Planning As Mixed-initiative Goal Manipulation

Michael T. Cox
Department of Computer Science & Engineering
Wright State University
Dayton, OH 45435-0001
mcox@cs.wright.edu

Abstract

Mixed-initiative planning systems attempt to integrate human and AI planners so that the synthesis results in high quality plans. In the AI community, the dominant model of planning is search. Operators having preconditions, post conditions, and variable bindings represent actions. Such operators can change the world through their effects. Planning therefore consists of search from an initial state to a goal state by means of operator and variable binding choices. The alternative that we propose is to model planning as a goal manipulation task. Here planning involves moving goals through a hyper space in order to reach equilibrium between available resources and the constraints of a dynamic environment. The users can establish and "steer" goals through a visual representation of the planning domain. They can associate resources with particular goals and shift goals along various dimensions in response to changing conditions as well as change the structure of previous plans. Users need not know details of the underlying technology, even when search is used within. Here we empirically examine user performance under both alternatives.

1. Introduction

The idea to treat planning as a goal manipulation process has much appeal, at least intuitively. Goals have long been recognized as important to a full understanding of both human and machine problem-solving abilities. A goal provides a focus for inference and learning, and it makes explicit the objectives of a system, either human or machine (Ram & Leake, 1995). By having explicit representation of goals, a planner can more easily reason not only about the domain of planning but also the direction of planning. Furthermore by including a means for explicitly representing goals at the interface of a computer problem-solving system, users are encouraged to express their specific intentions thereby forcing users to keep their "eyes on the prize."

Such focus is important in not only planning but also for learning within a planning system. Schank, Fano, Bell, and Jona (1993) argue that both problem solving and learning require a user to possess specific goals that motivate and direct these processes. They stress that the passive act of absorbing new information is not sufficient. Likewise, a mixed-initiative system requires a goal-driven user, if effective use is to be supported across problem domains. Yet in this paper, we view goals somewhat differently than do most cognitive or engineering approaches to intelligence. Goals are mutable. They will change appropriately with user direction according to the current context; they will change inappropriately when left to the vagaries of a dynamic environment.

Overwhelming combinatorics face fully autonomous planners attempting to solve such problems. A mixed-initiative approach inserts into the planning process an active human user whose knowledge is outside the transitive closure of course-grain tractable domains.

Section 2 describes the concept of goal transformations upon which mixed-initiative planners are implemented. Section 3 introduces the GTrans planner in addition to an alternative planner. Section 4 explains how user can plan for goal change by introducing the Bridges Problem. Section 5 summarizes an empirical study performed to compare the two different planning interfaces, and Section 6 closes the paper.

2. Goal Transformations

We start with the observation that goals often become obsolete given inherent uncertainties in a resource-limited world and a changing circumstance. For example the goal to have a package delivered may be abandoned because the recipient is no longer at the package address. Instead a goal to return the package to the sender may be substituted. Only under the closed world assumption can goals be defined as static input states to be achieved. Moreover, goals are often vague and ill-defined in the early stages of planning and require refinement throughout the planning process. Goals are not necessarily atomic, rather they may be semi-achievable. For instance Williamson (1996) examined time constraints on goals. A goal may not be achieved by a certain date, but may be achieved slightly later at a specific cost. In his conception, planning consists of maxi-

mizing a value function on goals. As Williamson further notes, a conjunctive goal is composed of multiple goals that all need to be satisfied. So satisfying some instead of all is better that satisfying none at all. We also prefer planners that do not simply fail under resource poor circumstances. Instead planners should not only be able to revise their plans when resources are limited, but they should also be capable of revising their goals in order to achieve solutions that are acceptable.[1] To address these problems we have created a theory of goal change formally represented as transformations.

A *goal transformation* represents a goal shift or change. Conceptually it is a change of position for the goal along a set of dimensions defined by some abstraction hyperspace (Cox & Veloso, 1998). The hyperspace is associated with two hierarchies. First the theory requires a standard conceptual type-hierarchy within which instances are categorized. Such hierarchies arise in classical planning formalisms. They are used to organize arguments to goal predicates and to place constraints on operator variables. Goal transformation theory also requires a unique second hierarchy.

In normal circumstances the domain engineer creates arbitrary predicates when designing operator definitions. We require that these predicates be explicitly represented in a separate predicate abstraction hierarchy that allows goals to be designated along a varying level of specificity. For example consider a military domain. The domain-specific goal predicate is-ineffective takes an aggregate force unit as an argument (e.g., (is-ineffective enemy-brigade1)). This predicate may have two children in the goal hierarchy such as is-isolated and is-destroyed. The achievement of either will then achieve the more general goal (Cox & Veloso, 1998). Furthermore if the predicate is-destroyed had been chosen to achieve in-effective, the discovery of non-combatants in the battle area may necessitate a change to is-isolated in order to avoid unnecessary casualties. Note also that to defer the decision, the movement may be to the more general is-ineffective predicate. Then when the opportunity warrants and further information exists, the goal can be re-expressed. In any case, movement of goals along a dimension may be upward, downward or laterally to siblings.

Goal movement may also be performed by a change of arguments where the arguments exist as objects of or members of the standard object type-hierarchy. The goal represented as the type-generalized predicate (inside-truck Truck1 PACKAGE) is more general than the ground literal (inside-truck Truck1 PackageA). The former goal is to have some package inside a specific truck (thus existentially quantified), whereas the latter is to have a particular package inside the truck. Furthermore both of these are more specific than (inside-truck TRUCK PACKAGE).[2] Yet movement is not fully ordered, because (inside-truck Truck1 PACKAGE) is neither more or less general than (inside-truck TRUCK PackageA).

A further way goals can change is to modify an argument representing a value rather than an instance. For example the domain of chess may use the predicate outcome that takes an argument from the ordered set of values {checkmate, draw, lose}. Chess players often opt for a draw according to the game's progress. Thus to achieve the outcome of draw rather than checkmate represents a change of a player's goal given a deteriorating situation in the game.

Generally planning is to achieve a desired state by managing the world, the resources, and the intended outcomes one wishes. Although a change in the world is delivered by acting upon it, the selection of actions is but one choice a planning agent has at its disposal. The choice of resources to commit to a given goal and the decision as to what state in which the world needs to be for the planner to be satisfied are as equally important as the actions themselves. Thus planning is really a context-dependent task of discovering, managing, and refining what one actually wants.

3. The GTrans Mixed-Initiative Planner

To directly support the goal manipulation model, we implemented a mixed-initiative interface to a planning system through which the user manipulates goals, the arguments to the goals, and other properties. The interface hides many of the planning algorithms and knowledge structures from the user and instead emphasizes the goal-manipulation process with a menu-driven and direct manipulation mechanism. The system, called *GTrans*[3] (Cox, 2000; Zhang, 2002), presents a direct manipulation interface to the user that consists of a graphical map with drag and drop capability for objects superimposed upon the map surface. GTrans helps the user create and maintain a problem file that is internally represented as follows. A planning problem consists of an initial state (a set of objects and a set of relations between these objects) and a goal state (set of goals to achieve). The general sequence is (1) to create a planning problem, (2) invoke the underlying planner to generate a plan, and then until satisfied given planning feedback either (3a) change the goals or other aspects of the problem and request a plan or (3b) request a different plan for the same problem.[4]

1.Our theory of goal change also recognizes upward revision to take advantage of opportunities to achieve more than originally intended. However we do not consider it here. See Cox and Veloso (1998), Edwin (2001), and Edwin and Cox (2001) for such discussion.

2.A type generalized predicate (p TYPE) is equivalent to the existentially quantified expression $\exists \dot{x} | (TYPE(x) \wedge p(x))$

3.The GTrans home page is `www.cs.wright.edu/~mcox/GTrans`.

4.Prodigy4.0 has the ability to iterate the planning process for a given problem generating a different solution depending upon the `:multi-sols` keyword parameter (Carbonell, et al., 1992). GTrans incorporated this into the interface. See Zhang (2002)

To support step 3a above, GTrans supports three general classes of goal transformations.

1. Goal type transformation
2. Goal argument transformation
3. Valence transformation

A *goal type transformation* enables the planner to manually transform a goal by moving the predicate of the goal along an abstraction hierarchy defined in the domain knowledge. The *goal argument transformation* involves an upward movement through an abstraction hierarchy on one or more goal arguments. A *valence transformation* is performed by toggling between a positive or negative truth value of the goal (i.e., whether to achieve the predicate or its negation).

The overall GTrans family of systems incorporates a number of versions including a multi-user collaborative planning system. This system configuration includes multiple networked copies for each user and a joint planning interaction mode along with information sharing capabilities. Another version integrates the mixed-initiative (human-directed) system with a multiagent (artificial agent) planning system called COMAS (Edwin, 2001; Edwin & Cox, 2001). COMAS (CoOrdination in MultiAgent Systems) also uses goal transformations for planning. Here we focus on a single-user version that interacts with one underlying planner.

The *GTrans User Interface Version 2.1* (Zhang, 2002) is the implementation that hides the planning algorithms and representations from the human user by focusing on the goal manipulation process. It provides the user with a mechanism to directly manipulate the set of objects, initial state of the planning world, as well as the objectives of the planning problem. The user is able to define the objectives or goals and to assign particular resources to achieve the goals. When the underlying planner fails to generate a plan because of insufficient resources or because the planning world changes, the user can asynchronously modify the goals and send them back to the planner for another round of planning, thus steering the planning process. See Figure 1 for an illustration of the interface. Discussion of the content of this window will follow in the next section.

The user interface is implemented in Java Version 1.2. The overall architecture includes two principle components: (1) Prodigy/Agent (written in Allegro Common Lisp) and (2) the user-interface single-agent component. The interface presents a graphical map display and a set of pull-down cascading menus to the human planner. The human can create objects, define initial states, set or transform goals, create problems, and solve problems. The *Prodigy/Agent* system[5] (Cox, Edwin, Balasubramanian, & Elahi, 2001; Cox, Elahi, & Cleereman, 2003) allows the underlying PRODIGY planner to communicate with the interface through a KQML protocol. The two programs use sockets to pass KQML performatives.

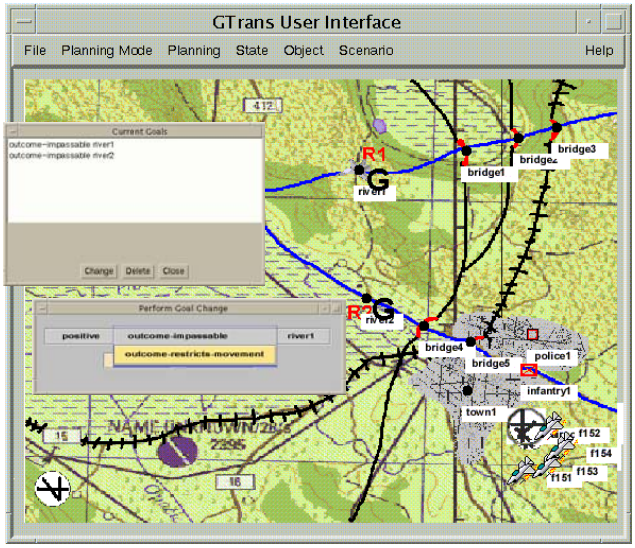

Figure 1. GTrans User Interface 2.1

PRODIGY (Veloso, Carbonell, Perez, Borrajo, Fink, & Blythe, 1995) is a domain-independent, nonlinear, state-space planner implemented at Carnegie Mellon University. It searches for a sequence of actions that transform the environment from an initial state into a final state containing the goal state. Like all state-space planners, its problem specification includes a set of objects existing in the planning environment, the initial state of the environment, and the goal state that needs to be achieved by the plan. Prodigy/Agent consists of a wrapper program and the PRODIGY planner. The wrapper program serves as a software interface between the PRODIGY planner and external agents.

PRODIGY has a standard user-interface that comes with the CMU system (see Figure 2). The *Prodigy 4.0 User Interface 2.0* (Cox & Veloso, 1997) provides access to the underlying data structures such as the search tree, the goal-subgoal graph (left canvas in Figure 22), and the emerging plan (right canvas in Figure 2). It also includes various pull-down menus and buttons to control planning and interface parameters, planning mode, and other facets. The interface displays the progress of the planning process in terms of search. Cox and Veloso claimed that the interface provides a mechanism that improves user-performance with the system, but this is a weak claim, because the previous interface was simply a LISP interpreter. Here we will compare the standard PRODIGY interface to the alterative GTrans interface.

When it loads a domain, the GTrans User Interface obtains the domain information from Prodigy/Agent through four types of requests. In response to an obj-request, Prodigy/Agent sends back to GTrans User Interface the types of objects that exist in the domain. After receiving a goal-

5.The Prodigy/Agent home is www.cs.wright.edu/~mcox/Prodigy-Agent/

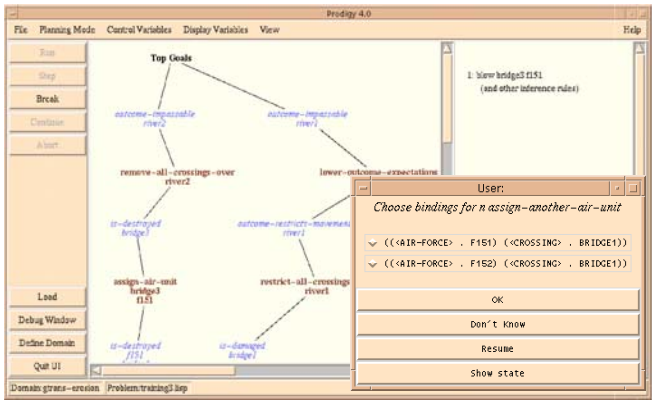

Figure 2. Prodigy 4.0 User Interface 2.0

request, Prodigy/Agent replies with all the possible goals that can be achieved in the domain. To support goal classification information PRODIGY reads each operator from a domain file and sends to GTrans a list of positive effects from each operator. The effects are represented as (predicate arg1-type arg2-type... argN-type). GTrans then can associate objects defined on the map with goals relevant to each**.** A tree-goal-request asks for all of the goal hierarchies in the domain. And a state-request requests all the possible initial states in the domain.

Unlike domain information that remains static for a given domain, problem information including object information, state information and goal information may change over time as the planning world changes. The communication of problem information between the Planning User Interface and Prodigy/Agent occurs after the human planner creates a set of objects, specifies the initial state, sets the goal state, and gets ready to run the problem. After receiving the request to search for a plan as well as the problem information, Prodigy/Agent runs the PRODIGY planner and sends the resultant plan, if any, back to the requesting Planning User Interface. In case of failure in generating a plan, Prodigy/Agent simply sends back a failure message indicating no plan can achieve the goal state of the problem.

4. Planning for Goal Change

Cox and Veloso (1998) introduced a problem that illustrates the trade-offs between resource allocation decisions in a military air campaign planning domain. The "Bridges Problem" is simply to make rivers impassable by destroying all bridges across them. This task universally quantifies the variable `<crossing>` with the relation (enables-movement-over `<crossing>` river) and requires a separate air unit for each crossing in order to achieve the goal. Therefore if a new crossing is discovered or an old resource becomes unavailable,[6] the constraints of the problem change thereby forcing dynamic replanning. When a goal state is composed of conjunctive goals for multiple rivers, an interesting trade-off exists for the user. To maximize the goal satisfaction, the optimal user will allocate resources to rivers with fewer crossings (Edwin & Cox, 2001). Figure 1 above presents a scenario with four air units (resources) and two rivers, the first with three bridges and the second with two. The user should allocate two resource units to the two-bridge river and two to the three-bridge one. By doing so the goal to make the first river impassable is achieved fully, whereas the second goal is more fully satisfied than if the reverse allocation was performed. Note that this determines a change to the second goal rather than the first. By transforming the goal (outcome-impassable river2) to (outcome-restricts-movement river2) the planner is able to achieve success.

The GTrans User Interface Version 2.1 provides a unique facility to transform or *steer* goals in such situations. When the user receives feedback from the underlying Prodigy/Agent planner (either a successful plan that may not be acceptable by the user, a failure to generate any plan, or an apparent problem due to the planner taking too long), the user can asynchronously modify the goals and send them back to the planner for another round of planing. The user does this by graphically manipulating the goals by selecting the "Change Goals" choice on the Planning pull-down menu.

Figure 1 shows two dialogue boxes that allow the user to change the goal of making river1 (R1) impassable by destroying all bridges across it. By highlighting the goal the users wished to change and then clicking on the "Change" button from the "Current Goals" pop-up menu, the goal can be modified by any of the general classes of transformations mentioned earlier. Because the erosion transformation moves the goal predicate itself down a level of abstraction, the goal changes to achieving the state of outcome-restricts movement rather than the state of outcome-impassability.

Fig. 2 above shows the standard PRODIGY interface with the same problem as that in Fig. 1. Instead of manipulating a representation of the goals, the user manages the problem in terms of the search technology that underlies PRODIGY. The figure shows the user with a choice of variable binding that determines which air unit will be used against BRIDGE1. In the goal manipulation model the user effects change directly to a representation of the goal. In the search model the change is represented by a PRODIGY inference rule. This rule is called lower-outcome-expectations and is visible in the goal-subgoal graph of Fig. 2. It asserts that if the goal to restrict movement over river1 is achieved, then the goal to have the river impassable is also achieved. The user must select this rule from among operator choices in during planning. Having made this change in the step prior to the step shown in the figure, the user must now

6.For example an air unit may break down during planning. In this paper we will not include a dynamic environment that changes (but see Cox & Veloso, 1998, for examples). Rather we concentrate on planning under limited resources and leave environmental change for future research.

decide which air unit to use for damaging river2 by specifying a binding for variable <AIR-FORCE> in Fig. 2.

5. Evaluation

Cox and Veloso (1997) made the claim that the Prodigy 4.0 User Interface 2.0 was an effective interface for mixed-initiative planning because of three characteristics. Firstly the interface allows both generative and case-based planning and as such is better suited to human cognitive constraints, because case-based reasoning represents and utilizes experience in the form of cases. Secondly the interface displays the planning process as well as the plan itself. This process is a runtime animation of the PRODIGY planning algorithm as shown in the goal-subgoal graph. Thirdly the interface has the potential to support both experts and novices in the planning technology. However these purported characteristics were never substantiated empirically.

Here we examine the first and third characteristics, although we do not test the user performance under the case-based mode in PRODIGY (i.e., Prodigy/Analogy. See Veloso, 1994). As noted by Cox and Veloso, however, the interface was primarily designed for the expert technologist. Some suggestions were made as to how the interface might be improved for novices. But instead of minor changes to the existing interface, we argue that a completely new approach is needed as incorporated in the GTrans Interface 2.1. Indeed, we expect that the goal-subgoal graph is of little help to the naive user who does not understand the search process. Instead it may actually interfere with decisions.

An experiment was performed with human subjects to compare and contrast the models each interface implements. The experiment is designed to evaluate the differences of the two models under differing amount of task complexity using both expert and novices.[7] This experiment uses 18 variations on the Bridges Problem in the military domain as test problems. In these problems, insufficient resources exists with which to solve the problem completely. Choices can be made, however, so that a solution is produced that achieves a partial goal satisfaction represented as a ratio of the subject's partial solution to the optimal partial solution.

The graph in Figure 3 shows the mean of the goal satisfaction ratio under the goal manipulation model and the search model. When presented with the goal manipulation model, subjects achieve over 95 percent goal satisfaction on average. When presented with the search model, subjects achieve about 80 percent goal satisfaction on average.

Given that the cognitive model itself is an important factor as concluded in previous analysis, we next examine the possible relationships among three independent variables: planning model, problem complexity, and subject expertise.

7.Experts are those 6 out of 13 subjects who had familiarity to search in AI.

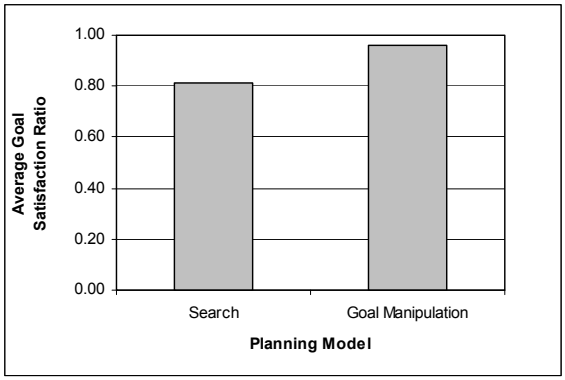

Figure 3. Goal satisfaction as a function of cognitive model

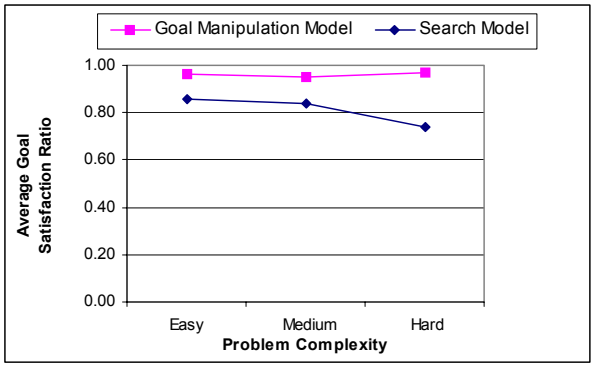

Figure 4. Goal satisfaction as funct. of problem complexity

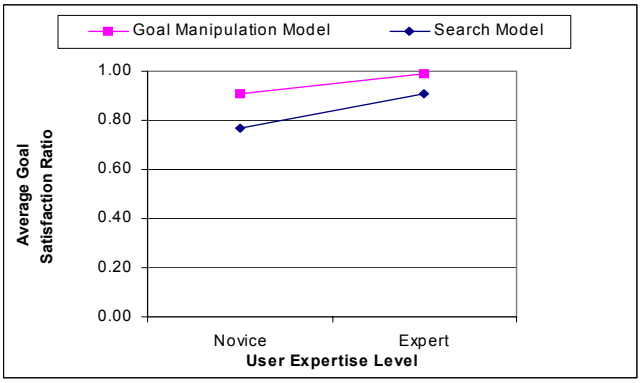

Figure 5. Goal satisfaction as a function of expertise

Figure 4 plots the average goal satisfaction ratio for each combination of the planning model and the problem complexity. As can be observed from the graph, when the goal manipulation model is presented to the user, the goal satisfaction ratio generally remains the same with increasing problem complexity; but when the search model is presented to the user, the goal satisfaction ratio decreases as the problem complexity increases. It is very likely that the effect of the planning model on the user performance depends on the problem complexity.

The next step of our analysis is to examine the possible interaction effects between the planning model and the user

expertise level. Figure 5 shows the average goal satisfaction ratio for each combination of the planning model and the user expertise level. It is apparent that experts perform better than novices under both planning models. But the two plot lines representing each planning model are not parallel, indicating the possible interactions between the two factors.

6. Conclusion

In the experiment subjects are indirectly learning the Bridges Problem evaluation function specified by Cox and Edwin (2001). This function calculates the optimal resource allocation policy for planning problems given resource limited environments. What is unique is that this resource allocation task can be conceptualized as a goal change task instead. The problem for the subjects in the goal manipulation model is to decide which goal most needs changing rather than to decide which resources are directly allocated to which subgoal. The automated planner will take care of these details instead of the subject. In the search model, however, the user is confronted with a more direct and overly detailed allocation decision. See Figure 2 for an example of the allocation choice presented to the user under the search model.

The results from the experiment clearly demonstrate that a strong effect exists on human planning performance and efficiency of performance due to the GTrans interface in contrast to the standard PRODIGY interface. We argue that these effects are due to the cognitive model of planning presented to the user. Because both software systems present a mixed-initiative interface to the same Prodigy/Agent planner underneath, the effects are not due to the autonomous planner itself. We claim that the metaphor of planning as goal manipulation is easier to understand than is search.

In summary this paper has contrasted two models of planning as a conflict of metaphors, but the models are not mutually exclusive. Search is still an important computational technique that can be used efficiently in mixed-initiative systems as illustrated in this paper. However, we argue that search is not an appropriate model to present to the user, especially a naive one. A stronger view is that human planning processes themselves are not strictly search but rather composed of process levels in which goal manipulation is a prime process. Certainly, however, more evidence is necessary to clarify the exact nature and composition of human cognitive planning abilities.

Acknowledgements

I thank Chen Zhang for her work on the GTrans implementation and the collection of subject data. Much credit is also due to anonymous reviewers and their suggestions.

References

Cox, M.T. (2000). A conflict of metaphors: Modeling the planning process. In *Proceedings of the 2000 Summer Computer Simulation Conference* (pp. 666-671). San Diego: The Society for Computer Simulation International.

Cox, M. T., Edwin, G., Balasubramanian, K., & Elahi, M. (2001). Multiagent goal transformation and mixed-initiative planning using Prodigy/Agent. In *Proceedings of the 4th International Multiconference on Systemics, Cybernetics and Informatics, Vol. 7* (pp. 1-6).

Cox, M. T., & Veloso, M. M. (1997). Supporting combined human and machine planning: An interface for planning by analogical reasoning. In D. Leake and E. Plaza (Eds.), *Proceedings of the2nd International Conf. on Case-Based Reasoning* (pp.531-540). Berlin: Springer.

Cox, M. T., Elahi, M., & Cleereman, K. (2003). A distributed planning approach using multiagent goal transformations. In Ralescu (Ed.), *Proc. of the 14th Midwest Art. Intel. and Cognitive Sci. Conf.* (pp. 18-23). Cincinnati: Omnipress.

Cox, M. T., and Veloso, M. M. (1998). Goal Transformations in Continuous Planning. In M. desJardins (Ed.), *Proceedings of the 1998 AAAI Fall Symposium on Distributed Continual Planning* (pp 23-30). Menlo Park, CA: AAAI Press / The MIT Press.

Edwin, G. (2001). *COMAS: Coordination in multiagent systems.* Masters dissertation, Wright State University, Computer Science and Engineering Department, Dayton, OH.

Edwin, G., & Cox, M. T. (2001). Resource coordination in single agent and multiagent systems. In *Proc. of the 13th IEEE International Conf. on Tools with Artificial Intelligence* (pp. 18-24). Los Alamitos: IEEE Computer Society.

Ram, A., & Leake, D. (1995). Learning, goals, and learning goal. In A. Ram & D. Leake (Eds.), *Goal-Driven Learning*. (pp. 1-37). Cambridge, MA: MIT Press/Bradford Books.

Schank, R. C., Fano, A., Bell, B., & Jona, M. (1993). The design of goal-based scenarios. *The Journal of the Learning Sciences, 3*(4), 305-345.

Veloso, M. M. (1994). *Planning and learning by analogical reasoning*. Berlin: Springer.

Veloso, M. M., Carbonell, J. G., Perez, A., Borrajo, D., Fink, E., & Blythe, J. (1995) Integrating planning and learning: The PRODIGY architecture. *Journal of Theoretical and Experimental Artificial Intelligence. 7(1)*: 81-120.

Williamson, M. (1996). *A value-directed approach to planning*. Doctoral dissertation, University of Washington, Computer Science Department, Seattle.

Zhang, C. (2002). *Cognitive models for mixed-initiative planning*. Masters dissertation, Wright State University, Computer Science and Engineering Department, Dayton, OH. (Available at URL `www.cs.wright.edu/~mcox/GTrans/zhang2002.pdf`)