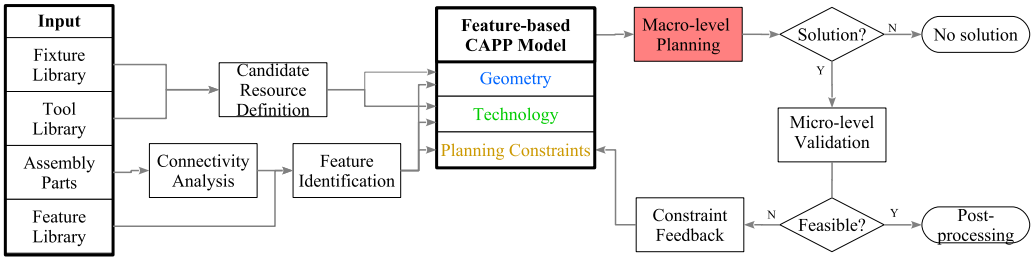


Fig. 1. The proposed CAPP workflow. The focus of the paper is on macro-level planning.

[32]. However, physical connectivity alone does not characterize fully the assembly operations, hence the feature-based assembly planning model is completed by technology-specific feature parameters, such as torque, lead or threaded depth for screwing features [34].

The connectivity graph provides input for *feature identification*. A feature describes how two parts can be assembled together, by defining the relative movement between the parts and the applicable resources (e.g., fixtures, tools). Hence, a feature corresponds to an edge in the connectivity graph with meaningful technological content. However, not every edge of the connectivity graph belongs to an assembly feature: e.g., two geometries touching at their edges cannot be joined by a feature. Consequently, feature identification assigns assembly features to some of the connectivity edges. This defines the so-called *liaison graph*, over the same nodes as the connectivity graph, with a subset of the edges.

Accordingly, each feature realizes exactly one edge in the connectivity graph *directly*. Nevertheless, the same feature may realize further edges *indirectly*: if the involved parts take place in previously constructed subassemblies, then the feature realizes all the connectivity edges between the parts of the two sub-assemblies.

The assembly process is complete when every edge in the connectivity graph is realized, directly or indirectly. In particular, the assembly of K parts can be completed with directly realizing $K - 1$ edges of the connectivity graph that form a spanning tree of the liaison graph.

Having a feature-based problem instance defined, the workflow follows with the solution loop, where first a macro-level combinatorial optimization problem is solved. This involves the selection of the features to be realized directly, their sequencing, as well as the assignment of resources. Its result is evaluated by detailed micro-level validation. Micro-level validation either fails and adds new constraints to the feature-based model (and thus to the macro-level), which ensure that the same failure cannot occur in subsequent iterations, or it succeeds and then the workflow is finished with post-processing.

This feature-based planning workflow is an improved version of that in [32]. A key new idea is reducing the necessary expert input to a pairwise part-to-part analysis during feature identification, which is more comprehensible for human experts. This is made possible by allowing an arbitrary structure for the liaison graph, instead of the previous tree structure. The benefit of this generalization is illustrated on the sample assembly in Fig. 2 a. The liaison graph has exactly the same edges as the connectivity graph (Fig. 2 b). However, an ill-defined spanning tree of this liaison graph can render the planning problem infeasible, as it is shown on the upper version of the liaison graph: features A-B and A-C mutually block each other, end hence, there is no feasible sequencing for them. Yet, a different spanning tree of the liaison graph, shown in the lower part of Fig. 2 b, leads to a feasible planning problem. In the previous approach, human decision was required to construct the liaison graph as a spanning tree of the connectivity graph, thus making room for such errors. These errors are now automatically avoided by the letting the solver select the features to realize.

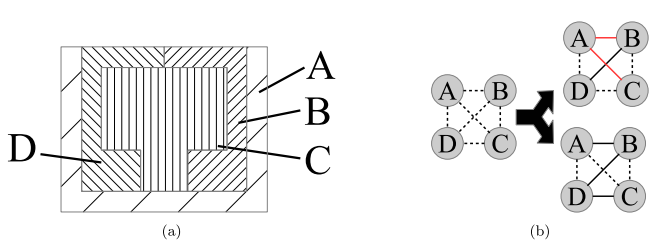


Fig. 2. A simple assembly with four parts (2a) illustrates how an ill-defined spanning tree (2b) can result in an unsatisfiable problem: realizing A-C feature blocks A-B and vice versa.

4. Problem statement

The approach proposed in the paper minimizes the total assembly time of the feature-based assembly planning (sequencing and resource assignment) problem and thus providing a solution to the macro-level planning problem of the iterative, hierarchical CAPP workflow. The elements of the resulting assembly sequence are represented as *tasks* where each task is composed of:

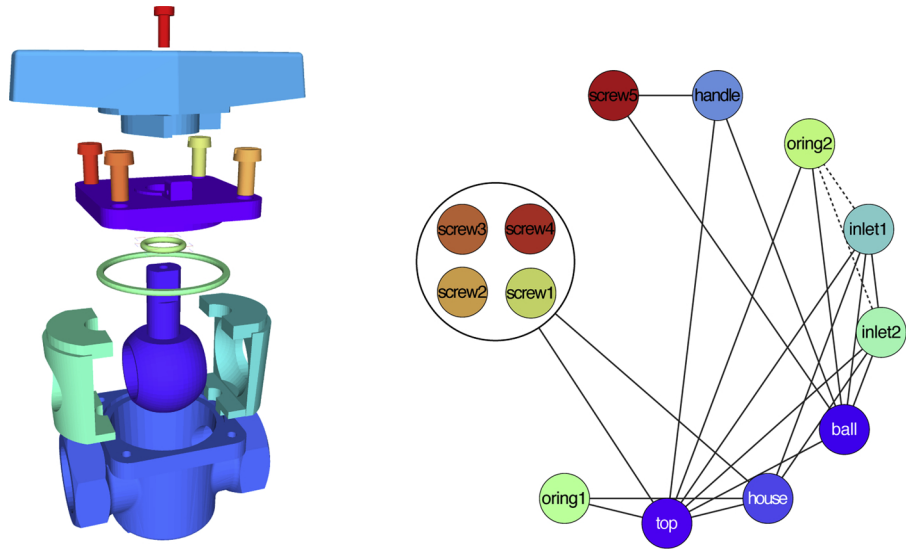
- the corresponding assembly feature,
- an assigned tool,
- an assigned fixture.

Taking two-handed assembly processes an assembly feature is $F(\omega, p, q, \theta, tc, fc)$ where ω is the feature type (placing, insertion, and screwing are handled in the current implementation), p and q are the two parts joined by the feature, while θ defines the motion required to join parts p and q. The set of candidate tools and candidate fixtures are denoted by tc and fc, respectively.

The input of assembly planning is a feature-based problem instance (see Fig. 1) and in the macro-level the following assumptions are made:

1. The assembly features represent two-handed operations.
2. The assembly process is monotonous.
3. There is a set of applicable tools and fixtures specified as candidate resources for each feature.
4. Each fixture can grasp a specific part. When that part takes place in a subassembly, the fixture can hold that subassembly up to a given weight limit.
5. The duration of realizing an assembly feature is independent from its position in the plan and the assigned resources.
6. The changeover times of fixtures and tools are given and fixed.

In addition, the micro-level geometrical validation applied in the paper assumes that parts are rigid, tolerance-free 3D objects, which allows using collision detection for the validation of the plan. Removing assumption 1 would require a slight generalization of the macro-level planning model, since features would correspond to hyper-edges (rather than classical edges) in the liaison-graph. Assumption 2 is necessary to maintain an upper bound on the length of the assembly plan. Assumptions 3 and 4 capture resource capacities in a realistic way. Replacing assumptions 5 and 6 with sophisticated functions for determining the durations of tasks and changeovers from the actual



(a) (b)

Fig. 3. Exploded view of the working example: a ball valve (3a). Connectivity graph of the ball valve (3b), where dashed lines mean connection without feature, solid lines indicate features. The grouped node containing screws 1-4 represents a composite.

Table 1
Notation applied in the macro-level problem

Parameters	
p_l, q_l	Two parts joined by feature l
r_l	The duration of feature l
y_k	Weight of part k
tt_m	Changeover time for tool m
ft_n	Changeover time for fixture n
g_n	The part grasped by fixture n
w_n	Weight limit of fixture n
$tc_{l,m}$	Indicates if tool m is a candidate tool for feature l
$fc_{l,n}$	Indicates if fixture n is a candidate fixture for feature l
Decision variables	
π_l	Position of feature l in the assembly sequence
ϕ_i	Feature at position i in the assembly sequence
τ_i	Tool at position i in the assembly sequence
χ_i	Fixture at position i in the assembly sequence
$\rho_{i,p,q}$	Indicates if part p and q are connected at position i

subassemblies is a relevant direction for future research.

Taking the above assumptions, minimizing the total assembly time means minimizing the time required for realizing the assembly sequence. The time has two components here: the time required for completing each of the selected features and the changeover time required for using the assigned tools and fixtures. For warranting the feasibility of the solution, the following constraints have to be satisfied:

- Every physical connection between the parts have to be realized. Accordingly, the features selected for direct realization have to form a spanning tree over the liaison graph of the assembly.
- The technological constraints (fixture weight limit, feature-tool and feature-fixture compatibilities) have to be fulfilled.
- The solution must be feasible on the micro level as well, which must be ensured by satisfying all feasibility cuts generated by micro-level validation.

4.1. Working example

In order to illustrate the proposed solution approach, a ball valve is used as a working example throughout the paper. The geometric model of this working example is shown in Fig. 3 a: it consists of 13 parts and 16 features. The connectivity graph of the assembly is shown in Fig. 3 b. The connectivity graph shows every physical connection between the parts as an edge, however, this does not necessarily result in a meaningful connection: e.g., between *oring2* and the two inlets the connection would not imply a technologically meaningful assembly operation. This means that the liaison graph is a subgraph of the connectivity graph, with dashed edges left out from the former. The screws holding the top in its place (screw1..4) are identical with the same feature parameters, and therefore they are bound together to form a composite part.

The working example includes three fixtures, where one is the human hand, without a geometry, and others are fixtures grasping the house and the top. The usage of the human hand is restricted by a strict weight limit which does not allow holding the whole assembly.

The working example uses the same geometric models as in [32]. However, due to the lifted earlier assumption that the liaison graph of the assembly is a tree, this version includes more features. As a consequence, the required input is more generic and easier to create for experts, but the problem is combinatorially more challenging.

5. Description of the CAPP model

The product must be assembled from K parts $\{p_k : k = 1, \ldots, K\}$, which can be either individual parts or composites during preprocessing (see Table 1 for the applied notation). Each part has a weight attribute $\{y_k : k = 1, \ldots, K\}$. The parts are joined by L features $\{f_l : l = 1, \ldots, L\}$. Each feature joins two parts $\{p_l, q_l : l = 1, \ldots, L\}$. For realizing the features there are M available tools $\{t_m : m = 0, \ldots, M\}$ and N available fixtures $\{f_n : n = 0, \ldots, N\}$, including a dummy tool t_0 and a dummy fixture f_0. Each tool and fixture has a changeover time indicating the time required to switch to the specific tool or fixture $\{tt_m : m = 0, \ldots, M\}$ and $\{ft_n : n = 0, \ldots, N\}$, respectively. For each fixture there is a weight limit on the sum of the weight of the grasped part and the parts attached to it $\{w_n : n = 0, \ldots, N\}$ where $w_0 = 0$. Each fixture is able to grasp a specific part $\{g_n \in \{0, \ldots, K\} : n = 0, \ldots, N\}$, and $g_0 = 0$.

For each feature the candidate sets of tools and the candidate sets of fixtures specify what are the allowed tools and fixtures for realizing that feature: $\{tc_{l,m} \in \{0, 1\} : l = 1, \ldots, L, m = 0, \ldots, M\}$ and $\{fc_{l,n} \in \{0, 1\} : l = 1, \ldots, L, n = 0, \ldots, N\}$.

For solving the two-handed, monotonous CAPP problem with K *unique parts*, the number of required features (i.e., positions in the plan) is $K - 1$. The goal of the CAPP model is to find an optimal task

sequence and resource assignment, which is given by finding the values of the following decision variables:

- position of a given feature: $\{\pi_l \in \{0, ..., K-1\} : l = 1, ..., L\}$, where $\pi_l = 0$ means that f_l is not realized directly.
- feature at a given position: $\{\phi_i \in \{1, ..., L\} : i = 1, ..., K-1\}$
- tool at a given position: $\{\tau_i \in \{0, ..., M\} : i = 0, ..., K-1\}$
- fixture at a given position: $\{\chi_i \in \{0, ..., N\} : i = 0, ..., K-1\}$
- connectivity that indicates if two parts (p and q) are connected after executing a feature at the ith position of the plan: $\{\rho_{i,p,q} \in \{True, False\} : p = 1, ..., K, q = 1, ..., K, i = 0, ..., K-1\}$

5.1. Model objective and constraints

The objective is minimizing the total time of completion, which means minimizing the sum of feature durations and changeover times: $\Sigma(r_{\tau_i} + (\tau_i \neq \tau_{i-1}) \cdot tt_{\tau_i} + (\chi_i \neq \chi_{i-1}) \cdot ft_{\chi_i})$.

The solution has to satisfy the following constraints. First of all, the integrity of the solution is ensured by:

- *alldifferent_except_0* constraint is applied to π_l to enforce that each valid (i.e., non-zero) position is only assigned to one feature.
- *alldifferent* constraint is applied to ϕ_i to enforce that every feature is realized at most once.
- channelling between π_l and ϕ_i enforces that $(\pi_l = i) \leftrightarrow (\phi_i = f_l)$, $\forall i, l$ if $\pi_l \neq 0$

In order to model the changeover needed for the first tool and fixture at zero position the dummy tool and fixture are assigned: $\tau_0 = 0$ and $\chi_0 = 0$, moreover $\tau_j \neq 0$, $\forall j \neq 0$ and $\chi_j \neq 0$, $\forall j \neq 0$.

As a redundant constraint, it is specified explicitly that part-to-part connections are symmetric: $\rho_{i,p,q} = \rho_{i,q,p}$, $\forall i, p, q$. After the completion of the last task, each pair of parts must be connected: $\rho_{K-1,p,q} = 1$, $\forall p, q$. Before the first task, each part is connected only to itself: $\rho_{0,p,q} \leftrightarrow (p = q)$, $\forall p, q$. A direct connection between two parts at a position means that the feature at that position joins those two parts. In general, two parts are connected at a given position in the plan if, directly or indirectly, they had been connected earlier by some feature:

$$\rho_{i,p,q} \leftrightarrow (\rho_{i-1,p,q} \vee (\rho_{i-1,p_{\phi_i},p} \wedge \rho_{i-1,q_{\phi_i},p}) \vee (\rho_{i-1,q_{\phi_i},q} \wedge \rho_{i-1,p_{\phi_i},p})) \quad \forall i, p, q \tag{1}$$

Here, the first term on the r.h.s. of the logical biconditional means that parts p and q have already been connected in the previous position of the plan. The second and third terms encode that the two parts have already been connected to the parts assembled directly by the feature in position i of the plan.

At every position in the plan, the part grasped by the assigned fixture must be connected to one of the parts directly involved in the realized feature (which includes the case that the involved part itself is grasped). For selecting an applicable fixture, the following conjunction must hold (which are illustrated in Fig. 4):

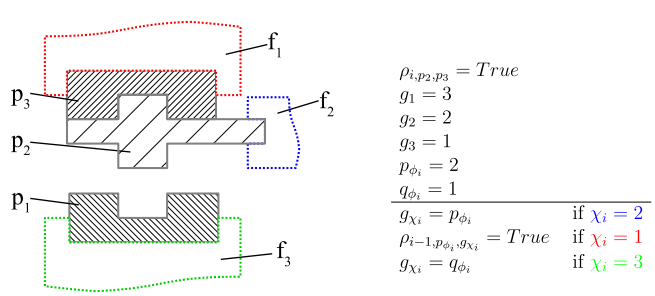


Fig. 4. An illustration of how grasping constraints take effect if different fixtures are applied: fixtures f_1, f_2, f_3 grasp parts p_3, p_2, p_1, respectively. In each case a different term of (2) yields true.

$$(\rho_{i-1,p_{\phi_i},g_{\chi_i}}) \vee (\rho_{i-1,q_{\phi_i},g_{\chi_i}}) \qquad \forall i \tag{2}$$

The assigned fixture must be among the candidate fixtures: $fc_{\phi_i,\chi_i} = 1$, $\forall i$ and the same for candidate tools: $tc_{\phi_i,\tau_i} = 1$, $\forall i$.

Finally, it has to be warranted that the total weight of the grasped part and the parts connected is below the weight limit of the assigned fixture: $\sum_k \rho_{i,g_{\chi_i},p_k} \cdot y_k \leq w_{\chi_i}$, $\forall i$.

5.2. Feasibility cuts

Feedback from micro-level validation takes the form of a set of *feasibility cuts*, each cut expressing a generalized precedence constraint. This constraint takes the form $\{\alpha_d, \{\beta_{d,1}, ..., \beta_{d,e}\}, \gamma_d, \delta_d\}$, for each cut $d \in D$, where:

- α_d is the preceding feature;
- $\{\beta_{d,e} \in 1, ..., L : e = 1, ..., E_d\}$ is the set of E_d succeeding features;
- $\gamma_d \in 0, ..., N$ is a fixture; and
- $\delta_d \in 0, ..., M$ is a tool in the generalized precedence constraint.

This generalized precedence constraint requires that either preceding task α_d precedes at least one of the succeeding tasks $\beta_{d,e}$ in the plan (3), or the preceding task α_d is left out of the plan (4), or one of the succeeding tasks $\beta_{d,e}$ is not present (5), or a different fixture (6) or a different tool (7) is used. As a special case, the dummy fixture $\gamma_d = f_0$ (respectively, the dummy tool $\delta_d = t_0$) can be used to encode that modifying the fixture (tool) is not an option to satisfy the constraint:

$$(4) \vee (5) \vee (3) \vee (6) \vee (7), \quad \forall d$$, where

$$\sum_e (\pi_{\alpha_d} < \pi_{\beta_{d,e}}) > 0 \tag{3}$$

$$\pi_{\alpha_d} = 0 \tag{4}$$

$$\sum_e (\pi_{\beta_{d,e}} = 0) > 0 \tag{5}$$

$$(\chi_{\pi_{\alpha_d}} \neq \gamma_d) \wedge (\gamma_d \neq 0) \tag{6}$$

$$(\tau_{\pi_{\alpha_d}} \neq \delta_d) \wedge (\delta_d \neq 0) \tag{7}$$

6. Micro-level evaluation

Micro-level evaluation provides the feedback for closing the CAPP solution loop. Its purpose is to verify the geometric feasibility of each movement defined by the macro-level plan, and in case of any failure, to generate constraints that preclude the repeated occurrence of the same failure. The sequence calculated on the macro level is used for building up the *Task-specific Liaison Graph* (TLG) for each task in the plan. It contains the parts that are already assembled, the applied fixture and the tool as nodes. The edges between the parts are the features directly realized in the previous and in the current assembly steps. Fixturing determines which part (and those already assembled to it) are grasped by the fixture and thus will be the *base components*, while the other part (and those attached to it) will be the *moved components*. In the TLG, the fixture node is connected to the grasped part and the tool is connected to the feature's moved part. Since the TLG only contains the directly realized features, it is a tree, and there exists only one path between any two nodes.

The geometric feasibility check of each feature in the macro-level plan is performed in two separate phases (see details in [32,35]):

1 For the *local motion* defined by the feature, which takes the moved component from its near position to the final relative position w.r.t. the base component.
2 For the *approach motion*, which brings the moved component from its remote location to the near position.

Since the local motion is perfectly defined by the feature, the colliding pairs of objects (parts, fixture, tool) can be unambiguously identified. Moreover, each colliding pair of objects determines an *invalid path* in the TLG that connects the nodes corresponding to two objects involved. Then, the generated cut encodes that this invalid path must not reoccur in any macro-level solution. Namely, in the format specified in 5.2, the preceding feature α is the current feature, the set of succeeding features $\{\beta_1, \ldots, \beta_e\}$ is the set of features in the invalid path (with the exception of α), whereas the tool γ and the fixture δ are the resources assigned to the current task (only if they take part in the collision).

A similar approach is taken for the verification of the approach motion, with the substantial difference that the existence or the lack of a collision-free approach motion can only be verified by solving a path planning problem. Moreover, the infeasibility of the path planning problem does not lead to a specific collision, and hence, the returned feasibility cut is a generic expression which states that the current task cannot be executed with the same (or larger) set of base and moved components and the same resources. In the generic format of 5.2, α is the current feature, the set $\{\beta_1, \ldots, \beta_e\}$ is the set of all features in the TLG (except for α), whereas γ and δ are the assigned resources. Since this cut contains more terms than the cut generated for local motions, it means a substantially weaker constraint on the macro-level plan. The details of the geometric evaluation are discussed in [32,35]. Examples of cuts generated from various micro-level conflicts are shown in Fig. 5 .

The micro-level evaluation generates *all* possible cuts based on the verification of the local motion of all features in the plan, since they can be computed efficiently. In contrast, time consuming path planning is performed for the approach motions only if all local motions in the plan are collision-free, up to the first failure.

7. Case study and experiments

For evaluating the presented approach, two additional case studies were selected further to that of the working example: an automotive supercharger assembly and a pneumatic cylinder assembly. Fig. 6 shows these assemblies, while problem sizes are summarized in Table 2 . The *raw* number of features and parts reflects the original CAD input, whereas the *processed* values characterize the model size after merging corresponding parts into composites. It can be noticed that besides the obvious geometric dissimilarities, the three cases differ also in their complexity. The supercharger assembly contains the highest number of parts and edges in the connectivity graph. However, during the analysis and instance definition, most of the connections can be discarded as they do not represent a meaningful feature. Moreover, several features can be grouped together as composites and therefore the resulting number of features from the macro-level perspective is only 17 for 18 parts. This means that every feature has to be assigned in the resulting sequence. In contrast, the ball valve and the cylinder examples have less parts and connections, but the number of features is higher in both cases, meaning that there are alternative ways to realize the assemblies and in the solution not every feature will be assigned to a position in the plan. A part of the fixtures and tools are characterized by geometries as well; collision detection on the micro level is performed only for these resources. Other, special resources, such as a human hand, do not have a solid geometry assigned, and they are disregarded on the micro level.

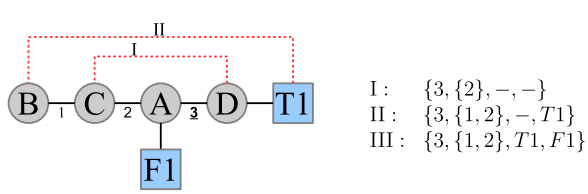


Fig. 5. Examples of cuts generated on the sample assembly of Fig. 2 a during the verification of feature 3. A collision between parts C and D during the local motion results in cut I, stating that feature 3 must precede feature 2, or one of them must be skipped. A collision between part B and tool T1 leads to cut II, with more preceding features and a tool in the cut. The failure of the approach motion results in cut III, which contains all previous features and all the assigned resources.

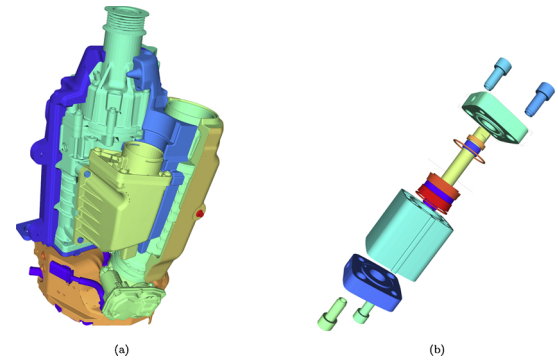


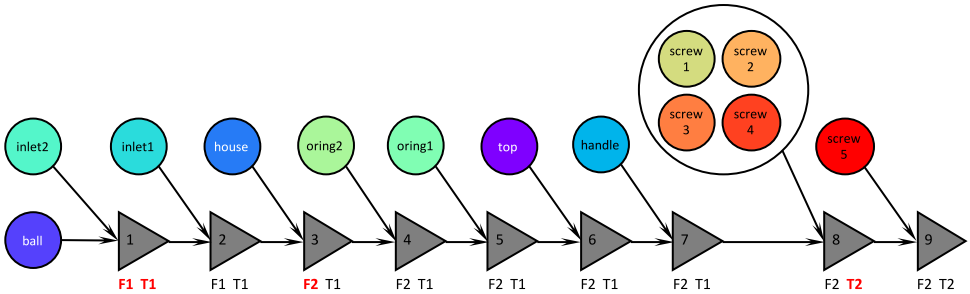
Fig. 6. Real-life case studies: automotive supercharger (6a) and a pneumatic cylinder (6b).

Table 2
Problem size and computational results for each case study.

Case study	Supercharger	Ball valve	Cylinder
	Problem description		
Parts (raw)	29	13	18
Parts (processed)	18	10	16
Connections	55	27	30
Features (raw)	28	19	26
Features (processed)	17	16	24
Init. constraints	5	2	2
Fixtures (w/ geom)	5 (0)	3 (2)	2 (0)
Tools (w/ geom)	3 (1)	2 (0)	2 (0)
Triangles	1300k	100k	70k
	Computational results		
Iterations	5	12	16
Macro-level planning			
total [s]	257.36	132.85	2120.44
per iteration [s]	51.47	11.07	132.53
Micro-level validation			
total [s]	9.21	5.15	29.67
per iteration [s]	1.84	0.43	1.85
Constraints			
total	40	58	121
per iteration	8	4.83	7.56

The CAPP solution loop was implemented using the Python programming language (v3.6) for building up a frame, which calls the macro-level solver developed using the MiniZinc (v2.2.3) constraint modelling language [36] with the Google's OR-Tools [37] open source optimization solver (v7.0, CP-SAT core, with three threads). The experiments were executed in a containerized environment [38] with three CPU cores and 2 GB of memory allocated from a regular computer (2015 Intel i7 processor with 4 cores). The experiments showed that finding an optimal solution is usually quick, however, proving its optimality can take an order of magnitude longer time. In order to control the computational load a time limit of 120 seconds was applied for finding an optimal macro solution per iteration. Since in the solution loop the macro solutions have to be validated on the micro level, it is reasonable to apply this time limit that also reduces the time spent on proving the optimality of solutions, later possibly dropped on the micro level. It is noted that macro-level planning times displayed in Table 2 also include the compilation of the MiniZinc model, and hence, may slightly exceed the specified time limit.

The results of all three case studies are shown in Table 2 . Even though the supercharger case study has the highest number of components, this problem instance can be solved relatively quickly, with




```
feature_at_position = [15, 8, 9, 2, 3, 1, 11, 4, 7]
fixture_at_position = [0, 5, 5, 2, 2, 2, 2, 2, 2, 2]
tool_at_position = [0, 1, 1, 1, 1, 1, 1, 1, 2, 2]
-------------------------------------------------------
"Put Ball into Fixture 1"
"Take Tool 1"
       1. "Assemble Ball and Inlet 2",
       2. "Assemble Ball and Inlet 1",
"Put House into Fixture 2",
       3. "Assemble House and Inlet 1",
       4. "Assemble Ball and Oring2",
       5. "Assemble House and Oring1",
       6. "Assemble House and Top",
       7. "Assemble Top and handle",
"Take Tool 2"
       8. "Assemble House and Top with 4 screws",
       9. "Assemble Ball and Handle with a screw"
-------------------------------------------------------
total_assembly_time = 56
-------------------------------------------------------
```

Fig. 7. The result of macro planning represented as a tree. The circular nodes are the parts, with colors matching to Fig. 3. The applied fixtures (F) and tools (T) are displayed below each assembly task node. The plan contains two fixture and two tool changeovers, highlighted with red font color.

the lowest number of iterations. The average time to get an optimal macro-level solution here is higher than for the ball valve. In turn, as there are no alternatives for any of the features, the number of iterations (5) and the number of constraints (40) fed back from the micro level is lower. From this aspect the pneumatic cylinder poses the greatest challenge to the solver as there are numerous alternatives for realizing the assembly, which results in the highest number of iterations (16) and constraints (121). The redundancy among features causes a significant increase in the complexity of the macro-level problem, and hence, the planner hits the time limit in each iteration. The experiments also showed that the best solution found by the solver within the time limit has the optimal value, but proving their optimality requires longer search.

With the lowest number of components and features, the ball valve case requires the least time for an optimal macro-level solution. However, due to the alternative features, the number of iterations and constraints generated is higher. Hence, alternative features increase complexity in two different ways: (1) by extending the search space in the macro-level planing problem and (2) by decreasing the efficiency of the feasibility cuts generated in the micro level. The solution to the ball valve assembly is shown as an assembly tree in Fig. 7 . The figure tells that the first two steps are completed in the same fixture, which is followed by a fixture changeover (due to weight limit of fixture F1), but from that point onward the fixture remains unchanged (fixture F2). This also suggests that the first two features cannot be done in fixture F2, due to micro-level constraints. The applied tools (T1–"human hand" and T2–"screwdriver") are also shown in the figure, and it can be seen that screwing features are put to the end of the plan in order to avoid unnecessary changeovers.

8. Conclusions

The paper introduced a constraint model for macro-level assembly planning for two-handed mechanical assembly processes with rigid parts. As a key step in a feature-based iterative assembly planning workflow, macro-level planning performs the selection of assembly tasks, their sequencing, as well as allocating appropriate resources to them, while minimizing the total assembly time.

Beyond enforcing classical macro-level constraints on the assembly plan, the proposed model takes feedback from micro-level validation in the form of feasibility cuts, which provides a means for ensuring, e.g., the geometrical or technological feasibility of the assembly plan on a detailed model of the assembly motions. A new method was proposed to generate cuts based on the results of collision checking, using a novel TLG model, to prevent the reoccurrence of the same collision in subsequent planning iterations.

The input of macro-level planning is a feature-based assembly model, partly based on the previous works of the authors. Nevertheless, the current contribution generalizes earlier models in several ways. In particular, it allows an arbitrary liaison graph among the parts, which may involve cycles corresponding to alternative features. Hence, assembly planning also involves the selection the features to be realized directly by the assembly tasks. This, on the one hand, demands less effort during problem definition and, on the other hand, helps avoiding the definition of infeasible problems. On the long term, along with the transition from assembly lines to islands the constraint-based model can be extended towards generating routing alternatives [39], and the planning workflow can accommodate physics-based modelling [40] as well.

The proposed approach was demonstrated on three industrial use cases from different sectors of the industry: a ball valve, an automotive supercharger, and a pneumatic cylinder. The results confirm the generality and practical applicability of the proposed constraint model. The strong expressive power of the CP representation and the efficiency of the solver allows finding exact optimum for real-life examples within acceptable computation time.

Acknowledgements

This research has been supported by the ED-18-2-2018-0006 grant "Research on prime exploitation of the potential provided by the industrial digitalisation" and by the European Commission through the H2020 project EPIC (www.centre-epic.eu) under grant No. 739592. A. Kovács acknowledges the support of the János Bolyai Research Fellowship.

References

[1] Sutton GP. Survey of process planning practices and needs. Journal of Manufacturing Systems 1989;8(1):69–71.
[2] ElMaraghy HA. Evolution and future perspectives of capp. CIRP Annals 1993;42(2):739–51.
[3] Ghandi S, Masehian E. Review and taxonomies of assembly and disassembly path planning problems and approaches. Computer-Aided Design 2015;67-68:58–86.
[4] Hu S, Ko J, Weyand L, ElMaraghy H, Lien T, Koren Y, Bley H, Chryssolouris G, Nasr N, Shpitalni M. Assembly system design and operations for product variety. CIRP Annals 2011;60(2):715–33.
[5] Rashid MFF, Hutabarat W, Tiwari A. A review on assembly sequence planning and assembly line balancing optimisation using soft computing approaches. The International Journal of Advanced Manufacturing Technology 2012;59(1):335–49.
[6] Jiménez P. Survey on assembly sequencing: a combinatorial and geometrical perspective. Journal of Intelligent Manufacturing 2013;24(2):235–50.
[7] Bahubalendruni MR, Biswal BB. A review on assembly sequence generation and its automation. Proceedings of the Institution of Mechanical Engineers. Part C: Journal of Mechanical Engineering Science 2016;230(5):824–38.
[8] Kardos C, Kovács A, Váncza J. Decomposition approach to optimal feature-based assembly planning. CIRP Annals 2017;66(1):417–20.
[9] Tsutsumi D, Gyulai D, Kovács A, Tipary B, Ueno Y, Nonaka Y, Monostori L. Towards joint optimization of product design, process planning and production planning in multi-product assembly. CIRP Annals 2018;67(1):441–6.
[10] Neb A. Review on Approaches to Generate Assembly Sequences by Extraction of Assembly Features from 3d Models. Procedia CIRP 2019;81:856–61.
[11] De Fazio T, Whitney D. Simplified generation of all mechanical assembly sequences. IEEE Journal on Robotics and Automation 1987;3(6):640–58.
[12] Wilson RH, Latombe J-C. Geometric reasoning about mechanical assembly. Artificial Intelligence 1994;71(2):371–96.
[13] Romney B, Godard C, Goldwasser M, Ramkumar G. An efficient system for geometric assembly sequence generation and evaluation. Computers in Engineering 1995:699–712.
[14] Thomas U, Barrenscheen M, Wahl F. Efficient assembly sequence planning using stereographical projections of C-space obstacles. Proceedings of the IEEE International Symposium on Assembly and Task Planning, 2003., IEEE. 2003. p. 96–102.
[15] Viganó R, Osorio Gómez G. Assembly planning with automated retrieval of assembly sequences from CAD model information. Assembly Automation 2012;32(4):347–60.
[16] Morato C, Kaipa KN, Gupta SK. Improving assembly precedence constraint generation by utilizing motion planning and part interaction clusters. Computer-Aided Design 2013;45(11):1349–64.
[17] Pan W, Wang Y, Chen X-D. Domain knowledge based non-linear assembly sequence planning for furniture products. Journal of Manufacturing Systems 2018;49:226–44.
[18] Watson J, Hermans T. Assembly planning by subassembly decomposition using blocking reduction. IEEE Robotics and Automation Letters 2019;4(4):4054–61.
[19] Bikas C, Argyrou A, Pintzos G, Giannoulis C, Sipsas K, Papakostas N, Chryssolouris G. An automated assembly process planning system. Procedia CIRP 2016;44:222–7.
[20] Tao S, Hu M. A contact relation analysis approach to assembly sequence planning for assembly models. Computer-Aided Design and Applications 2017;14(6):720–33.
[21] Hadj RB, Belhadj I, Trigui M, Aifaoui N. Assembly sequences plan generation using features simplification. Advances in Engineering Software 2018;119:1–11.
[22] Van Holland W, Bronsvoort WF. Assembly features in modeling and planning. Robotics and computer-integrated manufacturing 2000;16(4):277–94.
[23] Mascle C. Feature-based assembly model for integration in computer-aided assembly. Robotics and Computer Integrated Manufacturing 2002;18(5-6):373–8.
[24] Wang L, Keshavarzmanesh S, Feng H-Y. A function block based approach for increasing adaptability of assembly planning and control. International Journal of Production Research 2011;49(16):4903–24.
[25] Adamson G, Wang L, Moore P. Feature-based control and information framework for adaptive and distributed manufacturing in cyber physical systems. Journal of manufacturing systems 2017;43:305–15.
[26] Su Q. A hierarchical approach on assembly sequence planning and optimal sequences analyzing. Robotics and Computer-Integrated Manufacturing 2009;25(1):224–34.
[27] Ghandi S, Masehian E. Assembly sequence planning of rigid and flexible parts. Journal of Manufacturing Systems 2015;36:128–46.
[28] Pintzos G, Triantafyllou C, Papakostas N, Mourtzis D, Chryssolouris G. Assembly precedence diagram generation through assembly tiers determination. International Journal of Computer Integrated Manufacturing 2016;29(10):1045–57.
[29] Márkus A, Váncza J, Kovács A. Constraint-based process planning in sheet metal bending. CIRP Annals 2002;51(1):425–8.
[30] Dornhege C, Hertle A, Nebel B. Lazy evaluation and subsumption caching for search-based integrated task and motion planning. in: IROS'2013 Workshop on AI-based Robotics 2013.
[31] Rodriguez I, Nottensteiner K, Leidner D, Kasecker M, Stulp F, Albu-Schaffer A. Iteratively refined feasibility checks in robotic assembly sequence planning. IEEE Robotics and Automation Letters 2019;4(2):1416–23.
[32] Kardos C, Váncza J. Mixed-initiative assembly planning combining geometric reasoning and constrained optimization. CIRP Annals 2018;67(1):463–6.
[33] Rahmaniani R, Crainic TG, Gendreau M, Rei W. The benders decomposition algorithm: A literature review. European Journal of Operational Research 2017;259(3):801–17.
[34] Kardos C, Kovács A, Váncza J. Towards Feature-based Human-robot Assembly Process Planning. Procedia CIRP 2016;57:516–21.
[35] Kardos C, Kovács A, Pataki B, Váncza J. Generating human work instructions from assembly plans. in: Proc. of the 2nd ICAPS Workshop on User Interfaces and Scheduling and Planning (UISP2018) 2018:31–9.
[36] Nethercote N, Stuckey PJ, Becket R, Brand S, Duck GJ, Tack G. MiniZinc: Towards a standard CP modelling language. In: Bessiére C, editor. Principles and Practice of Constraint Programming–CP 2007, Vol. 4741. Heidelberg: Springer Berlin Heidelberg, Berlin; 2007. p. 529–43.
[37] Google, Google's OR-Tools. URL `https://developers.google.com/optimization/`.
[38] Merkel D. Docker: Lightweight Linux containers for consistent development and deployment. Linux J. 2014;2014(239).
[39] Moussa M, ElMaraghy H. Master assembly network for alternative assembly sequences. Journal of Manufacturing Systems 2019;51:17–28.
[40] Bourne D, Corney J, Gupta SK. Recent advances and future challenges in automated manufacturing planning. Journal of Computing and Information Science in Engineering 2011;11(2):021006.