Towards Integrated Planning and Scheduling: Resource Abstraction in the Planning Graph

Terry L. Zimmerman and **Stephen F. Smith**

The Robotics Institute, Carnegie Mellon University
5000 Forbes Avenue, Pittsburgh PA 15213
{wizim, sfs}@cs.cmu.edu

Abstract

We outline a approach for closely integrating classical automated planning with scheduling in a manner designed to maximize the opportunity for leveraging the strengths of each. A fundamental capability needed to implement the system is an efficient resource-lifted planning graph, and we introduce the process for achieving it here. We report experimental evidence that the resource-abstracted model functions well and provides an effective means of constraining the entailed search space via propagation through the planning graph of the set of valid resource instances for each proposition.

1. Introduction

The work reported here constitutes an important first step upon which we intend to base a novel integrated planning/scheduling system. We are pursuing an approach to solving a set of problem goals situated in time that separates casual reasoning needed for selecting actions to achieve goals (ala classical planning) from the resource allocation and situating actions in time (ala scheduling) in order to exploit strengths of both methodologies. Such an integrated planning/scheduling system might function as as a stand-alone temporal planner in its own right, but would also be well-suited for exploiting automated planning techniques in the service of solving sub-problems within a larger scheduling problem scenario [Smith & Zimmerman, 2004].

Automated planning can be described as the process of determining what actions to take and in what order so as to progress from an initial world state to a desired goal state. For a planning problem, besides the goals and the initial state, the input must define a set of domain operators, describing for each operator the set of effects produced when the action executes in a state containing certain preconditions. Planning focuses on making decisions about action (operator) choices, where choosing an action typically leads to a cascading series of entailed choices and these choices may interact in an arbitrary variety of complex ways (e.g. action duration, consistency with previously assigned actions, resource requirements). In the course of selecting actions automated planners implicitly assign resources via the same algorithm.

Scheduling can be viewed as the problem of assigning limited resources to tasks over time to optimize one or more objectives. In a scheduling problem, input demands provide top-level goals (e.g., produce 10 widgets by next Friday, transport package p from location a to location b by Thursday), and one or more activities (typically known at the outset) must be executed to satisfy a given demand. Activities require resources to execute and the scheduler must reason about which of a set of available resources should be assigned to perform each activity. These allocation decisions may be subject to an arbitrarily complex variety of constraints, such as precedence, resource capacity, activity performance duration, and other auxiliary conditions (sub-goals) entailed by the assignment of a given resource.

Planners and schedulers solve different though related problems, but each brings particular strengths to bear based on common aspects of their problem class. As scheduling systems extend their capabilities to handling some forms of sub-goaling and planning systems begin addressing more complex temporal problems with explicit resources there is an obvious incentive to examine integrations aimed at overcoming the weaknesses inherent in the biases of each approach.

We are pursuing an integrated planning/scheduling system, which we will refer to as GOzone, that combines a variation on classical, non-temporal Graphplan (G) [Blum & Furst, 1997] with Ozone [Smith et.al. 1996] a mature scheduling framework that employs a robust simple temporal network (STN). The closely integrated planning/scheduling system first seeks a parallel plan by building and searching on a specialized form of Graphplan's planning graph data structure in which resource objects are represented by variables that can represent equivalent resource instances. When the graph is extended to the first level where a classical solution is found GOzone enters a closely integrated scheduling and planning phase. We anticipate exploring various scheduling strategies for this phase to both allocate resources to support plan actions and to situate them in time. The anticipated payoff in this architecture is that by separating the causal reasoning as cleanly as possible from the temporal and resource-allocation reasoning, we can exploit proven and well-understood approaches in each mode.

This paper reports on the design, implementation, and testing of a resource-lifted planning graph structure that is foundational to this closely integrated planning/ scheduling system. Resource lifting represents a key step towards wresting the implicit resource allocation process away from a planning algorithm so that the more specialized resource allocation capabilities of the scheduler can be brought to bear. Whereas the original planning graph structure is composed of alternating levels of grounded state propositions and actions, we have designed and implemented a constructor that creates a lifted version of the planning graph in which any object belonging to a class (or type) identified by the user as a resource is converted to an appropriate resource variable. These resource variables will appear in the context of the propositions that are preconditions and effects of domain actions such as *(at ?r1 Paris)* where *?r1* might correspond to 'airplane'.

We then constrain the search space in a sound and complete manner by propagating the resource instances present in the initial world state propositions forward through all actions that require them. This results in a range of resource values for each proposition containing a resource variable in a given planning graph level.

We motivate the role of this resource-lifted planning graph in enabling effective use of scheduling and characterize the reduced planning graph size that results as well as the tradeoff in loss of consistency-enforcing constraints (mutexes). In the remainder of the paper we will first outline the approach to integrated planning/ scheduling we envision for GOzone and the role played by a resource-lifted planning graph. We then describe the methodology developed for implementing the resource abstraction such that it will effectively support the scheduling methods that we anticipate using when the GOzone system is fully implemented. We subsequently provide preliminary empirical evidence of the performance of the resource-lifted planning graph both with and without propagating the resource instances. In the section on related works we outline other studies that have implemented resource-lifting, and then we conclude with a discussion of the larger picture for this work and outline the anticipated direction for full implementation of the GOzone integrated planning/ scheduling system.

2. A Proposed Approach to Integrated Planning-Scheduling

Before turning to the resource-lifting process that is the focus of this paper, we here provide a high level description of the GOzone system it will support as well as a particular class of temporal goal satisfaction problem to illustrate the approach.

2.1 Problem Description

The temporal, resource-based planning/scheduling problem of interest has the following features:

- A set of goals *Goals,* each tagged with its latest finish time (*lft*) and a release time, in the general case.
- A simple time, *typed* planning domain theory: *Ops* :Actions extended ala PDDL 2.1 to include durations, preconditions must hold true for the *entire* duration (not strictly necessary, but this simplifies the algorithm)
- A set of resources possibly situated in a hierarchy of classes. Each resource's timeline may contain prior commitments or be empty at the start of planning.
- A list of object types in the domain that constitute resources (either provided by the user and identified via an automated method [c.f. Fox & Long, 2000])
- Each initial state proposition and each action precondition and effect has at most one resource term. In addition each action references at most one resource (not strictly necessary, but we adopt it for simplicity).

2.2 Motivation for the Approach

Given a temporal, resource-dependent planning problem, we seek to partition the solution search, as cleanly as possible, into aspects that are handled well by classical planning methods and those best handled by resource allocation strategies and temporal constraint networks. In this spirit we propose to address action selection issues having to do with causal ordering and goal satisfaction via purely classical planning (i.e. no explicit representation of time) while handling problem aspects primarily associated with resource availability and non-causal temporal constraints via scheduling techniques. These aspects are in fact connected; as discussed below the connection can vary greatly depending on the problem. There has been some work in such approaches in the planning community and the question of how to effectively separate these aspects remains an open issue [Srivastava 2000].

Causal reasoning ensures that for every action in a plan, its enabling preconditions are satisfied by some effect of another action that temporally precedes it. By themselves, causal relationships enforce a necessary set of orderings amongst actions in order to satisfy problem goals. In addition, they provide some constraints for assessing concurrency among actions. Resource reasoning ensures that all the resources needed for execution of an action are available for allocation without any resource conflicts. In addition to the temporal aspects associated with resource availability, there are a variety of other temporal characteristics; action duration, exogenous events, goal deadlines and maintenance goals that modern scheduling approaches handle effectively.

The classical planning graph first introduced as the key structure for Graphplan [Blum & Furst, 1997] embodies a succinct set of constraints on the search space for it's regression search strategy. A large body of subsequent research has shown that it can be mined for powerful heuristics to guide both classical and temporal state-space planners as well as partial-order planners without conducting search directly on the planning graph itself [Bonet & Geffner, 1999; Nguyen & Kambhampati, 2000; Gerevini & Serina, 2002]. More recent work has shown that by employing a succinct search trace, the PEGG planner can extract from the planning graph near-optimal plans in times that are competitive with the heuristic state-space planners [Zimmerman & Kambhampati, 2005]. Moreover, having found one plan on a k-length graph, the search trace can be leveraged to quickly generate an arbitrarily large number of k-length plans [Zimmerman & Kambhampati, 2002]. Though we do not provide the details in this paper, we expect to exploit both heuristics derived from the planning graph and the capabilities provided by PEGG's search trace enhanced methodology.

2.3 Search Process Overview

The search for a temporal, resource-consistent solution is conducted in two search phases.

1) Phase 1: Finding a 'classical' solution. This does not consider temporal constraints or (most) resource constraints. Here the planning graph is incrementally extended and search until a valid classical solution to the problem is extracted.
2) Phase 2: Finding a temporal, resource-consistent solution. This can be conducted using one or more of the following strategies:
    i. Attempt to schedule the classical solution using available resources and the temporal network. If this fails, exploit PEGG's ability to quickly generate additional plans, rank them based on the conflicts responsible for the scheduling failure, and attempt to schedule in order.
    ii. Conduct a 'temporally mapped' search from actions on the classical planning graph to the Ozone temporal network and resource timelines in order to find a temporally valid solution. This process essentially performs regression search on the planning graph (employing a subset of its mutex constraints) but augments the constraint checking to include durations and resource availability. (Elaboration of this process is beyond the scope of this paper.)

    In the process of either of the phase 2 strategies, the PEGG search algorithm may terminate search on a k-length plangraph to seek a k+1 step plan based on search-trace based heuristics.

Significant experimentation effort is expected to revolve around the interesting possibilities for the phase 2 solution search.

3. Enabling the Scheduling Phase: a Resource-Lifted Planning Graph

Planners that exploit a planning graph generally employ grounded actions (with an instance of a given action for *each* different resource alternative) in order to maximize the degree of constraint propagation (via binary mutexes). In classical (non-temporal) planning, these constraints embodied in the planning graph serve to efficiently trim the search space explored during the regression search on the graph (even if only a 'relaxed plan' is sought for heuristic purposes). However, given actions with durations and goals with deadlines the role and effectiveness of such mutexes becomes problematic [Smith & Weld, 1999]. The GOzone architecture is such that it can benefit from the power of mutex constraints for constraining search on the classical planning graph in Phase 1 of its search, but if we search on a fully grounded graph the action selection process inherently ends up also allocating resources. This motivates a plangraph construction approach that abstracts any action resource requirements. One means of achieving this is to use a 'resource lifted' representation for propositions and actions -i.e. When an action can be executed using any of several identical resources, we represent all these instances with a single action in the plangraph. With this representation the allocation of a specific resource and its temporal positioning can be reserved for the scheduling methods.

3.1 Grounded vs. Lifted representations: The tradeoffs

There has been a fair amount of work associated with identifying resources and using lifted representations [Fox & Long, 2000, Srivastava, et. al. 2001] and there are well-known tradeoffs. For example, constraint propagation is typically less powerful in lifted representations, in part because fewer violations can be identified. In the plangraph context this translates as fewer identifiable mutexes and possibly more difficult implementation of memoization (nogood learning). Exploiting mutexes is a powerful means of trimming the search space for standard Graphplan, so with a lifted plangraph we would be interested in any means of regaining some of what has been lost. If we abstract just the resource requirements from action instantiation[1] we will lose all mutexes associated with concurrent use of the same resource and any propagation that might result. However the scheduling methods that attempt to allocate selected actions will, of course, enforce such constraints. and we expect this to be a winning tradeoff.

A side benefit to moving to a lifted plangraph: graph size and construction cost (and possibly the search space?) can be greatly reduced. There are many larger problems that become infeasible for plangraph-based planners due to the shear cost of constructing the graph alone, so this can be a significant benefit.

3.2 Resource Lifting Implementation

Given a set of user-specified resource types the planning domain theory and the problem initial state is preprocessed as outlined in Figure 1.[2] We assume here for simplicity that each action requires at most one resource and the user identifies resources and resource classes in the domain

[1] We don't lift such things as alternate locations for an action-i.e. we would still have separate actions in the PG for *fly(airplane? JFK LAK)* and *fly(airplane? JFK Paris)* .

[2] Resources do not generally appear in a problem goal state as the final state of a resource is rarely of concern to the user specifying a problem.

1. For each proposition, P_k, in the Init State that contains a term, obj_m belonging to a resource class R_n:
    a. Replace the term with a resource variable unique to R_n
    b. Add obj_m to the value range of P_k
2. Combine all Init State propositions that are identical after step 1 by adding their value ranges to a single instance of the proposition.
3. For each action, A_k, in the domain theory that contains a term, obj_m in a precondition or effect belonging to a resource class R_n: Replace the proposition term with a resource variable unique to R_n

Figure 1: Preprocessing the problem and domain constraints for resource lifting

definition. The initial state is generated from the grounded problem initial state by merging all propositions within a resource equivalence class. That is, if there are two or more propositions that are term-wise identical except for terms identified as equivalent resources they are replaced by a single init state proposition with the resource terms replaced by variables.

We introduce here what proves to be an effective efficiency; propagation of the resource "value set" forward from propositions in the init state through actions that use them as preconditions, to the effects of the actions. As outlined in Figure 2, the resource instances in the equivalence class for an init state proposition constitute the resource set of all valid level 1 actions having the proposition as a precondition, and are then *added* to the value set for each effect proposition of the actions. Note that more than one action may contribute to the value set for a given level 1 proposition. Propagation continues in this manner for subsequent levels.

The regression search then uses the action resource sets to constrain the number of resources to consider in possible instantiations of a lifted action. Only the resource instances appearing in an action's resource set at a given level are feasible at that point. This can be particularly effective in domains with mobile resources.

Resource variables in actions are generally *not* instantiated (grounded) during plangraph construction, but are bound to a specific resource only during the regression search.

4. Preliminary Experimental Results

With the GOzone system still under development, we report here on some experiments we've run to characterize the size and performance of the lifted planning graph. The standard and resource lifted planning graphs were built until level-off (the level where no new actions or propositions appear and no dynamic mutexes relax) for four problems in

1. Build the first plangraph action level in standard Graphplan fashion, adding actions for which consistent bindings between initial state propositions and action preconditions can be found.
    - Resource types must match for a resource variable to bind.
    - Propagate precondition resource variable value ranges to any appearance of the variable in an action effect.
    - Determine action-action mutexes in standard fashion with one exception:
        - Ignore mutexes tied to resource variables unless there's only one value in each variable's range
2. The first plangraph proposition level is created from the union of the initial state and effects of newly added actions.
    - Determine prop-prop mutexes in standard fashion with one exception:
        - Ignore mutexes tied to resource variables unless there's only one value in each variable's range
3. Combine all level 1 propositions that are identical after step 2 by unioning their value ranges in a single instance of the proposition.

Figure 2: Building a resource-lifted planning graph

different domains (3 of which are from the International Planning Competition). Table 1 summarizes the most important impacts on the planning graph; The *rocket-ext-a* problem has 2 rockets as resources and the lifted version of the graph is reduced by roughly 1/3 in size over the standard graph. In addition converting the resources to variables causes a 42% reduction in the number of binary mutexes (dynamic and static). The logistics problem from the AIPS-00 International Planning Competition (IPC) has 4 trucks and 1 airplane as resources. The ZenoTravel problem features 3 aircraft as resources. The Depots

Problem (domain)	**Reduction in Propositions**	**Reduction in Actions**	**Reduction in Mutexes**
rocket-ext-a	31%	27%	42%
logistics-10-0 (AIPS-00)	41%	30%	37%
zenotravel-3-7a (AIPS-02)	33%	25%	44%
depotprob7654 (AIPS-02)	52%	39%	50%

Table 1. Comparison of the resource lifted planning graph relative to the grounded graph

problem from IPC-02, the 2 trucks and 5 hoists were declared resources resulting in the largest reduction overall; 52% in the number of plangraph propositions over the standard graph and a 50% loss of mutex constraints.

The practical implications of these results are an expected reduction in memory demands for large problems and faster planning graph construction (In fact we found the lifted graphs for the modest size problems of Table 1 were constructed ~24% faster than the standard planning graphs). On the downside, the lost of the mutex constraints has the potential to translate into less constrained search on the planning graph.

In order to get a feel for the impact of the lifted resource implementation in the absence of a completed planner/scheduler, we solved the Table 1 problems on both the grounded planning graph (with standard Graphplan) and on the resource-lifted graph. "Solving" the problems in the latter case involves a 2-phase search process; upon extraction of a resource-abstracted set of plan actions a consistent, resource-grounded set of actions must be found based on the possible values (resource instances) for the resource variables in the abstract plan. If this 2nd phase fails then the 1st phase must be resumed and/or the planning graph expanded until a valid solution is found.

We implemented this 2-phase approach by adding a simple, naive backtracking search to the Graphplan-style regression search that tries all combinations of resource instances for abstracted actions in a plan until either a resource-consistent plan is found or the search is passed back to Phase 1.

Table 2 then, compares the runtime for extracting a valid solution on the grounded planning graph (using Graphplan) against two variations of resource-lifting; one without the value constraining propagation described in Section 3.2 and one with it active.[3] Note that all planning processes reported here will return a step-optimal plan. Not surprisingly the resource-lifted search processes take considerably longer than search on the grounded planning graph. This is attributable to at least 3 factors

- The lifted graphs have fewer mutexes to constrain the search space (see Table 1)
- The 2-phase approach needed to find a grounded solution on the lifted planning graphs is inefficient as configured.
- The problems all involve multiple search episodes before Graphplan extends the graph to a length with an extant solution. The phase 1 search on the resource-lifted planning graphs finds 'solutions' multiple times in these intermediate episodes and enters the phase 2 search to no avail for each such partial solution.

[3] The "enhanced" Graphplan reported here as well as the phase 1of the resource-lifted search processes all employ a number of efficiency enhancements relative to the original Graphplan, such as EBL and DDB (see [Zimmerman & Kambhampati, 2005]).

Problem (domain)	**Graphplan** (enhanced)	**Resource-Lifted: No Value Constr. Prop**	**Resource-Lifted: w/ Constr. Propagation**
rocket-ext-a	3.5	12.2	5.7
logistics-10-0 (AIPS-00)	30.0	144.	75.9
zenotravel-3-8a (AIPS-02)	972	1488	1001
depotprob7654 (AIPS-02)	32.5	81	49

Table 2. Runtime comparison (in sec) of solution extraction on two versions of the resource lifted planning graph relative to Graphplan's search on a grounded graph.

This is not, in itself, a serious concern since GOzone will focus on finding a temporally consistent solution and time spent in finding such a classical solution is likely to be a small fraction of the final solution search time. Moreover, the Ozone scheduler should be able to handily outperform the naive backtracking phase 2 search implemented for this simple study.

Of greater interest is the boost that comes from the resource value propagation introduced in Section 3.2. This augmentation reduces the solution search time by factors of from 1.5 to 2.1 over these problems. This advantage should increase for larger problems with more resources that are asymmetrically distributed.

5. Related Work

Perhaps the most relevant previous work is associated with the development of RealPlan [Srivastava et.al. 2001, Srivastava 2000],a classical planning framework in which resource allocation is decoupled from planning and handled via scheduling in a subsequent phase. The system employs a version of Graphplan to generate an initial classical plan on a resource abstracted planning graph and passes this to the scheduling phase, which treats the resource allocation as a Constraint Satisfaction Problem (CSP) RealPlan can implement either a master-slave or peer-to-peer relationship between the planner and scheduler components, with the latter mode employing feedback to the planner when scheduling fails to allocate the generated plan actions.

The resource-lifted planning graph employed by RealPlan is similar to the approach reported here in that resource variables are substituted for objects identified by the user as resources. However, the system does not support identifying a possibly hierarchical set of resource *types* that are then used to automatically create classes of lifted propositions. Nor does RealPlan conduct the the resource value set propagation that we introduced here as an efficient boost to constrain the search space. Perhaps most importantly, RealPlan is purely a classical planner and is not concerned with temporal planning or, alternately, scheduling resources over time.

4. Discussion and Conclusions

As indicated at the outset, our broader, long term objective is the development of an integrated planner/scheduler, which combines a Graphplan planning mechanism (operating on a classical planning graph representation) with a incremental, constraint-based scheduling procedure (operating on a simple temporal network representation). We refer to this envisioned framework as GOzone, reflecting our current implementation effort to integrate a variation of the Graphplan (G) planning algorithm [Zimmerman & Kambhampati 2005] with the Ozone scheduling framework [Smith et.al. 1996]. In GOzone, the planning graph representation provides a basic mechanism for enforcing the causal constraints implicit in the satisfaction of various pending goals, while resources are allocated to extracted activity sequences over time via posting of additional sequencing constraints in an underlying temporal network.

Our emphasis in this paper on lifted planning graph representations aims at separating these complementary concerns. Given the non-temporal nature of classical planning graphs, a grounded representation of resources always runs the risk of generating highly sub-optimal action sequences in domains where action durations vary non-trivially. A lifted representation, alternatively, allows the planner to concentrate on establishing necessary causal relationships between activities that must be executed and minimizes this possibility; reasoning about activity durations is instead deferred to the scheduler and carried out in the context of resource allocation decisions with respect to an explicit temporal model.

We see several advantages to our proposed planner/ scheduler configuration:

Hybrid problem solving - By partitioning concerns as just summarized, we exploit distinct planning and scheduling techniques where they are best suited and avoid the combinatorics of a single integrated search space and solver.

Natural handling of limited resource availability and exogenous events – From a planning perspective, accommodation of exogenous events that limit resource usage and activity execution typically requires special treatment. Such events are naturally accommodated and enforced within the temporal constraint network representations employed by contemporary constraint-based schedulers such as Ozone.

Flexible reasoning about resource configuration – From a scheduling perspective, satisfaction of state-dependent constraints on resource usage (e.g., being at the right location, being adequately warmed up, etc.) typically requires simplifying modeling assumptions, which eliminate non-circumscribed solution possibilities. The capability to manage such state-dependent constraints is flexibly accommodated in a general way via goal expansion by contemporary planning approaches such as Graphplan.

At the same time, realization of the GOzone framework presents several interesting research and development challenges:

Integrating "plan step" reasoning with durative actions – Despite the use of a lifted planning graph representation, significant questions remain with regard to how well activity sequences that are generated according to a "plan step" notion of time can be integrated into resource time lines that depend on the actual temporal extent of various activities. Our intuition, for example, is that propagation of resource ranges in the lifted planning graph can restrict search in meaningful ways, but this remains to be shown.

Dealing with deadlines and release times – Another related issue concerns the association of deadlines (or other time constraints) with various goals to be accomplished; they should clearly impact the generation of requisite activity sequences. We are pursuing an approach that uses activity duration during the backward search through the planning graph, but there are other possibilities that may prove more effective.

Incremental, continuous operation – One of the hallmarks of constraint-based scheduling techniques is their incrementality; to accommodate new activities over time while keeping previously scheduled activities in place, and to locally respond to constraint conflicts introduced by unexpected execution results. The Graphplan planning framework alternatively depends on moving forward from a pre-stated initial state. Another open issue concerns how this requirement can be reconciled with a continuously changing execution state and the desire to keep existing schedules in place to the extent possible. For example, once a schedule has been established, can this information somehow be fed back into the planning graph (or into the planning graph construction process) to constrain subsequent causal reasoning?

Our current research is aimed at answering these general questions.

Acknowledgments: The work reported in this paper has been supported in part by the National Science Foundation under contract # 9900298, by the Department of Defense Advanced Research Projects Agency under contract # FA8750-05-C-0033 and by the CMU Robotics Institute.

References

Blum, A. and Furst, M.L. 1997. Fast planning through planning graph analysis. *Artificial Intelligence.* 90(1-2). 1997.

Bonet, B. & Geffner, H. 1999. Planning as heuristic search: New results. In *Proceedings of ECP-99.*

Long, D. and Fox, M. 2000 "Automatic Synthesis and Use of Generic Types in Planning." *Proceedings of AIPS 2000*. 2000. pp. 196-205

Gerevini, A. & Serina, I., 2002. LPG: A planner based on local search for planning graphs with action costs. In *Proceedings of AIPS-02.*

Nguyen, X. & Kambhampati, S. 2000. Extracting effective and admissible state space heuristics from the planning graph. In *Proceedings of AAAI-00.*

Smith, D. and Weld, D. 1999. Temporal Planning with Mutual Exclusion Reasoning. *Proceedings of IJCAI-99.*

Smith, S.F., O. Lassila and M. Becker, "Configurable, Mixed-Initiative Systems for Planning and Scheduling", in *Advanced Planning Technology*, (ed.) A. Tate, AAAI Press, Menlo Park, CA, May, 1996, ISBN 0-929280-98-9.

Smith, S.F. & Zimmerman, T.L. 2004. "Planning Tactics within Scheduling Problems", Workshop on Integrating Planning Into Scheduling, ICAPS-04, June. 2004.

Srivastava, B., Kambhampati, S. and Do, M. 2001. Planning the Project Management Way: Efficient Planning by Effective Integration of Causal and Resource Reasoning in RealPlan. *Artificial Intelligence Journal, 131 (p.73-134)*, 2001.

Srivastava, B. 2000. Efficient Planning by Effective Resource Reasoning. PhD dissertation, Arizona State University.

Zimmerman, T., Kambhampati, S. 2002. Generating parallel plans satisfying multiple criteria in anytime fashion. In 6th Intl. Conf. on AI Planning and Scheduling, Workshop on Planning and Scheduling with Multiple Criteria,. April, 2002.

Zimmerman, T. and Kambhampati, S. 2005. "Using Memory to Transform Search on the Planning Graph", Journal of Artificial Intelligence Research, Volume 23, pages 533-585.