Problem Topology, Texture and Objectives

Mark S. Fox

Center for Integrated Manufacturing Decision Systems
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213

DRAFT October 8, 1990

Abstract

We propose a model of problem solving that provides both structure and focus to search. The model achieves this by combining constraint satisfaction with heuristic search. We introduce the concepts of topology and texture to characterize problem structure and areas to focus attention respectively. The resulting model reduces search complexity and provides a more principled explanation of the nature and power of heuristics in problem solving.

This research has been supported, in part, by the Defense Advance Projects Agency under contract #F30602-88-C-0001, and in part, by grants from McDonnell Aircraft Company, Boeing Computer Services, and Schlumberger Corp.

1. Introduction

We propose a model of problem solving that provides both structure and focus to search in the problem space. The model achieves this by combining the process of constraint satisfaction (CSP) with heuristic search (HS). The resulting model both reduces search complexity and provides a explanation of the nature and power of heuristics in problem solving.

Our problem solving model, called *Constrained Heuristic Search* (CHS), retains heuristic search's synthetic capabilities and extends it by adding the structural characteristics of constraint satisfaction techniques. In particular, our model adds to the definition of a problem space, composed of states, operators and an evaluation function, by refining a state to include:

1. **Problem Topology**: Provides a structural characterization of a problem.
2. **Problem Textures**: Provide measures of a problem topology that allows search to be focused in a way that reduces backtracking.
3. **Problem Objective**: Defines a means for rating alternative solutions.

This model allows us to (1) view problem solving as constraint satisfaction, thus taking advantage of these techniques, (2), incorporate the synthetic capabilities of heuristic search, thus allowing the dynamic modification of the constraint model, and (3) extend constraint satisfaction to the larger class of optimization problems. In the following, we define the scheduling problem to be used as an example throughout the paper, followed by a definition of problem topology, textures, objectives and the CHS search process.

2. Factory Scheduling Example

[1] Factory scheduling involves the assignment of start times and resources to a set of activities. Each activity belongs to an order (i.e. job). Activities within the same order are subject to precedence constraints as specified by a process plan. Additionally no two activities are allowed to use the same resource at the same time (we assume resources of unary capacity). Each order has a release date and a latest acceptable completion date (which may be later than the due date), that can be used to determine an earliest start time and a latest start time for each activity in the order. Additionally each activity may require one or several resources, for each of which there may be several alternatives. For each activity, utility functions map each possible start time and each possible resource alternative onto a utility value (preference). The sum of these utilities over all the activities to be scheduled defines an objective function to be maximized. These utilities [Fox 87, Sadeh & Fox 88] arise from organizational goals such as reducing order tardiness, reducing order flowtime, using accurate machines, performing some activities during a specific shift, etc.

3. Problem Topology

It is our conjecture that an understanding of the structure of a problem will lead to more effective problem solving methods. Therefore, our goal is to formalize the concept of problem structure.

In an attempt to distinguish the problems to which AI was being applied, from problems which conventional algorithms were being applied, the notion of "well-structuredness" arose. Simon

[1] Excerpted from [Fox et al. 89].

defines a well-structured problem as follows [Simon 73]:

1. There is a definite criterion for testing any proposed solution, and a mechanizable process for applying the criterion.
2. There is at least one problem space in which can be represented the initial problem state, the goal state, and all other states that may be reached, or considered, in the course of attempting a solution to the problem.
3. Attainable state changes (legal moves) can be represented in a problem space, as transitions from given states to the states directly attainable from them. But considerable moves, whether legal or not, can also be represented -- that is, all transitions from one considerable state to another.
4. Any knowledge that the problem solver can acquire about the problem can be represented in one or more problem spaces.
5. If the actual problem involves acting upon the external world, then the definition of state changes and of the effects upon the state of applying any operator reflect with complete accuracy in one or more problem spaces the laws (laws of nature) that govern the external world.
6. All of these conditions hold in the strong sense that the basic processes postulated require only practicable amounts of computation, and the
7. information postulated is effectively available to the processes -- i.e., available with the help of only practicable amounts of search.

Most of the requirements focus on the feasibility of operationalizing, in a computational sense, the means for solving the problem, using the problem space model. But the sixth requirement focuses on the "strength" of the method. Newell defines a "weak" method as one that makes weak information demands and gives weak results, such as generate and test and hill climbing [Newell 73]. Using the big switch approach to problem solving where alternative methods exists for solving a class of problems, weak methods are the methods of last resort - that is, when other "stronger" methods fail we can use weak methods. Simon's sixth requirement reiterates the essence of Newell's definition of an ill-structured problem [Newell 69]: A problem is ill structured if there only exists weak methods to solve it; problem solving performance is the key concern.

Other definitions of a problem's structure have been proposed. Eastman defines a problem as being ill-structured if problem formulation proceeds concurrently with its solution, as typically found in design problems [Eastman 69]. Reitman defines a problem as being ill-structured when both the problem and goals are well defined by the method of solving it are not [Reitman 65]. None of these definitions provide insight into *how* to solve a problem more effectively.

During the last fifteen years, research in AI has begun to exploit various structural characteristics of problems. The success of expert systems relies upon the recognition and exploitation of recurring patterns in a problem [Waterman & Hayes-Roth 78]. More recently, the SOAR model [Laird et al. 87] and its interpretation in the RIME methodology [van de brug et al. 86] provides a structure for rules themselves, dividing them into rules for proposing, prioritizing, selecting and implementing operators.

Within the heuristic search model, a variety of techniques for structuring problems have been investigated. ABSTRIPS [Sacerdoti 74] demonstrated how hierarchical reformulation of the problem via the ranking and omission of variables reduces search complexity. Hearsay-II utilized data aggregation to reduce search complexity in a domain with high data uncertainty [Erman et al. 80].

MOLGEN utilized operator aggregation to restrict the set of operators used to construct detailed plans [Stefik 81]. ISIS demonstrated how hierarchical reformulation via omission of constraints reduces search complexity [Fox 87]. In each case, the systems provide an example of how the structuring of a problem can occur, but they do not provide a theory of structuring. What should be the basis of a theory of problem structure?

Problem structure has long been an interest in Operations Research. In particular, mathematical programming views problem solving as finding a solution to a set of ***constraints*** that optimizes an objective function. The success of linear programming stems from the recognition that a problem can be defined by a set of linear constraints that define a subspace in an n-dimensional euclidean space, and that the optimal solution, as defined by an objective function can be found at one of the vertices of delineated subspace. The same is true of nonlinear programming where the convexity of the solution space implies that hill climbing can be used to find an optimal solution.

We have seen the constraint perspective of problem solving reappear in the last fifteen years, within AI, in form of Constraint Satisfaction Problems (CSP). Constraint satisfaction techniques, as described in [Mackworth 77, Haralick & Elliott 80, Freuder 82, Dechter & Pearl 87], approach problem solving by constructing a constraint graph where nodes are variables with discrete domains and arcs are n-ary constraints. Problem solving is performed by sequentially choosing a variable and a value to assign to it that satisfies all constraints incident upon it. Backtracking occurs when an assignment cannot be found. Research has gone into methods for structuring the network so that the amount of backtracking can be reduced. Arc-consistency is one such technique that achieves local consistency between groups of variables via the elimination of incompatible values [Montanari 74, Mackworth 77, Davis 87].

The generality of the CSP model of problem solving, when extended to include continuous variables such as time and space, has been demonstrated across problems such as spatial planning [Baykan & Fox 87], scheduling [Dincbas et al. 88, Elleby et al. 88, Sadeh & Fox 88], diagnosis [Davis & Hamscher 88] and truth maintenance.

We can now return to the question posed earlier: What should be the basis of a theory of problem structure? The experience of both OR and CSP provides evidence that many problems can be adequately modeled by a constraint graph, and that the structure of these graphs can have significant impact on how to solve a problem. Therefore we adopt the view that a problem's structure is defined by its constraint graph.

We define ***problem topology*** as a graph G, composed of variables V and constraints C. Each variable may be a vector of variables whose domains may be finite/infinite and continuous/discrete. Constraints are n-ary predicates over variables. We distinguish between two types of problem topologies:

> **Definition 1**: A *completely structured problem* is one in which all non-redundant vertices and edges are known a priori.

This is true of all CSP formulations.

> **Definition 2**: A *partially structured problem* is one in which not all non-redundant vertices and edges are known prior to problem solving.

This definition tends to be true of problems in which synthesis is performed resulting in new

variables and constraints (e.g. the generation of new subgoals during the planning process).

We view the scheduling problem as an optimization version of the CHS model, where each activity is an aggregate variable whose values are reservations. A reservation consists of a start time and a set of resources to be allocated to the activity. Each activity constitutes a variable vertex in the problem topology. Activity precedence constraints are binary constraints represented by constraint vertices connected to two activity variable vertices. A capacity constraint vertex is associated to each physical resource of the domain and connected to all the variable vertices representing activities that can possibly use the resource. Each capacity constraint ensures that the corresponding resource will not be allocated to more than one activity at any given time. Accordingly we distinguish between two types of constraint interactions:

- the *intra-order* interactions defined by the precedence constraint vertices between activities belonging to a same order, and
- the *inter-order* interactions induced by the capacity constraint vertices between activities contending for a same resource.

Both types of interactions contribute to the contention of each activity.

Features of the problem topology are the types of variables and constraints (and their associated propagation algorithms). Davis [Davis 87] mentions two classes of what we view as topological features, namely the types of values the domain of a variable may contain, such as variables whose domains are discrete and finite (label and value inference), are intervals, have belief for each member (relaxation labeling), and are expressions (expression inference). The second class of features focus on the types of constraints, such as constraints that are unary predicates, order relations, bounded differences (e.g. $x - y \geq c$), linear equations with unit (i.e. -1, 0, 1) coefficients, linear equalities and inequalities with arbitrary coefficients, boolean combinations of constraints, algebraic equations, and transcendental equations. Additionally, domains may or may not have preferences for values (e.g. preferences for due dates of a job).

Informal notions of problem structure can be formalized by a problem's topology. Where and how to search can be guided by structure. For example, problem decomposition can be viewed as a decomposition of a topology into sub-graphs. Means for determining a decomposition may vary according to a problem's constraints and objectives [Alexander 65, Alexander 68, Courtois 77]. Situations in which search is efficient, such as backtrack free search in width 1 graphs, can also be identified [Freuder 82].

Problem situations can be identified by patterns in the topology. For example, difficult constraint situations can be identified by "knots" in the topology [Krishnan et al. 90].

Problems can be simplified by topology manipulation. Problem reformulation, by means of relaxation or abstraction can be explicitly defined in terms of topology. Relaxation is a process by which constraints are relaxed to admit more solutions. Abstraction can be defined in terms of variable or constraint aggregation or omission.

Another type of reformulation of the problem topology is the creation of a "contention graph" [Fox & Sadeh 90]. Consider the factory scheduling problem where many operations are contending for a small set of machines. The allocation of these machines over time must be optimized. This is equivalent to having a set of variables, with small discrete domains, each competing for the

assignment of the same value but linked by a disequality constraint. A contention graph replaces disequality constraints by a node for each value under contention, and links these "value nodes" to the variables contending for it by a demand constraint. We can now use these value nodes to measure the amount of contention for the value by variables. Contention is an important metric for variable and value ordering [Fox et al. 89].

What value do we derive from viewing a problem space state as a constraint graph? First, we have provided a more refined definition of a problem space state thereby reducing the looseness of its definition and allowing the definition of general measures of problem structure, i.e., textures. Second, properties can be proved about the nature of the problem, e.g., width-1 constraint networks that are arc consistent are backtrack free. Third, the process of problem reformulation can be viewed as transformations of problem topological primitives. A possible negative, is that the number of problem types that can be represented in the form of a constraint graph is limited. But this set is growing larger; in the factory scheduling example, we have shown how the representation can be extended to handle optimization [Sadeh & Fox 90]. By adding the power of heuristic search, we believe that we can apply the model to a broader class of problems.

4. Problem Textures

Evaluation functions play a prominent role in search; whether to identify the node to expand next in best first search, or to recognize an island to extend in opportunistic search. The problem is that most evaluation functions are "hand molded" for each problem, thereby limiting their reusability. The question is whether there are measurements of the problem topology that give rise to powerful heuristics and are problem invariant?

In CHS, for search to be well focused, there must be measures of the topology that differentiate one subgraph from another, and these measures must be related to the goals of the problem. We have identified and are experimenting with six such measures that we call problem *textures* [Sadeh & Fox 88, Fox et al. 89]:

- **Value Contention**: Degree to which variables are contending for the same value.

  In the factory scheduling domain, measuring the amount of contention there exists for resources is important in identifying bottlenecks. It is the activities associated with the bottleneck resource that are assigned first [Fox & Sadeh 90].

- **Value Conflict**: Degree to which a variable's assigned value is in conflict with existing constraints.

  In repairing schedules, focusing on activities that participate in the greatest number of conflicts reduces search [Minton et al. 90].

- **Value Reliance**: Degree to which a variable relies upon the availability of a particular value.

  Once contention is used to identify a resource and the activities contending for it, selecting an activity depends upon the degree to which the activity relies upon having the resource.

- **Value Goodness**: Probability that the value leads to the best solution.

  Once a resource is selected for an activity, deciding when the resource should be assigned for use depends upon how disruptive the assignment would be to subsequent assignments [Sadeh & Fox 90].

- **Constraint Tightness**: Degree to which a constraint reduces the number of solutions.

  If the scheduling problem is large and too complex to solve directly, then solving a simpler version may provide guidance in solving the original. Reformulating the problem can be accomplished by omitting "loose" constraints.

- **Variable Tightness**: Degree to which a solution to the problem is constrained by a particular variable.

  Variable tightness can be used to decide which variables to aggregate in a reformulation.

These textures generalize the notion of constraint satisfiability or looseness defined by [Nadel 86] and apply to both CHSs (and CSPs) with discrete and continuous variables. There exist many ways in which to perform these measurements. Textures may sometimes be evaluated analytically [Sadeh & Fox 88]. Such techniques may however be very costly. In general, for a given CHS, some textures are easier to approximate than others, and some are also more useful than others. Usually the texture measures that contain the most information are also the ones that are the most difficult to evaluate. Hence there is a tradeoff. Each domain may have its own approximation for a texture measure.

5. Problem Objectives

Many problems, such as design and scheduling, require the optimization of one or more objectives in addition to the satisfaction of a set of constraints. In linear programming, an objective function is used to evaluate each vertex visited, and in heuristic search an evaluation function is used to rate each state. In either case, objectives tend to be combinations of more primitive statistics. For example, in scheduling, the objective is a weighted combination of tardiness and flowtime. It has long been understood, that by moving more of the evaluation knowledge into the state generator, a more effective search can be performed. An analogous situation exists in CHS.

With CHS, we can view each objective as being a constraint with associated utilities. For example, a due date in the scheduling domain is a temporal constraint that specifies a set of acceptable dates for the completion of an activity. We can extend the representation to include preferences in the form of utilities for each due date [Fox 87]. This constraint, which is represented directly in the problem topology, represents a local preference. When selecting a variable and assigning a value to it, optimizing any local constraints is straight forward. But in order to optimize our decision making, each local decision in the constraint graph should be globally optimal. This can be achieved by constraint propagation. In particular, temporal preferences can be propagated such that local preferences can be transformed to represent preferences more globally optimal [Sadeh & Fox 88]. The power of representing a problem as a constraint graph enables the determination, by propagation, of how individual decisions will impact each via constraints.

6. CHS Problem Solving Process

The CHS model of problem solving is a combination of constraint satisfaction and heuristic search. Search is performed in the problem space where each state is defined by a problem topology. For most problems, the problem topology is only partially complete. Therefore, a goal of the search process is to acquire additional topology or modify existing topology. Optimization of the decision process occurs naturally as constraint propagation transform local preferences into more global

preferences.

The problem solving model we propose contains the following elements:

- An initial state is defined composed of a problem topology,
- Constraint propagation is performed within the state,
- Texture measures are evaluated for the state's topology,
- Operators are matched against the state's topology, and
- A variable node/operator pair is selected and the operator is applied.

The application of an operator results in either adding structure to the topology, further restricting the domain of a variable, or reformulating the problem (e.g., relaxation).

In our scheduling domain example[2]: Search begins with a single state where all activities still have to be scheduled and all resources are available. Scheduling an activity in a state with a reservation results in the creation of a new search state where new constraints resulting from the assignment of the reservation to to the activity are propagated. The propagation consists in updating the domain of start times and resources that remain possible for each unscheduled activity [Sadeh & Fox 88]. If an inconsistency is detected the system backtracks. Next the scheduler computes a contention/reliance measure for each unscheduled activity. The activity with the highest contention/reliance is selected to be scheduled next. A value goodness measure is computed to select the first reservation to be tried for that activity (among the reservations that are still possible). The process goes on until all activities have been scheduled or until all search states have been visited.

7. Conclusion

The creation of general models for problem solving has been of continuing interest to Artificial Intelligence researchers. The process is evolutionary, elaborating and/or creating new search methods and richer representations of knowledge. The SOAR architecture, for example, combines both the problem space and production system models and extends them with universal subgoaling and chunking, thus achieving a model with powerful learning capabilities. But within this model, there are two aspects of the problem space that remain ill-defined: the notion of structure and means of focusing attention within a structure. Our model, Constrained Heuristic Search, extends the problem space model in these directions. Problem topology provides a definition of structure in the form of a constraint graph. Problem textures provide a graph theoretic definition of the complexity and importance of decisions within a topology. Problem objectives define an objective function that after constraint propagation provide indicate the global optimality of a local decision. Together they enable the problem solver to direct search more economically towards a higher quality solution.

The model has been demonstrated in four domains: spatial planning [Baykan & Fox 90], factory scheduling [Fox & Sycara 90, Sadeh & Fox 90], transportation planning [Sathi et al. 90a], and resource configuration [Sathi et al. 90b]. In spatial planning, we demonstrated that CHS is more efficient in finding solutions than other comparable systems. In factory scheduling, we generalized constraint graphs to account for preferential temporal constraints, making it possible to represent the general job shop scheduling problem for the first time. Texture measures, based upon these

[2]Excerpt from [Fox et al. 89]

preferences, enabled the scheduler to opportunistically select the next best decision to make. They also provided an explanation of the power of domain heuristics like bottleneck analysis.

References

[Alexander 65] C. Alexander.
Notes on the Synthesis of Form.
Harvard University Press, Cambridge MA, 1965.

[Alexander 68] C. Alexander.
A City Is Not a Tree.
Ekistics 139:344-348, 1968.

[Baykan & Fox 87]
Baykan, C., and Fox, M.S.
An Investigation of Opportunistic Constraint Satisfaction in Space Planning.
In *Proceedings of the Tenth International Joint Conference on Artificial Intelligence*, pages 1035-1038. 95 First St., Los Altos, CA 94022, 1987.

[Baykan & Fox 90]
Baykan, C.A., and Fox, M.S.
Constraint Satisfaction Techniques for Spatial Planning.
Intelligent CAD Systems 3: Practical Experience and Evaluation.
Springer-Verlag, 1990.

[Courtois 77] Courtois.
Decomposability.
Academic Press, 1977.

[Davis 87] Davis, E.
Constraint Propagation with Interval Labels.
Artificial Intelligence 32:281-331, 1987.

[Davis & Hamscher 88]
Davis, R., and Hamscher, W.
Model-based Reasoning: Troubleshooting.
Exploring Artificial Intelligence: Survey Talks from the National Conferences on Artificial Intelligence.
Morgan Kaufmann Pub. Inc., 1988, pages 297-346.

[Dechter & Pearl 87]
Dechter, R. and Pearl, J.
Network-based Heuristics for Constraint-Satisfaction Problems.
Artificial Intelligence 34(1):1-38, 1987.

[Dincbas et al. 88]
Dincbas, Simonis, and Van Hentenryck.
Solving the Car-Sequencing Problem in CLP.
In *Proceedings of the 1988 European Conference on Artificial Intelligence.* 1988.

[Eastman 69] Eastman, C.
Cognitive Processes and Ill-defined Problems.
In *Proceedings of the International Joint Conference on Artificial Intelligence.* 1969.

[Elleby et al. 88] Elleby, P., Fargher, H.E., and Addis, T.R.
Reactive Constraint-Based Job-Shop Scheduling.
Expert Systems and Intelligent Manufacturing.
Elesevier Science Publishing Co., 1988, pages 1-10.

[Erman et al. 80] Erman, L.D., Hayes-Roth, F., Lesser, V.R., and Reddy, D.R.
The Hearsay-II Speech Understanding System: Integrating Knowledge to Resolve Uncertainty.
ACM Computing Surveys 12(2):213-253, 1980.

[Fox 87] Fox, M.S.
Constraint-Directed Search: A Case Study of Job-Shop Scheduling.
Morgan Kaufmann Publishers, Inc., 1987.

[Fox & Sadeh 90] Fox, M.S., and Sadeh, N.
Why Scheduling is Difficult: A CSP Perspective.
In *Proceedings of the European Conference on Artificial Intelligence*, pages 754-767. 1990.

[Fox & Sycara 90] Fox, M.S. and Sycara, K.
Overview of CORTES: A Constraint Based Approach to Production Planning, Scheduling and Control.
In *Proceedings of the Fourth International Conference on Expert Systems in Production and Operations Management*. May, 1990.

[Fox et al. 89] Fox, M.S., Sadeh, N., and Baykan, C.
Constrained Heuristic Search.
In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 309-316. Morgan Kaufmann Pub. Inc., 1989.

[Freuder 82] Freuder, E.C.
A Sufficient Condition for Backtrack-free Search.
Journal of the ACM 29(1):24-32, 1982.

[Haralick & Elliott 80]
Haralick, R.M., and Elliott, G.L.
Increasing Tree Search Efficiency for Constraint Satisfaction Problems.
Artificial Intelligence 14(3):263-313, 1980.

[Krishnan et al. 90]
Krishnan, V., Navinchandra, D., Rane, P., and Rinderle, J.R.
Constraint Reasoning and Planning in Concurrent Design.
Technical Report, Robotics Institute, Carnegie Mellon University, 1990.
CMU-RI-TR-90-03.

[Laird et al. 87] Laird, J.E., Newell, A., and Rosenbloom, P.S.,
SOAR: An Architecture for General Intelligence.
Artificial Intelligence 33(1):1-64, September, 1987.

[Mackworth 77] Mackworth, A.K.
Consistency in Networks of Relations.
Artificial Intelligence 8(1):99-118, 1977.

[Minton et al. 90] Minton, S., Johnston, M.D., Philips, A.B., and Laird, P.
Solving Large-Scale Constraint Satisfaction and Scheduling Problems Using a Heuristic Repair Method.
In *Proceedings of the Eighth National Conference on Artificial Intelligence*, pages 17-24. MIT Press, 1990.

[Montanari 74] Montanari, U.
Networks of Constraints.
Proc. IFIP Congress :727-732, 1974.

[Nadel 86] Nadel, B.A.
The General Consistent Labeling (or Constraint Satisfaction) Problem.
Technical Report DCS-TR-170, Department of Computer Science, Laboratory for Computer Research, Rutgers University, New Brunswick, NJ 08903, 1986.

[Newell 69] Newell, A.
Heuristic Programming: Ill-Structured Problems.
Progress in Operations Research :360-414, 1969.

[Newell 73] Newell, A.
Artificial Intelligence and the Concept of Mind.
Computer Models of Thought and Language.
W.H. Freeman and Co., 1973.

[Reitman 65] Reitman, W.
Cognition and Thought.
Wiley, 1965.

[Sacerdoti 74] Sacerdoti, E.D.
Planning in a Hierarchy of Abstraction Spaces.
Artificial Intelligence 5(2):115-135, 1974.

[Sadeh & Fox 88] Sadeh, N., and Fox, M.S.
Preference Propagation in Temporal Constraints Graphs.
Technical Report, Intelligent Systems Laboratory ,The Robotics Institute, Carnegie Mellon University, Pittsburgh, PA 15213, 1988.
CMU-RI-TR-89-2.

[Sadeh & Fox 90] Sadeh, N., and Fox, M.S.
Variable and Value Ordering Heuristics for Activity-Based Job-Shop Scheduling.
In *Proceedings of the Fourth International Conference on Expert Systems in Production and Operations Management.* May, 1990.

[Sathi et al. 90a] Camden, R., Dunmire, C., Goyal, R., Sathi, N., Elm, B., and Fox, M.S.
Distribution Planning: An Integration of Constraint Satisfaction and Heuristic Search Techniques.
In *Proceedings of the Conference on AI Applications in Military Logistics.* 1990.

[Sathi et al. 90b] Dunmire, C., Sathi, N., Goyal, R., Fox, M., and Kott, A.
Ammunition Inventory Planning: An Integration of Configuration and Resource Allocation Techniques.
In *Proceedings of the Conference on AI Applications in Military Logistics.* 1990.

[Simon 73] Simon, H.A.
The Structure of Ill-Structured Problems.
Artificial Intelligence 4:181-200, 1973.

[Stefik 81] Stefik, M.
Planning with Constraints (MOLGEN: Part 1).
Artificial Intelligence 16(2):111-140, 1981.

[van de brug et al. 86]
van de Brug, A., Bachant, J., and McDermott, J.
The Taming of R1.
IEEE Expert 1(3):33-42, 1986.

[Waterman & Hayes-Roth 78]
Waterman, D.A., and Hayes-Roth, F.
An Overview of Pattern-Directed Inference Systems.
Pattern-Directed Inference Systems.
Addison Wesley Pub. Co., 1978.

Table of Contents

1. Introduction 1
2. Factory Scheduling Example 1
3. Problem Topology 1
4. Problem Textures 5
5. Problem Objectives 6
6. CHS Problem Solving Process 6
7. Conclusion 7
References 8