Inductive and Numerical Methods in Knowledge Compilation

Giuseppe Cerbone
Thomas G. Dietterich
Computer Science Department
Oregon State University
Corvallis, OR 97331-3902

90-30-2

Inductive and Numerical Methods in Knowledge Compilation

March 14, 1990

Giuseppe Cerbone[†]
Thomas G. Dietterich[‡]

Oregon State University
Computer Science Dept.
Corvallis, OR 97331
Phone (503) 737-3273

cerbone@cs.orst.edu
tgd@cs.orst.edu

Abstract

Many important application problems can be formalized as constrained non-linear optimization tasks. However, numerical methods for solving such problems are brittle and do not scale well. Furthermore, they do not provide much insight into the structure of the problem space. This paper describes a method for discovering efficient rules that can bypass the numerical methods and produce solutions directly. The rules also provide some insight into the problem structure. The method is illustrated in the domain of 2-dimensional structural design. To discover design rules in this domain, a numerical optimizer is run on example problems. The resulting optimal designs are then analyzed to detect geometric patterns. These patterns can then be applied to reformulate the search space and the objective function and construct a more direct and efficient problem-solving procedure. As a side-effect of this process, new geometric features are defined that can be used to help discover additional design rules for more complex problems.

[†]This research was supported by the National Science Foundation under Presidential Young Investigator Award IRI-86-57316 with additional support from SUN Microsystems. During part of this research, G.Cerbone was supported by a research fellowhip (CNR 203.01.43) from the Italian National Research Council.

[‡]The authors wish to thank Prasad Tadepalli for insightful comments on drafts of this paper.

1 Introduction

Many important applications can be formalized as constrained optimization tasks. For example, we are studying the engineering domain of two-dimensional (2-D) structural design. In this task, the goal is to design a structure of minimum weight that bears a set of loads and whose components do not enter any forbidden regions. Figure 1 shows a non-optimal solution to a design problem in which there is a single load (L1), two stationary support points (S1 and S2), and one forbidden region (FR). The solution consists of six members, E1, E2, E3,E4, E5 and E6, that connect the load to the support points. In principle, optimal solutions to problems of this kind can be found by numerical optimization techniques. However, in practice ([Pik86]) these techniques are very slow and they can only solve problems with a limited number of unknowns. Hence, their applicability to real-world problems is severely restricted.

In contrast, human experts are able to construct satisfactory, albeit sub-optimal, solutions to these problems. One group ([NGa89]) has constructed an expert system, MOSAIC, for this task by encoding the knowledge of expert designers. Rather than employing numerical techniques, MOSAIC parses the input problem to identify abstract features. Using these features, it is able to describe and construct a solution.

The goal of our research is to develop knowledge compilation techniques that can automatically discover rules of optimal design in this domain. Our approach is to solve simple problems and transfer the knowledge acquired from these problems to help solve more complex problems. By this boot-strapping process, we hope to obtain a high-performance design system with broad coverage.

The key question that we must answer is to determine what form of knowledge can be usefully transferred from simple problems to complex ones. The individual design rules learned by the system are unlikely to transfer. However, in the process of formulating design rules, it is helpful to define new abstract features of the kind used in MOSAIC. We hypothesize that these features will prove useful in acquiring design rules for more complex problems. Hence, these features will provide the vehicle for transferring knowledge from simple problems to complex ones.

To discover design rules for a particular problem, we first solve the problem via numerical optimization. Then, the optimal solution is analyzed inductively to discover geometrical patterns. The observed patterns are employed as constraints on the search for an optimal design. In fact, the patterns can often be exploited to reformulate the search space. For example, the search space may be restricted to a small finite set of candidate solutions. The objective function may also be simplified. Even when the search space cannot be converted to a space of finite candidates, it can often be reduced in dimensionality — that is, the number of unknown coordinates of connection points. All of these reformulations substantially improve the speed with which a problem solver can find solutions to similar design problems.

The remainder of the paper is organized as follows. Section 2 presents the 2-D structural design task. This is followed in Section 3 by an overview of the method we have developed and an example of the kind of rules we want the system to learn. Section 4 describes the geometric domain knowledge supplied to the system. We employ a bottom-up search strategy to discover useful patterns and constraints, and this is described in Section 5. Finally, in Section 6, we give an example of how abstract features learned in one problem can be usefully transferred to a more complex problem.

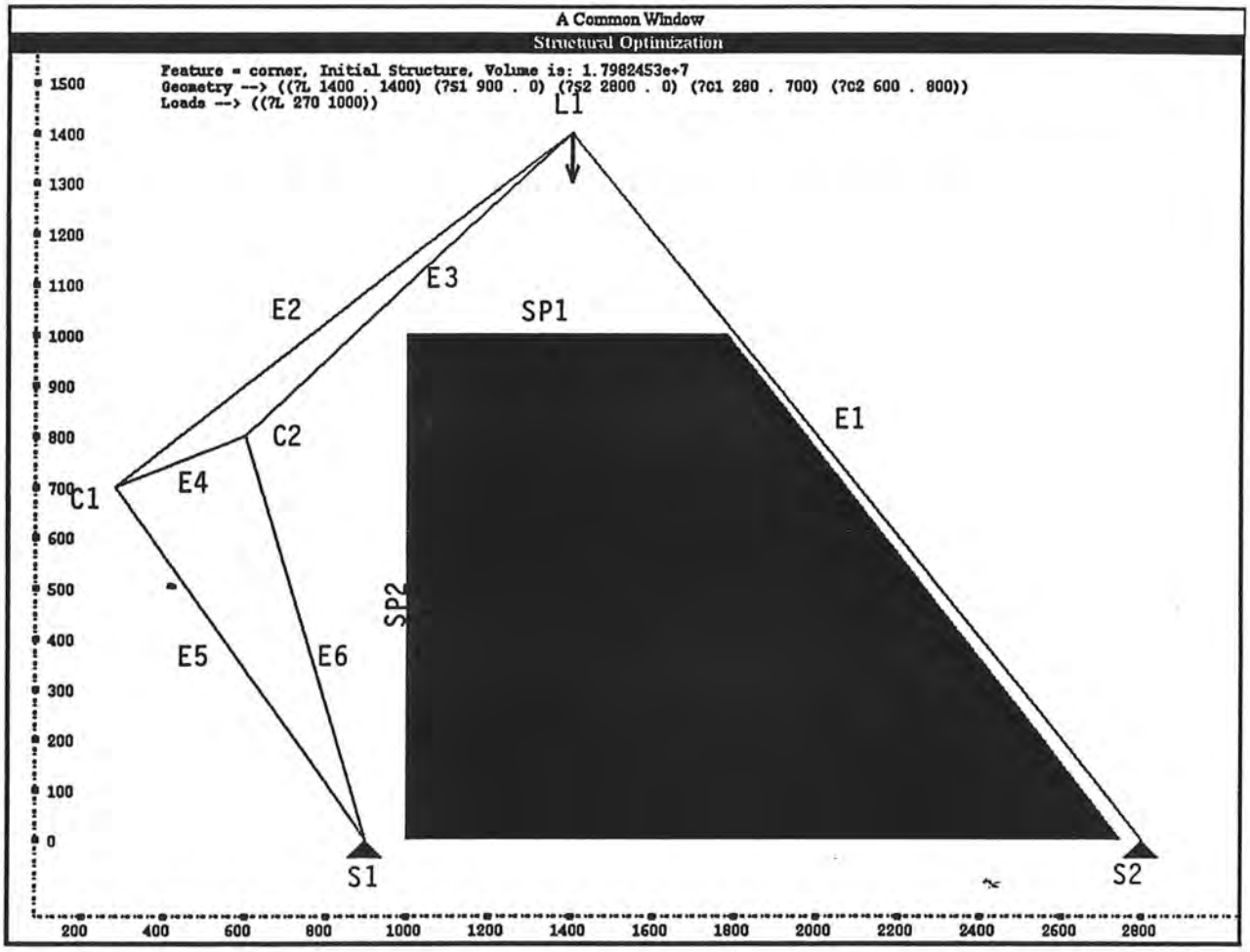

Figure 1: A non-optimal solution to a simple 2-D structural design problem.

2 Task description

Table 1 describes the task of 2-D structural design. A complete design consists of a connected collection of columns and rods. Columns withstand compressive forces while rods oppose tensile stresses. As we have already seen, Figure 1 is an example of a structural design problem and solution in which L1 is the load, S1 and S2 are two supports, C1 and C2 are intermediate connection points, and the forbidden region is the filled polygon. Although not indicated in the figure, members E2 and E5 are in tension, and members E1, E3, E4, and E6 are in compression. The solution shown here is far from optimal.

We have introduced several simplifying assumptions to provide a tractable testbed for developing and testing knowledge compilation methods. Specifically, we assume that structural members are joined by frictionless pins, only statically determinate structures [1] are considered, only one material is used for both columns and rods, the cross section of a column is square, columns and rods of any length and cross sectional area are available, supports have no freedom of movement along either axis, and a local minimum suffices. We hope to relax these assumptions as the work progresses.

Given these assumptions, the weight of each member is proportional to the volume of the member, which is in turn proportional to the product of the magnitude of the force acting in the member and the length of the member. Hence, numerical optimization can simply minimize the volume of the structure.

One common procedure ([PS70]) for solving problems of this kind consists of the three steps shown in Table 2. First, the problem solver chooses what is loosely termed a "topology" for the solution structure. A topology is a graph whose edges are members and whose nodes are loads, supports, and intermediate connection points. The topology also indicates the configuration of the graph with respect to the forbidden regions. The topology does not further specify the locations, lengths, or cross-sectional areas of the members.

Once a topology is selected, the locations of the loads and supports are known, so the second step is to determine the locations of the connection points (and hence the lengths, locations, internal forces, and cross-sectional areas of the members) so as to minimize the weight of the structure. This is usually accomplished by numerical non-linear optimization techniques.

The third and final step in the process optimizes the shapes of the individual members. This can often be accomplished by linear programming.

In this paper, we consider only the second step, which can be formulated as a non-linear constrained optimization problem. The objective function is the volume of the structure, and the constraints are the equilibrium equations from the method of joints and the inequalities derived from the requirement that no member enters the forbidden regions. The unknown parameters are the (x, y)-coordinates of the intermediate connection points.

In future work, we plan to develop topology selection rules (step 1) by analyzing the geometrical design rules learned using the methods described in this paper.

[1] A statically determinate structure contains no redundant members, and hence, the geometrical layout of the structure completely determines the forces acting in each member. These forces can be computed using the *method of joints* ([WS84]).

Table 1: The 2-D Design Task.

Given:	A 2-dimensional region R A set of convex **polygonal** forbidden regions within R A set of stable points (supports) A set of external loads with application points within R
Find:	A structural configuration made of straight columns and rods that has minimum weight, is stable with respect to all external loads, uses as many supports as necessary from the given set, does not contain members that cross one another, and does not enter any of the forbidden regions.

Table 2: Optimization stages for the structural design task.

Optimization	*Explanation*
Topological	which considers as variables the number of connection points and the connectivity among them. For instance, the frames in Figures 1 and 2 are topologically equivalent.
Geometrical	in which the topology is assumed and the variables to be optimized are the locations of the extra connection points. Figure 2 shows the result of geometrical optimization applied to the structure in Figure 1.
Physical	in which topology and geometry are assumed and physical characteristics of the frame are optimized. Examples of parameters optimized in this phase are: material and shape of members.

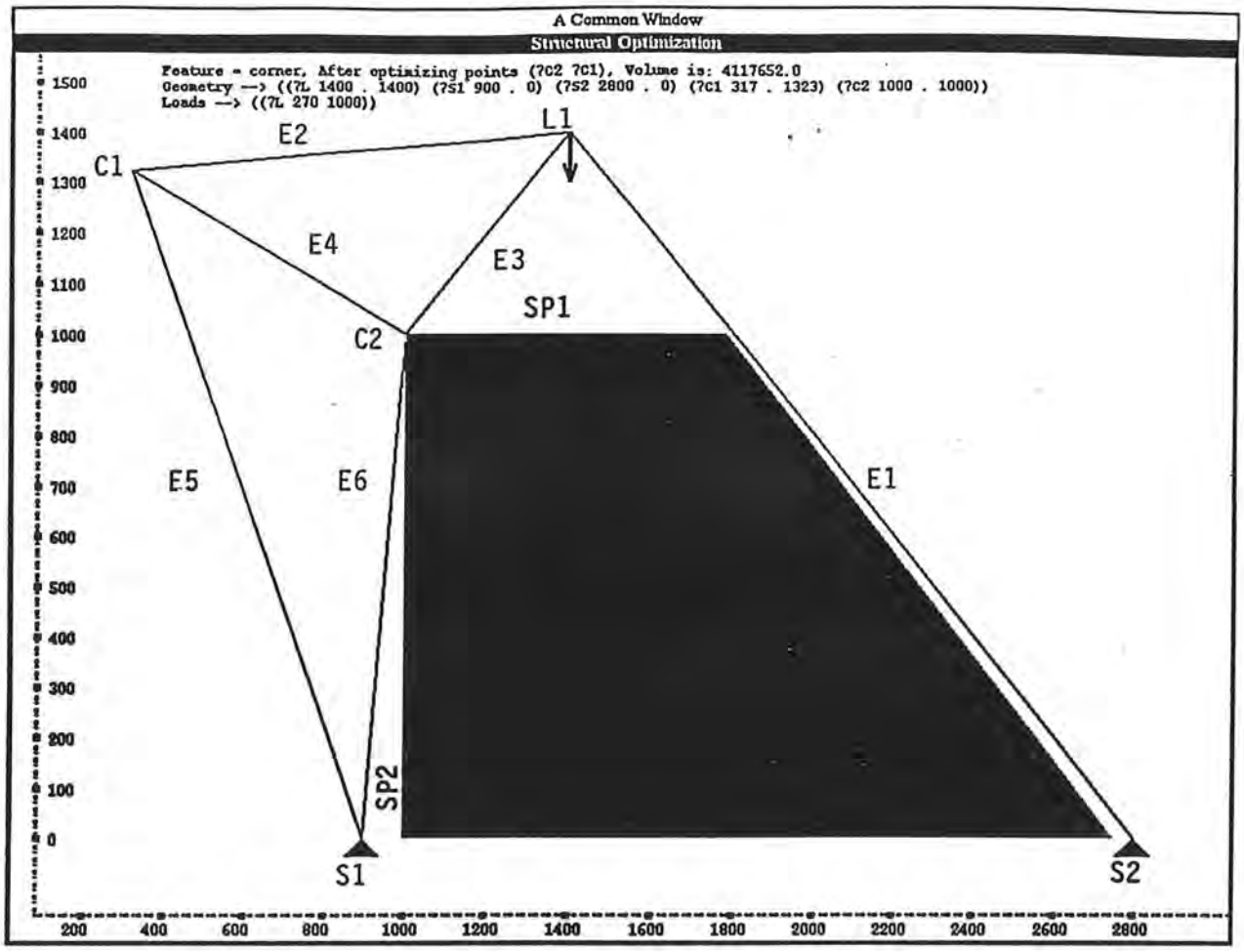

Figure 2: An optimal solution to the simple problem from Figure 1.

3 Discovering Optimal Design Rules

Our approach to discovering geometrical design rules involves three main steps: (a) apply numerical optimization techniques to find the optimal solution to a given example problem, (b) apply geometric knowledge to discover patterns that relate the optimal solution to the given geometric objects, (c) treat these patterns as constraints and incorporate them into the solution generator.

To see how these three steps work, consider again the example problem in Figure 1 which we shall refer to as the "corner" example. During the first step, this initial structure is optimized to produce the structure shown in Figure 2. This optimization process can be formalized[2] as follows:

[2]We employ Prolog-like logical notation throughout.

```
If    T= λ(C1,C2)
        topology([load(L1)],
                 [support(S1),support(S2)],
                 [region(FR)],
                 [point(C1),point(C2)],
                 [edge(E1,L1,S2),edge(E2,L1,C1),
                  edge(E3,L1,C2),edge(E4,C1,C2),
                  edge(E5,C1,S1),edge(E6,C2,S1)])

Then ⟨C1,C2⟩ =
        min(⟨P1,P2⟩,
            [P1 ∈ ℜ², P2 ∈ ℜ², legal(T(P1,P2))],
            weight(T(P1,P2))).
```

This states that the points C1 and C2 should be chosen as those points P1 and P2 (drawn from $\Re^2$) that minimize the weight of the given topology and do not violate any constraints on legal structures. The min operator takes three arguments: the variables to consider, a conjunction of predicates defining a set of values from which those variables should be drawn, and the objective function to be minimized. Notice that a topology is a very complex term. Lambda-binding is used to parameterize the term with respect to the two unknown connection points.

The topology symbol groups together loads, supports, connection points, forbidden regions, and edges. Because we are only concerning ourselves with geometric optimization rules, this topological information is assumed to be given.

The second step involves looking for patterns relating the locations of the connection points C1 and C2 to the given geometric objects. For now, let us focus on C2. Our system notices that C2 lies on what we would call a "corner" of the forbidden region. However, "corner" is not one of the geometric concepts initially provided to the system. Instead, the system constructs the following chain of geometric relations. C2 lies at the intersection of two lines. These two lines are the boundary lines of two semiplanes. The two semiplanes are chosen from the semiplanes that define the forbidden region. There is only one forbidden region, and it is a given. This chain is used to define a new feature, g-corner, which we call "generalized corner" because, unlike a corner, a g-corner may lie outside the forbidden region (at any point where two boundary lines intersect).

The system also notices that in order to build a structure of the desired topology, it is necessary that there be a clear line-of-sight from the load L1 to C2 and from the stable support S1 to C2.

In the third step, we incorporate these observed patterns to reformulate the generator. Specifically, we obtain the following design rule:

```
If    T= λ(C1,C2)
        topology([load(L1)],
                 [support(S1),support(S2)],
                 [region(FR)],
                 [point(C1),point(C2)],
                 [edge(E1,L1,S2),edge(E2,L1,C1),
                  edge(E3,L1,C2),edge(E4,C1,C2),
                  edge(E5,C1,S1),edge(E6,C2,S1)]),
        line(Ln,L1,S1)

Then  C2=min(P2,
             [g-corner(P2,FR),see(P2,L1,[FR]),see(P2,S1,[FR])],
             distance(P2,Ln)),
      C1=min(P1,
             [P1 ∈ ℜ², legal(T(P1,C2))],
             weight(T(P1,C2)))
```

The rule states that in problems with the same topology, the problem solver should construct a line Ln from L1 to S1 and identify all g-corners of the forbidden region. It should then place C2 at a g-corner that is visible from L1 and S1 and that minimizes the distance to Ln. C1 is then located to minimize the weight of the entire structure. The new feature g-corner is defined as follows:

```
g-corner(Gc, FR)  ⇔  semiplane(SP1, FR), boundaryline (LN1, SP1),
                     semiplane(SP2, FR), boundaryline (LN2, SP2),
                     intersect(LN1,LN2,Gc)
```

The relation intersect(A,B,C) says that the intersection of objects A and B is object C.

Of these three steps, the second step is the most complex and difficult. Hence, the remainder of the paper discusses the geometric knowledge given to the system and the search methods employed to discover geometric patterns.

4 Geometric Knowledge Base

In this section we describe the geometric knowledge base that is applied to discover patterns in the results of the numerical optimization process. There are three parts to this knowledge base: (a) primitive geometric entities, (b) relationships among geometric entities (and the definitions of those relationships in terms of the primitives), and (c) relationships between topology and geometry.

There are seven primitive geometric entities: points, lines, line segments, semiplanes, regions, distances, and angles. Each entity has a standard form. For example, points are represented as (x, y) coordinate pairs. Lines are represented by the three coefficients in the equation $ax + by + c = 0$. Semiplanes are represented by a line and a comparison symbol (either $\geq$ or $\leq$). By substituting the comparison symbol for the $=$ sign in the linear equation, one obtains the appropriate semiplane. Forbidden regions are represented as sets of semiplanes. Line segments are represented by a distinguished endpoint and a direction vector that when added to the distinguished endpoint produces the other endpoint.

Table 3: Relations among geometrical objects.

Relation	*Explanation*
angle(α,Sg1,Sg2)	α is the angle between segments Sg1 and Sg2. The segments share the same designated endpoint.
line(Ln,P1,P2)	Ln is the line that passes through points P1 and P2.
line-polar(Ln,Sg,α)	Ln is the line that passes through the designated endpoint of the segment Sg at angle α from the segment.
segment(Sg,P1,P2)	Sg is the segment with endpoints P1 and P2. P1 is the designated endpoint.
semiplane(Sp,FR)	Sp is a semiplane defining region FR.
boundaryline(Ln,Sp)	Ln is the boundary of semiplane Sp.
intersect(Obj1,Obj2,Obj3)	Obj1 and Obj2 intersect to produce Obj3, which may be nil.
same(Obj1,Obj2)	Obj1 and Obj2 are the same geometric object.

There are many relationships that can be defined among these geometric primitives. The full domain theory provides rules for defining each relationship. Table 3 lists each of these relations.

In addition to the geometric primitives and relations among primitives, there is one rule concerning the topology of a structure:

If edge(E,P1,P2)
Then see(P1,P2,FRs)

This states that if the topology contains an edge that connects P1 to P2, then there must be a line-of-sight relationship between P1 and P2 with respect to all forbidden regions FRs. In other words, the line segment connecting P1 and P2 must not intersect any forbidden region in FRs.

5 Finding Geometric Patterns and Constructing Rules

Given this knowledge base, the system searches for patterns by reasoning forward from the given geometrical objects and the unknown objects (i.e., the intermediate connection points) in an effort to find a general, constraining relationship between them. For expository purposes, we will divide this search into four phases, although they are interleaved in the implementation.

1. Reason forward from givens to identify geometric objects related to the givens.
2. Look for patterns relating these geometric objects to the unknowns.
3. Define new features that encapsulate the discovered patterns.
4. Accumulate see constraints from the topology.

Let us follow the corner example through each of these phases.

Phase 1: Reason forward from the givens. Reasoning begins with the load, the supports, and the forbidden region specified in the given topology. It then progresses by chaining together the relationships listed in Table 3. Each geometric object extracted by this process can be labelled as given. Here is a listing of some of the resulting chains of relations:

given(FR),
semiplane(SP1, FR), semiplane(SP2, FR),
semiplane(SP3, FR), semiplane(SP4, FR),
boundaryline(LN1, SP1), boundaryline(LN2, SP2),
boundaryline(LN3, SP3), boundaryline(LN4, SP4).

Phase 2: Find patterns. The second phase involves reasoning forward from the unknowns and the objects identified in Phase 1 looking for additional instances of the relations from Table 3.

For the corner example, the system notices that intersect(LN1,C2,C2) and intersect(LN2,C2,C2)—that is, C2 lies on both lines LN1 and LN2. Reasoning forward through the rules defining intersect, it can conclude that intersect(LN1,LN2,C2). This phase terminates either when no more rules are applicable or when the unknowns have been related to the givens.

Phase 3: Define new features. In the third phase, we collect all of the relations connecting the unknown to the original givens and use this to define a new abstract feature. In the corner example, this leads us to define the g-corner concept:

g-corner(Gc, FR) ⇔ semiplane(SP1, FR), boundaryline (LN1, SP1),
semiplane(SP2, FR), boundaryline (LN2, SP2),
intersect(LN1,LN2,Gc).

All instances of this new feature are detected via forward reasoning as in Phase 1 and added to the collection of given geometric objects. In particular, we obtain g-corner(C2,FR). In Figure 2, there are five g-corners, the four corners and the intersection of the two non-parallel boundaries of the forbidden region.

Phase 4: Accumulate line-of-sight constraints. The topology includes two edges connecting C2 to the givens, namely to L1 and S1. Hence, by applying the line-of-sight rule, we can infer the constraints:

see(C2,L1,[FR]), see(C2,S1,[FR]).

This completes the forward search for patterns and constraints. To obtain an efficient design rule, we can now reformulate the solution procedure to incorporate these constraints.

Because the g-corners can be constructed from the given geometric objects, they can serve as a generator of candidate points for C2. This suggests a generator of the form:

C2 = min(P2,
[g-corner(P2,[FR]),see(C2,L1,[FR]),see(C2,S1,[FR])],
objective-function)

where we need to find some objective function. We cannot use the weight function, because that requires values for both C1 and C2. One strategy we are investigating is to apply the "minimum perturbation principle", which says that the solution to a complex problem should be a minimum perturbation of the solution to a simpler problem. Specifically, we can generate a "relaxed" version of the problem in which the forbidden region has been removed. The optimal solution to this relaxed problem includes a member connecting the load L1 directly to support S1. We can then use the distance between P2 and the line connecting L1 and S1 as the objective function.

This gives us the first part of the final design rule:

```
If    T= λ(C1,C2)
        topology([load(L1)],
                 [support(S1),support(S2)],
                 [region(FR)],
                 [point(C1),point(C2)],
                 [edge(E1,L1,S2),edge(E2,L1,C1),
                  edge(E3,L1,C2),edge(E4,C1,C2),
                  edge(E5,C1,S1),edge(E6,C2,S1)]),
        line(Ln,L1,S1)

Then  C2=min(P2,
             [g-corner(P2,FR),see(P2,L1,[FR]),see(P2,S1,[FR])],
             distance(P2,Ln))
```

This strategy is an instance of the general technique ([Gas79], [MP89]) of using the solution to a relaxed version of a difficult problem as a heuristic guide for solving the original problem.

Now that we have an efficient procedure for finding C2, the point can be marked as a given geometric object.

Let us turn our attention to C1, the other connection point. To construct an efficient generator for C1, we repeat the four steps given above. This explanation depends on discovering that the angles L1 – $\widehat{C2}$ – C1 and C1 – $\widehat{C2}$ – S1 are equal.

In Phases 1 and 2, among others, the following relations are derived:

segment(Sg1,C2,L1), segment(Sg2,C2,C1), segment(Sg3,C2,S1),
angle(α,Sg1,Sg2), angle(β,Sg2,Sg3), angle(γ,Sg1,Sg3),
$\alpha = \beta$, $\alpha + \beta = \gamma$, $\alpha = \frac{\gamma}{2}$,
line-polar(Lna,Sg1,α), intersect(Lna,C1,C1)

In Phase 3, the new feature bisector-point(P,C2,L1,S1) is defined to be any point that lies on the line bisecting the angle L1 – $\widehat{C2}$ – S1:

bisector-point(P,C2,L1,S1) $\Leftrightarrow$ segment(Sg1,C2,L1), segment(Sg2,C2,S1),
angle(γ,Sg1,Sg2), $\alpha = \frac{\gamma}{2}$, line-polar(Lna,Sg1,$\alpha$),
intersect(Lna,P,P).

Forward inference identifies C1 as a bisector-point: bisector-point(C1,C2,L1,S1).

In Phase 4, we can obtain the following line-of-sight constraints:

see(C1,L1,[FR]), see(C1,C2,[FR]), see(C1,S1,[FR]).

It is easy to incorporate these constraints into the solution procedure, because we can use the weight function as the objective function. The final solution procedure is

```
If    T= λ(C1,C2)
        topology([load(L1)],
                 [support(S1),support(S2)],
                 [region(FR)],
                 [point(C1),point(C2)],
                 [edge(E1,L1,S2),edge(E2,L1,C1),
                  edge(E3,L1,C2),edge(E4,C1,C2),
                  edge(E5,C1,S1),edge(E6,C2,S1)]),
        line(Ln,L1,S1)

Then  C2=min(P2,
             [g-corner(P2,FR),see(P2,L1,[FR]),see(P2,S1,[FR])],
             distance(P2,Ln)),
      C1=min(P1,
             [P1 ∈ ℜ², bisector-point(P1,C2,L1,S1),
              see(P1,L1,[FR]),see(P1,C2,[FR]),see(P1,S1,[FR])],
             weight(T(P1,C2))).
```

6 Knowledge Transfer from Simple to Complex Problems

As we discussed in the introduction, our overall strategy involves transferring knowledge acquired from analyzing simple problems to solve more complex problems. This is accomplished by transferring the new features and relations that are defined during Phase 3 of the explanation-generation process.

To see how this works, consider Figure 3, which we shall refer to as the "tangent" example. The design rule constructed in the last section does not transfer to this problem because there are no g-corners with line-of-sight relations to L1 and S1. Hence, it is necessary to derive a new rule to handle problems of this kind. Fortunately, the features and relations defined in the previous example can be applied to this new problem.

To derive the new rule, we begin by applying the numerical optimizer to produce an optimal solution (shown in Figure 4). Then, we reason forward from the givens to find patterns. Here are some of the relationships discovered in this process:

given(FR), semiplane(SP1, FR), semiplane(SP2, FR),
semiplane(SP3, FR), semiplane(SP4, FR),
boundaryline(LN1, SP1), boundaryline(LN2, SP2),
boundaryline(LN3, SP3), boundaryline(LN4, SP4),
g-corner(CR1, FR), g-corner(CR2, FR), g-corner(CR3, FR),
g-corner(CR4, FR), g-corner(CR5, FR), g-corner(CR6, FR),
line(LNa,L1,CR1), line(LNb,S1,CR2),
intersect(Lna,Lnb,C2)

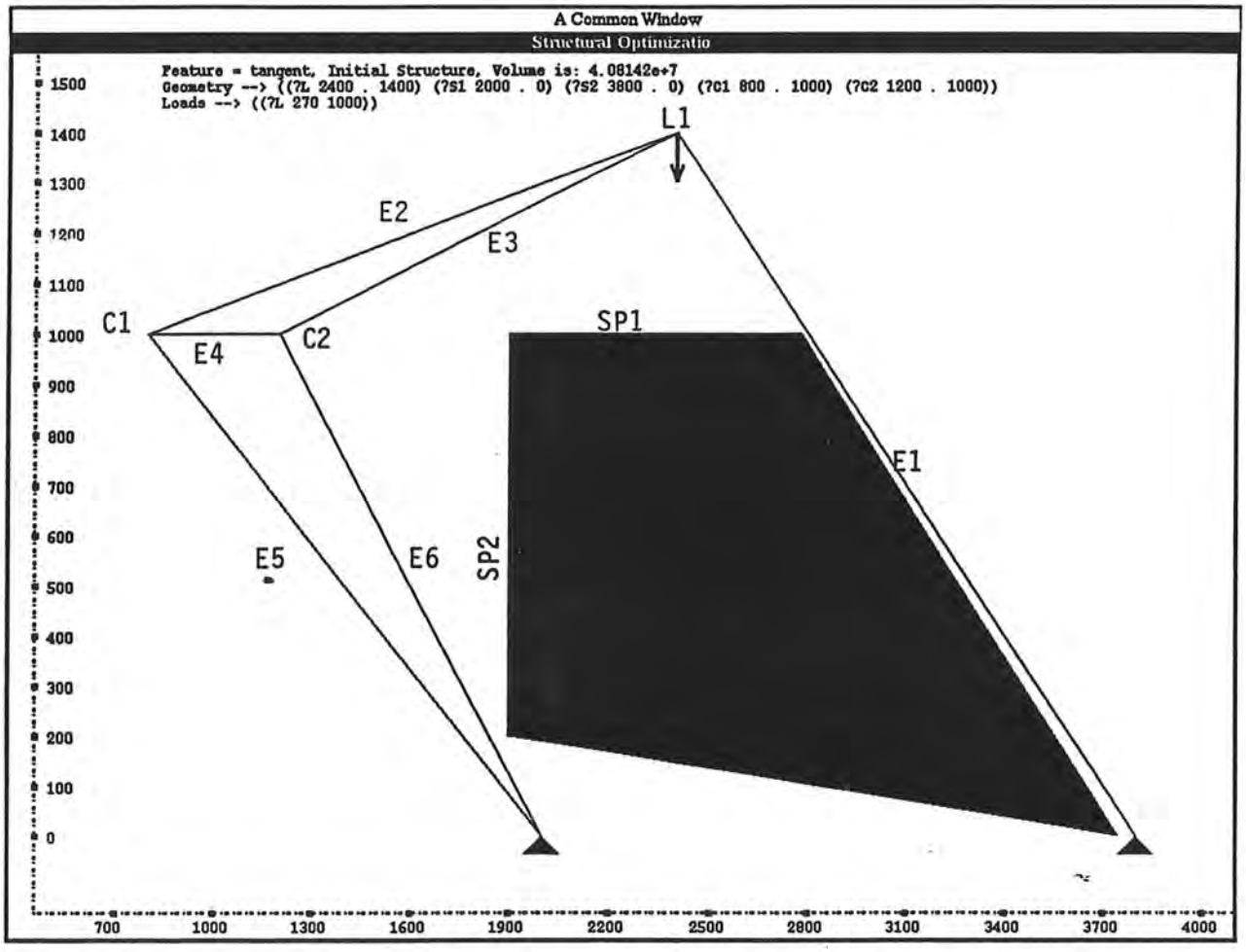

Figure 3: A non-optimal solution to the tangent problem.

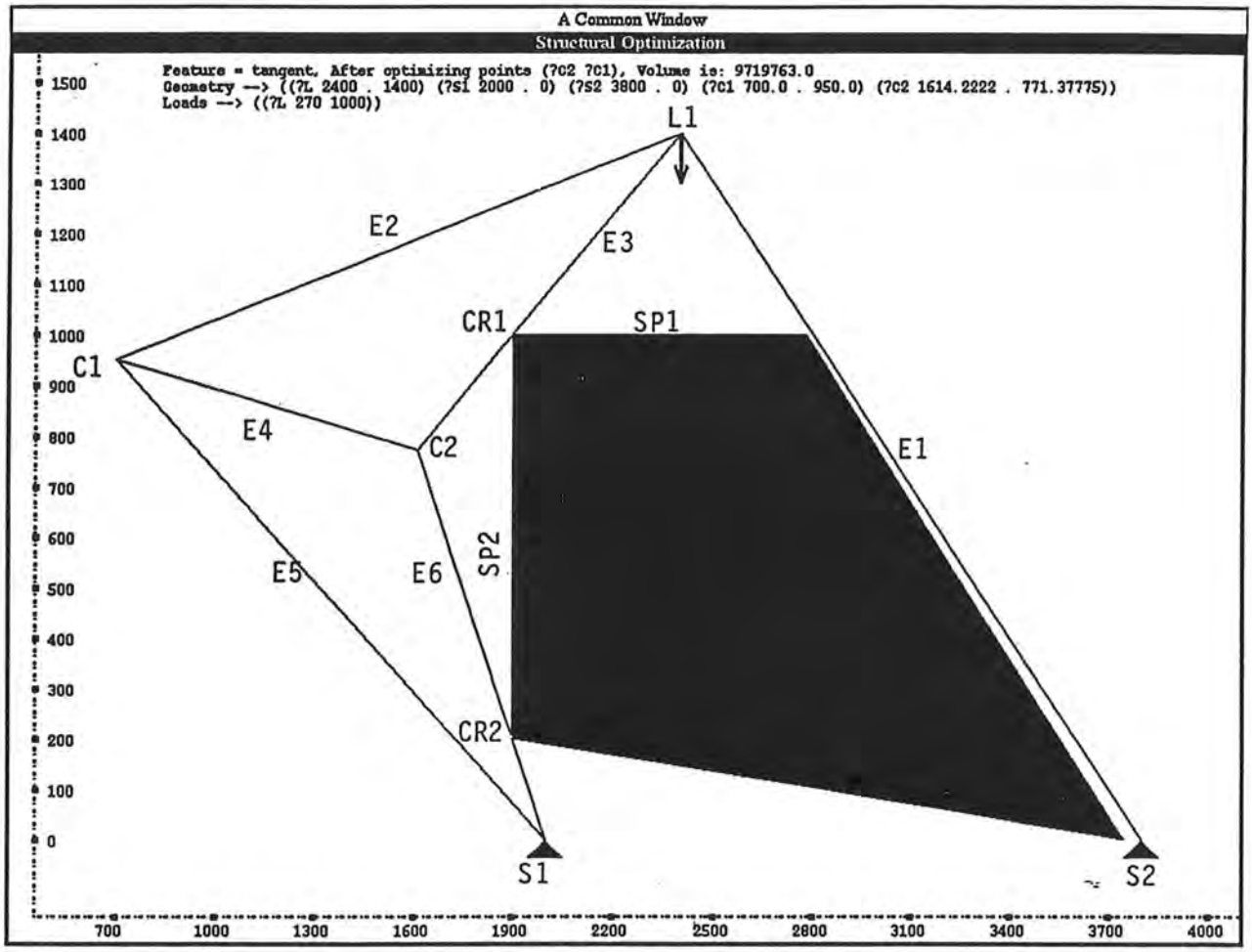

Figure 4: Example in Figure 3 after numerical optimization.

Notice that six g-corners are found. These enable us to easily construct the two lines LNa and LNb (and many other useless lines as well). Lines LNa and LNb intersect at point C2. Hence, the g-corner concept allows us to rapidly detect a new pattern connecting the unknowns to the givens.

From these relationships, the system can define the new concept tangent-intersection:

```
tangent-intersection(Tg,FR,P1,P2)  ⇔  region(FR),
                                      g-corner(CR1,FR), line(LNa,CR1,P1)
                                      g-corner(CR2,FR), line(LNb,CR2,P2)
                                      intersect(LNa,LNb,Tg).
```

When this is combined with the line-of-sight constraints, we obtain

```
tangent-intersection(C2,FR,L1,S1), see(C2,L1,[FR]), see(C2,S1,[FR]).
```

To construct the final solution procedure for this case, we need to gather constraints for C1 as well. It turns out that as in the previous problem, C1 is a bisector-point. Hence, the final procedure looks like:

```
If    T= λ(C1,C2)
        topology([load(L1)],
                 [support(S1),support(S2)],
                 [region(FR)],
                 [point(C1),point(C2)],
                 [edge(E1,L1,S2),edge(E2,L1,C1),
                  edge(E3,L1,C2),edge(E4,C1,C2),
                  edge(E5,C1,S1),edge(E6,C2,S1)]),
        line(Ln,L1,S1)

Then  C2=min(P2,
             [tangent-intersection(P2,FR),see(P2,L1,[FR]),see(P2,S1,[FR])],
             distance(P2,Ln)),
      C1=min(P1,
             [P1 ∈ ℜ², bisector-point(P1,C2,L1,S1),
              see(P1,L1,[FR]),see(P1,C2,[FR]),see(P1,S1,[FR])],
             weight(T(P1,C2))).
```

This example shows how features learned from solving one problem can substantially aid the discovery of new design rules and additional features.

Note that to learn efficient design procedures for large problems, it is still necessary to first solve at least a representative set of problems using numerical optimization methods. Hence, while the new features do permit the rule-construction process to scale, they do not completely solve the scaling problem. This remains an important problem for future work.

7 Concluding Remarks.

The methods outlined in this paper show how an inductive analysis of the results of numerical optimization can produce efficient optimal design rules. Because there is an inductive component to this process, however, these rules are not guaranteed to be correct. Fortunately, it is quite inexpensive to test whether a proposed solution (suggested by the rules) is in fact locally optimal. Hence, errors in the rules can be detected easily, and this should allow the rules to be repaired and refined (by adding conditions). Furthermore, even when a rule proposes a suboptimal design, that design may provide a good starting point for performing numerical optimization. The convergence of non-linear optimizers depends critically on the quality of such starting points.

This paper also shows how new features can provide a vehicle for transferring knowledge from simple problems to more complex ones. The new features substantially reduce the cost of analyzing the results of numerical optimization in larger problems.

Our future work will focus on using the geometric design rules to discover topology selection rules. We believe that with each geometric design rule it will be possible to associate a formula that predicts the approximate weight of the optimal design. These formulas can then be analyzed to determine conditions under which one topology will be lighter than another. This kind of analysis would be too expensive to perform using only numerical optimization techniques. Topological optimization is nearly impossible to perform using current numerical methods, so a solution to this problem will provide an important tool for mechanical designers.

References

[Gas79] J. Gaschnig. A problem-similarity approach to devising heuristics. In *Proceedings of IJCAI-6*, pages 301–307, 1979.

[MP89] J. Mostow and A. E. Prieditis. Discovering admissible heuristics by abstracting and optimizing: a transformational approach. In *Proceedings of IJCAI-11*, pages 701–707, 1989.

[NGa89] Nevill, G.E. Jr., Garcelon, J.H., and al. Automating preliminary mechanical configuration design: The MOSAIC perspective. In *NSF Engineering Design Research Conference*, 1989.

[Pik86] Ralph W. Pike. *Optimization for Engineering Systems.* Van Nostrand, 1986.

[PS70] A.C. Palmer and D.J. Sheppard. Optimizing the shape of pin-jointed structures. In *Proc. of the Institution of Civil Engineers*, pages 363–376, 1970.

[WS84] Chu-Kia Wang and Charles G. Salmon. *Introductory Structural Analysis.* Prentice Hall, New Jersey, 1984.