J. Symbolic Computation (2001) **31**, 409–427
doi:10.1006/jsco.2000.0403
Available online at http://www.idealibrary.com on IDEAL®

Decomposition Plans for Geometric Constraint Problems, Part II: New Algorithms

CHRISTOPH M. HOFFMAN[†§], ANDREW LOMONOSOV[‡¶]
AND MEERA SITHARAM[‡‖]

[†] *Computer Science, Purdue University, West Lafayette, IN 47907, U.S.A.*
[‡] *CISE, University of Florida, Gainesville, FL 32611-6120, U.S.A.*

We systematically design two new decomposition–recombination (DR) planners, geared to perform well with respect to several performance measures. The DR-planning problem and the performance measures were formally defined in Part I of this paper to closely reflect specific requirements of CAD/CAM applications. As expected, in analysis and comparison based on all of these performance measures, one of the new DR-planners, the modified frontier algorithm (MFA), represents a significant improvement over existing planners based on SR (constraint shape recognition) and MM (maximum matching) that were analyzed in Part I. We also present salient heuristics and data structures used in the implementation of MFA.

1. Preliminaries

We present two decomposition–recombination (DR) planning algorithms or DR-planners, a notion that was formally defined in Part I (Section 3) of this paper. The new planners follow the overall structural description of a typical DR-planner, based on the DR-solver $\mathcal{S}$ given in Part I (Section 1). Furthermore, the new DR-planners adopt and adapt features of older decomposition methods such as SR (shape recognition) and MM (generalized maximum matching) that were analyzed and compared in Part I (Section 4). In particular, those methods as well as the new planners are based on *degree of freedom* analysis of geometric constraint *hypergraphs*—these concepts are reviewed in Part I (Section 3).

It should be noted that the SR- and MM-based algorithms (Owen, 1991, 1996; Hoffmann and Vermeer, 1994; Bouma *et al.*, 1995), (Hoffmann and Vermeer, 1995; Latham and Middleditch, 1996; Fudos and Hoffmann, 1996, 1997), (Serrano and Gossard, 1986; Serrano, 1990; Kramer, 1992; Ait-Aoudia *et al.*, 1993; Pabon, 1993), were being developed even as the issue—of efficient decomposition of constraint systems for capturing design intent in CAD/CAM—was still in the process of crystallization; in fact the DR-planning problem has been precisely formulated for the first time in Part I (Section 3).

In contrast, our development of the new DR-planners is *systematically guided* by the

[§]Supported in part by NSF Grants CDA 92-23502 and CCR 95-05745, and by ONR Contract N00014-96-1-0635.
[¶]Supported in part by NSF Grant CCR 94-09809.
[‖]Corresponding author: E-mail: `sitharam@cise.ufl.edu`

new performance measures that were formally defined for the first time in Part I (Section 3), to closely reflect several desirable characteristics $\mathcal{C}$ of DR-planners for CAD/CAM applications that were informally discussed in Part I (Section 1).

NOTE. As we mentioned above, our methods are based on degree of freedom analysis. As was described in Part I (Section 3), currently there are no known purely combinatorial characterizations of solvable constraint systems. In fact there are known counterexamples where degree of freedom analysis incorrectly characterizes a generically unsolvable constraint system as a generically solvable one. In Part I (Section 3) we discuss this issue and describe the scope of our methods.

An important *building block* of both the new DR-planners is the routine used to isolate the solvable subsystems S_i at each step i. (Recall the description $\mathcal{S}$ of the typical DR-planner in Part I (Section 1).) In both new DR-planners, the solvable subsystem/subgraph S_i is isolated using an algorithm developed by the authors based on a subtle modification of incremental network flow: this algorithm, called "Algorithm Dense," first *isolates* a dense subgraph, and then finds a minimal dense subgraph inside it, which ensures its solvability (see Part I (Section 3)). The interested reader is referred to earlier papers by the authors: Hoffmann *et al.* (1997) for a description as well as implementation results, and Hoffmann *et al.* (1998) for a comparison with prior algorithms for isolating solvable/dense subgraphs. Here, we shall only note several desirable features of Algorithm Dense.

(a) A useful property of the dense subgraph G' found by the Algorithm Dense (in time $O(n(m+n))$) is that the densities of all proper subgraphs are strictly smaller than the density of G'. Therefore, when G' corresponds to a well-constrained subsystem, then G' is in fact minimal, and hence it is unnecessary to run the additional steps to obtain minimality. Ensuring minimality crucially affects completeness of the new DR-planners.

(b) A second advantage of Algorithm Dense is that it is much simpler to implement and faster than standard max-flow algorithms. This was borne out by our C++ implementation of Algorithm Dense both for finding dense and minimal dense subgraphs. By making D—in the inequality defining dense subgraphs in Part I (Section 3)—a parameter of the algorithm, our method can be applied uniformly to planar or spatial geometry constraint graphs. Furthermore, the new algorithm handles not only binary but also ternary and higher-order constraints which can be represented as *hyperedges*.

(c) A third specific advantage of our flow-based algorithm for isolating dense subgraphs (solvable subsystems) is that we can run the algorithm on-line. That is, the constraint graph and its edges can be input continuously to the algorithm, and for each new vertex or edge the flow can be updated accordingly. This promises a good fit with interactive, geometric constraint solving applications, i.e. criterion (viii) of $\mathcal{C}$ discussed in Part I (Section 1) of this paper.

As will be seen in Sections 2 and 3, the main *difference* between the two new planners, however, lies in their simplifier maps T_i, i.e. the way in which they abstract or simplify a solvable subgraph S_i once it has been found (recall Part I (Section 3)).

MAIN RESULTS AND ORGANIZATION

In Section 2, we use the new performance measures to guide the systematic development of two new DR-planners CA (condensing algorithm) and MFA (modified frontier algorithm), which use—as a crucial building block—a fast method based on network flow for *locating* solvable subgraphs of constraint graphs. This method was presented by the authors in Hoffmann *et al.* (1997) and compared with existing methods in Hoffmann *et al.* (1998). One of the new DR-planners, MFA, represents a significant improvement over existing decomposition algorithms with respect to all of the performance measures. Furthermore, it performed well in initial implementations. Thus, it promises to develop into a DR-planner with all of the nine characteristics $\mathcal{C}$ desirable for CAD/CAM applications, which were laid out in Part I (Section 1) of this paper.

In Section 4, we sketch the important data structures and heuristics of the initial MFA implementation.

As a prelude to the analysis of the new DR-planners CA and MFA in Sections 2 and 3, we give a table below which extends the comparison between the SR and MM to the new DR-planners CA and MFA. The performance measures in the left-most column were formally defined in Part I (Section 3).

The "complexity" entries for the two new DR-planners are directly based on the complexity of a building block Algorithm Dense (briefly discussed above) for isolating minimal dense subgraphs S_i. "Under-const" refers to the ability to deal with underconstrained systems, "Design-dec" refers to the ability to incorporate design decompositions specified by the designer, "Solv." and "Strict solv." refer to (strict) solvability preservation, "Worst and Best approx." refer to the worst and best choice approximation factor.

Perf. meas.	SR	MM	CA(new)	MFA(new)
Generality	No	Yes†	Yes	Yes
Under-const	No(Yes*)	No	Yes	Yes
Design-dec	No(Yes$^{*,\circ}$)	No	No(Yes$^\circ$)	Yes
Validity	No(Yes*)	Yes$^+$	Yes	Yes
Solv.	No(Yes*)	Yes$^+$	Yes	Yes
Strict solv.	No(Yes*)	Yes$^+$	No	Yes
Complete	No(No*)	No	Yes	Yes
Worst approx.	$0\ \left(O\left(\frac{1}{n}\right)^*\right)$	$O\left(\frac{1}{n}\right)$	$O\left(\frac{1}{n}\right)$	$O\left(\frac{1}{n}\right)$
Best approx.	$0\ \left(O\left(\frac{1}{n}\right)^*\right)$	$O\left(\frac{1}{n}\right)^\dagger$	$O\left(\frac{1}{n}\right)$	$O\left(\frac{1}{2}\right)$
Church–Rosser	No(Yes*)	Yes†	Yes	Yes
Complexity	$O(s^2)$	$O(n^{D+1}s)^\dagger$	$O(n^3s)$	$O(n^3s)$

NOTE. The variable s in the complexity expressions denotes the number of vertices, n, plus the number of edges, m, of the constraint graph. Recall that D refers to the number of degrees of freedom of a rigid object in the input geometry (in practice, this could be treated as a constant).

As mentioned in Part I (Section 4), the superscripts "$*$" and "$+$" refer to narrow classes of DR-plans: those that require the solvable subsystems S_i to be based on triangles or a fixed repertoire of patterns, or to represent rigid objects that are fixed or grounded with respect to a single coordinate system. The superscript "$\dagger$" refers to results that were left unproven by the developers of the MM-based algorithms (Ait-Aoudia *et al.*, 1993; Pabon,

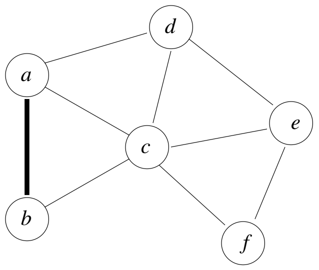

Figure 1. Constraint graph with vertices of weight 2 and edges of weight 1. The minimal dense subgraph {a, b} can be extended sequentially by the other elements, in alphabetic order.

1993) and proved in this paper through a crucial modification of MM described in Part I (Section 4). The modification also results in the improvement of the complexity of the best MM algorithm to $O(n(s = n + m))$.

The superscript “$\circ$” refers to a strong restriction on the design decompositions that can be incorporated into DR-plans by SR and the new DR-planner CA. In fact, the other new DR-planner MFA also places a (however, much weaker) restriction on the design decompositions that it can incorporate, as will be discussed in Section 2.1.

2. Condensing Algorithm (CA)

This DR-planner was already sketched by the authors in Hoffmann *et al.* (1998). The authors' flow-based algorithm discussed above (Hoffmann *et al.*, 1997) is applied repeatedly to constraint graphs to find minimal dense subgraphs or clusters (which we know to be generically solvable—see Part I (Section 3)) containing more than one vertex. CA consists of two conceptual steps. A minimal dense cluster can be *sequentially extended* under certain circumstances by adding more geometric objects one at a time, which are rigidly fixed with respect to the cluster. After a cluster has been thus extended, it is then simplified into a single geometric object, and the rest of the constraint graph is searched for another minimal dense subgraph. The following example illustrates sequential extension. Consider the constraint graph G of Figure 1. We assume that all vertices have weight 2 and all edges have weight 1. The geometry-dependent constant $D = 3$. The vertex set $\{a, b\}$ induces a minimal dense subgraph of G which could be chosen by CA as the initial minimal dense cluster, which could be extended sequentially by the vertices c, d, e, f, one vertex at a time, until it cannot be extended any further. The resulting subgraph is called an *extended dense subgraph or cluster.*

The simplification of an extended cluster is taken to be a single geometric object with D degrees of freedom. This is done as follows: an extended cluster A is replaced by a vertex u of weight D; all edges from vertices in A to a vertex w outside A are combined into one edge (u, w), and the weight of this induced edge is the sum of the weights of the combined edges. After the simplification, another solvable subgraph is found, and the process is continued until the entire graph is simplified into a single vertex.

This is illustrated by the sequence of simplifications of Figure 2. Initially all vertices have weight 2, all edges have weight 1. The vertices connected by the heavy edges constitute minimal or sequentially extended clusters. After four simplifications the original graph is replaced by one vertex.

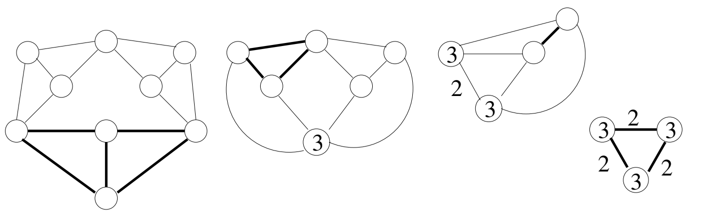

Figure 2. Sequence of simplifications from left to right.

DEFINING SUBSYSTEM SIMPLIFIERS

We capture the transformations performed by the DR-planner CA by describing simplifier maps (recall the definitions in Part I (Section 3)). Let G be the input constraint graph; the first graph G_1 in the DR-plan is the original graph G. Let $G_i = (V, E)$ be the current graph and let S_i be a cluster found at the current stage. Let A be any subgraph of G_i. Then $T_i(A)$ is defined as follows.

- If $A \cap S_i = \emptyset$, then $T_i(A) = A$.
- If $A \cap S_i \neq \emptyset$, then $T_i(A) = (V_{TA}, E_{TA})$, where V_{TA} is the set of all vertices of A that are not vertices of S_i plus one vertex c_i of weight D which represents the simplification of the cluster S_i. The set of edges E_{TA} is formed by removing edges with all endpoints in S_i, and combining edges with at least one endpoint outside S_i, (as well as their weights) as described earlier in this section.

PERFORMANCE ANALYSIS

In this section, we analyze the CA algorithm with respect to the various performance measures defined in Part I (Section 3).

CLAIM 2.1. *CA is a valid DR-planner with the Church–Rosser property. In addition, CA finds DR-plans for the maximal solvable subgraphs of underconstrained graphs.*

PROOF. If the graph G is not underconstrained, then it will remain so after the replacement of any solvable subgraph by a vertex of weight D, i.e. after a simplification step by CA. Thus, if $G = G_1$ is well-constrained, it follows that all of the G_i are well-constrained. Moreover, we know that if the original graph is solvable, then at each step, CA will in fact find a minimal dense cluster S_i that consists of more than one vertex, and therefore G_{i+1} is strictly smaller than G_i for all i. Thus the process will terminate at stage k when G_k is a single vertex. This is independent of which solvable subgraph S_i is chosen to be simplified at the ith stage, showing that CA has the Church–Rosser property.

On the other hand, if G is underconstrained, since the subgraphs S_i chosen to be simplified are guaranteed to be dense/solvable, the process will not terminate with one vertex, but rather with a set of vertices representing the simplification of a set of maximal solvable subgraphs (such that no combination of them is solvable). This completes the proof that CA is a DR-planner that can adapt to underconstrained graphs.

The proof of validity follows straightforwardly from the properties of the simplifier map. □

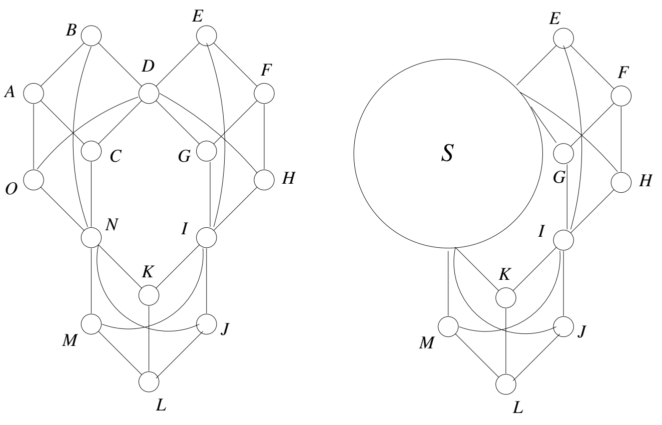

Figure 3. Original and simplified graphs.

Claim 2.2. *CA is solvability preserving.*

Proof. The simplifier maps T_i do not affect subgraphs outside of S_i. □

Claim 2.3. *CA is not strictly solvability preserving.*

Proof. Consider the constraint graph of Figure 3. The vertex weights are 2, the edge weights are 1, and the geometry-dependent constant $D = 3$. The graphs $ABCDNO$, $EFHGID$ and $NMKLIJ$ are all dense/solvable. Suppose that first the cluster $S_1 = ABCDNO$ was found and simplified into one vertex S of weight 3. Now the graph $SEFHGI = T_1(EFHGID)$ is not dense/solvable anymore. □

Intuitively, the reason why CA is not strictly solvability preserving is that the removal of the vertices D and N loses valuable information about the structure of the solvable graph.

Claim 2.4. *CA is complete.*

Proof. This is because CA finds minimal dense subgraphs at each stage. □

Claim 2.5. *Best-choice (and worst-choice) approximation factor of CA is at most $O(1/n)$.*

Note. This proof mimics the MM approximation factor proof in Part I, except that now the subgraph S_i is not a strongly connected component.

Proof. To prove the bound on the best-choice approximation factor consider Figure 4. The left and right columns contain $n/2$ vertices each. The weights of all the vertical edges are 2, the weights of all other edges are 1, the weights of the vertices are as indicated, and the geometry-dependent constant $D = 3$.

Note that all solvable subgraphs in Figure 4 could be divided into three classes. The first class consists of the subgraphs $CL_1L_2; CL_1L_2L_3; \dots; CL_1L_2, \dots, L_{n/2-1}L_{n/2}$. The

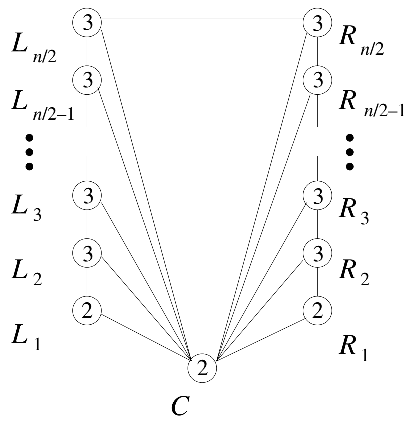

Figure 4. Bad best-choice approximation.

second class consists of the subgraphs $CR_1R_2; CR_1R_2R_3; \ldots ; CR_1R_2, \ldots , R_{n/2-1}R_{n/2}$. The third class contains the solvable subgraphs that contain both L and R vertices. There is only one element in this class—the entire graph $CL_1L_2, \ldots , L_{n/2}R_1R_2, \ldots , R_{n/2}$. There is an optimal DR-plan of constant size that takes $S_1 = CL_1L_2, S_2 = S_1 \cup L_3, \ldots , S_{n/2-1} = S_{n/2-2} \cup L_{n/2}$. After that it takes $S_{n/2} = CR_1R_2, S_{n/2+1} = S_{n/2} \cup R_3, \ldots , S_n = S_{n-1} \cup R_{n/2}$. Finally, it takes $S_{n+1} = S_{n/2-1} \cup S_n$.

However all DR-plans found by CA will have size $O(n)$. The reason for this is that CA is unable to simplify solvable subgraphs on the left of Figure 4 independently from the solvable subgraphs on the right. More formally, let S_1 be the first subgraph simplified by CA under some DR-plan Q. If S_1 belongs to the third class of solvable subgraphs then the size of Q is $O(n)$. Otherwise, without loss of generality, we can assume that S_1 belongs to the first class. According to the definition of CA, $T_1(S_1)$ is a single vertex of weight 3 that replaces several vertices including vertex C. Now for any graph A that belongs to the second class, $T_1(A)$ is not solvable anymore (it has density -4). Hence there is an S_i in Q such that $R_1R_2, \ldots , R_{n/2} \subset S_i$. Hence the size of Q is $O(n)$. □

Next, we consider the last performance measure discussed in Part I (Section 3).

Observation 2.6. *In general, CA cannot incorporate a design decomposition of the input graph for reasons similar to those given for the SR algorithm in Part* I *(Section* 4*). It can incorporate a design decomposition, only if every pair A and B of subgraphs in the decomposition are either completely disjoint or $A \subseteq B$ or $B \subseteq A$.*

2.1. frontier algorithms (FA and MFA)

Intuitively, the reason why CA is not strictly solvability preserving is that the simplification of a minimal or extended dense cluster into a single vertex loses valuable information about the structure of the cluster. The algorithms described in this section preserve this information at least partially by designing a simplifier that keeps the structure of the *frontier vertices* of the cluster, i.e. those vertices that are connected to vertices outside of the cluster. However, the way in which the minimal dense clusters and their

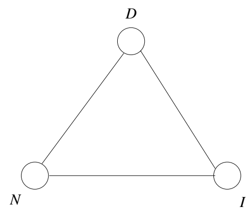

Figure 5. The simplified graph after three clusters have been replaced by edges.

sequential extensions are found is identical to that of CA—i.e. by using the authors' Algorithm Dense from Hoffmann *et al.* (1997).

Informally, under FA, all internal (i.e. not frontier) vertices of the solvable subgraph S_i found at the ith stage are simplified into one vertex called the *core* c_i. The weight of c_i is equal to the geometry-dependent constant D. The core vertex is connected to each frontier vertex v by an edge whose weight is equal to that of v. All other edges of S_i are removed. This is repeated until the solvable subgraph S_m found is the entire remaining graph G_m.

If the solvable subgraph S_i has only two frontier vertices, then all internal vertices of S_i should be removed and no new core vertex created. Instead, the two frontier vertices of S_i should be connected by an edge whose weight w is chosen so that the sum of the weights of the two frontier vertices less w is equal to the constant D. This ensures that the graphs G_i become steadily smaller as i increases. For instance, Figure 3 is such a special case, where every cluster has only two frontier vertices. Hence after three iterations it would be simplified by the FA into Figure 5.

DEFINING THE SUBSYSTEM SIMPLIFIER

We capture the transformations performed by the FA DR-planner by describing simplifier maps (recall the definitions in Part I (Section 3)).

Let S_i be the solvable subgraph of G_i found at stage i, SI be the set of inner vertices of S_i, FI be the set of frontier vertices of S_i, and A be any subgraph of G_i. Then the simplifier map $T_i(A)$ is defined as follows.

- $T_i(S_i) = c_i$, where the weight of c_i is equal to the geometry-dependent constant D.
- If $A \cap S_i = \emptyset$, then $T_i(A) = A$.
- If $A \cap S_i \neq \emptyset$ and $A \cap SI = \emptyset$, then the image of A under the map T_i is A minus those edges that A shares with S_i.
- If $A \cap SI \neq \emptyset$, then $T_i(A) = (V_{TA}, E_{TA})$, where V_{TA} is the set of all vertices of A that are not vertices of SI, plus the vertex c_i representing the simplification of S_i. The set of edges E_{TA} is formed by removing edges of A that have both endpoints in S_i and creating new edges of A that have one endpoint in SI and another in FI as described earlier in this section.

The following observes that FA suffers from the same drawback as CA.

OBSERVATION 2.7. *FA is solvability preserving but not strictly solvability preserving.*

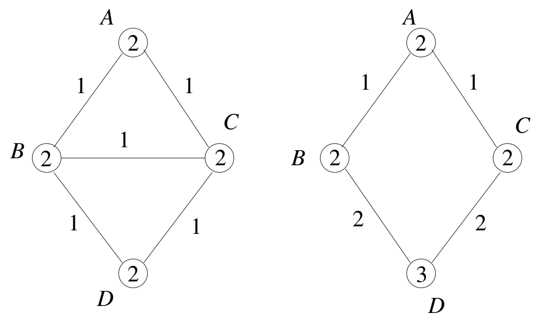

Figure 6. Density of ABC is destroyed by transformation of BCD.

Proof. See for example Figure 6. □

To correct this shortcoming, we will modify the FA algorithm.

modified frontier algorithm (MFA)

Once a minimal or extended dense cluster S_i is discovered, as in FA, the subgraph induced by its internal vertices is contracted into one vertex (the core vertex) c_i. However, this core vertex is connected to each frontier vertex v of S_i by a combined edge whose weight is the the sum of the weights of the original edges connecting internal vertices to v. *The frontier vertices, edges connecting them, and their weights remain unchanged.* The weight of the core vertex is chosen so that the density of the entire simplified cluster is exactly equal to $-D$ where D is the geometry-dependent constant. This process of finding solvable S_i and simplifying them is repeated, until the solvable S_m found is the entire remaining graph G_m.

defining subsystem simplifiers

We capture the transformations performed by the MFA DR-planner by describing simplifier maps (recall the definitions in Part I (Section 3)). Let S_i be the solvable subgraph of G_i found at stage i, SI be the subgraph induced by the inner vertices of S_i, FI be the subgraph induced by the frontier vertices of S_i, A be any subgraph of G_i. The mapping $T_i(A)$ is defined as follows.

- If $A \cap SI = \emptyset$, then $T_i(A) = A$.
- If $A \cap SI \neq \emptyset$, then $T_i(A) = (V_{TA}, E_{TA})$, where V_{TA} is the union of all vertices of $A \setminus SI$ and all vertices of FI plus the core vertex c_i. The set of edges E_{TA} is the union of all the edges of A and of all the edges of S_i, with the exception of edges that have at least one endpoint in SI. Edges that have at least one endpoint outside SI are combined (their weights are combined as well) as described in the previous subsection. Edges that have all endpoints in SI are removed completely.
- The weight assigned to c_i is such that the density of $T_i(S_i)$ becomes exactly $-D$.

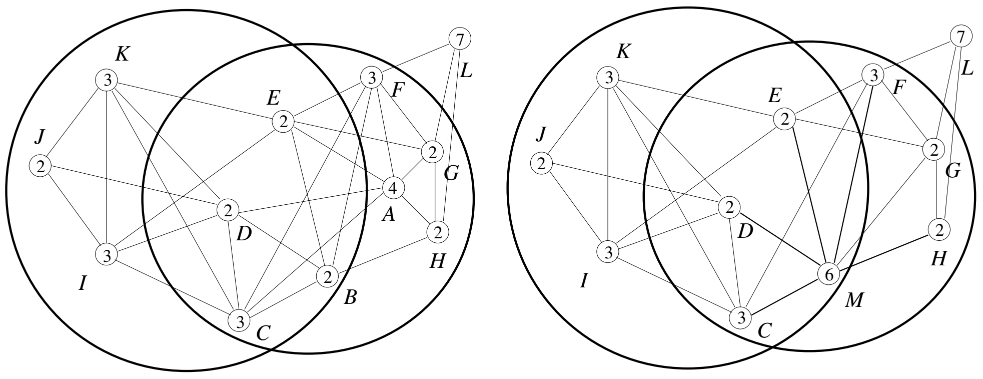

Figure 7. Original graph $BCDEIJK$ is dense, new graph $MCDEIJK$ is not.

PERFORMANCE ANALYSIS

In this section, we analyze the MFA algorithm with respect to the various performance measures defined in Part I (Section 3).

CLAIM 2.8. *MFA is a valid DR-planner with the Church–Rosser property. In addition, MFA finds DR-plans for the maximal solvable subgraphs of underconstrained graphs.*

PROOF. The proof is identical to the one used for CA. □

Before we discuss the solvability preserving property of MFA, we would like to consider the following example shown in Figure 7. All edges have weight 1, vertices as indicated. Initially, the graph $BCDEIJK$ is solvable. The graph $ABCDEFGH$ is also solvable, vertices A and B are its inner vertices, vertices C, D, E, F, G and H are its frontier vertices. After $ABCDEFGH$ has been simplified into the graph $MCDEFGH$, the new graph $MCDEIJK$ is no longer dense (edges MC, MD, ME, MH and MF have weight 2 now). However, note that according to the definition of the MFA simplifier map, the image of $BCDEIJK$ is not the graph $MCDEIJK$, but the graph $MCDEFGHIJK$ which is dense. This graph $MCDEFGHIJK$ is well-overconstrained, since it has density -1 and it could be made well-constrained by say removing constraints FG and FH. Thus the image of $ABCDEFGH$ is also solvable.

In general, the following claim holds.

CLAIM 2.9. *Let A and B be solvable subgraphs such that $A \cap B \neq \emptyset$ and $A \cap B \neq v$ where v is a single vertex of weight less than the geometry-dependent constant D, then $A \cup B$ is solvable.*

PROOF. Since A is solvable, the density of $A \cap B$, $d(A \cap B) \leq -D$ (a solvable graph cannot contain an overconstrained subgraph, unless it is a well-overconstrained graph in which case it can be replaced by an equivalent well-constrained graph). Hence the density of $A \cup B$, $d(A \cup B) = d(B) + d(A \setminus B) = -D + d(A) - d(A \cap B) \geq -D$. Equality occurs when $d(A \cap B) = -D$, otherwise $A \cup B$ is overconstrained. If $A \cup B$ is overconstrained it is well-overconstrained, since it could be converted into well-constrained by reducing weights of some edges of $B \setminus A$ or $A \setminus B$. □

This property can be used to show that:

Claim 2.10. *MFA is solvability preserving as well as strictly solvability preserving.*

Proof. Let B be a solvable graph, and suppose that the solvable graph S_i was simplified by MFA. Then B would only be affected by this simplification if B contains at least one internal vertex of S_i (recall that frontier vertices of S_i remain unchanged). But then, by the definition of the MFA simplifier, $T_i(B) = T_i(B \cup S_i)$. Since $T_i(B \cup S_i)$ is obtained by replacing S_i by solvable $T_i(S_i)$, and according to the previous claim, the union of two solvable graphs is solvable, thus $T_i(B) = T_i(B \cup S_i)$ is also solvable. □

Claim 2.11. *MFA is complete.*

Proof. This is because MFA (just as CA) finds minimal dense subgraphs at each stage. □

Claim 2.12. *MFA has worst-choice approximation factor $O(1/n)$.*

Proof. Consider Figure 8—the solvable constraint graph G. Initially MFA will locate the minimal dense subgraph ABC (since this is the only minimal dense subgraph of G). It will not be able to locate any dense subgraphs disjoint from ABC, or include only frontier vertices of ABC. If it attempts to locate a dense subgraph that includes the entire (simplified) cluster ABC, and does so by inspecting the other vertices in the sequence $A, B, C, H, I, \ldots, F, E$, (i.e. going counterclockwise), then MFA would not encounter any dense subgraphs after ABC, until the last vertex of the sequence E is reached. The minimal dense subgraph found at this stage is the entire graph G. Thus the size of the DR-plan corresponding to this choice of vertices is proportional to n. On other hand, there is a DR-plan of constant size. This DR-plan would first locate the minimal dense subgraph $S_1 = ABC$ and simplify it. After that it would simplify $S_2 = ABCE = S_1 \cup \{E\}$, after that $S_3 = S_2 \cup \{F\}$, etc. going clockwise until the vertex H is reached. At every stage i, the size of S_i is constant, hence the size of this DR-plan is constant. □

Claim 2.13. *The best-choice approximation factor of MFA is at least $\frac{1}{2}$.*

Proof. Let G be the weighted constraint graph. Let P be an optimal DR-plan of G, let p be the size of P (i.e. the size of every cluster S_i simplified under the optimal DR-plan is less than $p+1$). We will show that there is a DR-plan P' that is "close to" P. Complete resemblance ($P = P'$) may not be possible, since the internal vertices of the cluster $S_i^{'}$, found by MFA at the ith stage, are simplified into one core vertex, thereby losing some information about the structure of the graph. However we will show that there is a way of keeping the size of P' within the constant D of the size of P.

Suppose that MFA is able to follow the optimal DR-plan up to the stage i, i.e. $S_i = S_i^{'}$. Suppose that there is a cluster S_j in the DR-plan P such that $i < j$ and S_j contains

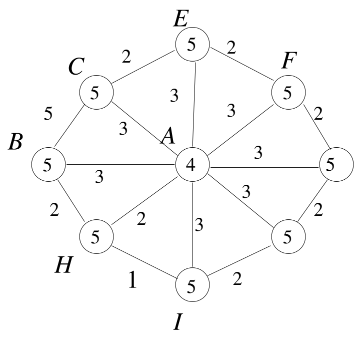

Figure 8. $1/n$ worst-choice approximation factor of MFA.

some internal vertices of S_i. Therefore the simplification of S_j by MFA may be different from simplification of S_j by P. However, since the union of S_i and S_j is solvable, MFA could use $S_j^{'} = T_i^{'}(S_i) \cup S_j$ instead of S_j. The size of $S_j^{'}$ differs from the size of S_j by at most D units, where D is the constant depending on the geometry of the problem. Hence the size of P' is at most $p + D$, and since p is at least D, the result follows. □

Next, we consider the last performance measure discussed in Part I (Section 3).

Observation 2.14. *MFA can incorporate a design decomposition of the input graph if and only if all pairs of subgraphs A and B in the given design decomposition satisfy: the vertices in $A \cap B$ are not among the internal vertices of either A or B.*

Note. This condition on the design decomposition puts no restriction on the sizes of the intersections of the subgraphs in the decomposition, and is far less restrictive than the corresponding conditions for SR and CA in Part I (Section 4).

Proof. The proof is similar to the case of the SR algorithm. For the "if" part, we find a topological ordering O of the given design decomposition P—which is a set of solvable subgraphs of the input graph G, partially ordered under the subgraph relation—such that O is embedded as a subplan of the final DR-plan generated by MFA; i.e. O forms a subsequence of the sequence of solvable subgraphs S_i, whose (sequential) simplification gives the DR-plan.

We take any topological ordering of the given design decomposition P and create a DR-plan for the first solvable subgraph A in P, i.e. while constructing the individual DR-plan for A, we "ignore" the rest of the graph. This individual DR-plan induces the first part of the DR-plan for the whole graph G. In particular, the last graph in this partial DR-plan is obtained by simplifying A using the simplifier described in Section 2.1 (and treating each A exactly as MFA would treat a cluster S_j found at some stage j). Let G_i be the last graph in the DR-plan for G created thus far. Next, we consider the next subgraph in the ordering O, and find an individual DR-plan for it, treating it not as a subgraph of the original graph G, but as a subgraph of the simplified graph G_i. This individual DR-plan is added on as the next part of the DR-plan of the whole graph G.

The crucial point is that the simplification of any subgraph, say A, will not affect any of the unrelated subgraphs B in P. This is because by the requirement on the decomposition

P, A and B share at most frontier vertices. As a result, by the functioning of the MFA algorithm, when the core vertex for A is created, none of the solvable subgraphs inside B are affected.

The process—of constructing individual DR-plans for subgraphs in the decomposition P and concatenating them to the current partial DR-plan—is continued until a partial DR-plan for the input graph G has been produced, which completely includes some topological ordering of the decomposition P as a subplan. Again, let G_k be the last graph in this partial DR-plan. The rest of the DR-plan of G is found by running the original MFA on G_k.

For the "only if" part, we consider a DR-plan Q produced by MFA.

We first observe that the sequence of (original) solvable subgraphs whose sequential simplification gives Q can never contain two subgraphs A and B such that $A \cap B$ contains both internal vertices of A and internal vertices of B. This is because if, for instance, A is simplified before B, then B (on its own) cannot be simplified at a later stage (although $A \cup B$ could), since an internal vertex of B that is also an internal vertex of A will disappear from the graph (will be replaced by a core vertex representing everything internal to A), the moment A has been simplified.

Next, we consider the remaining case where $A \cap B$ contains some internal vertices of A but only frontier vertices of B. In this case, potentially B could be simplified before A, and A will not be affected, since the frontier vertices of B are unchanged by the simplification. However, since a given design decomposition P could contain an arbitrary number of overlapping subgraphs, we can choose decompositions P such that all topological orderings of P are infeasible, i.e. no DR-plan can incorporate them as a subsequence. For instance, if there are three subgraphs A, B and C in P such that $A \cap B$ contains only frontier vertices of B but some internal vertices of A; $B \cap C$ contains only frontier vertices of C, but some internal vertices of B; and $C \cap A$ contains only frontier vertices of A, but some internal vertices of C. This forces the DR-plan to simplify B before A, C before B and A before C, which is impossible. □

3. Implementation of MFA

Before sketching the data structures used in the implementation of MFA, we describe certain heuristics that improve MFA's performance.

3.1. KEY HEURISTICS FOR MFA

- As in the case of CA, once a cluster A has been found, MFA checks whether there are any sequential extensions of A, i.e. a neighboring vertex v s.t. $A \cup \{v\}$ forms a cluster, in this case, the cluster A is replaced by the cluster $A \cup \{v\}$.
- MFA checks whether there is another previously found cluster B such that A and B share two or more frontier vertices in the 2d case, or three or more in the 3d case. If such B exists then $A \cup B$ forms a cluster and the cluster A is replaced by the cluster $A \cup B$. Note that we may choose not to exercise both or either of the above heuristics while incorporating input design decompositions.
- Every cluster A is assigned a value, the so-called *depth*, defined as

$$1 + \max_k depth(B_k),$$

where the B_k are the previously found clusters that comprise A. The depth of each vertex of the original graph G is equal to zero. The depth of a sequential extension of the cluster A is equal to the depth of A.

The advantages of using the depth of a cluster are the following.

(1) The depth naturally defines the level of a cluster in the partial ordering (by the subgraph relation) of the subgraph simplifications in a DR-plan. Thus, it can be used to efficiently incorporate input design decompositions. Thus, as mentioned in Part I (Section 3), one can view the depth as a *priority rating* which specifies which component of the design decomposition has most influence over a given geometric object. In other words, a given geometric object is *first* fixed/manipulated with respect to the local coordinate system corresponding to the lowest level cluster containing it. Thereafter, the entire cluster is manipulated in the local coordinate system corresponding to the next level cluster containing it, etc. In fact, the notion of depth can be naturally extended to include more general priority ratings as well.

(2) When MFA's search for clusters to simplify is stratified by depth, i.e. when MFA finds all clusters of depth 1 first, then those of depth 2, depth 3 and so on, it allows the cluster S_i selected for simplification to be uniformly distributed in the constraint graph G_{i-1}, rather than "growing" new clusters around an initially found cluster. In many natural examples of constraint graphs, this heuristic often helps in keeping the size of S_i small, and thus keeping the DR-plan close to optimal.

3.2. FINDING AND SIMPLIFYING A CLUSTER

Recall that MFA constructs a DR-plan of a constraint graph G by iteratively repeating the following two steps.

- Find a new cluster S_i by using the authors' Algorithm Dense from Hoffmann *et al.* (1997).
- Simplify S_i as described in Section 2.

This section briefly explains the notion of "distribution" that Algorithm Dense relies on. Informally, *distributing* an edge or a vertex in a graph is an operation that determines whether there is a solvable subgraph that contains the given edge or vertex. A graph that has been distributed is one where the weights of all edges have been distributed into (or balanced off by) the capacities of the incident vertices to within the geometric constant D (here the weights of the vertices are treated as capacities). This process involves finding the maximum flow in a certain auxiliary graph. Thus, a graph that *cannot* be distributed must contain a solvable graph, and vice versa, a vertex or an edge that cannot be distributed are contained *in* a solvable graph.

The general framework for finding clusters is the same in MFA as in the algorithm of Hoffmann *et al.* (1997). Let G be a weighted constraint graph $G = (V, E), V = \{v_1, \ldots, v_n\}$. A graph K_1 is created, $K_1 = \{v_1\}$. Vertex v_2 is distributed in K_1. If there are no clusters in the union of $K_1 \cup \{v_2\}$, then $K_2 = K_1 \cup \{v_2\}$, vertex v_3 is distributed in K_2, etc. If no clusters have been found when $K_n = G$ is reached, then the algorithm terminates. If the cluster S_i has been located at stage i, then it should be simplified and the simplification $T_i(S_i)$ should be represented appropriately in the graph K_{i+1} and iterations continue.

3.3. DATA STRUCTURES

The major difference between MFA and CA is the way in which the clusters are simplified and represented. While CA simplifies the cluster into a vertex, MFA uses the simplifier described in Section 2 and simplifies the cluster into a graph consisting of a core vertex and frontier vertices. Furthermore, MFA employs heuristics for incorporating input design decompositions and stratification of clusters by depth. These features of MFA are implemented by the following three sets of cluster structures and a new graph structure F for bookkeeping.

Before we describe these structures, we additionally need to extend the notion of distribution to simplified clusters as well. A *simplified cluster C is distributed* in graph F if the only solvable subgraph in $C \cup F$ that contains any part of C is C itself. After distribution, the cluster C is included in F.

Now we describe the three sets of cluster structures. The first is a set of simplified clusters that have already been distributed, this set is implemented as a list of simplified clusters and is denoted by *ClustersInF*. The vertices of the graph F are the core and frontier vertices of the clusters in *ClustersInF*. The second set is a set of simplified clusters that are still waiting to be distributed, and which have the same depth. This set is denoted by *CQueue* and is implemented as a queue of clusters. The third set also consists of the simplified clusters that are still waiting to be distributed: these have larger depth than the clusters currently under consideration. This set is denoted by *NewCQueue* and is also implemented as a queue of clusters.

These three sets of simplified clusters are used as follows. Initially *CQueue* contains the vertices of the original weighted graph G, and the depth bound is set to 0. Every simplified cluster C in *CQueue* is distributed in F, and is removed from *CQueue*.

- If distribution of a simplified cluster C has not yielded any solvable subgraphs of $C \cup F$, then C is added to F.
- If a solvable subgraph S_i was located, then it is simplified and the depth of the simplification $T_i(S_i)$ is computed as described in Section 3.1. MFA temporarily removes the core vertex of $T_i(S_i)$ from the graph F, if the depth of $T_i(S_i)$ is greater than the current depth bound. The cluster corresponding to $T_i(S_i)$ is added to the *NewCQueue*. If the depth of $T_i(S_i)$ is no greater than the current depth bound, then $T_i(S_i)$ is added to the *ClustersInF* and the core of $T_i(S_i)$ remains in F. When all simplified clusters are removed from *CQueue*, the current depth bound is incremented and the clusters in *NewCQueue* that have this new depth bound are moved from *NewCQueue* to *CQueue*. The process repeats until both *CQueue* and *NewCQueue* are empty.

The above description is altered slightly in order to incorporate an input design decomposition, as will be described in the example that follows.

NOTE. The ith graph G_i in the output DR-plan (defined in the Part I (Section 3)), is the union of frontier and core vertices of clusters in *ClustersInF, CQueue*, and *NewCQueue*, at the ith stage.

In order to describe any single simplified cluster A the following data structure is used.

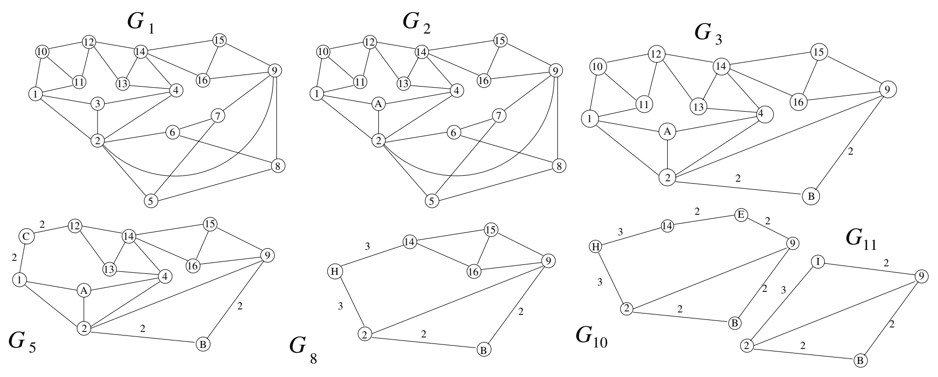

Figure 9. The sequence of simplifications giving the output DR-plan.

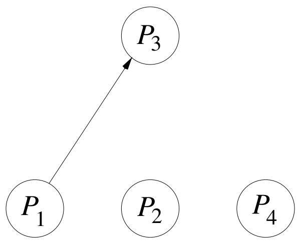

Figure 10. Partial ordering of the input design decomposition P.

Vertex	*Core*;	
Vertexlist	*FV*;	list of frontier vertices
Edgelist	*EInner*;	list of edges between core and frontier vertices
Edgelist	*EOuter*;	list of edges between pairs of frontier vertices
Clusterlist	*CP*;	list of previously found clusters comprising A
Vertexlist	*OV*;	list of the vertices of the original graph comprising A (this may be useful for satisfying characteristic (v) of $\mathcal{C}$ described in Part I of this paper)
Int	*Depth*;	

3.4. EXAMPLE

We will demonstrate how MFA generates a DR-plan for an example constraint graph G and design decomposition P, by describing various stages of simplification of Figure 9.

The weight of all vertices in $G = G_1$ is equal to 2, the weight of all the edges is equal to 1, the geometric constant D is equal to 3.

Suppose that the following input design decomposition $\mathbf{P}$ is given, $\mathbf{P}$=$\{\mathbf{P_1}, \mathbf{P_2}, \mathbf{P_3}, \mathbf{P_4}\}$ where $\mathbf{P_1} = \{\mathbf{1}, \mathbf{2}, \mathbf{3}, \mathbf{4}\}$, $\mathbf{P_2} = \{\mathbf{2}, \mathbf{5}, \mathbf{6}, \mathbf{7}, \mathbf{8}, \mathbf{9}\}$, $\mathbf{P_3} = \{\mathbf{1}, \mathbf{2}, \mathbf{3}, \mathbf{4}, \mathbf{10}, \mathbf{11}, \mathbf{12}, \mathbf{13}, \mathbf{14}\}$, $\mathbf{P_4}$ $= \{9, 14, 15, 16\}$.

Then the corresponding partial ordering O of P is shown in Figure 10.

The tables below contain information about the current cluster S_i and the queues of clusters *CQueue, NewCQueue* as well as *ClustersInF* at the ith stage. Recall the definitions of $S_i.Core$, $S_i.FV$, $S_i.CP$, $S_i.OV$ and $S_i.Depth$ from the preceding section.

NOTE. For every component P_i of the input design decomposition P, a copy of each of $ClustersInF_i$, $CQueue_i$, $NewCQueue_i$ is created. These copies are induced in the natural way, for example, initially $CQueue = \{1, 2, \ldots, 16\}$, $CQueue_1 = \{1, 2, 3, 4\}$.

i	$S_i.Core(weight)$	$S_i.FV$	$S_i.CP$	$S_i.OV$	$S_i.Depth$
1	$\emptyset$	$\{1,2,3\}$	$\{1,2,3\}$	$\{1,2,3\}$	1
2	A (2)	$\{1,2,4\}$	$\{S_1,4\}$	$\{1,2,3,4\}$	1
3	B (4)	$\{2,9\}$	$\{2,5,6,7,8,9\}$	$\{2,5,6,7,8,9\}$	1
4	$\emptyset$	$\{1,10,11\}$	$\{1,10,11\}$	$\{1,10,11\}$	1
5	C (3)	$\{1,12\}$	$\{S_4,12\}$	$\{1,10,11,12\}$	1
6	$\emptyset$	$\{12,13,14\}$	$\{12,13,14\}$	$\{12,13,14\}$	1
7	D (2)	$\{12,14,4\}$	$\{S_6,4\}$	$\{12,13,14,4\}$	1
8	H (5)	$\{2,14\}$	$\{S_2,S_5,S_7\}$	$\{1,2,3,4,10,\ldots,14\}$	2
9	$\emptyset$	$\{14,15,16\}$	$\{14,15,16\}$	$\{14,15,16\}$	1
10	E (3)	$\{14,9\}$	$\{S_8,9\}$	$\{14,15,16,9\}$	1
11	I (5)	$\{2,9\}$	$\{S_9,S_{10}\}$	$\{1,2,3,4,9,\ldots,16\}$	3
12	J (3)	$\{\emptyset\}$	$\{S_{11},S_3\}$	$\{1,\ldots,16\}$	4

i	$ClustersInF$	$CQueue$	$NewCQueue$
1	$\{1,2,3\}$	$\{4,5,\ldots,16\}$	$\emptyset$
2	$\{1,2,4\}$	$\{5,\ldots,16\}$	$\{S_2\}$
3	$\{1,2,4,9\}$	$\{10,\ldots,16\}$	$\{S_2,S_3\}$
5	$\{1,2,4,9,12\}$	$\{13,\ldots,16\}$	$\{S_2,S_3,S_5\}$
8	$\{2,9,14\}$	$\{15,16\}$	$\{S_3,S_8\}$
10	$\{2,9,14\}$	$\{\emptyset\}$	$\{S_3,S_8,S_{10}\}$
11	$\{2,B,9\}$	$\{\emptyset\}$	$\{S_{11}\}$

MFA will proceed as follows.

First MFA constructs a DR-plan for P_1.

Vertices 1–3 are distributed and cluster $S_1 = \{1, 2, 3\}$ is discovered. Vertex 4 forms a sequential extension of cluster S_1, together they form the cluster $S_2 = P_1$. There are no further sequential extensions of S_2, so S_2 is added to *NewCQueue.* The DR-plan for P_1 has been found (so data structures of P_1 could be removed at this stage).

Now MFA constructs a DR-plan for P_2.

After vertices 5–9 are distributed, cluster $S_3 = \{2, 5, 6, 7, 8, 9\} = P_2$ is discovered. Since S_3 has no sequential extensions, it is added to *NewCQueue.* Note that some weights in the graph F have changed. The weight of the vertex B is 4, the weights of the edges $(B, 2)$ and $(B, 9)$ are 2. The DR-plan for P_2 has been found.

Now MFA constructs a DR-plan for P_3.

After vertices 10 and 11 are distributed, cluster $S_4 = \{1, 10, 11\}$ is discovered. Vertex

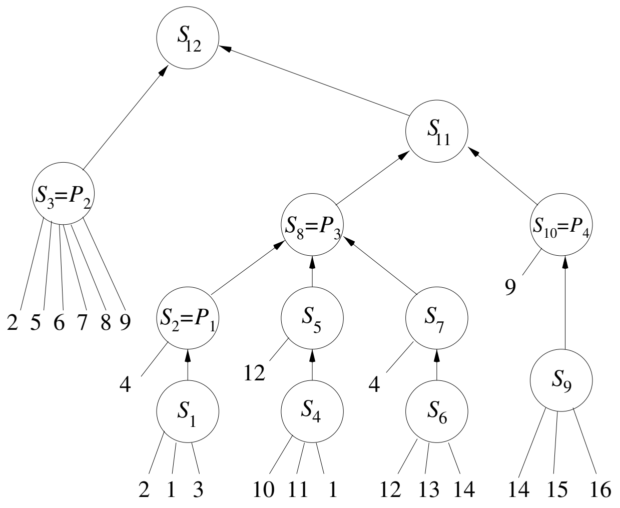

Figure 11. Partial ordering of the S_i in the output DR-plan.

12 forms a sequential extension of cluster S_4, together they form cluster S_5. Cluster S_5 is added to *NewCQueue*.

After vertices 13 and 14 are distributed, cluster $S_6 = \{12, 13, 14\}$ is discovered. Vertex 4 forms a sequential extension of cluster S_6, together they form a cluster S_7. Cluster S_7 is added to *NewCQueue*.

Since $CQueue_3$ is now empty, $Depth_3$ is incremented and is set equal to 1 (*Depth* for the entire DR-plan remains 0). The new $CQueue_3 = \{S_2, S_5, S_7\}, NewCQueue_3 = \{\emptyset\}, CQueue = \{S_2, S_5, S_7, 15, 16\}, NewCQueue = \{S_3\}$.

After the clusters S_2, S_5 and S_7 are distributed, cluster $S_8 = \{S_2, S_5, S_7\} = P_3$ is discovered. It is added to *NewCQueue*. The DR-plan for P_3 has been found.

Now MFA constructs a DR-plan for P_4.

After vertices 15 and 16 are distributed, cluster $S_9 = \{14, 15, 16\}$ is discovered. Vertex 9 forms a sequential extension of cluster S_9, together they form a cluster $S_{10} = P_4$. Cluster S_{10} is added to *NewCQueue*. The DR-plan for P_4 has been found.

Since *CQueue* is now empty, *Depth* is incremented and is set equal to 1. The new $CQueue = \{S_3, S_{10}\}, NewCQueue = \{S_8\}$.

After clusters S_3 and S_{10} are distributed, *Depth* is set equal to 2. After the cluster S_8 is distributed, the cluster $S_{11} = \{S_8, S_{10}\}$ is discovered. Cluster S_{11} is added to the *NewCQueue*.

Since the clusters S_{11} and S_3 share more than one frontier vertex, the cluster $S_{12} = \{S_3, S_{11}\}$ is discovered by using the second heuristic described in the beginning of this section.

Since S_{12} is the entire remaining graph, the DR-plan terminates successfully, incorporating the design decomposition P. The partial ordering of all S_i of the DR-plan is shown in Figure 11.

References

Ait-Aoudia, S., Jegou, R., Michelucci, D. (1993). Reduction of constraint systems. In *Compugraphics*, pp. 83–92. Alvor, Portugal.

Bouma, W., Fudos, I., Hoffmann, C., Cai, J., Paige, R (1995). A geometric constraint solver. *Comput. Aided Des.*, **27**, 487–501.

Fudos, I., Hoffmann, C. M. (1996). Correctness proof of a geometric constraint solver. *Int. J. Comput. Geom. Appl.*, **6**, 405–420.

Fudos, I., Hoffmann, C. M. (1997). A Graph-Constructive Approach to Solving Systems of Geometric Constraints. *ACM Trans. Graph.*, **16**, 179–216.

Hoffmann, C. M., Lomonosov, A., Sitharam, M. (1997). Finding solvable subsets of constraint graphs. In *Proceedings of Principles and Practice of Constraint Programming '97, Linz, Austria, LNCS* **1330**, pp. 463–477. Berlin, Springer-Verlag.

Hoffmann, C. M., Lomonosov, A., Sitharam, M. (1998). Geometric constraint decomposition. In Brud-erlin, Roller eds, *Geometric Constraint Solving*. Berlin, Springer-Verlag.

Hoffmann, C. M., Vermeer, P. J. (1994). Geometric constraint solving in R^2 and R^3. In Du, D. Z., Hwang, F. eds, *Computing in Euclidean Geometry,* 2nd edn, Singapore, World Scientific Publishing.

Hoffmann, C. M., Vermeer, P. J. (1995). A spatial constraint problem. In *Workshop on Computational Kinematics, France.* Sophia-Antipolis, INRIA.

Kramer, G. (1992). *Solving Geometric Constraint Systems.* Cambridge, MA, U.S.A., MIT Press.

Latham, R., Middleditch, A. (1996). Connectivity analysis: a tool for processing geometric constraints. *Comput. Aided Des.*, **28**, 917–928.

Owen, J. (1991). Algebraic solution for geometry from dimensional constraints. In *ACM Symposium on the Foundations of Solid Modeling, Austin, TX*, pp. 397–407. New York, ACM.

Owen, J. (1996). Constraints on simple geometry in two and three dimensions. *Int. J. Comput. Geom. Appl.*, **6**, 421–434.

Pabon, J. A. (1993). Modeling method for sorting dependencies among geometric entities. *US States Patent*, 5,251,290.

Serrano, D. (1990). Managing constraints in concurrent design: first steps. In *Proceedings of Computers in Engineering, Boston, MA, 1990*, pp. 159–164.

Serrano, D., Gossard, D. C. (1986). Combining mathematical models with geometric models in cae systems. *Computers in Engineering, Chicago, IL. ASME*, **1**, 277–284.

Originally Received 16 December 1998
Accepted 5 June 2000