An Investigation of Opportunistic Constraint Satisfaction In Space Planning

Can A. Baykan and Mark S. Fox

The Robotics Institute
Carnegie Mellon University
Pittsburgh, PA 15213

ABSTRACT

We are investigating constraint directed heuristic search as a means for performing design in the field of space planning. Space planning is selecting, dimensioning, locating and shaping design units to create two dimensional layouts based on functional, topological and geometrical considerations. Search is carried out using operators at different abstraction levels and design objects at different levels of detail. Constraints are used to represent domain knowledge, to define the search space by specifying operators in means ends analysis manner, and to rate the partial candidate solutions using importances associated with each constraint

Search is carried out opportunistically. The philosophy behind opportunism is that understanding the approximate topology of the search space will lead to efficient search. Uncertainty associated with constraints is derived and used to identify islands of certainty in the search space, which are used as starting points and anchors for search. The knowledge that enables us to identify opportunistic decisions are interactions between constraints and the usefulness of a constraint in different situations. The resulting uncertainty measure will be tested by observing the problem solving behavior it causes in different search spaces.

I INTRODUCTION

This is an investigation of constraint directed heuristic search as a means of performing design. The philosophy behind the formulation of design as opportunistic search is the topological assumption that understanding a problem's search space will enable efficient search (Fox 1986). Constraints help in understanding the approximate topology of the search space by allowing us to identify islands of certainty through which the solutions must pass.

Space planning covers the set of problems which humans solve using orthographic drawing. Typical space planning problems are layout of floor plans, arrangement of equipment and furniture in rooms, and site planning. In these problems topological relations such as adjacency, and geometrical properties such as shape, dimension, distance, and other functions of spatial arrangement are a principal concern (Eastman 1973). It is natural to express space planning problems in terms of constraints.

The domain of application we have selected is kitchen design. An appropriately designed kitchen should have well-defined centers for serving, cooking, mixing, and the sink. The centers should be arranged to reflect the natural sequence of use during food preparation. The three distances between the front mid-points of the sink, range, and refrigerator usually form a work-triangle. The sum of these distances should be greater than 12 feet and less than 22 feet. Since most of the time spent in the kitchen is spent at the sink, placing the sink against the window provides light and view while working. This knowledge can be represented by constraints of the form:

- Mix center should be next to and to the left of refrigerator.
- There should be a continuous counter of 36 inches for mixing and food preparation.
- Distance between the front mid-points of sink and range should be 4 to 6 feet.
- There should be no traffic or furniture interfering with the work triangle.

Different abstractions have been used in space planning: discrete locations in the form of a grid of cells, adjacency graphs and networks, rectangular dissections, and drawing based representations using closed polygons. Search methods that have been used range from generate and test, backtracking, branch and bound, hill-climbing, means-ends analysis, to hierarchical planning. Heuristics are applied to selection of design units, sequencing the operators and tests, and if more than one abstraction is used, to the selection of abstractions.

Search architectures that have been used are classified as *sequencing by design unit* and *priority solution method* (Eastman 1973). In sequencing by design unit, a design unit is selected to enter the design, locations are generated tor it and tested against all applicable constraints. Heuristics for selecting the next design unit to enter the design are: select the design unit with the most restrictive set of constraints, select the largest remaining design unit, select the one most strongly connected to those already placed. The most efficient scheme for applying the constraints is in ascending order of cost of test execution divided by probability of failure. Priority solution method involves creating macro-objects in unbounded space by considering a subset of the constraints. This is useful if groups exist in terms of severity or importance of constraints and strength of interaction between design units, or if a constraint proves hard to satisfy otherwise. Experience with space planning programs indicate that time consumption is affected not so much by the number of rooms as by the strength of constraints and by sequencing. The generation process should utilize all constraints to limit combinatorial search. Constraint-directed search attempts to formulate general models for the representation of constraints (Fox 1983). The objectives are to identify and represent the variety of constraints, and interactions between constraints for effective utilization during search. Interactions between constraints such as conflict, competition and cooperation, relaxation of constraints are some of the issues addressed.

WRIGHT is a knowledge based space planning system that uses multiple abstractions opportunistically. Designs are represented at different levels of detail, and in qualitatively different ways using relation graphs and adjacency networks, and design objects at different levels. Constraints are modeled by a goal tree. Search operators are associated with constraints in means ends analysis manner. Satisfying a constraint leads to posting and propagation of other constraints within and across levels. Opportunism is making highly constrained decisions early, and least commitment is postponing those that are not. Opportunism is a resource allocation problem where the constraints are scheduled in order to generate solutions efficiently (Hayes-Roth and Lesser 1977). Uncertainty measures associated with each constraint based on the search state is used to identify islands of certainty in the search space.

II ARCHITECTURE OF WRIGHT

A. Components of WRIGHT

WRIGHT consists of a knowledge base, a problem solver and a user interface. The knowledge base contains knowledge about the application domain i.e. the vocabulary of design objects and desired attributes and relationships between them, and general knowledge about space planning abstractions and search states. Problem solver focuses attention on different aspects of space planning such as locating, dimensioning and shaping the objects of design opportunistically, based on uncertainty measures associated with constraints. Each constraint specifies search operators. The knowledge base can be modified to reflect different styles and preferences, and to design in other domains of space planning.

B. Design Units

Layouts are created by configurations of design units. It is possible to consider design units at different levels of detail. In the case of kitchen design, we deal with work centers of the kitchen and with elements such as sinks, refrigerators and counters. The work centers consist of some combination of applicances, counter area and storage space. The design units at both levels of abstraction form a hierarchy through which there is inheritance of variables, values and constraints. At the top of the hierarchy are conceptual categories such as spaces, solids, architectural elements, and boundary elements. At the lower level of abstraction, there are categories such as sink, refrigerator, and counter. These are refined by types of sinks such as single bowl sink, double bowl sink etc., or different types of refrigerators. The leaves of the hierarchy may contain specific models selected from manufacturers catalogues.

The knowledge base, consisting of design units and constraints, design abstractions and search states are represented by schemata using Knowledge Craft.* A schema is the unit for representing objects, relations and concepts. The sink schema inherits all of the variables from architectural-du, which is above it in the hierarchy. It inherits length and width slots and constraints that may have been specified, i.e., it may not overlap another solid. It has slots specific to itself, i.e. number-of-bowls and number-of-drainboards. Below sink are the types of sinks: single bowl, double bowl and triple bowl sinks, and single and double drainboard sinks. For refrigerators, the types are: conventional refrigerators and built in refrigerators which do not require ventilation space at the back and are the same width as standard counters. Below different types of sinks are specific sinks from manufacturers catalogues. Once a specific sink is selected, only its location needs to be specified. Its length, width, shape, number of bowls, and number of drainboards are either inherited or specified for itself. But in accordance with least commitment principle, this selection is not made until constraints on the attributes of sink help to make this decision with high certainty.

*Knowledge Craft is a trademark of Carncgic Group Inc.

C. Problem Solving Abstractions

Design abstractions are useful for considering some aspects of designs and to defer dealing with other aspects, limiting combinatorial explosion. A design in a higher level abstraction represents an equivalence class of designs in a lower level abstraction.

For interactive design and the user interface, design units are represented by their shapes such as rectangle or L-shape, dimensions and x, y locations, and orientations. During search, adjacency networks and spatial relations are used.

1. Region Line Adjacency Network

The elements used in region line adjacency network abstraction are horizontal and vertical lines and regions. Each line is unique, and extends from *-infinity* to *+infinity.* Incidence relations between a region and the lines which define it are: north-line, south-line, east-line and west-line. Their inverses define relations from lines to regions. There are also relations between two lines of the same type. Relations between vertical lines are: line-east-of and line-west-of. Relations between horizontal lines are: line-north-of and line-south-of. These are transitive relations, in terms of relations between lines and regions and lines and lines.

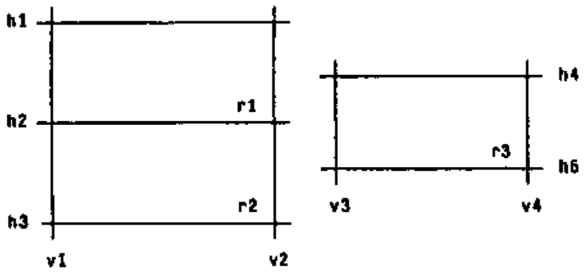

Figure 1: Example configuration of regions and lines

```
{{ r1
      INSTANCE: region
      NORTH-LINE: h1
      SOUTH-LINE: h2
      WEST-LINE: v1
      EAST-LINE: v2 }}
{{ h2
      INSTANCE: horizontal-line
      NORTH-REGIONS: r1
      SOUTH-REGIONS: r2
      LINE-NORTH-OF: h5
      LINE-SOUTH-OF: h4}}
```

Figure 2: Representation of region r1 and line h2

Figure 1 shows a configuration of regions and lines, and figure 2 shows how it can be represented in the region line adjacency network using the relations described above.

Networks are also used for determining the dimensions of regions and locations of lines. These values are represented as intervals with minimum and maximum allowable values. Initially an interval for the location of a line such as the x coordinate of a vertical line, extends from *-infinity* to *+infinity,* an interval for the x or y dimension of a region extends from *0* to *+infinity.* As design progresses minimums may only increase and maximums may only decrease, until the two are equal.

The operators for creating the topology and determining the dimensions of a network are:

- Create new region.
- Merge two lines.
- Put a line south or west of another line.
- Post minimum/maximum values for an interval.

These operators propagate new bounds to related intervals. A new relation can be satisfied if it does not contradict existing relations and dimensions. An operation can be contradictory, redundant or constraining. When a contradiction is detected, propagation stops with failure. In case of a redundant value or relation, the operation succeeds and stops. A constraining operator leads to further propagation of effects.

One dimensional relations between regions in the horizontal or vertical direction are defined in terms of relations between lines and regions, and lines and lines. These are similar to Allen's temporal relations (Allen 1983). Horizontal relations between regions are below, and their corresponding vertical relations are in parenthesis. The inverse of a relation is also defined if it is not symmetric, but not given below.

- *region-west-of (region-south-of)*
- *region-west-acjjacent (region-south-adjacent)*
- *horizontally-inside (vertically-inside)*
- *horizontally-overlapping (vertically-overlapping)*

A horizontally-inside B is defined as west line of A being cast of west line of B, and east line of A being west of east line of B. *A south-adjacent B* is defined as region A sharing its south line with the north line of region B.

In this representation dimensions and topology are separate yet related. Decisions can be made in any order. Every state created using the operators defined above is an approximative abstraction that defines an equivalence class of solutions in terms of the decisions made up to that point. A constraint that has been satisfied will not be violated later using these operators. All spatial relations can be explicitly represented,-which makes it possible to design the layout of work centers in a kitchen at one level of detail and to design the layout of the applicances in the work center at another level. It is possible to generate packed layouts using the merge lines operation, or loosely packed layouts putting lines relative to each other.

2. Spatial Relations

Spatial relations are defined in terms of the one-dimensional relations between regions in the adjacency network. Types of relations considered are: adjacency relations, spatial-overlap relations, location relations, position relations, orientation relations and distance relations. Adjacency relations are: next-to, completely-next-to and covers. Next-to is defined as the regions being region-west(south)-adjacent in one direction and vertically(hori2ontally)-overlapping in the other. Spatial-overlap relations are inside, has-inside, overlap and non-overlap. Location relations are north-of, south-of, east-of and west-of. Position relations are at-front, at-back, at-left and at-right Location relations are defined with respect to the global coordinates and position relations are defined with respect to object centered coordinates of the design units. Most design units have fronts, backs and sides which have to be treated differently in a layout. There are orientation relations: parallel-to, perpendicular-to, opposite, and distance relations.

Existing relations between design units are used in inferring new relations, and determining which new relations indicated by constraints arc already satisfied and which are contradicted. Some relations are mutually exclusive i.e.,north-of and south-of. Given a relation between two design units, relations that arc mutually exclusive arc contradicted i.e., *A north-of B* contradicts *A south-of B.* Some relations satisfy others, such as inside satisfies overlap, and next-to satisfies non-overlap. This type of reasoning uses the definitions of the relations. Another way of reasoning uses the transitivities of the relations. Some relations are transitive, so that: if *A inside* £, *B inside C* and *C north-of D* then *A north-of D.* Transitivity of each relation is expressed by a path grammar and used in reasoning via relations. A third way of reasoning involves making inferences. Two relations are used to infer a third: if A *at-front B* and *orientation of A is 90* then *A west-ofB.*

Relations level enables reasoning while more complete but combinatorialy expensive reasoning at the adjacency network level is deferred. Domain constraints expressed in object-centered coordinates are converted to design coordinates as successive design decisions enable inferences. Some relations can be shown to be satisfied or contradicted due to interactions between relations.

D. <u>Constraints</u>

Constraints represent design goals and control knowledge. The categories of knowledge that need to be known in order to use constraints as a representation of knowledge and as a search technique are (Fox 1983):

- How to generate states satisfying a constraint.
- How to test if a constraint is satisfied
- Possible relaxations of a constraint.
- Relative importance of a constraint.
- Uncertainty of a constraint.

Constraints encoding knowledge of the design domain are posted to prototype design units. Design unit instances inherit constraints from their prototypes and the prototypes above it in the hierarchy. The constraint values inherited from different prototypes for a constraint type can be added on, i.e. sinkl may inherit constraint values from solid, architcctural-du, and sink for the same variable and constraint type, or a value placed at sink may replace those specified above it

Domain constraints which specify a relation between two prototypes such as: sink should be next to wall, is inherited at sinkl and refined into: sinkl should be next-to walll or wall2. How the instances walll and wall2 should be combined is specified at each constraint or inherited from the type of constraint, for example non-overlap constraints are combined with and. Similar relations are posted between constraints to indicate that the actual instances selected should be the same, i.e., if sink is placed next-to walll, it should also be parallel-to walll, and not to another wall instance.

Constraints at the network level express an instantiated constraint in terms of one-dimensional relations between regions. For example, sinkl inside sink-center 1 is refined into: sinkl is horizontally inside sink-centerl and sinkl is vertically inside sink-centerl. The goal tree is an and/or tree of constraints starting with domain constraints between prototype design units and ending with one dimensional relations between instances.

E. <u>Problem Solver</u>

A space which serves as the envelope and its adjacent spaces are the minimum givens that make up the initial state. Any features that should be fixed in the design and some information about the design context such as house area, household population, and building type should also be given. First stage of problem solving is pre-search analysis.

Based on givens, additional constraints are posted by rules of the form: If kitchen area is greater than or equal to *15%* of the house area, then the kitchen should be an eat-in kitchen with a table and adequate seating. Design unit types are selected: there should be a sink, a refrigerator, a cooker etc. The goal tree is set up using the instances selected to be in the design and those present in the initial state. The interactions between constraints that can be determined are found. A geometric modeling program finds the existing relations between design units, and creates a region line adjacency network representation of the initial state.

The second stage in problem solving is opportunistic search. Search control operates by selecting a state and a constraint to generate new states. Each search step consists of the following operations:

- select a state
- select a constraint
- generate new states
- update constraints
- update states

Constraints are selected based on their uncertainties. Identifying interactions between constraints is useful to identify opportunistic decisions. Constraints are said to be independent if achieving one goal has no effect on the second, cooperating if achieving one goal makes it easier to achieve the second, competing if one goal can only be achieved at the expense of the other, and conflicting if achieving one goal must take the second into account due to an interaction (Mostow 1985).

Other information for determining the uncertainties of constraints are: importance of a constraint, severity of a constraint the type and state of the variable(s) constrained, the size of design unit(s) affected by the constraint. Uncertainty is used as a measure for rating the opportunism of constraints, and determining where to focus attention during search.

Opportunism is focusing the attention of the problem solver on the most constrained operation that can be performed at each step instead of taking a pre-determined strategy for control such as top down, goal driven, or data driven (Stefik 1980). A top down approach locates work centers first and uses this to guide the location of applicances. In an opportunistic approach, if the location of an applicance is more constrained than the location of a work center, then the applicance is placed first and this guides decisions at higher levels regarding the location of work centers. Islands of certainty through which the solutions must pass can be identified by understanding the approximate topology of the search space [4] Search proceeds by using the islands as starting points and expands them.

Since design involves relaxing and adding constraints as well as searching for a solution defined by a set of constraints, we find a representative sampling of significantly different solutions. Significance of differences are based on which constraints are satisfied and the values used to satisfy them.

III TEST RESULTS

Version 0 of WRIGHT uses reasoning at the spatial relations and user interface abstractions, but not at the adjacency network level. Selection of levels is fixed. A variable is selected based on the uncertainties of its constraints. Then all the constraints of that variable are applied in order of certainty to generate all valid ranges of values for that variable. Each constraint is applied first at the relational level. If there is no conflict then the constraint is used to generate a range of values. After all the constraints of the variable are applied, then a discrete set of values are selected from the range of the best state to continue the search with other variables.

Version 1 of WRIGHT about to be completed uses reasoning at all the levels described. Levels are searched opportunistically and all constraints of a variable do not have to be considered together.

WRIGHT has been tried on different test cases. This approach works better when the solutions are tightly constrained. Research on the types of knowledge that enable us to identify opportunistic decisions is continuing.

IV CONCLUSIONS

WRIGHT version 0 results indicate that starting search with more strongly constrained design units efficiently eliminates many dead ends and leads search towards the better solutions. The ability to defer uncertain aspects of a decision while being able to reason at a higher abstraction level is needed to use islands of certainty as guidance for search. WRIGHT version 1 is formulated to achieve this capability.

One of the limitations of WRIGHT is the types of constraints that can be considered. The goal tree should start with general performance issues such as daylighting, spatial order and energy efficiency and derive specific constraints from these. This level of the goal tree is omitted in this research, to concentrate on using space planning constraints opportunistically. We attempt to generate a sampling of good solutions so that the designer can compare in the light of other constraints.

REFERENCES

1. Allen J.F. "Maintaining knowledge about temporal intervals". *Communications of the ACM 26,*11 (November 1983), 832-843.

2. Eastman C M. "Automated space planning". *AI4* (1973), 41-64.

3. FoxM.S. Constraint-directed search: A case study of job-shop scheduling. Tech. RepL CMU-RI-TR-83-22, CMU-CS-83-161, Computer Science Department, Carnegie-Mellon University, December, 1983.

4. Fox M.S. Observations on the role of constraints in problem solving. Proceedings Sixth Canadian Conference on Artificial Intelligence, May, 1986, pp. 172-187.

5. Hayes-Roth F. and Lesser V.R. Focus of attention in the HEARSAY-U speech understanding system. Proc UCAI 5th, 1977, pp. 27-35.

6. Mostow J. "Toward better models of the design process". *AIMagazine* 6,1 (1985), 44-57.

7. Stefik M.J. *Planning with Constraints.* Ph.D.Th., Stanford University, January 1980.