WORKING PAPER
ALFRED P. SLOAN SCHOOL OF MANAGEMENT

Planning with Resources

David A. Rosenblitt

CCSTR#117    SSWP# 3259-91-msa
Planning with Resources

David A. Rosenblitt

CCSTR#117    SSWP# 3259-91-msa

2/91
Abstract

The STRIPS-based (Fikes, 1971) propositional planning framework is extended for operators that consume and produce quantities of resources, which are ubiquitous in many real-world domains. In this framework, step preconditions must specify, in addition to a set of propositions, the amounts of resources that must be available in order for a step to be performed. Similarly, effects must specify the amounts of resources that are consumed or produced by a step. McAllester's formally precise, provably correct nonlinear propositional planner (McAllester, 1991) is extended for resource-manipulating operators, and is proved correct. The nonlinear planning with resources algorithm is implemented in a prototype system, Synapse.

1. Introduction

A planning problem may be represented as a triple \( <I, G, \{S\}> \), where \( I \) is the initial state, \( G \) is the goal state, and \( \{S\} \) is a set of steps. (The terms step, operator, and task are synonyms.) A linear plan consists of a totally ordered set of operators; a nonlinear plan consists of a partially ordered set of operators. In the STRIPS model (Fikes, 1971), which has been the basis of all classical planners (e.g., Sacerdoti, 1975, Tate, 1977, Vere, 1983, Chapman, 1987, Wilkins, 1988), each step has a set of preconditions and effects. Preconditions are propositions representing conditions that must be true before the step is performed. Effects are divided into an add-list, containing propositions that are made true by the step, and a delete-list, containing propositions that are made false by the step. The initial state, \( I \), specifies the initial truth values of propositions; the goal state, \( G \), specifies the desired final truth values of propositions.

Situation-dependent operators, or operators whose effects are dependent on the situation in which they are performed, are disallowed in most previous planners as the associated reasoning mechanisms become computationally intractable (Chapman, 1987). SIPE (Wilkins, 1988) is the notable exception, and provides a variety of heuristics for managing the complexity resulting from an expressive representation. However, if we restrict our focus to situation-dependent operators that consume and produce resources, which are ubiquitous in many real-world domains (e.g., construction planning, process planning), efficient and provably correct planners can be developed.

It is often natural to express goals (e.g., the desired quantity of some product) in terms of resources. The STRIPS representation may be extended as follows to
allow the specification of effects to consume or produce quantities of resources.\(^1\) Step preconditions now specify, in addition to a set of propositions, the amounts of resources that must be available in order for a step to be performed. Effects now specify the amounts of resources that are consumed or produced by a step. Similarly, the initial state, I, must now specify the initial resource allocations, and the goal state, G, must specify the desired final amounts of resources.

Most classical planners did not emphasize resource considerations, primarily because resource management was viewed as part of the scheduling problem, which was viewed as distinct from the planning problem. For example, ISIS (Fox, 1983), a job-shop scheduling system assumes that resources are allocated and tasks scheduled only after the task network has been planned. However, there are good reasons for including resource considerations during planning: 1) it is often natural to express some planning goals (e.g., the desired quantity of some product) in terms of resources; 2) resource constraints may determine whether a given planning problem is solvable; 3) resource considerations may be useful in comparing and evaluating alternate plans.

Few planners have a non-trivial planning with resources capability. SIPE (Wilkins, 1988) is the best-known example.\(^2\) SIPE provides a powerful constraint mechanism that can represent complex numerical constraints among resources. However, SIPE only detects one limited type of resource conflict: when a binary resource is required by possibly simultaneous steps. A binary resource may be in one of two states: available or unavailable. Binary resource conflicts in SIPE are always handled by step reordering, whereas in Synapse, it is also possible to address a conflict by increasing the supply of the resource.

Several other planning systems are able to resolve simple binary resource conflicts, similar to those resolved by SIPE (e.g., Corkill, 1979, Georgeff, 1983, Lee and Chung, 1989, Lee and Chung, 1989). Callisto (Sathi, Morton, and Roth, 1986), a project management system, resolves resource scheduling conflicts. von Martial (von Martial, 1990) also resolves resource scheduling conflicts, using a framework based on Allen's (Allen, 1984) temporal logic.

Section 2 describes how McAllester's nonlinear propositional planner (McAllester, 1991) may be extended for resource-manipulating operators. I used McAllester's planner as the starting point for Synapse because it is provably correct, and its search control mechanism is more efficient than TWEAK (Chapman, 1987).

---

\(^1\) Note that a "reusable" resource may be modeled as a resource that is both consumed and produced in equal amounts by every step that uses it. For example, a step may first "consume" a piece of equipment, and then "produce" it upon completion, making it available for use by other steps.

\(^2\) SIPE is one of the few planners to be successfully applied to practical problems of a non-trivial size (Wilkins, 1989).
Section 3 proves that Synapse is correct. Section 4 illustrates the planning with resources framework on an example taken from an actual construction project. Finally, Section 5 briefly discusses two extensions of the planning with resources framework that are likely to be useful in practice.

2. A Nonlinear Resource-Manipulating Planner

2.1. Nonlinear Planning Terminology

Definition. A nonlinear plan (for a planning with resources problem) consists of:

1) A set of nodes. The initial node I has no preconditions, and has the effect of asserting the propositions and allocating the resources specified in the initial state, and the final node G has no effects, and has preconditions specifying the propositions and resource quantities specified in the goal state (McAllester, 1991).

2) A partial order < on nodes, represented by a set of precedence constraints. If i < j then i must precede j and j must follow i, otherwise i might follow j and j might precede i. The initial node I must precede every other node, and the final node G must follow every other node.

3) A set of causal links.

4) A resource availability table, which indicates guaranteed resource availability (GRA) values (defined below) for each node with respect to each resource.

Definition. A causal link in a nonlinear plan is of the form <i, p, j> where i and j are nodes such that i must precede j, i asserts p, and p is a precondition of j. i is the establisher of j, the establishee.

Definition. An open precondition is a pair <p, j> where p is a precondition of j and there is no causal link of the form <i, p, j>.

Definition. A causal link <i, p, j> is unsafe if there is some clobberer k that deletes p which might occur between i and j.

Definition. A causal link <i, p, j> in a linear (totally ordered) plan is never unsafe. That is, there is no step k, such that k is between i and j, and k negates p.

Definition. A producer S of some resource r is a supplier for each consumer or utilizer C of r that must follow S.
Definition. A consumer $C$ of some resource $r$ is a competitor of any consumer $C'$ of $r$ that might follow $C$.

Definition. The guaranteed resource availability (GRA) of a node $S$ with respect to a resource $r$ is the amount of $r$ produced by the suppliers of $S$ minus the amount of $r$ consumed by the competitors of $S$.

Since the exact order of execution is unspecified in a nonlinear plan (i.e., it can't always be determined whether $i$ precedes $j$, or vice versa), the best we can do is compute a lower bound on the guaranteed minimum level of a resource store in a given state (whereas exact GRA values may be computed for a linear plan).

Definition. A resource deficit occurs when there is a consumer $C$ of some resource $r$ such that $\text{GRA}(C, r) < \text{PRECONDITION}(C, r)$, where $\text{PRECONDITION}(C, r)$ is the amount of $r$ required by node $C$.

2.2. Soundness and Completeness

The nonlinear planner maintains a list PARTIAL-PLANS that contains all of the candidate plans under construction. The following soundness lemma captures the notion of correctness for nonlinear resource-manipulating plans. The intuition is that all preconditions must be satisfied in a correct plan: propositional preconditions are satisfied when there is a safe causal link, and resource preconditions are satisfied when there is no deficit.

Soundness Lemma. If $P$ is a nonlinear plan with no open preconditions, unsafe causal links, or resource deficits, then any linearization $f$ of $P$ is a solution to the planning problem.

This soundness lemma removes the need for a soundness invariant on the search process. A nonlinear planner can be based on the following completeness invariant (McAllester, 1991).

Completeness Invariant. If $f$ is a minimal finite sequence of steps that solves the planning problem, then there exists some nonlinear plan $P$ in PARTIAL-PLANS such that $f$ is a linear interpretation of $P$.

Intuitively, $f$ is a linear interpretation of a nonlinear plan $P$ if $f$ has at least as many steps as there are nodes in $P$, such that every node in $P$ corresponds to a step in $f$, $f$ respects the partial order of $P$, and every causal link in $P$ has a corresponding causal link in $f$.

---

3 The GRA for nonlinear plans is essentially the resource-cliche truth criterion discussed by Chapman (Chapman, 1987).
**Definition.** A sequence of operations \( f = a_1; \ldots; a_n \) is a *linear interpretation* of a nonlinear plan \( P \) if there are at least as many steps in \( f \) as nodes in \( P \) and there exists an assignment of integers to the nodes of \( P \) such that (McAllester, 1991):

1) node \( I \) has number zero, node \( G \) has number \( n + 1 \) (where \( n \) is the length of the sequence \( f \)), and every other node is assigned a number between 1 and \( n \) inclusive.

2) the step \( a_i \) in the sequence \( f \) is associated with node \( i \) in \( P \).

3) the partial order of \( P \) is respected (and no two nodes have the same number). Thus, if \( P \) contains the precedence constraint \( i < j \), then \( a_i \) must precede \( a_j \) in \( f \).

4) every causal link \( <i, p, j> \) in \( P \) has a corresponding causal link \( <a_i, p, a_j> \) in \( f \). That is, if \( i \) is an establisher of \( j \), then \( a_i \) is an establisher of \( a_j \).

The search procedure runs by iteratively removing elements from PARTIAL-PLANS. When a plan \( P \) is removed from PARTIAL-PLANS, new plans are added to PARTIAL-PLANS, where each new plan is obtained by either adding a new precedence constraint to \( P \) or adding a new node and a new precedence constraint to \( P \). This removal process makes the plans that appear in PARTIAL-PLANS get longer. A breadth first search (in which PARTIAL-PLANS is treated as a queue) will eventually remove all the short plans and, assuming that some solution exists, the completeness invariant ensures that a solution will eventually be found.\(^4\)

### 2.3 The Nonlinear Planning Algorithm

The nonlinear planning with resources algorithm proceeds as follows:

1) Initialize PARTIAL-PLANS to the set \( \{P_0\} \), where \( P_0 \) contains only the nodes \( I \) and \( G \).

2) Remove some plan \( P \) in PARTIAL-PLANS.

3) Return \( P \) if it has no open preconditions, unsafe causal links, or resource deficits.

4) If \( P \) has unsafe causal links, explore all ways of eliminating these unsafe links as follows:

\(^4\) Korf (Korf, 1985) has shown that an iterative-deepening-A* search is significantly more space efficient (it uses linear space) than breadth-first-search (which uses exponential space). However, to simplify the description of the Synapse algorithms, I have used breadth-first-search.
a) Select some causal link \(<i, p, j>\) and a node C that deletes p and might be between i and j.

b) If C might precede i add the plan P' to PARTIAL-PLANS where P' is P with the additional precedence constraint \(C < i\) (this is called demoting the clobberer).

c) If C might follow j add the plan P' to PARTIAL-PLANS where P' is P with the additional precedence constraint \(j < C\) (this is called promoting the clobberer).

d) Go to step 2.

5) If P has open preconditions, explore all ways of eliminating these open preconditions as follows:

a) Select some open precondition \(<p, j>\).

b) For each node i in P that adds p and might precede j, create a new plan P' by adding the new causal link \(<i, p, j>\), and add P' to PARTIAL-PLANS (i.e. choose an existing node i, and make it the establisher of j).

c) For each step a that asserts p, create a new plan P' by adding a new node i that corresponds to step a, the precedence constraint \(i < j\), and the causal link \(<i, p, j>\), and add P' to PARTIAL-PLANS (i.e. add a new node i, and make it the establisher of j).

d) Go to step 2.

6) If P has resource deficits, explore all ways of reducing these deficits as follows:

a) Select a deficit at some node i with respect to resource r.

b) For each competitor node C of i, add the plan P' to PARTIAL-PLANS where P' is P with the additional precedence constraint \(i < C\) (i.e. choose a competitor C, and force it to follow i).

c) For each node S in P that produces r and might precede i, add the plan P' to PARTIAL-PLANS where P' is P with the additional precedence constraint \(S < i\) (i.e. choose a producer S, and force it to precede i, thus becoming a supplier of i).

d) For each step a that produces r, create a new plan P' by adding a new node S that corresponds to step a, and the precedence constraint \(S < i, \text{ and}\)
add P' to PARTIAL-PLANS (i.e. add a new producer S, and make it a supplier of i).

e) Go to step 2.

The algorithm initially adds new nodes at step 5, if there are any open preconditions, or step 6, if there are resource deficits and no remaining open preconditions. If adding a step causes an existing causal link to become unsafe, the algorithm will add precedence constraints at step 4 until all causal links are safe.

Whenever a plan is modified, the set of unsafe links, open preconditions, and GRA values (and therefore the set of resource deficits) may change. For example, if a new node is added, its preconditions will denote new open preconditions or resource deficits, unless they are satisfied by the initial state (the only node that is guaranteed to precede a new node). The new node may also cause new unsafe links by clobbering a propositional precondition of some other node, or resource deficits by competing for the same resource as some other node. Similarly, when a new precedence constraint i < j is added, open preconditions, unsafe causal links, and resource deficits may be eliminated (e.g., a deficit at node j will be reduced if i now supplies j, and a deficit at node i will be reduced if j no longer competes with i).

3. Proving Correctness

Nonlinear Completeness Theorem. The nonlinear planning with resources algorithm preserves the completeness invariant.

Proof. We need to show that after each iteration of the algorithm, there is a plan P in PARTIAL-PLANS such that some minimal solution f is a linear interpretation of P. I prove this by induction. The basis is established when PARTIAL-PLANS contains only the initial plan, P0, which consists of nodes I and G. The completeness invariant is satisfied, since any solution f must be a linear interpretation of this skeletal plan. Assume that PARTIAL-PLANS satisfies the completeness invariant at some point in the search. Now assume that the problem is solvable, and there exists a minimal length solution f = a1;...;an, such that f is a linear interpretation of some nonlinear plan P in PARTIAL-PLANS. We now need to show that if the nonlinear planning algorithm removes P from PARTIAL-PLANS, such that a solution f is a linear interpretation of P, it always adds back to PARTIAL-PLANS some plan P', an extension of P, such that f is also a linear interpretation of P'.

If P is not a solution, then either P has one or more unsafe causal links, which are made safe in step 4, or P has one or more open preconditions, which are satisfied in
step 5, or P has one or more resource deficits, which are reduced in step 6. These three cases are now examined in turn.

Step 4 selects an unsafe causal link <i, p, j> and clobberer C in P, and: 1) if C might precede i, it constructs an extension P containing the additional precedence constraint C < i (step 4b); 2) if C might follow j, it constructs an extension P" containing the additional precedence constraint j < C (step 4c), where nodes i, j, and C in P correspond to the steps ai, aj, and aC in f. Since f is a linear interpretation of P, it must contain a causal link <ai, p, aj> corresponding to causal link <i, p, j> in P. Since the semantics of causal links in linear plans requires that there cannot be a step ak in f, between aj and aj, such that ak negates p, then either aC precedes ai, and f must be a linear interpretation of P", or aj precedes aC, and f must be a linear interpretation of P'. Thus step 4 guarantees that f is a linear interpretation of an extension of P.

Step 5 selects an open precondition <p, j> in P, and constructs all possible extensions of P containing the causal link <i, p, j> such that i is some node already in P that asserts p (step 5b), or i is a new node corresponding to a step ai that asserts p (step 5c). Let h be the suffix of f beginning with aj, the step in f corresponding to the node j in P. Since f is a solution, the weakest precondition for h to achieve G must be satisfied. Therefore there must be some step aj before aj that asserts p, and either aj corresponds to a node i already in P, and f must be a linear interpretation of some plan constructed in step 5b, or aj does not correspond to a node already in P, and f must be a linear interpretation of some plan constructed in step 5c. Thus step 5 guarantees that f is a linear interpretation of an extension of P.

Step 6 selects a deficit at node i with respect to resource r, such that GRA(i, r) < PRECONDITION(i, r), and constructs all possible extensions of P containing the new precedence constraint i < C for each competitor C of i in P (step 6b), or containing the new precedence constraint S < i, where S is either some node already in P that produces r (step 6c) or a new node corresponding to a step aS that produces r (step 6d). Let h be the suffix of f beginning with aj, the step in f corresponding to the node i in P. Since f is a solution, the weakest precondition for h (or any suffix of f) to achieve G must be satisfied. Therefore, GRA(ai, r) >= PRECONDITION(ai, r), which implies that GRA(i, r) < GRA(ai, r), which implies that at least one of the following must be true:

1) There is some competitor aC in f corresponding to a node C in P, such that aC follows aj, but C might precede i in P (i.e. a competitor has been removed from competition in f, but not in P). In this case, f will be a linear interpretation of some extension constructed by step 6b that adds i < C.
2) There is some producer $a_S$ in $f$, corresponding to a node $S$ in $P$, such that $a_S$ precedes $a_i$ in $f$ but $S$ might follow $i$ in $P$ (i.e. a supplier exists in $f$, but is not guaranteed in $P$). In this case, $f$ will be a linear interpretation of some extension constructed by step 6c that adds $S < i$.

3) There is some producer $a_S$ in $f$ that does not correspond to a node in $P$, such that $a_S$ precedes $a_i$. In this case, $f$ will be a linear interpretation of the extension constructed by step 6d that adds the new node $S$ (which corresponds to $a_S$) and $S < i$.

Thus step 6 guarantees that $f$ is a linear interpretation of an extension of $P$.

Therefore steps 4, 5, and 6 of the planning with resources algorithm always construct some plan $P'$, an extension of $P$, such that $f$ is also a linear interpretation of $P'$. Thus the algorithm preserves the completeness invariant.

**Nonlinear Completeness Lemma.** *The nonlinear planning with resources algorithm is complete (i.e. it will find a solution, provided one exists).*

**Proof.** By the previous theorem, there is always some plan $P$ in PARTIAL-PLANS such that a minimal solution $f$ is a linear interpretation of $P$ (if a solution exists). The number of possible plan modifications at any given point in the search is always finite, since there are only a finite number of nodes to add, precedence constraints to add, or nodes to enlarge. Furthermore, the maximum number of nodes in $P$ is bounded by the number of steps in $f$, the number of precedence constraints in $P$ is also finite ($O(N^2)$ in the worst case, where $N$ is the number of nodes in $P$), and the number of rescale operations is bounded by the resource quantities (which are constrained to be integral) produced in $f$. Therefore, the planner must eventually find a solution plan $P_f$ such that $P_f$ solves the problem $<I, G, \{S\}>$ (provided such a solution exists), and the nonlinear planning algorithm is complete.

**4. An Example: Planning the Construction of a Warehouse and a School**

This section illustrates the process of plan construction for a plan to build a warehouse, based on an actual project (Barrie and Paulson, 1984). The Synapse planner (running on a Macintosh II with 8 Megabytes of memory) required 11.08 seconds of CPU time to run on this example.

The initial state of the warehouse construction plan specifies the initial allocations of four resources: 1 cement mixer, 2 cranes, 600 tons of steel, and 10000 tons of concrete. The goal state specifies that the following propositions relating to the
construction of the warehouse should be true: building finish complete, electricity installed, fire protection complete, roofing complete, and special floors complete. The tasks used in this planning problem include some generic construction tasks (e.g., lay concrete, roofing), as well as some tasks that are specific to the particular building under construction (e.g., start special floors, start special rooms). All of these tasks are described in (Rosenblitt, 1991).

Successive iterations of the planning algorithm will add (at step 5c, which addresses open preconditions) new task instances to establish the goal propositions. Each of these tasks, in turn, may have open preconditions, which require additional tasks to establish those preconditions, etc., until the plan shown below (computed by Synapse) in Figure 1 is obtained:
5. Extensions

The full Synapse implementation, described in (Rosenblitt, 1991), allows operators to consume and produce variable (rather than fixed) amounts of resources, as specified by consumption and production constraints. Consumption constraints associated with an operator schema specify how the amount of resource consumed depends on the level of some other quantity. Consumption constraints are analogous to function constraints in SIPE (Wilkins, 1988), where the value of a variable is constrained to be the value of some function. Production constraints specify limitations on a step's production capacity, which may be represented as restrictions on the amount of resource produced to either a discrete set of alternative values (e.g., discrete lot-sizing constraints) or a continuous range of alternative values (this is analogous to numerical range constraints in SIPE). Such generalized operators more closely model the behavior of real-world operations (e.g., in manufacturing).

Synapse also includes the capability for problem reformulation in response to unsolvable planning problems. Although many planners re-order goals to avoid conflicting interactions and reduce search (e.g., Dawson and Siklossy, 1977, Tate, 1977, Vere, 1985, Hayes, 1986, Wilkins, 1988, Drummond and Currie, 1989, Irani and Cheng, 1989, Cheng and Irani, 1989), they all fail when a solution cannot be found. When the planning framework includes quantities of consumable, reusable, and producible resources, incremental changes in production goals or initial resource allocations may yield a solvable problem.

Conclusion

I have described a provably correct planner that can reason about resources. Although most previous planners included only a limited capability for reasoning about resources, there are several advantages to including resource considerations when planning: 1) it is often natural to express some planning goals in terms of resources; 2) resource constraints may determine whether a given planning problem is solvable; 3) resource considerations may be useful in comparing and evaluating alternate plans.
Bibliography


Hayes, C., Using goal interactions to guide planning, Proceedings AAAI-87, Seattle, WA, 224-228.


McAllester, D.A., Systematic nonlinear planning (submitted to AAAI-91).


