Planning geometric constraint decompositions via optimal graph transformations

Christoph M. Hoffmann[1]

Andrew Lomonosov[2]

Meera Sitharam[2,3]

Abstract. A central issue in dealing with geometric constraint systems that arise in Computer Aided Design and Assembly is the generation of an optimal decomposition recombination *plan* that is the foundation of an efficient solution of the constraint system. Though complex, this issue has evolved and crystallized over the past few years, permitting us to take the next important step: for the first time, in this paper, we formalize, motivate and explain the optimal decomposition-recombination (DR) planning problem as a problem of finding a sequence of graph transformations T_i that maximizes an objective function subject to a certain criteria. We also give several performance measures phrased as graph transformation properties by which DR planning algorithms can be analyzed and compared. These measures include: generality, validity, completeness, Church-Rosser property, complexity, best and worst choice approximation factors, (strict) solvability preservation, and the ability to deal with under-constrained systems.

The clear and precise formulation of the problem and performance measures in terms of graph transformations allow us to systematically develop a new DR-planner which we call the Frontier Algorithm (FA), which represents a significant improvement over existing algorithms in analysis and comparison based on all of these performance measures. We analyze the performance of FA and compare FA to previously existing algorithms.

1 Introduction and Motivation

This paper shows that a core problem in geometric constraint solving is to find a sequence of graph transformations that satisfies certain formal requirements. We refer to it as Decomposition-recombination-planning problem to be described later.

A geometric constraint problem consists of a finite set of geometric objects and a finite set of constraints between them. Geometric objects include points,

[1] Computer Science, Purdue University, West Lafayette, Indiana 47907, USA. Supported in part by NSF Grants CDA 92-23502 and CCR 95-05745, and by ONR Contract N00014-96-1-0635.

[2] CISE, University of Florida Gainesville, FL 32611-6120, USA

[3] Supported in part by NSF Grant CCR 94-09809.

lines, planes, circles, and so on. Constraints between them include parallel, perpendicular, distance, tangency, and so on. Some of these constraints are logical, such as incidence or tangency; others are dimensional such as distance or angle. *A solution* to a geometric constraint problem is a placement of geometric objects that satisfies the constraints. For example solving the geometric constraint problem shown in left half of Figure 1 is equivalent to determining coordinates of the three points in 2d, given the distances between them. This reduces to finding real solutions of a system of quadratic equations. In general solving a geometric constraint problem reduces to solving a nonlinear algebraic system over reals.

Industrial relevance. Geometric constraints are at the heart of computer aided engineering applications (see, e.g., [11, 12]), and also arise in many geometric modeling contexts such as virtual reality, robotics, molecular modeling, teaching geometry etc. For recent reviews of the extensive literature on geometric constraint solving see, e.g, [3, 18, 5].

In particular, the constraint decomposition approach to geometric constraint solving is so successful that most of the major CAD systems such as I-DEAS or Pro/ENGINEER have licensed a commercial solver based on this principle. This solver uses a repertoire of subgraph patterns and construction rules to decompose the constraint graph and break it into small subsets that can be solved easily. It is highly successful in planar constraint solving, but barely adequate for spatial constraint solving.

As pointed out in [2], the pattern repertoire quickly becomes unmanageable when extending the geometric coverage from a simple subset of planar constraint configurations to more complex ones. In spatial constraint solving, using a repertoire of patterns is not very satisfactory, as pointed out in [15], and a more general decomposition strategy is needed. This general strategy, first approached in [13], is the subject of this paper. It manages to break the barrier that keeps the pattern approach to constraint solving from becoming fully effective in spatial constraint solving and in more general planar constraint solving. We would not be surprised if a commercial implementation would be attempted based on this work. The first and third authors have past and ongoing confidential industrial contracts related to this work.

1.1 Need for decomposition-recombination plans

The major issue in solving geometric constraint problems is efficiency: computing the solution of the nonlinear algebraic system that arises from geometric constraints is computationally challenging, and except for very simple geometric constraint systems such as the example in Figure 1, this problem is not tractable in practice without further machinery. The overwhelming cost in a geometric constraint solving is directly proportional to the *size of the largest subsystem* that is solved using a direct algebraic/numeric solver. This size dictates the practical utility of the constraint solver, since the time complexity of the constraint solver is at least exponential in the size of the largest such subsystem.

Therefore, the geometric constraint solver should use geometric domain knowledge to develop a *Decomposition-Recombination (DR) plan* (to be formally defined in section 2.2) for decomposing the constraint system into small subsystems, whose solutions are recombined thereby simplifying the original system on which the decomposition-recombination is applied recursively until the system is fully solved. To facilitate the recombination, the small subsystems in the decomposition should be geometrically *rigid*. A rigid or *discretely solvable* subsystem of the geometric constraint system is one for which the set of real-zeroes of the corresponding algebraic equations is discrete (i.e. the corresponding real-algebraic variety is zero dimensional), *after* the local coordinate system has been fixed arbitrarily. Discretely solvable systems of equations are also called *wellconstrained* or (consistently) *overconstrained*. A system of equations that is not rigid is also called *underconstrained*.

Example. There are many ways to fix a local coordinate system in Figure 1, one way for example is to place a point, say P1 at origin and place the point P3 so that the line segment (P1,P3) coincides with the x-axis. After the local coordinate system is fixed, the corresponding algebraic equations either have 2 real solutions (placing P2 above or below x-axis) or no real solutions (for example if $A + B < C$).

Note. It is important to distinguish "discretely solvable" from "has a real solution." Although overconstrained (or even certain wellconstrained) systems may have no real solutions at all, by our definition, since their set of real zeroes is discrete, they would still be considered "discretely solvable."

An important performance measure of a DR-plan is that the subsystems in the decomposition should be as small as possible: as previously mentioned, the complexity of solving a subsystem by an algebraic/numeric solver is proportional to the size of the subsystem. The optimal DR-plan is the one that minimizes the size of the largest such subsystem.

A problem of finding a DR-plan if one exists is called a DR-problem and we differentiate it from the problem of finding the optimal DR-plan. An algorithm that solves the DR-problem by constructing a DR-plan for an input geometric constraint system is called a DR-planner. To our knowledge, despite its long-standing presence, the DR-problem has not yet been clearly isolated or precisely formulated, although there have been many prior, specialized DR-planners that utilize geometric domain knowledge, e.g.[2, 21, 22, 15], [16, 19, 6, 7, 4, 10, 17], [1, 23, 18, 27, 26].

1.2 Using graph rewriting for solving DR-problems

In order to solve a DR-problem we employ a geometric constraint graph structure (described in the following section) to represent a geometric constraint system. Using this graph structure the DR-problem can be informally described as decomposing the entire graph into small subgraphs of certain type and replacing these small subgraphs by other subgraphs until the resulting graph is small

enough. Hence, using terminology of [20] it could be said that DR-plan is a sequence of applications of graph productions, where the left hand side is the small subgraph to be replaced, the right hand side is the new subgraph that replaces it and the embedding transformation specifies the edges or constraints between the new subgraph and the rest of the graph. Thus solving the DR-problem requires defining a graph grammar that on one hand is general enough to allow rewriting of any subgraph that represents a solvable subsystem, and on the other hand is meaningful with respect to the geometric context, i.e graph transformations should not change solvability of the underlying system. Also, the right hand side of the graph production should be smaller than the left hand side, since this influences the time performance of the DR-planner. Finally, the total number of grammar rules should be small for the computer implementation to be efficient.

In this paper we describe an efficient DR-planner, that uses a particular graph grammar. We would like to note that while we did not use theoretical foundations of graph rewriting theory when proving correctness and convergence properties of our particular DR-planner, we anticipate that a rewriting framework will be helpful for studying general DR-planners. However we are aware of very little previous work [8, 24, 25] on applying graph grammars to CAD systems. One example is [8] that uses graphs as a data structure, so nodes and edges of the graph represent certain objects and relations between these objects respectively. Thus the task of adding new objects or modifying relations between objects can be stated as graph productions. This allows for natural implementation and maintenance of the data structure, however the question of actually solving constraint systems is not addressed.

1.3 Organization of the paper

Section 2 describes the geometric constraint graph structure that is used for solving DR-problems, formally defines a DR-plan in terms of this structure and introduces various performance measures for comparison of various possible DR-plans. Section 3 describes new DR-planner, called Frontier Algorithm, which was systematically developed to excel in the performance measures defined in Section 2, as shown in last subsection where FA is compared to previously known algorithms.

2 Using graph transformations for finding DR-plans

2.1 Geometric constraint graph and degree of freedom analysis

A geometric constraint system C could be represented by a geometric constraint graph. A geometric constraint graph $G = (V, E, w)$ is weighted and undirected with n vertices V (corresponding to the geometric objects of C) and m edges E (corresponding to the geometric constraints of C); weight $w(v)$ is the number of degrees of freedom available to a vertex v and $w(e)$ is the number of degrees of freedom removed by an edge e. The number of degrees of freedom of a rigid

object is roughly the minimum number of independent variables needed to fix this object (or its local coordinate system) in space.

Example. A point in 2d has 2 translational degrees of freedom and a distance constraint in 2d determines 1 degree of freedom. See Figure 1 for a geometric constraint graph corresponding to the geometric constraint problem described in the previous section.

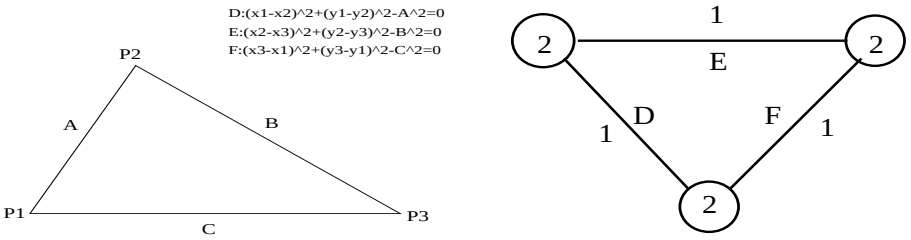

Fig. 1. Geometric constraint problem and corresponding constraint graph

Note that the constraint graph could be a hypergraph, since geometric constraints that involve more than two geometric objects are represented as hyperedges.

A vertex-induced subgraph $A \subseteq G$ that satisfies

$$d(A) = \sum_{e \in A} w(e) - \sum_{v \in A} w(v) \geq -D \tag{1}$$

is called *dense*. The function $d(A)$ is called the *density* of A and meaning of constant D will be explained in the next few paragraphs.

The density of a graph could be used to analyze discrete solvability of a corresponding geometric constraint system because of the following arguments. In the *generic* case we assume that if a number of equations in constraint system is greater than or equal to the number of its variables,then this system is discretely solvable. Similarly, the genericity assumption for constraint graphs is the following: if the total number of degrees of freedom removed by constraints (plus a constant number D) is greater than or equal to the total number of degrees of freedom available to the objects, then the corresponding constraint system is discretely solvable. This is because generically the geometric objects can then be placed rigidly with respect to each other by using only the constraints between them. The absolute position of these objects in space can be fixed by removing D more degrees of freedom. In 2d, D is equal to 3: the number of translational and rotational degrees of freedom of a rigid planar object. In 3d, D is equal to 6 in general: 3 rotational and 3 translational degrees of freedom. If the object and

its local coordinate system are already fixed with respect to a global coordinate system, then $D = 0$.

Therefore we call a *minimal* dense subgraph (i.e one that does not contain a proper dense subgraph) discretely solvable since it generically represents discretely solvable (i.e wellconstrained or consistently overconstrained) subsystems of the corresponding geometric constraint system. A subgraph that is not dense generically represents a system that is not discretely solvable i.e underconstrained.

If the dense graph A is not minimal, the system corresponding to it could still be underconstrained, even in the generic case: density of A could be the result of embedding an overconstrained subgraph $B \subseteq A, d(B) > -D$. Hence dense subgraphs do not in general represent discretely solvable subsystems unless they are minimal. For a non-minimal dense subgraph A to generically correspond to a discretely solvable subsystem, A should remain dense even after replacing any of its overconstrained subgraphs B by any (other) wellconstrained subgraph of density exactly equal to $-D$.

2.2 DR-plans in terms of graph transformations

Consider Figure 2. The top depicts a geometric constraint graph G, where the

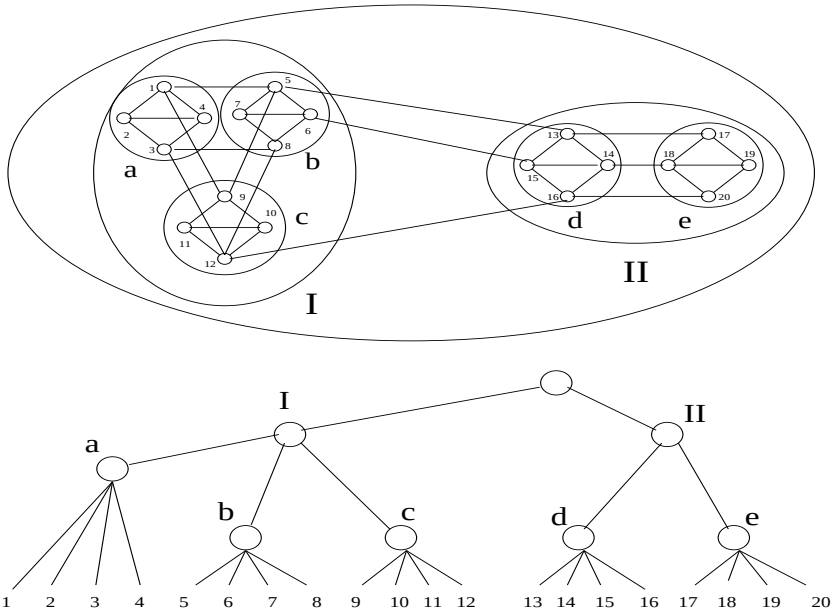

Fig. 2. Geometric constraint graph and a tree of dense subgraphs

weight of each edge is 1, the weight of each vertex is 2, and the dimension dependent constant D is equal to 3. One of the plans for decomposing and recombining

G (and the geometric constraint system that G represents) into small discretely solvable subgraphs (representing subsystems), is to decompose into dense subgraphs $a = \{1, 2, 3, 4\}, b = \{5, 6, 7, 8\}, c = \{9, 10, 11, 12\}, d = \{13, 14, 15, 16\}, e = \{17, 18, 19, 20\}$, recombine their solutions appropriately into a simplified graph so that they can be recombined, say by representing them as vertices a, b, c, d, e; recursively decompose the simplified graph into $I = \{a, b, c\}, II = \{d, e\}$; represent and recombine their solutions as vertices I, II; and so on until the entire graph is recombined and represented as a single vertex. Hence a DR-plan should indicate a sequence of subgraphs or subsystems chosen at every stage whose containment relationships are shown at the bottom of Figure 2.

Formally, a DR-plan Q of a geometric constraint graph G is a sequence of graphs G_i. This sequence has the following properties.

1. $G_1 = G$
2. **Decomposition.** For all i, there is a minimal dense subgraph $S_i \subseteq G_i$
3. **Recombination.** Graph G_{i+1} is a modification of G_i, constructed by replacing subgraph S_i by an *abstraction or simplification* subgraph $T_i(S_i)$ which induces a transformation $T_i(G_i) = G_{i+1}$. This transformation T_i, defined for all subgraphs of G_i, is also called *simplifier* (this is analogous to substituting the solution of S_i into the remaining equation system).
4. $S_n = G_n$

Typically the DR-plan Q is specified not just by the sequence of graphs G_i, but also by the corresponding sequence of discretely solvable subgraphs S_i and the corresponding sequence of graph simplifier transformations T_i.

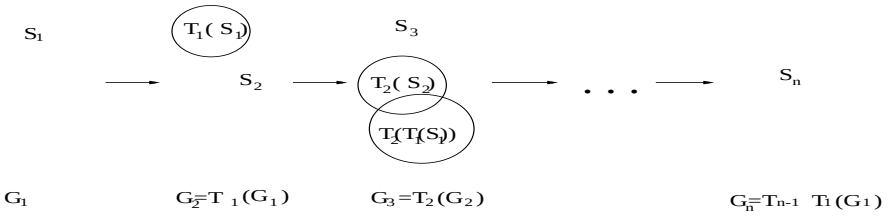

Fig. 3. DR-plan

Figure 3 shows a general DR-plan. Note that there could be many DR-plans for a given geometric constraint graph G. These plans depend upon the choices of S_i and $T_i(S_i)$ at every iteration i. In the next subsection, we will formalize what these choices could be.

2.3 Properties of graph transformations

Graph simplifiers T_i must satisfy the following three natural requirements.

(1) If A is a subgraph of B then $T_i(A)$ is a subgraph of $T_i(B)$.
(2) $T_i(A) \cup T_i(B)$ is the same as the graph $T_i(A \cup B)$.
(3) $T_i(A) \cap T_i(B)$ is the same as $T_i(A \cap B)$.

Note. Any subgraph is understood to be vertex induced. The union of (resp. intersection) of two subgraphs A and B is the graph induced by the union (resp. intersection) of the vertex sets of A and B.

According to the definition of the DR-plan, every DR-plan must satisfy following four requirements, together called *validity*.

(4) Every constraint graph G_i in the DR-plan can be written as $S_i \cup R_i$, where S_i is discretely solvable, S_i and R_i do not have common edges.
(5) For every $A \subseteq R_i$, $T_i(A)$ is isomorphic to A
(6) The initial map T_0 is an identity mapping upon the subsets of G_1
(7) All the pre-images of every S_i, i.e $T_j^{-1}T_{j+1}^{-1}...T_{i-1}^{-1}(S_i)$ for all $1 \leq j < i-1$, are discretely solvable.

Another natural rule for the DR-plans is to prevent the graph simplifiers T_i from mapping discretely solvable subgraphs to not discretely solvable ones and vice versa. The DR-plan Q of G is *discretely solvability preserving* (or *solvability preserving* for short) if for all i and for all subgraphs $A \subseteq G_i$, A is discretely solvable and whenever $A \cap S_i = \emptyset$ or $A \subseteq S_i$ then the corresponding simplified subgraph $T_i(A)$ is discretely solvable and viceversa. The DR plan Q of G is *strictly solvability preserving*, if for all i and for all subgraphs $A \subseteq G_i$, A is discretely solvable implies $T_i(A)$ is discretely solvable and viceversa. The DR plan Q of G is *complete* if for all i and for all discretely solvable subgraphs B of the subgraphs S_i chosen by Q it holds that $B = T_{i-1}T_{i-2}...T_j(S_j)$ for some $j \leq i-1$.

A *conceptual design decomposition* P of a geometric constraint graph G is a collection of discretely solvable subgraphs $P_i \subseteq G$, which are partially ordered with respect to the subgraph relation. A DR-plan Q is said to *incorporate a design decomposition* P, if the sequence of discretely solvable subgraphs S_i in Q embeds a topological ordering of P as a subsequence (a topological ordering is a linear order that is consistent with the natural partial order given by the subgraph relation on P).

In order to define an optimal DR-plan we need to first define the *size* of a geometric constraint graph A and the size of a DR-plan Q. The size of an arbitrary subgraph $A \subseteq G_1 = G$ is equal to $\sum_{v \in A} w(v)$. The size of an arbitrary subgraph $A \subseteq G_i$ is computed by adding the appropriate constant D (3 in 2d, 6 in 3d, as explained earlier) for any of the images of S_j contained in A and adding weights of the vertices of A that are not contained in any such image.

The size of a DR plan of G is the maximum of the sizes of the corresponding discretely solvable subgraphs S_i. The *optimal size* of the constraint graph G is the minimum of sizes of all DR-plans of G. The *optimal DR-plan* of G is the DR-plan that has size equal to the optimal size of G. The *approximation factor*

of a DR-plan Q of the graph G is defined as the ratio of the optimal size of G to the size of Q. The *optimal DR-planning problem* is a problem of finding an optimal DR-plan.

Note. The optimal DR-planning problem is NP-hard. This follows from our result in [13] showing that the problem of finding a *minimum* dense subgraph is NP-hard, by reducing this problem to the CLIQUE. The CLIQUE problem is extremely hard to approximate [9], i.e, finding a clique of size within a $n^{1-\epsilon}$ factor of the size of the maximum clique cannot be achieved in time polynomial in n, for any constant ϵ (unless P=NP). However our reduction of CLIQUE to the optimal DR-planning problem is not a gap-preserving reduction thus the polynomial time approximability of this problem is still an open question.

In addition to the above performance measures for the DR-plans, next we define several performance measures for DR-planning algorithms or *DR-planners*. We assume that all DR-planners use *randomized choices* at each step where an arbitrary selection of a vertex or an edge is needed.

The *worst-choice approximation factor* of a DR-planner on input graph G is the minimum of the approximation factors of all DR-plans Q obtained over all possible random choices. The *best-choice approximation factor* of the DR-planner on input graph G is the maximum of the approximation factors of all the DR-plans Q obtained over all possible random choices.

A DR-planner is *general* if it terminates with a DR-plan when given a discretely solvable system as input. A DR-planner is said to have a *Church-Rosser* property, if it terminates with a DR-plan irrespective of which discretely solvable graph S_i is chosen at the i^{th} stage. Given the Church-Rosser property, at each step, a discretely solvable subgraph S_i can be chosen greedily to satisfy the requirements of the planner. This prevents exhaustive search.

A DR-planner X *adapts to underconstrained constraint graphs* G if every (partial) DR-plan produced by X terminates with a G_n consisting of a set $W = \{A_1, \ldots, A_k\}$ of discretely solvable subgraphs (instead of a single subgraph S_n in case of a well-constrained graph G), such that W is *maximal*, i.e no union of subset of W gives a discretely solvable subgraph.

3 A new DR-planner: Frontier Algorithm

In this section, we present a new DR planner whose development is systematically guided by the performance measures discussed in the previous section. This new algorithm called Frontier Algorithm (*FA*) is designed by apriori choosing a particular type of graph transformation T_i for the DR-plan.

FA uses an earlier algorithm developed by the authors for locating the minimal dense subgraphs S_i at each stage i. This algorithm is 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 discrete solvability. The interested reader is referred to [13] for a description as well as implementation results, and to [14] for an extensive comparison with prior algorithms for isolating discretely solvable/dense

subgraphs with respect to performance measures discussed in the previous section.

The found discretely solvable subgraph S_i is simplified or abstracted as follows. The subgraph of S_i induced by its *internal vertices* (that are not adjacent to any vertex outside of S_i) is replaced by one vertex (the *core vertex*) c_i. The *frontier vertices* of S_i (i.e not internal), edges connecting them, and their weights remain unchanged. The core vertex is connected to each frontier vertex v of S_i by a combined edge e whose weight is the sum of the weights of the original edges connecting internal vertices to v. The weight of the core vertex is chosen so that the density of $T_i(S_i)$ is exactly equal to $-D$ where D is the geometry-dependent constant explained earlier.

For example, consider Figure 4. The graph on the left is the graph G_i, with all edge-weights being 1, all vertex-weights being 2, the dimension dependent constant D in this case is equal to 3. Assume that the discretely solvable subgraph $ABCDEF$ is chosen to be S_i at the current stage. Then B and D are frontier vertices of S_i, A, C, E, F are internal ones. Thus S_i will be simplified by FA into a graph consisting of three vertices c_i, B, D, with the weight of B and D being 2 (the same as in G_i), the weight of edges (c_i, B) and (c_i, D) being 3 (given by $w(AB) + w(CB) + w(EB)$ and $w(CD) + w(ED) + w(AD)$ respectively) and the weight of c_i being 5; whereby the density of $T_i(S_i)$ is $w(c_iB) + w(c_iD) - w(c_i) - w(B) - w(D) = -3$, as required for rigid bodies in 2d.

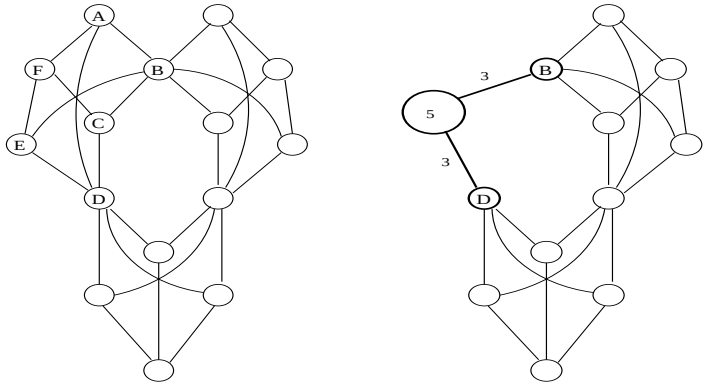

Fig. 4. Geometric constraint graph before and after simplification

Formally, the DR-planner FA could be described by specifying the simplifier transformations. Let S_i be the minimal dense subgraph of G_i found at stage i, let SI be the subgraph induced by the inner vertices of S_i, let FI be the subgraph induced by the frontier vertices of S_i, and let A be any subgraph of G_i. The simplifier transformation T_i 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). 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 $-D$.

3.1 Performance analysis of FA

In this section, we state properties of the new DR-planner FA with respect to the various performance measures defined in Section 2.

Proposition 1. *FA is a valid DR-planner with the Church-Rosser property. In addition, FA finds DR-plans for the maximal discretely solvable subgraphs of underconstrained graphs.*

Proof. If the graph G is not underconstrained, then it will remain so after the replacement of any discretely solvable subgraph by a subgraph of density $-D$, i.e, after a simplification step by FA. Thus, if $G = G_1$ is dense, it follows that all of the G_i are dense. Moreover, we know that if the original graph is discretely solvable, then at each step, FA will in fact find a minimal dense subgraph S_i that consists of more than one vertex, and therefore the size of G_{i+1} is strictly smaller than the size of G_i (using our definition of size) for all i. Thus the process will terminate since the entire graph will eventually be minimal dense, and at that stage, the termination condition $S_n = G_n$ holds. This is independent of which discretely solvable subgraph S_i is chosen to be simplified at the i^{th} stage, showing that FA 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/discretely solvable, the process will not terminate with one vertex, but rather with a set of vertices representing the simplification of a set of maximal discretely solvable subgraphs (such that no combination of them is discretely solvable). This completes the proof that FA is a DR-planner that can adapt to underconstrained graphs.

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

Proposition 2. *FA is solvability preserving, but not strictly solvability preserving in the general case.*

Proof. Consider for example Figure 5. Initially ABC and BCD are dense. After ABC has been simplified into EB (B is frontier vertex, E is core vertex), the image of dense subgraph BCD is EBD which is no longer dense. □

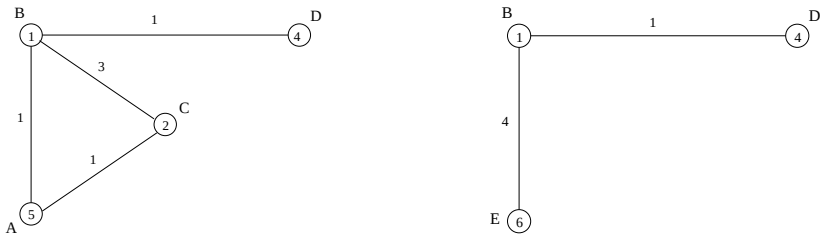

Fig. 5. Original graph is dense, simplified by FA is not

However in geometric applications, situations like this do not arise, i.e where the "inner" part BC imposes some constraints on the "outer" part BD. A more illuminating analysis of FA can be accomplished by assuming that the input is restricted to graphs that have geometrically meaningful interpretations. The dense graph G is said to be *geometrically consistent* if and only if for any two rigid or discretely solvable subgraphs A and B of G such that A contains some inner vertices of B, the union of A and B is rigid as well.

Note. For example, consider case of vertices representing points and edges representing distances in 2d. Here if any two rigid clusters share internal vertices, then they must share more than one frontier vertex, and from 2d geometry, this would mean that their union is rigid as well. Similarly in 3d, if two rigid clusters share internal vertices, then they must share more than 2 frontier vertices, and by 3d geometry, this would mean that their union is rigid as well.

Proposition 3. *If the graph G is geometrically consistent, then FA is solvability preserving and strictly solvability preserving.*

Proof. Let B be a dense graph, and suppose that the dense cluster S_i was simplified by FA. 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 simplifier, $T_i(B)$ is the same as $T_i(B \cup S_i)$. Now, by definition of T_i, $T_i(B \cup S_i)$ is obtained by replacing S_i by $T_i(S_i)$, which has weight exactly $-D$. Due to geometric consistency, $B \cup S_i$ is discretely solvable, and a discretely solvable graph remains dense even after any of its discretely solvable (well or overconstrained) subgraphs is replaced by any (other) subgraph of density exactly $-D$. Thus $T_i(B) = T_i(B \cup S_i)$ is also discretely solvable. □

Proposition 4. *FA is complete*

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

Proposition 5. *FA has worst-choice approximation factor $O(1/n)$ (even for geometrically consistent graphs) (proof can be found in [14]).*

Proposition 6. *The best-choice approximation factor of FA is at least $\frac{1}{2}$ for geometrically consistent graphs.*

Proof. Let G be the weighted constraint graph. Let Q be an optimal DR-plan of G, let q be the size of Q (i.e the size of every cluster S_i simplified under the optimal DR-plan is less than $q+1$). We will show that FA could produce a DR-plan Q' that is "close to" Q. Complete resemblance ($Q = Q'$) may not be possible, since the internal vertices of the cluster $S_i^{'}$, found by FA at the i^{th} 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 Q' within an additive constant of the size of Q.

Suppose that FA is able to follow the optimal DR-plan Q up to the stage i, i.e $S_i = S_i^{'}$. Suppose that there is a cluster S_j in the DR-plan Q such that $i < j$ and S_j contains some internal vertices of S_i. Therefore the simplification of S_j by FA maybe different from the simplification of S_j in Q. However, since the union of S_i and S_j is discretely solvable (due to geometric consistency), FA 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 Q' is at most $q + D$, and since q is at least D, the result follows. □

Proposition 7. *FA 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 (proof can be found in [14]).*

3.2 Example operation of FA

Consider a geometric constraint graph G and design decomposition $P = \{P_1, P_2, P_3, P_4\}$ of Figure 6, where the weight of all vertices in G is equal to 2, the weight of all the edges is equal to 1, the dimension dependent constant D is equal to 3.

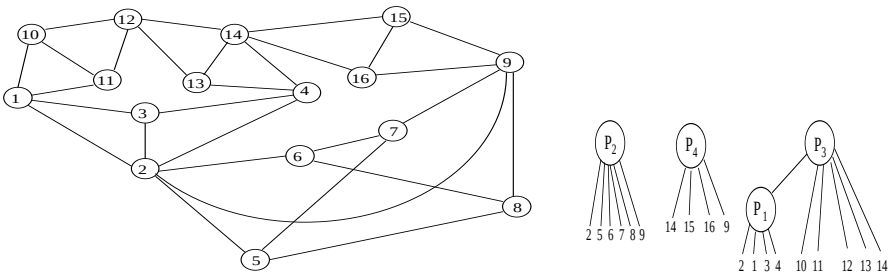

Fig. 6. Input constraint system and design decomposition

For this G and P, the FA planner will construct a DR-plan Q shown in Figure 7. Crucial intermediate graphs G_i in the DR-plan output by FA are shown in Figure 8.

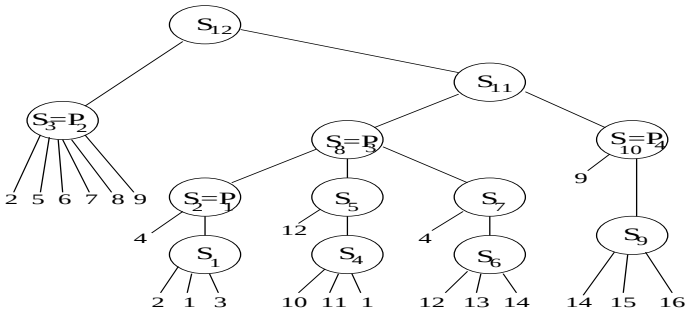

Fig. 7. The structure of discretely solvable subgraphs S_i in DR-plan output by FA for input in Figure 6

The description of discretely solvable subgraphs S_i found at the i^{th} stage is given in the table below ($S_i.Core$ is a new vertex in G_{i+1} that replaces the inner vertices of S_i after simplification, $S_i.FV$ is a list of frontier vertices of S_i, $S_i.CP$ is a list previously found discretely solvable subgraphs that comprise S_i, $S_i.OV$ is a list of vertices of G that has been transformed into S_i).

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

3.3 Comparison of performance

There are 3 previously known types of DR-planners. One type of DR-planner is based on ideas in [2, 15, 16, 21, 22]. During decomposition phase DR-planners of this type locate subgraphs of certain shape (for example triangles). We denote such DR-planners by SR, which stands for "Shape Recognition". Another type of DR-planner is based on ideas in [1, 23, 23, 19, 18]. DR-planners of this type involve finding maximum matching in a bipartite graph formed by geometric objects and geometric constraints. We denote such DR-planners by MM, which stands for "Maximum Matching". The third type of DR-planner is based on

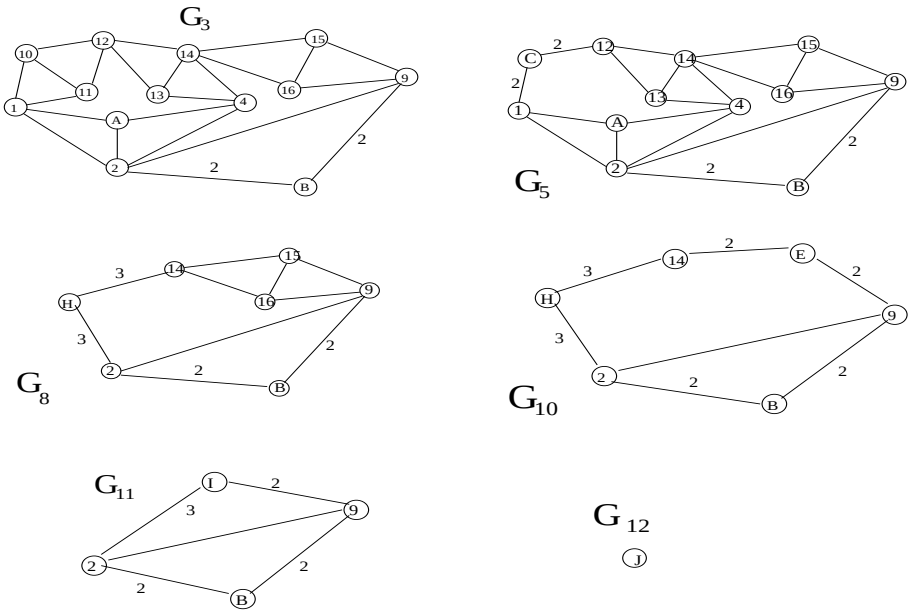

Fig. 8. Crucial G_i in DR-plan

ideas in [13]. This DR-planner behaves similarly to FA during decomposition phase, however during the recombination phase it condenses the entire dense subgraph S_i into a single vertex. We denote such DR-planners by CA, which stands for "Condensing Algorithm".

The detailed descriptions of these DR-planners in terms of graph simplifiers and proofs for their performance analysis are given in [14]. Here we briefly outline main advantages of FA. A DR-planner of type SR can only produce DR-plans that require the dense subgraphs S_i to consist of triangles or other fixed repertoire of patterns. Even when restricted to such inputs, a DR-planner of type SR is still not general and not complete. A DR-planner of type MM could only produce DR-plans that require the discretely solvable subsystems S_i to represent rigid bodies that are fixed or grounded with respect to a single coordinate system. Even when restricted to such inputs, a DR-planner of type MM still cannot handle underconstrained graphs, cannot incorporate an input design decomposition and is not complete.

In contrast, FA places no restrictions on inputs, it is general, it can handle underconstrained graphs, it can incorporate design decomposition, is valid, complete, solvability preserving and strictly solvability preserving for the geometrically consistent graphs. While all four types of DR-planners have worst-choice approximation factor of $O(\frac{1}{n})$, FA is the only algorithm that has constant best-choice approximation factor.

Acknowledgements. We would like to thank the anonymous reviewers for their helpful suggestions.

References

1. S. Ait-Aoudia, R. Jegou, and D. Michelucci. Reduction of constraint systems. In *Compugraphics*, pages 83–92, 1993.
2. W. Bouma, I. Fudos, C. Hoffmann, J. Cai, and R. Paige. A geometric constraint solver. *Computer Aided Design*, 27:487–501, 1995.
3. B. Bruderlin and R. Roller ed.s. *Geometric constraint solving and applications*. Springer-Verlag, 1998.
4. G. Crippen and T. Havel. *Distance Geometry and Molecular Conformation*. John Wiley & Sons, 1988.
5. I. Fudos. *Geometric Constraint Solving*. PhD thesis, Purdue University, Dept of Computer Science, 1995.
6. I. Fudos and C. M. Hoffmann. Correctness proof of a geometric constraint solver. *Intl. J. of Computational Geometry and Applications*, 6:405–420, 1996.
7. I. Fudos and C. M. Hoffmann. A graph-constructive approach to solving systems of geometric constraints. *ACM Trans on Graphics*, pages 179–216, 1997.
8. H. Gottler, J. Gunther, and G. Nieskens. Use graph grammars to design cad-systems! In *Lect. Notes in CS, Vol 532*, pages 396–410. Springer-Verlag, 1991.
9. J. Hastad. Clique is hard to approximate within $n^{1-\epsilon}$. In *Proceedings of Foundations of Computer Science*. IEEE press, 1996.
10. T. Havel. Some examples of the use of distances as coordinates for Euclidean geometry. *J. of Symbolic Computation*, 11:579–594, 1991.
11. C. M. Hoffmann. Solid modeling. In J. E. Goodman and J. O'Rourke, editors, *CRC Handbook on Discrete and Computational Geometry*. CRC Press, Boca Raton, FL, 1997.
12. C. M. Hoffmann and J. Rossignac. A road map to solid modeling. *IEEE Trans. Visualization and Comp. Graphics*, 2:3–10, 1996.
13. Christoph M. Hoffmann, Andrew Lomonosov, and Meera Sitharam. Finding solvable subsets of constraint graphs. In *Constraint Programming '97*, Linz, Austria, 1997.
14. Christoph M. Hoffmann, Andrew Lomonosov, and Meera Sitharam. Decomposition of geometric constraints systems: Part i and part ii. In *51 page Manuscript submitted to a journal, University of Florida technical report*, 1998.
15. Christoph M. Hoffmann and Pamela J. Vermeer. Geometric constraint solving in R^2 and R^3. In D. Z. Du and F. Hwang, editors, *Computing in Euclidean Geometry*. World Scientific Publishing, 1994. second edition.
16. Christoph M. Hoffmann and Pamela J. Vermeer. A spatial constraint problem. In *Workshop on Computational Kinematics*, France, 1995. INRIA Sophia-Antipolis.
17. Ching-Yao Hsu. *Graph-based approach for solving geometric constraint problems*. PhD thesis, University of Utah, Dept. of Comp. Sci., 1996.
18. G. Kramer. *Solving Geometric Constraint Systems*. MIT Press, 1992.
19. R. Latham and A. Middleditch. Connectivity analysis: a tool for processing geometric constraints. *Computer Aided Design*, 28:917–928, 1996.
20. M. Nagl. A tutorial and bibliographical survey on graph grammars. In *Lect. Notes in CS, Vol 73*, pages 70–126. Springer-Verlag, 1979.

21. J. Owen. Algebraic solution for geometry from dimensional constraints. In *ACM Symp. Found. of Solid Modeling*, pages 397–407, Austin, Tex, 1991.
22. J. Owen. Constraints on simple geometry in two and three dimensions. In *Third SIAM Conference on Geometric Design*. SIAM, November 1993. To appear in Int J of Computational Geometry and Applications.
23. J.A. Pabon. Modeling method for sorting dependencies among geometric entities. In *US States Patent 5,251,290*, Oct 1993.
24. Detlev Ruland. Edm—a data model for electronic cad/cam-applications. In Gottfried Tinhofer and Gnther Schmidt, editors, *Graph-theoretic concepts in computer science*, volume 246, pages 290 – 305. Lecture Notes in Computer Science, Springer-Verlag, 1986.
25. Detlev Ruland. Cadula—a graph-based model for monitoring cad-processes. In M. Nagl, editor, *Graph-theoretic concepts in computer science*, volume 411, pages 63–77. Lecture Notes in Computer Science, Springer-Verlag, 1989.
26. D. Serrano. Managing constraints in concurrent design: first steps. In *Proc. Computers in Engineering*, pages 159 –164, Boston, MA, 1990.
27. D. Serrano and D.C. Gossard. Combining mathematical models with geometric models in cae systems. In *Proc. Computers in Engineering*, pages 903 –909, Chicago, IL, 1986.