Proceedings of DETC'97:
1997 ASME Design Engineering Technical Conferences
September 14-17,1997, Sacramento, California

DETC97/DFM-4333

TOWARD HYBRID VARIANT/GENERATIVE PROCESS PLANNING

Alexei Elinson
Dept. of Computer Science
and Institute for Systems Research
University of Maryland
College Park, MD 20742
elin@cs.umd.edu

Jeffrey W. Herrmann
Dept. of Mechanical Engineering
and Institute for Systems Research
University of Maryland
College Park, MD 20742
jwh2@eng.umd.edu

Ioannis E. Minis
Dept. of Mechanical Engineering
and Institute for Systems Research
University of Maryland
College Park, MD 20742
minis@isr.umd.edu

Dana S. Nau
Dept. of Computer Science
and Institute for Systems Research
University of Maryland
College Park, MD 20742
nau@cs.umd.edu

Gurdip Singh
Dept. of Mechanical Engineering
and Institute for Systems Research
University of Maryland
College Park, MD 20742
gurdip@glue.umd.edu

ABSTRACT

This paper describes our ongoing work on a hybrid approach to process planning, that attempts to combine the best characteristics of both variant and generative process planning while avoiding the worst limitations of each. Our approach uses a database of designs and process plans that are classified using *design signatures*, graphical structures based on detailed product design attributes that are more meaningful and accurate than GT codes and can be computed automatically from the designs stored in the database. We are developing ways to use design signatures to classify and retrieve "slices" of designs and plans, so that when a process plan is needed for a new design, we will be able to retrieve the plan slices that are most relevant, and combine them and modify them to produce a plan for the new design.

INTRODUCTION

In process planning practice, variant techniques are the tools of choice: they currently support almost all practical implementations of computer-aided process planning. Several variant process planning systems are commercially available and have provided significant benefits—but despite the relative popularity of this approach, variant process planning has some well known drawbacks.

A generative process planner that provides realistic process plans for a reasonably wide spectrum of products would make a great impact on industrial practice. Thus, a great deal of research has been done on generative approaches, and a number of experimental systems have been developed for various aspects of process planning. However, generative process planning has proved quite difficult. Most existing systems work only in restricted domains, and have not really achieved significant industrial use.

This paper describes a hybrid process planning approach that we are developing. This approach attempts to combine the best characteristics of both variant and generative process planning while avoiding the worst limitations of each. As shown in Figure 1, our approach involves the following steps:

- Create a database of designs and process plans similar to a variant database—but instead of using GT codes to index and classify the entries in the database, use detailed product design attributes that are more meaningful and accurate than GT codes and can be computed automatically from the designs stored in the database.
- Given a new design for which a process plan is needed, retrieve relevant process planning information from the database—but unlike traditional variant process planning, this information is not a single plan but includes instead portions or "slices" of several plans, each of which is relevant for a different portion of the design.
- Use generative plan-merging techniques to combine and modify the retrieved plan slices in order to synthesize a process plan for the new design.

This paper describes the basic approach we are developing for each of these steps. Some of the steps are implemented and working, and some of them are still work in progress.

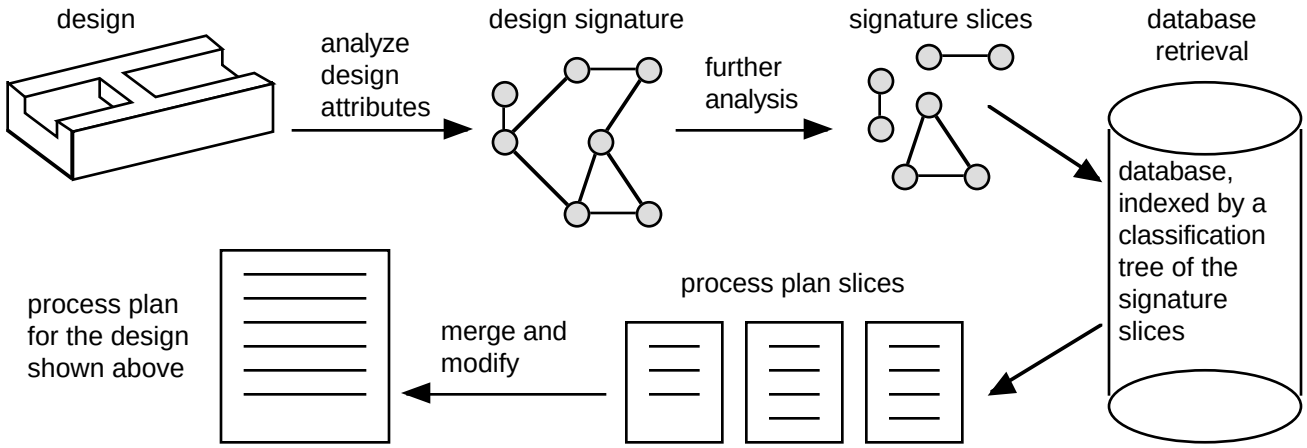

Figure 1. Hybrid variant/generative process planning.

BACKGROUND

Classifying Designs

Group Technology. Group Technology (GT) is a manufacturing philosophy that was first introduced by Russian gun manufacturers during World War I and now widely used in industry [Mitr66]. Group Technology involves classifying similar products into groups in order to achieve economies of scale normally associated with high-volume production. In order to implement GT, one must have a concise coding scheme for describing products and a method for grouping (or classifying) similar products. Many researchers have developed GT coding and classification systems, including the popular Opitz, DCLASS [Bond88], and MICLASS [Orga86] schemes. In each case the basic idea is to capture critical design and manufacturing attributes of a part in an alphanumeric string, or GT code, that is assigned to that part.

The typical GT code [Hout75] consists of two types of positions. In one case, a position describes some global property of the design such as material, size, type, functionality, etc., and its meaning is completely independent of what values are stored elsewhere. In the other case, a position represents some details that are relevant only for certain types of designs, and thus its meaning depends on the values of other positions.

GT classification schemes are essentially tables and rules that help a designer determine the GT code of a part from a drawing manually. One can use a database of the GT codes for design retrieval, variant process planning, and other manufacturing applications. Since the 1980's several researchers [Shah89, Hend88, Srik94] have worked on automating this manual process for classes of machined parts.

One issue is the relevance of GT methods to specific real world design retrieval problems. As it has been used during the last 35 years it works—but it has an inherited drawback: describing designs as short strings creates a coarse classification scheme. Moreover, from the beginning GT coding was intended to be human interpretable, hence the typical questions describe somewhat subjective human impressions of 2D drawings. This has caused difficulty in automating the generation of GT codes.

Geometric Approaches. Another possible basis for classifying designs is to use geometric properties of solid and CAD models. Most of today's CAD/CAM systems use either constructive or boundary models to represent solids.

The use of CSG trees as a way to classify designs has two appealing characteristics: the analogy between volumetric CSG primitives and the volumes of material removed by machining operations, and the ready availability of CSG trees as a basic representational scheme in several geometric modelers. However, the approach suffers from two drawbacks. First, the CSG representation for a design is not unique and a robust method for computing a unique CSG representation for a design has not yet been found (many believe that such a method simply cannot be found [Lee87]). Second, the CSG primitives that would be involved in such a representation do not necessarily correspond to the manufacturing operations that would be used to manufacture the design—and thus the classification might not be very useful for manufacturing practice. As far as we know, no methods to measure similarity on the basis of CSG trees were developed.

Sun *et al.* [Sun95] have described a similarity measure for solids based on properties of their boundary representations. The approach involves representing a polyhedral approximation of a solid using a graph, in which the vertices correspond to faces of a solid and have labels capturing the faces' orientation and area, and the edges correspond to the adjacency relation between solid faces, and are labeled by the corresponding solid angles. To compare two solids they use a sophisticated algorithm to take the graphs of these solids and map them into each other in such way that the area and orientation of corresponding vertices are as close as possible. The results of such mapping are expressed as a real number in a range from 0 to 1. As a new measure of "relaxed" geometrical similarity their work looks very interesting, but there are several difficulties to be overcome before it can be useful as a classification scheme for manufacturing:

- As described in [Sun95], the method works only with polyhedral objects—any non-planar faces of the designs must first be replaced with planar approximations. This may cause difficulty in classifying solids with a significant number of cylindrical or sculptured surfaces.
- The measure of similarity is not symmetrical (similarity between solids A and B is not equal to the similarity

between B and A). This will cause difficulties in using it as the basis for a traditional database indexing scheme, since such schemes assume a symmetrical measure.

- As described in [Sun95], the method does not incorporate (or reflect) manufacturing considerations, such as approachability, fixturing, and operation interference; and we do not see any obvious way to add them.

Process Planning

The two primary approaches to computer-aided process planning are the variant and generative approaches, which are described briefly below. For a more detailed survey of variant and generative approaches, see [Ham86, Ham88].

Variant Process Planning. Variant process planning (see Figure 2) is based on the use of the Group Technology coding schemes described earlier. Given a new design *D* for which a process plan is needed, the process engineer first determines a GT code for *D*, and then uses this code as an index into a database to retrieve a process plan *P'* for a design *D'* similar to *D*. Once this is done, the process engineer modifies the retrieved process plan manually to produce a plan *P* for the design *D*. Some of our group's work on variant process planning includes [Cand95, Cand96, Iyer95].

In process planning practice, variant techniques are the tools of choice: they currently support almost all practical implementations of computer-aided process planning. However, variant process planning also has some significant drawbacks. If the part mix varies over time, then for a new proposed design it may be difficult to find existing designs in the database that satisfy similar design specifications or require similar manufacturing processes. Furthermore, if the process plan retrieved by the variant system uses out-of-date processes, then these will propagate to the process plan for the new design unless the process engineer makes a point of replacing them. Finally, a drastic reduction in the batch size of an existing product may require re-planning to derive an economically sensible process plan.

Generative Process Planning. Given a new design *D* for which a process plan is needed, a generative process planning system attempts to synthesize a process plan directly for *D*. For machined parts, the typical approach is to do the planning on a feature-by-feature basis by retrieving candidate processes from the manufacturing knowledge repository, selecting the feasible processes on the basis of geometric and manufacturing-related constraints, and combining the chosen processes in a proper sequence.

A great deal of research has been done on generative approaches, and a number of experimental systems have been developed for various aspects of process planning [Mant89, Kamb93, Gupt94a, Yue94]. However, generative process planning has unfortunately proved quite difficult. Difficulties arise from interaction among various aspects of the problem, such as workpiece fixturing, process selection, and process sequencing. As a result, most existing systems work only in restricted domains. Although one generative system, the PART system [Geel95]) is being marketed commercially, generative systems have not really achieved significant industrial use.

Even in the absence of complete and comprehensive solutions to the entire process planning problem, generative process planning techniques can be useful in design for manufacturing [Boot94], in which the designer tries to take manufacturability considerations into account during the design stage. For example, by generating and evaluating operation plans for a part, it is possible to give feedback to designers about possible manufacturability problems with the part, and/or to suggest changes to the part that may improve its manufacturability [Mant89, Gupt94a, Gupt95, Das95, Lam95, Hebb96].

Hybrid Approaches. By a hybrid approach, we mean any approach that attempts to exploit knowledge in existing plans while generating a process plan for a new design. Though some approaches have been proposed (two are described below), researchers have not yet developed comprehensive solutions:

- Park *et al.* [Park93] describe an approach for acquiring knowledge useful for generating process plans. Given a process plan for a design, it uses inference rules to find the explanations behind the plan (what part of the plan did what). Then it stores the knowledge as a schema, which describes how in general to make some collection of features. Planning is done by seeking the relevant schema and inserting the necessary values to construct a valid plan. A relevant schema is one with the same collection of features. This is a very simple design similarity measure: it uses no other manufacturing information (such as precedences or tolerances) to identify the relevant schema.
- Marefat and Britanik [Mare94] propose a hybrid approach that captures plan knowledge that specifies the processes

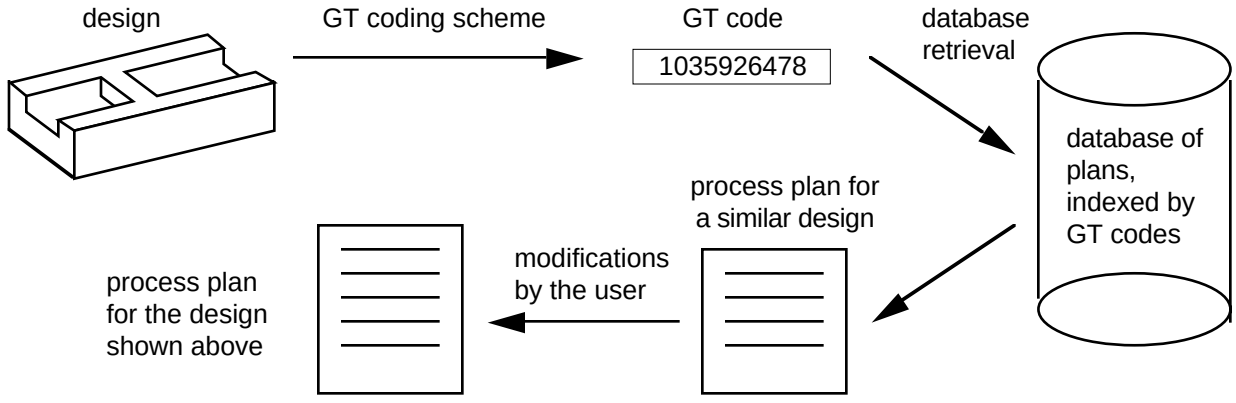

Figure 2. Variant process planning.

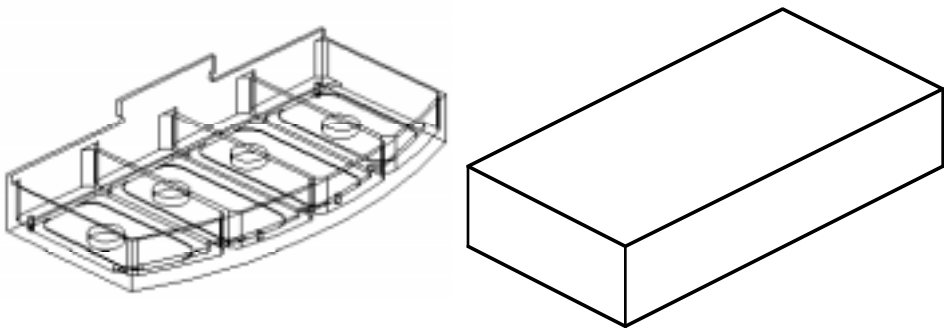

Figure 3: A machined part d_0, and a piece of stock from which to make it. The part is an adaptation of a part design that Boeing Aircraft contributed to the NIST design repository.

necessary to make a certain feature (with a specific size, hardness, surface finish, and tolerances). Planning decomposes a design by generating subplans for each feature and then searching the old subplans for the most appropriate one. The most appropriate old plan is the one that makes a feature that is most similar to the new design's feature. Similarity here is hierarchical: the feature must be the same type, then the same dimensions, then the same tolerances. Corresponding to each level of feature properties are process capabilities. Because the new and old features will be different at some level, the old plan is modified: the planner keeps the process information that corresponds to the levels at which the old and new features are identical, discards the remainder, and generates new information using process capability rules. This approach assumes that each feature can be made independently and thus does not group features in any way.

As these examples show, the existing hybrid approaches have limited capabilities. A robust hybrid approach must consider feature interactions, precedences, tolerances, and other critical design information that impact process planning. In addition it must consider how to store, classify, and retrieve useful design and process planning information.

OUR APPROACH

Overview

As described in the following sections, the overall approach includes two major phases:

- *Preprocessing:* given a database of existing designs and process plans, build an indexing and classification structure for search and retrieval.
- *Planning for new designs:* given a new design, use the classification structure to retrieve relevant planning information from the database, and use this information to synthesize a plan for the new design.

Preprocessing

Design Signatures. Given a set of CAD designs and process plans for those designs, we want to organize them into a database similar to a variant database. However, instead of using GT codes to index and classify the entries in the database, we are developing a more detailed structure called a *design signature*. A design signature is a graph structure that

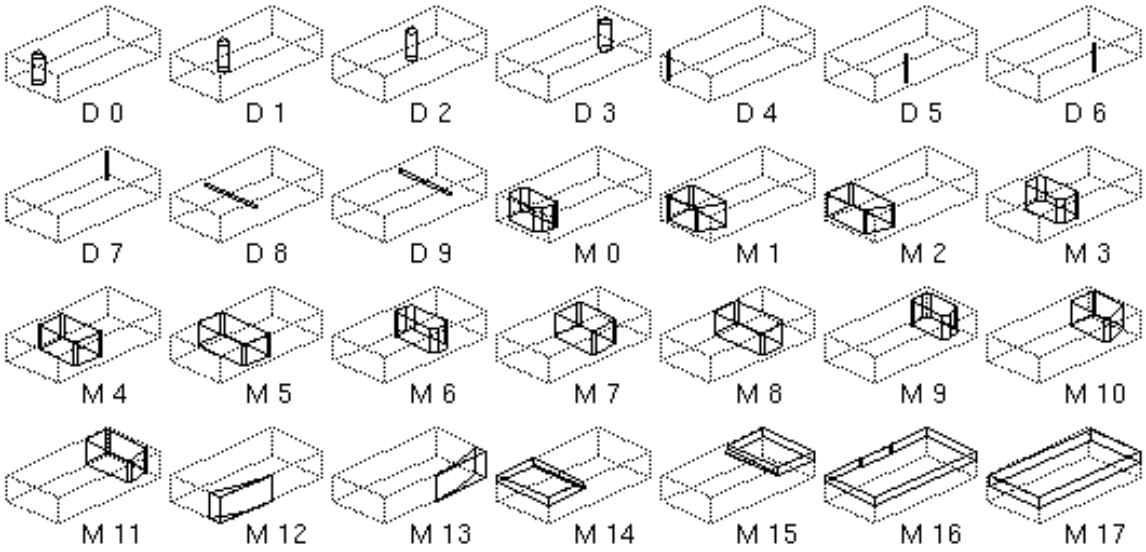

Figure 4: Volumetric machining features for the part d_0 of Figure 3, found by the F-Rex feature extractor.

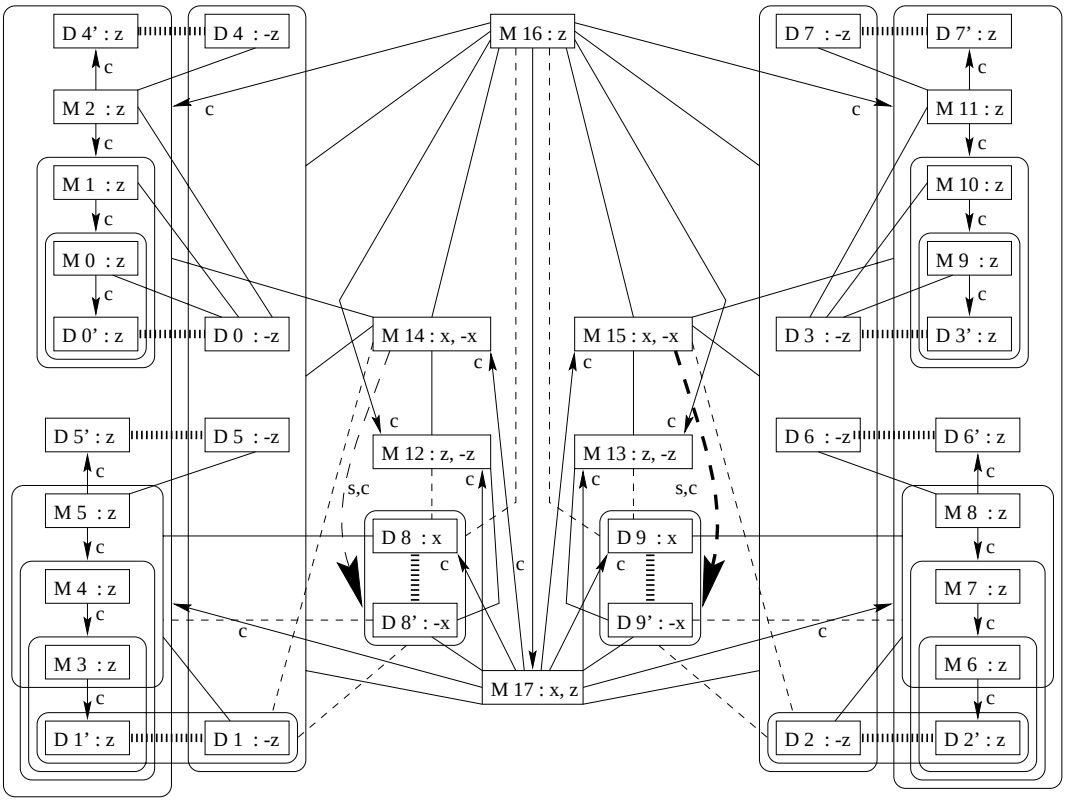

Figure 5: The basic signature $s_0(d_0)$ for d_0. The vertices represent the features in Figure 4, and the boxes and edges represent some of the relationships among them. For brevity, the other design attributes have been omitted.

represents a number of design attributes that are relevant for manufacturing and can be computed automatically from the designs. Below is a summary of the idea; for additional details we refer the reader to [Elin97].

A design can have multiple signatures, that represent the design at different levels of detail. The most detailed signature for a design is the design's *basic* signature. The basic signature represents all properties of the design that are relevant for the current application. Thus, if two designs have basic signatures that are isomorphic, then for the purposes of the current application, the two designs are identical. To reason about a design at a higher level of abstraction, we may compute other signatures for the design. These signatures will be simplifications of the basic signature, produced by removing some of the less important details from the basic signature.

The precise attributes one might want to represent in a design signature will depend on the particular manufacturing application—but for each manufacturing application, the attributes should be things that are specified explicitly in the CAD model or can be deduced automatically from it. For machined parts (our current focus), the attributes that we use include volumetric machined features and various relationships among them, tolerance information, and many of the usual kinds of properties measured in GT coding schemes, such as material, quantity, and so forth.

As an example, consider the machined part d_0 shown in Figure 3. Using the F-Rex feature extractor developed by Regli *et al* [Regl95, Regl97], we get the volumetric machining features shown in Figure 4. By augmenting this set of features to include the design attributes mentioned above, we get d_0's basic signature $s_0(d_0)$, which is shown in Figure 5.

Design Similarity. Figure 6 shows a design d_1 that is different from d_0 but similar to it. The similarity between the two designs is reflected in similarities between their basic signatures $s_0(d_0)$ and $s_0(d_1)$, which are shown in Figure 5 and Figure 7, respectively. If we simplify the basic signatures by removing some of the less important details, this will produce simplified design signatures for d_0 and d_1 that are isomorphic, as shown in Figure 8 and Figure 9. The more similar two designs are, the fewer the simplifications needed in order to produce simplified design signatures that are isomorphic—so we can judge how similar two designs by looking at how much simplification is needed. We formalize this idea as follows.

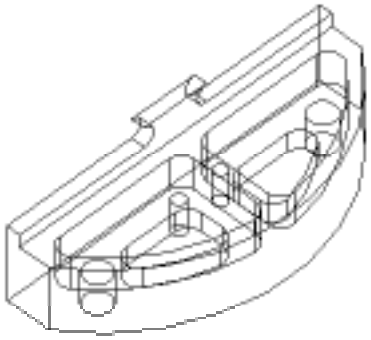

Figure 6: A machined part d_1 similar to d_0.

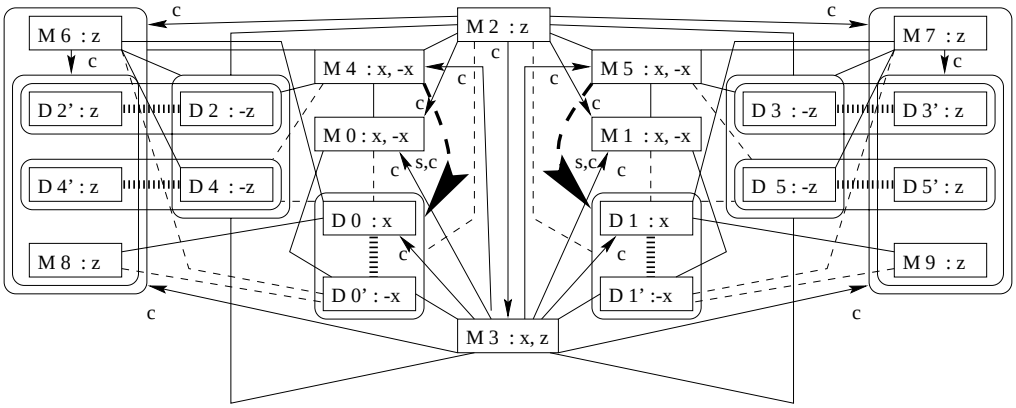

Figure 7: The basic design signature $s_0(d_1)$ for d_1. Just as in Figure 5, the vertices represent the features found by the F-Rex feature extractor, the boxes and edges represent some of the relationships among these features, and other design attributes have been omitted for brevity.

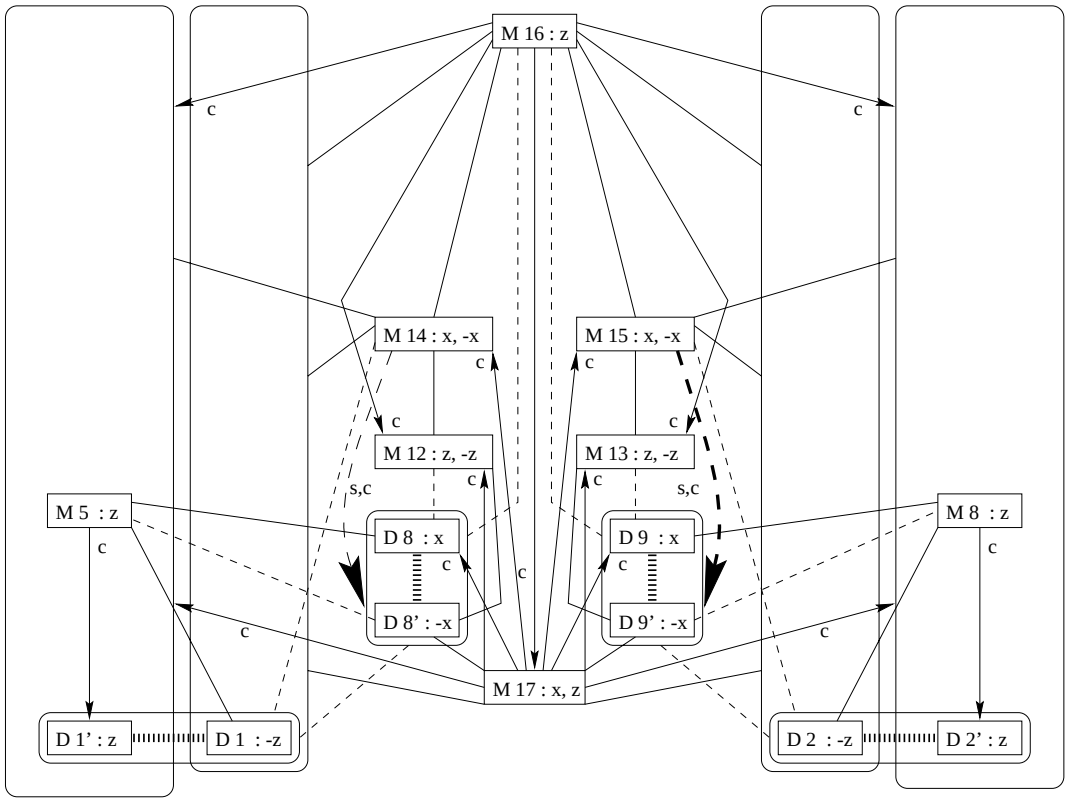

Figure 8: A simplified design signature for d_0.

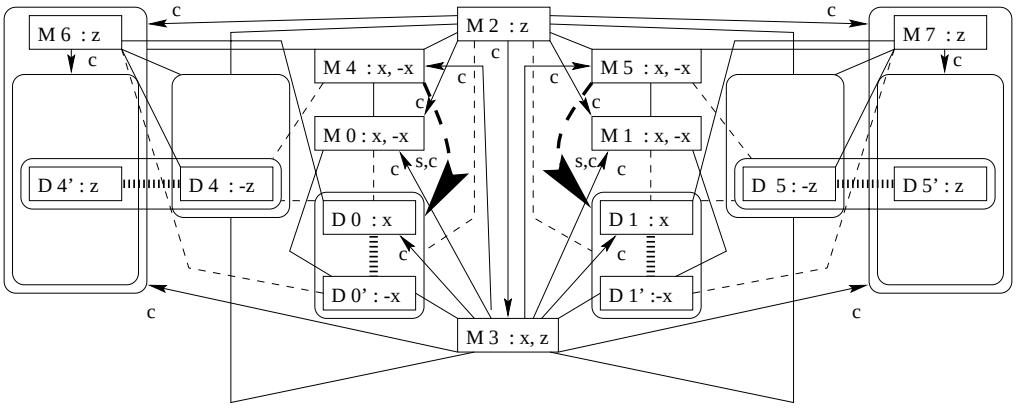

Figure 9: A simplified design signature for d_1.

Let D be a set of designs. Let P be the set of all design properties that are relevant for the current application, and suppose we have partitioned P into subsets P_1, P_2, …, P_n; where P_1 is the set of design properties that we consider least important, P_2 is the set of design properties that we consider second-least important, and so forth. Let d be a design in D, and let $s_0(d)$ be its basic signature, which represents the values that d has for *all* of the properties in P. Then we can produce a sequence of progressively simpler design signatures $s_1(d)$, $s_2(d)$, …, $s_k(d)$, by taking

$s_1(d)$ = the parts of $s_0(d)$ that represent properties in $P_2,\ldots,P_n$;

$s_2(d)$ = the parts of $s_1(d)$ that represent properties in $P_3,\ldots,P_n$;

…;

$s_{n-1}(d)$ = the parts of $s_{n-2}(d)$ that represent properties in P_n;

$s_n(d) = \varnothing$.

Let d' be another design in D, and let $s_0(d')$, $s_1(d')$, …, $s_n(d')$ be its design signatures, computed in the same manner as the corresponding signatures for d. Then for $k = 0, 1, \ldots, n$, we define the relation $R_k(d,d')$ to hold if and only if $s_k(d) = s_k(d')$. Thus, $R_k(d,d')$ is an equivalence relation that has following basic properties:

- $R_0(d,d')$ holds only if d and d' are identical for the purposes of the current application. Thus the equivalence classes of R_0 are singleton sets;
- Since $s_n(d) = s_n(d') = \varnothing$, $R_n(d,d')$ holds for all designs d,d' in D. Thus R_n has a single equivalence class consisting of D itself.
- For k=1,…,n, the equivalence classes of R_{k-1} are subsets of the equivalence classes of R_k.

From these properties, it follows that the equivalence classes of R_0, R_1, …, R_n can be arranged hierarchically in a tree structure such as the one shown in Figure 10. We call this tree a *classification tree*.

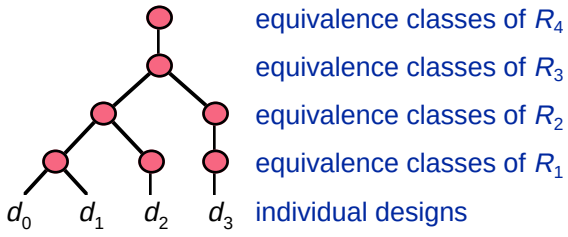

Figure 10. A simple example of a classification tree. Each vertex represents an equivalence class.

Finding similar designs. Given a new design d, the classification tree provides an easy way to find existing designs similar to d. For each vertex v of the tree, we say that v *matches* d if d is a member of the equivalence class represented by v. It immediately follows that if v matches d, then at most one of v's children matches d. Furthermore, the deepest vertex matching d represents the set of designs that are most similar to d, and we can find this vertex using the following search algorithm:

procedure *find-best-match(d)*
v = the root vertex of the classification tree
loop
    if d does not match any of v's children
    then exit, returning v.
    else v := whichever child of v matches d
repeat

Figure 11 gives an example of the operation of this algorithm.

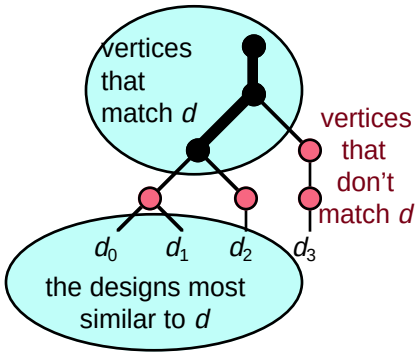

Figure 11. Using the search algorithm to search the classification tree of Figure 10.

Computational Issues. To determine whether a vertex v of depth k in the classification tree matches a design d, the above algorithm compares the design signatures $s_{n-k}(d)$ and $s_{n-k}(d')$, where d' is some design in the equivalence class represented by v. This is a special case of the subgraph isomorphism problem, which is well known to be computationally difficult. It is not known whether or not the subgraph isomorphism problem is NP-complete, but no known algorithm for it runs in less than exponential time.

In our case, there are ways to speed up the computation considerably. The details of how we do this appear in [Elin97], but here are two examples:

- We make use of special properties of our problem that do not occur in the general subgraph isomorphism problem. For example, by keeping track of whether corresponding edges have the same edge labels, we can often very quickly detect cases in which d and d' do not match.
- The algorithm will not even attempt to check whether d matches a vertex v at depth k of the tree unless d matches v's parent at depth k-1. Since the design signatures used at depth k–1 are simpler than those used at depth k, checking whether d matches v's parent can take exponentially less time than checking whether d matches v.

Validation. To validate our classification system, we wanted to show that for a reasonable collection of realistic objects, the techniques correctly and efficiently (1) find identical objects and (2) return reasonably intuitive estimations of similarity between different objects.

One major obstacle is the lack of a generally available large and varied data set of CAD models for mechanical designs. A further complication is the lack of an agreed-upon standard for what it means to be *similar* from the *manufacturing point of*

view—-answers vary depending on the individual to whom the question is posed. Hence, large-scale validation was not possible and we chose to perform controlled experiments to determine how well the prototype system operates on reasonable examples.

To perform our study, we used a collection of solids taken from the NIST Design, Process Planning, and Assembly Repository and employed the experimental F-Rex [Regl95, Regl97] feature recognizer previously developed at the University of Maryland at College Park.

The prototype system used few feature parameters but was able to identify identical designs, and its similarity estimations corresponded closely to the design families. The experiments, once pre-processing was performed, ran in moderate user-time, and we did not observe that the isomorphism checks created any computational bottlenecks. Interestingly, we noticed that the approach worked even on those parts where the F-Rex feature recognizer had difficulty or produced spurious results. We believe that we can enhance the implementation by using the design features in the CAD model and integrating a commercially tested features tool.

Planning for New Designs

If one wants to construct a process plan for a new design using the traditional approach to variant process planning, one begins by retrieving (from the variant database) a process plan for an existing design that is similar to the new design. One drawback to this approach is that although some portions of the existing design may closely match the corresponding portions of the new design, other portions may not match so well. Our goal is to find close matches to as much of the new design as possible, using the following approach:

- decompose the design signature for the new design into *slices* that are meaningful from the point of view of process planning;
- retrieve *process plan slices* that correspond to similar design slices;
- combine and modify the plan slices to produce a process plan for the new design.

Below, we describe the techniques that we are currently developing for these tasks.

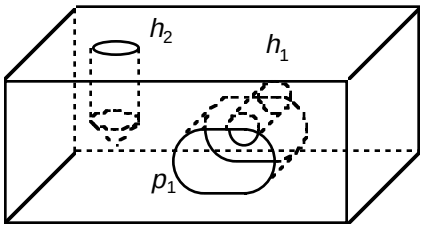

Figure 12. In this design, the hole h_1 and the pocket p_1 are dependent on each other for manufacturing purposes, but both are independent of the hole h_2.

Slicing Designs. Given a design, some features of a design may be independent of each other for manufacturing purposes, and some features may depend on other features. For example, in the simple design shown in Figure 12, the hole h_1 and the pocket p_1 depend on each other, because the process details for each will depend on which of them is machined first. However, no such dependency exists between these two features and the hole h_2. The idea of a design slice is that it is a collection of d's features that depend on each other for manufacturing purposes.

Manufacturing dependencies can arise in a number of ways; below are a few examples:

- if two features intersect (like h_1 and p_1 in Figure 12) , then they will usually depend on each other; this dependence may form a precedence constraint if one feature establishes or denies the approachability, accessibility, or emergence conditions of another feature.
- One feature will depend on another if the first feature has a tolerance attribute that is defined relative to a datum that is created by the second feature.
- One feature will depend upon another if there is a thin section between them.
- One feature will depend upon another if the same tool is used to produce both.
- One feature will depend upon another if both have the same approach direction.

In practice, it is not appropriate to try to capture *all* manufacturing dependencies in a design slice, because we want our characterization of a design slice to depend only on the design and the design signature, and some dependencies would be quite difficult to detect without knowing the process plan as well. Thus, for practical purposes, we will consider a feature f to depend on another feature g only under the following conditions:

- f and g intersect;
- a face created by f has a tolerance that is defined relative to a datum that is created by g;
- f and g have the same corner radii;
- f and g have the same approach direction.

Given this set of conditions, there are several possible things that we might mean by a design slice. The first possible meaning is that given a design d and a feature f of d, the slice of d by f is the set of all features in d that f depends on. This can be computed using the algorithm *slice1* shown below; and the set of all such slices of d is thus $S = \{slice1(d,f) : f \text{ is a feature of } d\}$.

procedure *slice1(d,f)*
$s := \emptyset$
$t := \{f\}$
while t is not a subset of s do
  $s := s \cup t$
  $t := \bigcup_{g \in t}$ {all features of d that feature g depends on}
  repeat
return s

Not every set of features found by *slice1* will actually be of interest to us. For example, if one feature f depends on another feature g, then we will not care about the set *slice1(d,g),*

because it will be a subset of the set *slice1(d,f)*. This leads to the second possible meaning of a design slice: that it is any set of features found by *slice1* that is not a subset of some other set of features found by *slice1*. The set of all such slices of *d* can be found using the algorithm *slice2* below.

procedure *slice2(d)*
S := ∅
for each feature *f* of *d* do
    s := *slice(d,f)*
    if *s* is not a subset of any of the sets in *S*
    then do
        remove from *S* every set that is a subset of *s*
        S := *S* ∪ {*s*}
    end
repeat
return *S*

For the design shown in Figure 12, *slice2* will return a set *S* consisting of two slices: $\{h_1,p_1\}$ and $\{h_2\}$. In this case the slices are disjoint, but in other cases they may overlap. This leads to a third possible meaning of a design slice: that the slices of *d* consist of the sets found by *slice2*, but modified by taking the unions of all overlapping sets. These sets can be computed using the algorithm *slice3* shown below.

procedure *slice3(d)*
S := *slice2(d)*
let the members of *S* be called $s_1,s_2,\ldots,s_k$
for i = 1 to k do
    if s_i is still in *S*
    then for $j = i+1$ to k do
        if s_j is still in *S* and $s_i \cap s_j \neq \emptyset$
        then remove s_j from *S* and set $s_i := s_i \cup s_j$
    repeat
repeat
return *S*

We have not yet determined whether it is preferable to use the sets computed by*slice2* or *slice3*. Deciding this will be an important topic for future work, as it has consequences for the details of the plan retrieval and combining techniques described in the following sections.

Retrieving Plans and Extracting Plan Slices. Once we have computed the design slices for a new design *d* as described above, the next step for each slice *s* of *d* is to match *s* to a corresponding slice *s'* of some design *d'* in the database. We can do that by using the techniques described earlier in the "Preprocessing" section to create a classification tree of design slices and search for a design slice similar to *s*.

Once we have found *s'* and *d'*, we need to take the process plan *p'* for *d'* and extract the plan slice *q'* of *p'* that corresponds to the design slice *s'*. This plan slice consists of the set of all operations in *p'* that create the features in *d'*, as well as any other operations in *p'* that are needed to establish the necessary preconditions. The conditions under which we consider an operation *m* to establish a precondition of another operation *o* are similar to the conditions for feature dependency above:

- *m* is the setup needed for operation *o*.
- *m* creates a feature *f* that must precede the feature *g* that operation *o* creates.

We can use this set of conditions to extract the plan slice *q'* using the following algorithm:

procedure *slice-plan(p',s')*
q' := ∅
r' := {the operations of *p'* that create the features in *s'*}
while *r'* is not a subset of *q'* do
    q' := *q'* ∪ *r'*
    for each operation *m* of *r'* do
        $r' := \bigcup_{m \in r'}$ {all operations of *p'* on which *m* depends}
    repeat
repeat
return *q'*

Combining Plan Fragments. Suppose that for a design *d*, we have used the techniques of the previous sections to compute design slices $s_1, s_2, \ldots, s_k$; and suppose that using these slices we have retrieved plan slices $q_1, q_2, \ldots, q_k$. Each plan slice q_i consists of a linear sequence of operations, and we wish to interleave these sequences to form a single plan *q*; and change the parameters of the operations in *q* so that they will fit the requirements of the design *d*. In doing this, we need to be careful about the way in which we interleave the operations, for not every possible interleaving will work correctly.

As an example, consider the design shown in Figure 12. Regardless of which design-slicing algorithm we use, we will get the slices $s_1 = \{h_1,p_1\}$ and $s_2 = \{h_2\}$. Suppose that from a database of plans for machining prismatic parts on a vertical machining center, we retrieve the following plan slices (for brevity, many details have been omitted from these plans):

plan slice q_1:
    Step 1. setup the part so that h_1 and p_1 are vertical
    Step 2. end-millp_1
    Step 3. drill h_1

plan slice q_2:
    Step 1. setup the part so that h_2 is vertical
    Step 2. drill h_2

In the plan slice q_2, setting up the part so that h_2 is vertical is one of the preconditions needed in order to drillh_2 on a vertical machining center. If we interleave the two plan slices in such a way that the first step of q_1 comes between the first and second steps of q_2, then the first operation of q_1 will deny this precondition, and thus it will not be possible to drill h_2.

If the design slices are independent and each plan slice fully creates the corresponding design slice, then one simple plan-merging approach is to do each plan slice in turn. That is, the new process plan would equal the sequence $q_1, q_2, \ldots, q_k$. However, it may be possible to create a better plan by reducing the number of setups and tool changes, as described below.

Note that in the machining domain, each plan slice consists of one or more subslices (which must be done in the specified order). Each subslice begins with a setup followed by one or more machining operations performed in that setup. The next subslice begins with a setup in a different direction. In the new plan formed by merging the plan slices, it is possible for a subslice to precede one or more subslices with the same setup direction. In this case we can often merge the subslices into a single subslice, by removing the following subslices' setups (since the first subslice's setup would be sufficient). Having

done this, we may now also be able to remove tool changes if consecutive machining operations within the subslice use the same cutting tool.

More generally, we are interested in choosing an order for interleaving the plan slices that is consistent with the relevant manufacturing constraints and maximizes the amount of subslice merging that can be done. Finding an optimal solution to this problem is NP-hard—but we been able to solve a closely related problem quite efficiently using using branch-and-bound techniques [Yang92], and we plan to adapt that approach to the current problem.

SUMMARY AND CONCLUSIONS

Current Status

At this point we have made significant progress towards implementing our hybrid approach. We have created routines that allow the user to define the design and the relevant feature relationships. The feature extractor is complete, and we are developing the program to create the design graph of these features. We need to construct routines that divide a design graph into design slices and search for similar design slices. We will devise a design classification scheme that reflects the need to find useful process plans. We have a routine for creating the process plan slices. Future work includes defining algorithms and developing programs that combine the plan slices and modify them to form a new process plan.

Anticipated Benefits

Our approach, when completed, will have the following primary benefits:

- Accelerating the product development process. Like traditional variant process planning, our hybrid approach will construct process plans that the process engineer may need to improve. However, by automatically adapting the retrieved plans to the new design requirements, our hybrid approach will minimize the need for such improvements.
- Our approach will utilize, in an innovative way, the strengths of both variant and generative process planning. This approach also includes sophisticated feature recognition and plan-based design evaluation in an integrated methodology.
- Our approach will provide the designer feedback about the achievable product quality, the cost, and time needed to manufacture the product. By identifying those design elements that are especially difficult to produce in a cost-effective manner, our approach will help the designer develop products that are easy to manufacture. This will reduce the need for redesign during the production run, resulting in reduced lead time and product cost.

In summary, adaptive process planning and plan-based design evaluation will support agile manufacturing by supporting a quick response to ever-changing market opportunities. By using our hybrid variant-generative approach, a firm will be able to develop a new product quickly and manufacture a small production run economically.

ACKNOWLEDGMENTS

This work is supported in part by ONR grant DABT63-95-C-0037, NSF grants NSF EEC 94-02384, IRI-9306580, and DDM-9201779, ARPA grant DABT63-95-C-0037, and in-kind contributions from Spatial Technologies and Bentley Systems. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the funders.

REFERENCES

[Bond88] A. Bond and R. Jain, "The Formal Definition and Automatic Extraction of Group Technology Codes," *Proceedings of the ASME Computers in Engineering Conference*, pp. 537-542, 1988.

[Boot94] G. Boothroyd. Product design for manufacture and assembly, *Computer-Aided Design* 26:7, 1994, pp. 505–520.

[Bour95] D. Bourne and C. Wang. Design and manufacturing of sheet metal parts: using features to resolve manufacturability problems, in A. Busnaina, Ed., *Computers in Engineering* 1995, ASME, 1995, pp. 745-753.

[Cand95] A. Candadai, J. W. Herrmann, and I. Minis. "A Group Technology-based variant approach for agile manufacturing," in Concurrent Product and Process Engineering, edited by A.R. Thangaraj, R. Gadh, and S. Billatos (ASME, New York, 1995), 289-306.

[Cand96] A. Candadai, J.W. Herrmann, and I. Minis, "Applications of group technology in distributed manufacturing," *Journal of Intelligent Manufacturing*, Volume 7, pages 271-291, 1996.

[Chan90] Chang, T. *Expert Process Planning for Manufacturing*, Addison-Wesley, Reading, MA, 1990.

[Das95] D. Das, S. Gupta, and D. Nau. Generating redesign suggestions to reduce setup cost: a step towards automated redesign. *Computer Aided Design*, to appear.

[Elin97] A. Elinson, D. S. Nau, and W. C. Regli. Feature-based similarity assessment of solid models. *Proc. ACM Solid Modeling Conference*, May 1997.

[Geel95] R. Geelink, O. Salomons, F. van Slooten, F. van Houten, and H. Kals. Unified feature definition for feature based design and feature based manufacturing, in A. Busnaina, Ed., *Computers in Engineering 1995*, ASME, 1995, pp. 517–533.

[Gupt94a] S. K. Gupta, D. S. Nau, W. C. Regli, and G. Zhang. A methodology for systematic generation and evaluation of alternative operation plans. In Jami Shah, Martti Mantyla, and Dana Nau, editors, *Advances in Feature Based Manufacturing*, pages 161--184. Elsevier/North Holland, 1994.

[Gupt95] S. K. Gupta and D. S. Nau. A systematic approach for analyzing the manufacturability of machined

parts. *Computer Aided Design*, 27(5):343--342, 1995.

[Ham86] I. Ham., D. Marion., and J. Rubinovich, "Developing a Group Technology Coding and Classification Scheme," *Industrial Engineering*, Vol. 18, No. 7, pp. 90-97, 1986.

[Ham88] I. Ham and S. Lu (1988), "Computer Aided Process Planning, the Present and Future," *Annals of CIRP*, 37(2).

[Hebb96] K. Hebbar, S. J. Smith, I. Minis, and D. Nau. Plan-based evaluation of designs for microwave modules. *Proc. ASME Design Technical Conference*, August 1996.

[Hend88] M. Henderson and S. Musti, "Automated Group Technology Part Coding from a Three-Dimensional CAD database," *Journal of Engineering for Industry*, Vol. 110, No. 3, pp. 278-287, 1988.

[Hout75] A. Houtzeel. MICLASS, a classification system based on group technology. Technical Report Working Paper MS #75-721, Society of Manufacturing Engineers, 1975.

[Iyer95] S. Iyer and R. Nagi, "Identification of Similar Parts in Agile Manufacturing," in Concurrent Product Design, edited by R. Gadh (ASME, New York, 1994), 87-96.

[Kamb93] S. Kambhampati, M. Cutkosky, J. Tenenbaum, and S. Lee. Integrating general purpose planners and specialized reasoners: case study of a hybrid planning architecture. *IEEE Transactions on Systems, Man, and Cybernetics* 23:6,1993.

[Lam95] G. Lam, *High Level Process Planning and Manufacturability Analysis in Agile Manufacturing*, M.S. Thesis, Department of Mechanical Engineering, University of Maryland, College Park, Maryland, 1995.

[Lee87] Y. C. Lee and K. S. Fu. Machine understanding of CSG: extraction and unification of manufacturing features. *IEEE Computer Graphics & Applications*, Vol. 7, No. 1, pp. 20—32, 1987.

[Mant89] M. Mäntylä, J. Opas, and J. Puhakka. Generative process planning of prismatic parts by feature relaxation, in B. Ravani, Ed., *Proc. 15th ASME Design Automation Conf.*, ASME, 1989 pp. 49–60.

[Mare94] M. Marefat and J. Britanik, "Case-based process planning with hierarchical plan merging," IEEE, 1994.

[Mitr66] S. P. Mitrofanov, *Scientific Principles of Group Technology*, English Translation, National Library for Science and Technology, Washington, D.C., 1966.

[Orga86] Organization for Industrial Research, "OIR Multi-M Code Book and Conventions," Waltham, Massachusetts, 1986.

[Park93] S.C. Park, M.T. Gervasio, M.J. Shaw, and G.F. DeJong, "Explanation-based learning for intelligent process planning," IEEE Transactions on Systems, Man, and Cybernetics, Volume 23, Number 6, pages 1597-1616, 1993.

[Regl95] W. C. Regli, Satyandra K. Gupta, and D. S. Nau. Extracting alternative machining features: An algorithmic approach. *Research in Engineering Design* 7:3,1995, pp. 173-192.

[Regl97] W. C. Regli, S. K. Gupta, and D. S. Nau. Toward multiprocessor feature recognition. *Computer Aided Design*, 1997, to appear.

[Srik94] A. Srikantappa and R. Crawford. Automatic part coding based on interfeature relationships, in J. Shah, M. Mäntylä, and D. Nau, Eds., *Advances in Feature Based Manufacturing*, Elsevier, Amsterdam, 1994, pp. 215–237.

[Shah89] J. J. Shah and A. Bhatnagar, "Group Technology Classification from Feature-Based Geometric Models," *Manufacturing Review*, Vol. 2, No. 3, pp. 204-213, 1989.

[Sun95] T.-L. Sun, C.-J. Su, R.J. Mayer, and R.A. Wysk. Shape similarity assessment of mechanical parts based on solid models. *ASME Design Engineering Technical Conferences* 83:2, 1995, pp. 953—962.

[Wang91] H.-P. Wang and J.-K. Li, Computer-Aided Process Planning. *Advances in Industrial Engineering* 13, Elsevier Science Publishers, 1991.

[Wozn94] M., Wozny, M., Pratt, and C. Poli, Topics in feature-based design and manufacturing, in J. Shah, M. Mäntylä, and D. Nau, Eds., *Advances in Feature Based Manufacturing*, Elsevier, Amsterdam, 1994, pp. 481–510.

[Yang92] Q. Yang, D. S. Nau, and J. Hendler, Merging Separately Generated Plans with Restricted Interactions, *Computational Intelligence* 8:2, 1992, pp. 648-676.

[Yue94] Y., Yue and J. Murray, Workpiece selection and clamping of complex 2.5D components, in J. Shah, M. Mäntylä, and D. Nau, Eds., *Advances in Feature Based Manufacturing*, Elsevier, Amsterdam, 1994, pp. 185–214.