Fox, M.S. (1989), "Reflections on the Relationship between Artificial Intelligence and Operations Research", Proceedings of the NASA Conference on Space Telerobotics,, Vol. 3, pp. 383-394

Reflections on the Relationship Between Artificial Intelligence and Operations Research

Mark S. Fox

Robotics Institute and Computer Science Department
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213

Abstract

Historically, part of AI's roots lie in Operations Research. It is the goal of this paper to explore how AI has extended the problem solving paradigm developed in OR. In particular, by examining how scheduling problems are solved using OR and AI, we demonstrate that AI extends OR's model of problem solving through the opportunistic use of knowledge, problem reformulation and learning.

1. Introduction

Artificial Intelligence (AI) has come along way during the last 35 years. Like any "new" decision technology, the publicity that heralds its appearance tends to sweep aside technologies, such as Operations Research (OR), that have come before[1]. Not surprisingly, the mere mention of the phrase Artificial Intelligence is liable to elicit from the Operations Research practitioner, a variety of responses from adulation to incredulity. In my experience, neither responses are appropriate, but are due primarily to a lack of knowledge of the field's basic methods. The goal of this paper is to dispel many of the misconceptions that surround AI technology. By examining both approaches to problem solving, both the AI and OR practitioner should obtain a better appreciation of each other's approaches. The ultimate goal being to merge the approaches.

Both OR and AI investigate problem solving methods. As Simon has noted, they share the same roots [19], but diverged at an early point in their history. A common distinction between AI and OR has been that OR deals with well structured problems and AI deals with ill structured problems. I believe it is time to put this distinction to rest.

Looking back at what has been written on the topic, Simon [18] defines a problem as being *well structured* if the means to solving the problem can be operationalized, in a computational sense. This tended to be a concern in the early days of computation[2]. Another view is that an ill structured problem is one which cannot be completely defined. E.g., objective function, constraints. Rapid prototyping is a means of elicting problem structure. A third view [16] is that a problem is ill structured if there only exists *weak* methods to solve it. Weak methods make weak demands on the task environment.

It is the last view that I believe is important. It primarily focuses on the performance of the problem solver. Weak methods tend to perform poorly because they are unable to reduce search complexity. Weak methods are not unique to AI nor OR. They each have their share of weak and strong methods. The real issue is what is the **nature** of the **strength** behind OR and AI problem solving theories?

One popular view is that AI problem solving draws its strength from heuristics. Glover explores the role of heuristics in [11], Grant also extolls the importance of heuristics in constructing an aircraft engineering management system, FIXER [12]. Another view is that AI methods are based on heuristic search and symbolic models vs OR's optimization a la linear programming and

[1] A concern also raised by Sutherland [22].

[2] for example, the issue of effective computability as described in Church's thesis [3].

quantitative modeling [19]. The view that will be explored in this paper is that the AI's strength is related to but more subtle than either of these notions.

The rest of this paper addresses this issue. The next two sections explore the strength of the OR approach and AI approach to problem solving respectively. A vareity of methods are described for each applied to the problem of factory scheduling. Based upon these examples the subsequent section explores the relationshiop of AI to OR.

2. OR Approach To The Scheduling Problem

Operations Research draws its problem solving strength from two inter-related sources: the modeling of problems using algebraic constraints, and the optimization of these models by means of mathematical programming -- linear programming being the core problem solving technique. There are other important areas of OR, such as queuing theory, simulation and network analysis, but it is mathematical programming that represents the bulk of the research.

In this section, we investigate the representation and solution of the one machine, non-preemptive weighted tardiness scheduling problem: Given a set of n jobs, sequence them through a single machine so that the total tardiness of all the jobs is minimized. We examine a number of problem solving techniques from an adequacy and efficiency perspective.

2.1. Modeling

The one machine scheduling problem can be modeled by a set of algebraic constraints. Given n jobs each with processing time P_i, due date D_i, and weight w_i:

n jobs.
P_i: processing time for job i
D_i: due date for job i
w_i: weight of job i

The objective is to minimize the total weigthed tardiness **T** subject to the following set of constraints:

$$min\ \mathbf{C} = \sum_{j=1}^{n} w_j T_j$$

$$s.t. \sum_{i=1}^{n} \left(\sum_{k=1}^{j} P_i X_{ik} \right) = C_j \qquad for\ all\ j$$

$$C_j - D_j = T_j - E_j \qquad for\ all\ j$$

$$\sum_{i=1}^{n} X_{ij} = 1 \qquad for\ all\ j$$

$$\sum_{j=1}^{n} X_{ij} = 1 \qquad for\ all\ i$$

$$X_{ij} \in \{0,1\} \qquad integrality$$

$$All\ Variables \geq 0$$

The core of this model is the variable $\mathbf{X}_{ij}$ which specifies the scheduling solution using a positional notation. That is, variable $\mathbf{X}_{ij}$ is one if the job is initially in position *i* is scheduled finally in position *j*, otherwise it is zero. The constraints restrict the use of **X** so that no job can be in more than one position.

While not necessarily obvious, the representation is powerful none-the-less.

2.2. Linear Programming

The first step in solving this problem is to see whether linear programming can be applied directly. In this case, the variable $\mathbf{X}_{ij}$ is an integer and cannot take on fractional values. Any solution generated using linear programming may assign fractional values to **X** which does not have an interpretation here.

2.3. Branch and Bound

Since **X** is integer, the most appropriate approach is to consider integer programming. The simplest algorithm is to generate a tree of alternative assignments to the variables $\mathbf{X}_{ij}$. Starting with the root node, the first two arcs would assign the variable $\mathbf{X}_{1,1}$ the values 0 and 1 respectively. Each of their binary branches would assign the variable $\mathbf{X}_{1,2}$ the values 0 and 1 respectively, and so on. At each node, constraints are tested to see if the variables generated to that point satisfy them. In particular, a node may assign a job to more than one position in the queue. Nodes that violate constraints are pruned. The full decision tree has $n!$ leaves. Search, even with pruning, is exponential in the size of the problem.

Another approach is to estimate the optimal solution and use it as stopping criteria in the branching search. This is can be accomplished by relaxing the integer constraint. This creates an L.P. which is easy to solve. The solution is a lower bound. If this solution is very close to our currently best feasible solution, then we have verified we are very near optimal, and do not have to continue the search.

Or we may use the L.P. solution as a starting "superoptimal," and try to make small changes restoring it to feasibility. That is, we search for a nearby feasible point. If so, we are done; if not, do more extended search by dual ascent of similar technique.

The branching procedure can be extended to full branch and bound if there was a way to prove that parts of the tree are inferior, then prune these parts away without ever checking. We can use the L.P. solution as the lower bounding procedure, but would have to solve an L.P. at each node in the search tree.

In each case, the size of the search tree still grows exponentially.

2.4. Lagrangian Relaxation

Another approach to solving the scheduling is to employ Lagrangian relaxation.

> "Lagrangian relaxation is based upon the observation that many difficult integer programming problems can be modeled as a relatively easy problem complicated by a set of side constraints. To exploit this observation, we create a Lagrangian problem in which the complicating constraints are replaced with a penalty term in the objective function involving the amount of violiation of the constraints and their dual variables." [8]

The idea is that the resulting integer program is supposed to be easier to solve. Since some constraints are missing, we will get a lower bound on the solution for any value of the lagrange multipliers.

$$min C = \sum_{j=1}^{n} W_j T_j + (1 - \sum_{i=1}^{n} X_{ij})$$

$$s.t. \sum_{i=1}^{n} (\sum_{k=1}^{j} P_i X_{ik}) = C_j \qquad \text{for all } j$$

$$C_j - D_j = T_j - E_j \qquad \text{for all } j$$

$$X_{ij} \in \{0,1\} \qquad \text{integrality}$$

$$\text{All Variables} \geq 0$$

Now we vary the multipliers, looking for values that will make the subsumed constraints true, so that the lower bound at that point will in addition be optimal.

For large problems the complexity of the search is prohitibitive.

2.5. Dispatch Simulation

Another approach to solving the scheduling problem is to use dispatch rules as part of a simulation. A rule is a heuristic that prioritizes jobs in the queue for a machine. In the case where there is a single statistic to be optimized, such as tardiness, dispatch rules have been shown to perform well. But when additional statistics are to be optimized, such as work in process, setups, etc., dispatch rule perform poorly [2].

2.6. OR Methodology

As Simon has noted [19], the emphasis of OR is on mathematical models and their optimization. It appears, at least from our examples, that the dominant optimization method for problems that cannot be solved directly using L.P. is to recast them as linear problems by the process of reformulation. Solutions to this reformulation are then used to guide a more general search process. The solution procedure is as follows:

1. Optimization via Constraint Satisfaction: In this case, a set of linear constraints entail a structure whose properties are known and can be taken advantage of when performing search (e.g., simplex).
2. Reformulation: For complex integer problems, simpler reformulations of the problem are solved in order to provide guidance in solving the more complex version.
3. Simulation: Myopic rules are used to make local decisions hopng that one or more macro statistics will be optimized (e.g., SPT minimizes weighted tardiness).

3. AI Approach To The Scheduling Problem

As in OR, Artificial Intelligence has more than one method of problem solving: deduction, constraint satisfaction and heuristic search. But the "key" insight from which AI draws its strength is best articulated in the Physical Symbol System definition of Newell & Simon [17]. A physical symbol system has the necessary and sufficient means for general intelligent action. A physical symbol system is "a machine which produces through time an evolving collection of symbol structures" and is capable of: Designation and Interpretation.

From this, the Heuristic Search Hypothesis is derived: Solutions to problems are represented as symbol structures. A physical symbol system exercises its intelligence in problem solving by search -- that is, by generating and progressively modifying symbol structures until it produces a solution structure [17]. The Problem Space operationalizes the concept of a phsycial symbol system. A problem space is composed of

- **States** which are collections of features that define some situation,
- **Operators**, that transform one state into another, and
- An **evaluation function**, that rates each state in the problem space.

Search begins at an **initial** state, and the problem is solved when a path is found from it to a **goal** state.

In this section we explore a more complex version of the factory scheduling problem. In this version:

- There are n jobs.
- Each job has a process graph, each node in the graph represents an operation, and each path through the graph is a possible process plan.
- Each operations requires zero or more resources, including machines, tooling, material and resources.
- Alternatives may be specified for each required resource.
- Jobs arrive at random times into the factory and have varying due dates.

As in the section on OR, we will first look at the representation of the problem, followed by a successive set of models to solve it.

3.1. Modeling

AI Knowledge representation research focuses on the development of ontologies and semantics that:

- span the set of concepts required to solve a problem,
- precisely and unambiguously represent concepts at all levels of granularity,
- provide a single representation that is understood and used by more than one application (other, more efficient representations may be constructed from this representation as required by an application), and
- be easily understood by the people who construct them.

Consider the following paragraph describing a factory activity:

> The milling operation precedes the drilling operation. It is composed of two steps: setup and run. Setup takes one hour and run time is 10 minutes. Two resources are required. A five poind wrench and an operator. The wrench is only required during setup. The operation is peformed in cost center 48.

Figure 3-1 is an example of the representation of the knowledge in that paragraph. It is relational in form. Figure 3-1 divides the knowledge into two types: activity and state. A state description describes a snapshot of the world before an activity is performed. For example "cost center 48 possess a wrench" is a state description. It must be true in order to enable the milling activity to occur. States and activities are linked via causal relations. A state describes what must be true of the world to enable an activity to occur. In addition, activities may be defined at multiple levels of abstraction. Milling is refined into two sub-activities: the setup and running of the machine. Lastly, we must represent time. Setup, in time, occurs before the milling run. Time is not absolute, it is relative. When describing the factory floor, it is atypical to use absolute time periods, instead, activities are described as preceding each other; and hence, once the time of one activity is determined the time of all the other activities related to it can be determined.

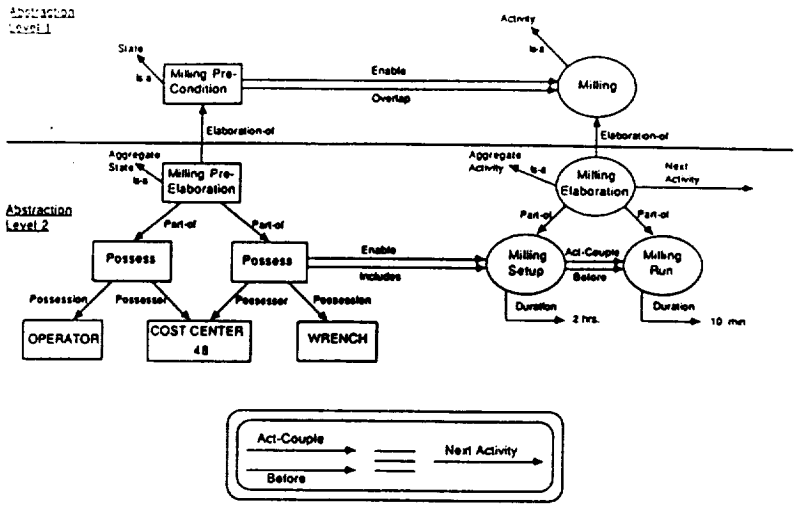

Figure 3-1: Activity Semantic Network

3.2. Constraint Directed Search

Scheduling can be viewed as search through a problem space, where states represent partial schedules, operators extend a partial schedule defined by a state into a new state, and the evaluation function rates each state in the problem space according to the known constraints. Constraint directed search is a form of search where constraints can be used to specify operators (e.g., operation precedence constraints specify the next operations) and terms of the evaluation function (e.g., a due date constraints measures slack in the schedule). The efficacy of this approach depends on the ability of the constraints to identify the more profitable paths to pursue. Experience has shown that this tends not to be the case.

3.3. Expert Systems

Another important insight in AI, due to Feigenbaum [6], is that the combinatorics of moving through the problem space can be reduced by smarter selection of operators. That is by utilizing knowledge of the domain. Domain knowledge can be represented as elaborations on the conditions of operators, making the application of operators more sensitive to the current context. This leads to the "knowledge-search dichotomy": the more knowledge one has the less search needs to be performed; the less knowledge one has the more search has to be performed to solve the problem.

One source of knowledge is from an expert. For example, rather than generate all possible possible alternative operations everytime scheduling is performed, expertise suggests that we consider alternatives only when capacity becomes scarce on the machines used in the standard plan:

```
IF      Milling is to be scheduled
        AND Capacity is scarce
        AND Queue time is high
        AND Due date is tight

THEN    Consider
                1. Grinding
                2. Subcontracting
```

There are two problems with the expert sytems approach:

1. Problems like factory scheduling tend to be so complex that they are beyond the cognitive capabilities of the human scheduler. Therefore, the schedules produced by the scheduler are poor; nobody wants to emulate their performance.
2. Even if the problem is of relatively low complexity, factory environments change often enough that any expertise built up over time becomes obsolete.

Expert systems appear to be appropriate only when the problem is both small and stable.

3.4. Hierarchical Constraint Directed Search

Since the problem cannot be solved using expert systems, more sophisticated search techniques are required. On approach is to reformulate the problem as a simpler problem whose solution can be used to guide the solution of the original problem. Hierarchical constraint directed search (HCDS) is one method that embodies this approach.

Search is divided into four levels: order selection, capacity analysis, resource analysis, resource assignment. Each level is composed of three phases: a pre-search analysis phase which constructs the problem, a search phase which solves the problem, and a post-search analysis phase which determines the acceptability of the solution. In each phase, ISIS uses constraints to bound, guide, and analyze the search.

Level 1 is responsible for selecting the next unscheduled order to be added to the existing shop schedule. Its selection is made according to a prioritization algorithm that considers order type and requested due dates. The selected order is passed to level 2 for scheduling.

Level 2 represents the simpler reformulation of the original problem. It simplifies the problem by removing both resources and constraints from consideration. It performs a dynamic programming analysis of the plant based on current capacity constraints. It determines the earliest start time and latest finish time for each operation of the selected order, as bounded by the order's start and due date. The times generated at this level are codified as operation time bound constraints which serve to influence the search at the next level.

Level 3 solves the original scheduling problem. It selects a particular routing for the order and assigns reservation time bounds to the resources required to produce it. Pre-search analysis begins with an examination of the order's constraints, resulting in the determination of the scheduling direction (either forward from the start date or backward from the due date), the creation of any missing constraints (e.g. due dates, work-in-process), and the selection of the set of search operators which will generate the search space. A beam search is then performed using the selected set of search operators. The search space to be explored is composed of states which represent partial schedules.

Once a set of candidate schedules have been generated, a rule-based post search analysis examines the candidates to determine if one is acceptable (a function of the ratings assigned to the schedules during the search). If no acceptable schedules are found, then diagnosis is performed. Intra-level repair may result in the re-instantiation of the level's search. Pre-analysis is performed again to alter the set of operators and constraints for rescheduling the order. Inter-level repair is initiated if diagnosis determines that the poor solutions were caused by constraint satisfaction decisions made at another level.

Level 3 outputs reservation time bounds for each resource required for the operations in the chosen schedule. Level 4 then establishes actual reservations for the resources required by the selected operations which minimize the work-in-process time.

This approach performs well when there exists adequate capacity in the factory. But in situations where contention for resources was high, the performance of the system was no better than the human scheduler.

3.5. Macro-Opportunistic Constraint Directed Search

In situations where resources are highly contended for, experience has shown that optimizing resource allocation by scheduling operations incident with the resource produces better results than scheduling jobs one at a time. Macro-opportunistic constraint directed search [21, 20] extends HCDS by first analyzing the capacity requirements of all jobs in order to identify whether there is a high degree of resource contention. If contention exists for a resource, then a resource centered scheduler is chosen to schedule the operations incident with it (ususally a dispatch rule simualtion). The job cenered scheduler (i.e., HCDS) is used to scheduler the jobs out from the resource. Opportunism arises out of the systems ability to dynamically determine at any point during the construction of schedules, primary/secondary bottlenecks and take the opportunity to schedule them rather than pursue a strictly job centered approach.

Experiments with this approach has shown that it outperforms both hierarchical constraint directed search and dispatch rule approaches. Secondly, it can efficiently solve typically complex factory scheduling problems.

3.6. Observations

Based on the approaches described above, it is clear that AI problem solving is based upon search and incorporates the following refinements:

- Knowledge of constraints can be used to both guide search (in the form of preferences) and can be used to prune alternatives.
- Reformulation is used to define simpler versions of a problem whose solutions are used to guide search in the more complex case.
- Introspection, that is the systems analysis of its own performance, is used to modify how the system solves the problem in subsequent trials. For example, knowledge of constraint violations can be used to automatically reformulate a problem.
- Opportunism is used to decide where the search is to proceed from. Opportunism can exist at more than one level. At the micro level it can be used to select the next state to extend. At the macro level, it can be used to select the perspective to apply next (e.g., resource vs job).

4. Reflections

It is clear from the earlier sections that both AI and OR are developing strong methods for performing search in the problem space. OR techniques are generally, but not exclusively, constraint directed and quantitatively modeled. AI techniques are generally, but not exclusively, pattern directed and symbolically modeled. Part of the reason for this difference is due to ORs focus on *optimization* and AI on *satisficing*. The former strives to select problems for which it can define high quality, hopefully optimal, and efficient solutions while the latter strives to find high quality and efficient solutions for everyday, messy problems.

It is my view that much of the work in representations and search in AI can be viewed as a natural extension of OR.

Observation 1: AI representations extend OR representations by the processes of abstraction and differentiation.

AI Knowledge representations are qualitative abstractions of underlying quantitative models. Abstractions can be very powerful. They enable the answering of questions that an underlying quantitative model cannot. Lets assume that a quantitative model of time has for each activity to be performed a start time and end time specified. With this model the following question could be answered: "find the start and end time of activity 3 that has the longest duration given start and end times of activity 1 and 2, and knowing that Activity 1 is during activity 2 which is during activity 3.

OBJECTIVE FUNCTION: MAX $et_3 - st_3$

CONSTRAINTS:

$st_1 < et_1$ $st_1 >= st_2$ $st_2 >= st_3$
$st_2 < et_2$ $st_1 <= et_2$ $st_2 <= et_3$
$st_3 < et_3$ $et_1 <= et_2$ $et_2 <= et_3$

WHERE

st_1, et_1, st_2, et_2 are known

ALGORITHM: Simplex

One can view Allen's relational model of time [1] as an abstraction of a quantitative model. In the relational model the temporal relation **during** denotes that one activity is performed during the time that another activity is performed. This relationship can be asserted without knowledge of the actual times of each activity. If we did not know any of the start times or end times for each activity but knew only that Activity 1 is during activity 2 which is during activity 3, we would be able to answer the following question: "Is activity 1 during activity 3?"

GIVEN: A_1, A_2, A_3 are activities, and

A_1 -----d-----> A_2 -----d-----> A_3

where d is the during relation, and
d is transitive

ALGORITHM: Hypothesis Introduction

By its nature, an abstraction is a partial model of an underlying more complete model. In fact, there is a contiuum of partial models one can construct. As demonstrated, each partial model enables the answering of a different set of questions with more or less efficiency.

AI representations also enable the differentiation of concepts that are contained in a quantitative representation. For example, there are many types of temporal relations that can be defined between activities in addition to *during*, for example:

- *before* specifies that the end of one activity occurs before the beginning of another
- *overlap* specifies that the end of one activity occurs after the beginning of anther but before the other ends

Allen [1] identifies 13 types of temporal relations.

Being able to differentiate among concepts contained within but not easily identified in a quantitative model is a requirement for the construction of knowledge based search; the more precise the representation, the more specific the definition of situations for which actions can be defined.

Observation 2: Search is the core method employed in both the AI and OR problem solving models.

By definition, the core problem solving technique of AI is search; it forms the basis of the problem space model. Search plays a similar role in OR. Linear programming is a search technique where vertices in a n dimensional space are visited to find an assignment that optimizes an objective function. Both simplex and karmarkar algorithms exploit knowledge of the structure of the problem space to more quickly find a solution. For integer problems, branch and bound is directly related to AI's AI* algorithm [15]. For non-linear problems, the General Reduced Gradient method is yet another form of search where gradients are used to decide where in the n dimensional space to move to next.

Observation 3: AI extends the model of problem solving by the opportunistic application of situational knowledge of the domain.

The definition of a problem space includes the generation of new states through the application of operators. Conditions of operators are actually descriptions of situations at which a particular decision is to be made. Operators are created as a result of a situational analysis of how to solve a problem.

Operators in AI are opportunistically applied; at each step in the search the operator that best matches the current situation is chosen to extend the search. The transition from one state to another in a problem space in AI programs is rational; at each step the next step is opportunistically made, jumping from one island of certainty to another, and one decision process to another, within the problem space. This is in contrast to how the next decision is selected using OR algorithms. By careful analysis of the strcuture of the problem, OR has devised a more mechanistic approach in deciding what to do next in the search process.

Observation 4: AI extends the model of problem solving from analysis to design.

As Simon has noted [19], OR optimization techniques perform analysis but are unable to perform design. Optimization searches for an assignment of values to variables that optimize an objective function. All variables and their domains are known apriori. But in design new variables and constraints among them are generated during the search process. For example, in the design of a mechanical part, the refinement of the part into subparts introduces new variables (representing the subparts) and constraints among them.

Observation 5: AI extends the model of problem solving to include the automation of problem formulation and re-formulation.

Within OR, problem formulation is the purview of the analyst. AI methods have extended problem solving to include the formulation of problems by the reasoned analysis of hypotheses and the implications. For example, methods of Truth Maintenance [5, 4] enable the representation and management of sets of hypotheses that support deductions. If deductions are found to be unacceptable, hypotheses and their dependent deductions can be withdrawn and alternative hypotheses explored. Consequently, the selection of model parameters are subject to change by the program itself.

In constraint directed search [9, 10], pre and post-analysis of the search space allows for the formulation and re-formulation of the problem via the modification of constraints and the selection of search operators.

Lastly, the SOAR model [14] of problem solving defines the concept of "universal subgoaling" in which new problem spaces are created where re-formulations of the original problem are worked on. The SOAR model supports the opportunistic movement from one problem space to another at each step during problem solving.

Observation 6: AI extends the model of problem solving to include the acquisition of expertise.

L.P. algorithms do not learn from their experience. Running them on the same or similar problem does not result in a net speed up in search. The use of prior experience in increasing a problem solver's performance is never the less important. A variety of methods for learning from prior experience have been used to increase an AI problem solver's performance over time. These include a variety of macro-operations [7, 13] and chunking mechanism [14].

5. Conclusion

Both AI and OR are trying to develop strong methods for performing search in the problem space. OR techniques are generally (but not exclusively) constraint directed and quantitatively modeled. AI techniques are generally (but not exclusively) pattern directed and symbolically modeled. AI methods represent a powerful extension to the OR model of problem solving in two senses. First, the use of situational knowledge in opportunistically guiding search reduces the combinatorics of search in many cases. Secondly, the model of problem solving is extended to include the automation of problem formulation/re-formulation and learning from prior experience.

Acknowledgements

This paper is based on a seminar prepared by Thomas Morton and myself. This work was supported, in part, by the Defense Advanced Research Projects Agency under contract #F30602-88-C-0001, and in part by a grant from Digital Equipment Corporation.

References

[1] Allen, J.F.
Towards a General Theory of Action and Time.
Artificial Intelligence 23(2):123-154, 1984.

[2] Baker, K.R.
Introduction to Sequencing and Scheduling.
John Wiley & Sons, 1974.

[3] Brainerd, W.S., and Landweber, L.H.
Theory of Computation.
John Wiley & Sons, 1974.

[4] de Kleer, J.
An Assumption-based TMS.
Artificial Intelligence 28(2):127-162, 1986.

[5] Doyle, J.
A Truth Maintenance System.
Artificial Intelligence :231-272, April, 1979.

[6] Feigenbaum, E.
The Art of Artificial Intelligence.
In *Proceedings of the International Joint Conference on Artificial Intelligence.* Morgan Kaufman, 1977.

[7] Fikes, Hart, and Nilsson.
Learning and Executing Generalized Robot Plans.
Artificial Intelligence 3:251-288, 1972.

[8] Fisher, M.L.
An Applications Oriented Guide to Lagrangian Relaxation.
Interfaces 15(2):10-21, 1985.

[9] Fox, M.S.
Constraint-Directed Search: A Case Study of Job-Shop Scheduling.
Technical Report, Carnegie-Mellon University, 1983.
CMU-RI-TR-85-7,Intelligent Systems Laboratory,The Robotics Institute, Pittsburgh,PA.

[10] Fox, M.S.
Observations on the Role of Constraints in Problem Solving.
In *Annual Conference of the Canadian Society for the Computational Studies of Intelligence*. University of Quebec Press, 1986.

[11] Glover, F.
Future Paths for Integer Programming and Links to Artificial Intellience.
Technical Report 85-8, Center for Applied Artificial Intelligence, Graduate School of Business, Univesity of Colorado, 1985.

[12] Grant, T.J.
Lessons for O.R. from A.I.: A Scheduling Case Study.
Journal of the Operational Research Society 37(1):41-57, 1986.

[13] Korf, R.E.
Macro-Operators: A Weak Method of Learning.
Artificial Intelligence 26(1):35-77, April, 1985.

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

[15] Nau, D.S., Kumar, V., and Kanal, L.
General Branch and Bound and Its Relation to A* and AO*.
Artificial Intelligence 23(1):13-28, 1984.

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

[17] Newell, A., and Simon, H.A.
Computer Sciences as Empirical Inquiry: Symbols and Search.
Communications of the ACM 19(3):113-126, 1976.

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

[19] Simon, H.A.
Two Heads Are Better than One: The Collaboration between AI and OR.
Interfaces 17(4):8-15, 1987.

[20] Smith, S.F., and Ow, P.S.
The Use of Multiple Problem Decompositions in Time Constrained Planning Tasks.
In *Proceedings of the Ninth International Conference on Artificial Intelligence*, pages 1013-1015. 1985.

[21] Smith, S., Fox, M.S., and Ow, P.S.
Constructing and Maintaining Detailed Production Plans: Investigations into the Development of Knowledge-Based Factory Scheduling Systems.
AI Magazine 7(4):45-61, Fall, 1986.

[22] Sutherland, J.W.
Assessing the Artificial Intelligence Contribution to Decision Technology.
IEEE Transactions on Systems, Man and Cybernetics SMC-16(1):3-20, 1986.