WORKING PAPER
ALFRED P. SLOAN SCHOOL OF MANAGEMENT

Supporting Collaborative Planning:

The Plan Integration Problem

David A. Rosenblitt

CCSTR#116 SSWP#3258-91-msa

MASSACHUSETTS
INSTITUTE OF TECHNOLOGY
50 MEMORIAL DRIVE
CAMBRIDGE, MASSACHUSETTS 02139

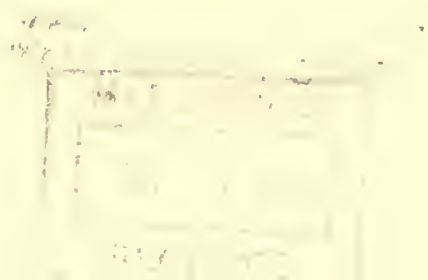
Supporting Collaborative Planning:

The Plan Integration Problem

David A. Rosenblitt

CCSTR#116 SSWP#3258-91-msa

2/91



Abstract

When different members of a work group develop their own individual plans, or sets of tasks to achieve desired goals, there may be conflicting and synergistic interactions among these plans. Conflicts may arise when one task negates the effect of another task, or two tasks compete for the same resource. Synergies may arise if the desired effects of some tasks are also accomplished by other tasks, allowing some of the tasks to be deleted. In many organizations, plans are often poorly integrated: conflict detection and resolution are performed late in the planning cycle, resulting in costly revisions and delays, and potential synergies are overlooked and unexploited, resulting in wasted resources.

This paper details a framework for solving the *plan integration problem*, and shows how this capability can support an important aspect of cooperative work: *collaborative planning*. The utility of plan integration in supporting collaborative planning is illustrated in a construction planning scenario, based on an actual project. The planning framework is domain-independent and provably correct. Unlike previous work in AI planning theory, it includes a general mechanism for reasoning about *resources*. The planning algorithms are implemented in Synapse, a prototype collaborative planning tool.

1. Introduction

When different members of a work group develop their own individual plans, or sets of tasks to achieve desired goals, there may be conflicting and synergistic interactions among these plans. In many organizations, plans are often poorly integrated: conflict detection and resolution are performed late in the planning cycle, resulting in costly revisions and delays, and potential synergies are overlooked and unexploited, resulting in wasted resources. Although existing computer-supported cooperative work tools may be used to support the development and execution of individual plans (Croft, 1988), no system to date is capable of integrating separately developed plans so that conflicts are resolved and synergies are exploited. This paper describes a framework for solving the *plan integration problem*, a capability that can support an important aspect of cooperative work: *collaborative planning*. Collaborative planning occurs when multiple agents iteratively develop and exchange their plans, as they discover and integrate conflicting and synergistic interactions. The planning framework is domain-independent, provably correct, and includes the capability for reasoning about *resources*. Most previous planners either ignored resources, or were only able to detect a limited type of resource conflict: when a binary reusable resource (i.e. a resource that is either available or unavailable) is utilized by possibly simultaneous steps.

This paper illustrates how a plan integration system can support one aspect of collaborative work: *collaborative planning*. An implemented prototype, Synapse, has been used to demonstrate collaborative planning in a scenario based on an actual construction project (Barrie and Paulson, 1984). In this scenario, two contractors develop plans to achieve their respective goals: the construction of a warehouse and a school. Each plan includes generic construction tasks, as well as tasks that are specific to warehouse or school construction. The following conflicting and synergistic interactions occur in this scenario:

1) A task in the warehouse plan competes with a task in the school plan for a shared resource. For example, both the roofing task in the warehouse plan and the precast walls task in the school plan may require use of the crane at the same time. This situation represents a conflict that may be resolved by imposing a precedence constraint between the two tasks, or by increasing the supply of the shared resource.

2) Although the initial supply of some resource is sufficient to meet the demand of either plan, it is insufficient to meet their combined demand. For example, there is enough concrete and steel on hand to construct either the warehouse or the school, but not both. This represents a conflict that may be resolved by increasing the initial allocation of the resource.

3) There is a setup task that is present in both plans and redundant. For example, there is a concrete setup task in both the warehouse and school construction plans, that need only be done once. This situation represents a potential synergy that may be exploited by deleting the redundant task from one of the plans.

4) A pair of tasks may be merged to achieve an economy of scale. For example, it may be cheaper or quicker to perform the lay concrete task in the warehouse plan simultaneously with the lay concrete task in the school plan. This situation represents a potential synergy that may be exploited by merging the two tasks into a single task that accomplishes the results of the original tasks.

Plan integration is useful in domains where the development of high-quality plans is important. Plan integration may be especially useful in domains characterized by long periods of plan development (e.g., process planning, construction planning), or repeated execution of plans in a relatively static environment (e.g., manufacturing). In contrast, plan integration techniques are not likely to be applicable in domains requiring a real-time response to a dynamically changing world. These domains require a reactive planning approach (Agre and Chapman, 1987, Georgeff and Lansky, 1987), where an intelligent plan execution module is crucial, and the planner itself may be of secondary importance, because assumptions made by a planner may be invalidated by later changes in the execution environment, and most planning problems are too complex to solve in real-time (Chapman, 1987). However, like Agre and Chapman (Agre and Chapman, 1988), I view plans as a reference guide that may be improvised upon, rather than as a script that is rigidly followed.

Previous hierarchical planners (e.g., NOAH (Sacerdoti, 1975), NONLIN (Tate, 1977), DEVISER (Vere, 1983, Vere 1985), and SIPE (Wilkins, 1988)) and multi-agent planners (e.g., Corkill, 1979, Rosenschein, 1982, Durfee and Lesser, 1987, and Georgeff, 1987) did not provide a general mechanism for reasoning about conflicting and synergistic interactions involving resources, as the notion of resource interaction has been restricted to *binary resource conflicts*. A binary resource may be in one of two states: available or unavailable. A conflict occurs when the resource is required by possibly simultaneous steps. Binary resource conflicts are handled by step reordering, whereas in Synapse, it is also possible to address a conflict by increasing the supply of the resource. 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; 4) there may be conflicts (e.g., a resource's aggregate demand exceeds its aggregate supply) and synergies (e.g., mergeable and redundant steps), resulting from resource interactions between plans.

Section 2 describes the plan representation that is used by the plan integrator, which is described in Section 3. Section 4 illustrates plan integration in a collaborative construction planning scenario, based on an actual project (Barrie and Paulson, 1984). Finally, Section 5 briefly describes future extensions to the plan integration framework.

2. Planning Representation and Terminology

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.

Definition. A *nonlinear plan* consists of (McAllester, 1991):

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.

---

[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.

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 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.[2]

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 GRA(C, r) < PRECONDITION(C, r), where PRECONDITION(C,r) is the amount of r required by node C.

3. A Plan Integrator

3.1. The Plan Integration Problem

The *plan integration problem* is the problem of combining two plans to resolve potential conflicts and exploit potential synergies, where the two plans are solutions to their own respective planning problems.

Definition. A *plan integration problem* $\langle P_A, P_B \rangle$ is the problem of combining two plans, P_A and P_B, which are minimal solutions to the planning problems $\langle I_A, G_A, \{S_A\} \rangle$ and $\langle I_B, G_B, \{S_B\} \rangle$, into an *integration* P_{int} that solves the problem $\langle I_{int}, G_{int}, \{S_{int}\} \rangle$, such that:

---

[2] The GRA for nonlinear plans is essentially the resource-cliche truth criterion discussed by Chapman (Chapman, 1987).

1) I_A and I_B are consistent. That is, they are either identical, or they describe different subsets of the initial state relevant to their respective planning problems. In the latter case, I_{int} will be the union of the propositions and resource quantities specified in I_A and I_B.

2) G_{int} includes the aggregate goals of G_A and G_B: the conjunction of the propositional goals of G_A and G_B[3], as well as the aggregated production goals of G_A and G_B. (i.e. if both G_A and G_B specify lower bounds on the production of some resource, then the corresponding lower bound in G will be the sum of the lower bounds in G_A and G_B).

3) $\{S_{int}\} = \{S_A\} \cup \{S_B\}$. The steps available in the integration problem include the steps available in the individual planning problems.

Definition. A plan integration problem $<P_A, P_B>$ is *solvable* if the planning problem $<I_{int}, G_{int}, \{S_{int}\}>$ is solvable.

Definition. A plan integration problem $<P_A, P_B>$ is *satisfiable* if planning problem $<I_{int}, G_{int}, \{NODES_{AB}\}>$ is solvable, where $\{NODES_{AB}\}$ is the union of the sets of nodes contained in P_A and P_B.

Thus the concept of satisfiability is more restrictive than the concept of solvability, since the choice of nodes to be added to a plan is restricted to a smaller set.

3.2. Conflicts and Synergies

The plan checker detects potential conflicts and synergies between two plans. Two types of conflicts[4] may arise during plan integration: unsafe causal links, if a step in one plan negates a precondition of a step in the other plan, and resource deficits, if a step in one plan competes for the same resource as a step in the other plan. Synergies arise due to *redundant* nodes, or nodes whose desired effects are achieved by other nodes in the plan.

Definition. A node i is redundant if:

1) There is an *alternate establisher* e for each causal link <i, p, j> that i establishes such that:

   a) e asserts p

   b) e might precede j

   c) for each clobberer node k, such that k is possibly before j and k negates p: either e might follow k, or k might follow j.

2) For each resource r produced by i and each consumer C of r that must follow i, it is possible to reorder or rescale nodes as follows to avoid a deficit of r at C (i.e. if i is deleted, then no additional deficits will be incurred with respect to any resources it produces):

   a) adding precedence constraints of the form C < C' (where C' is a competitor of C with respect to r)

---

[3] If G contains a contradiction (i.e. G_A contains the negation of a propositional goal of G_B), then the plans A and B are unintegrable, and one of the contradictory goal propositions must be removed from G_A or G_B in order to enable successful integration.

[4] Open preconditions cannot be caused by interactions between two plans.

b) adding precedence constraints of the form S < C (where S is a producer of r)

c) rescaling (i.e. increasing the production of) some supplier S of C

If a node meets the above requirements, then all of its desired effects may be achieved by other nodes in $\{NODES_{AB}\}$.

For example, in Figure 1 below, both i and k are redundant because both nodes assert p and might precede j.

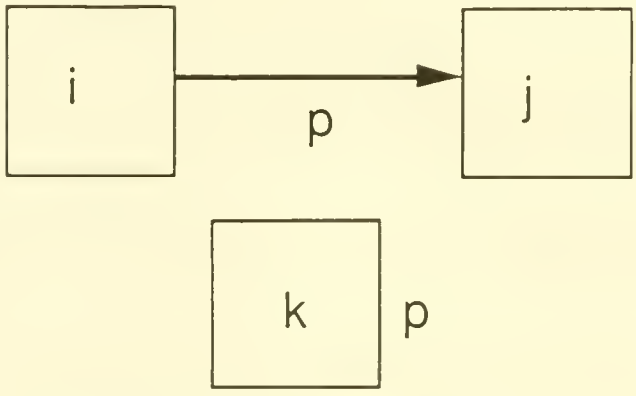
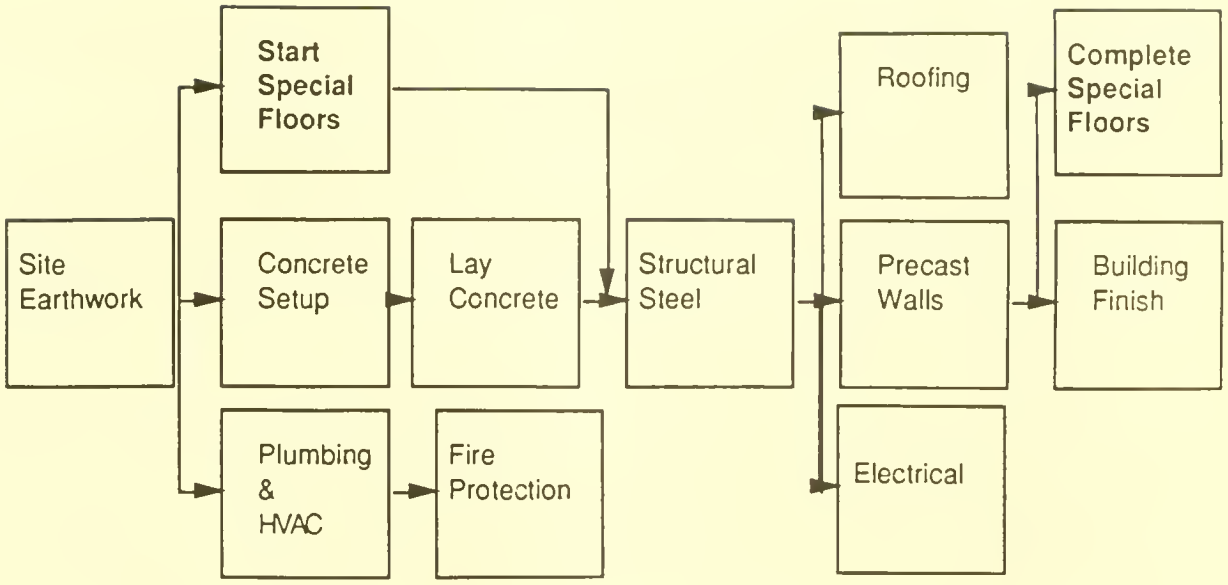


FIGURE 1: Two Redundant Nodes

Redundant nodes commonly occur when two plans each contain a setup operation that only needs to be performed once. Other common examples include any pair of nodes that are different instantiations of the same operator schema. For example, both the warehouse and school plans shown below in Figures 2 contain several instances of generic construction tasks, such as roofing, lay concrete, etc. Other examples might include tool or machine setup operations in manufacturing, telescope positioning operations in astronomical observation applications (e.g., Johnston, 1990), etc.



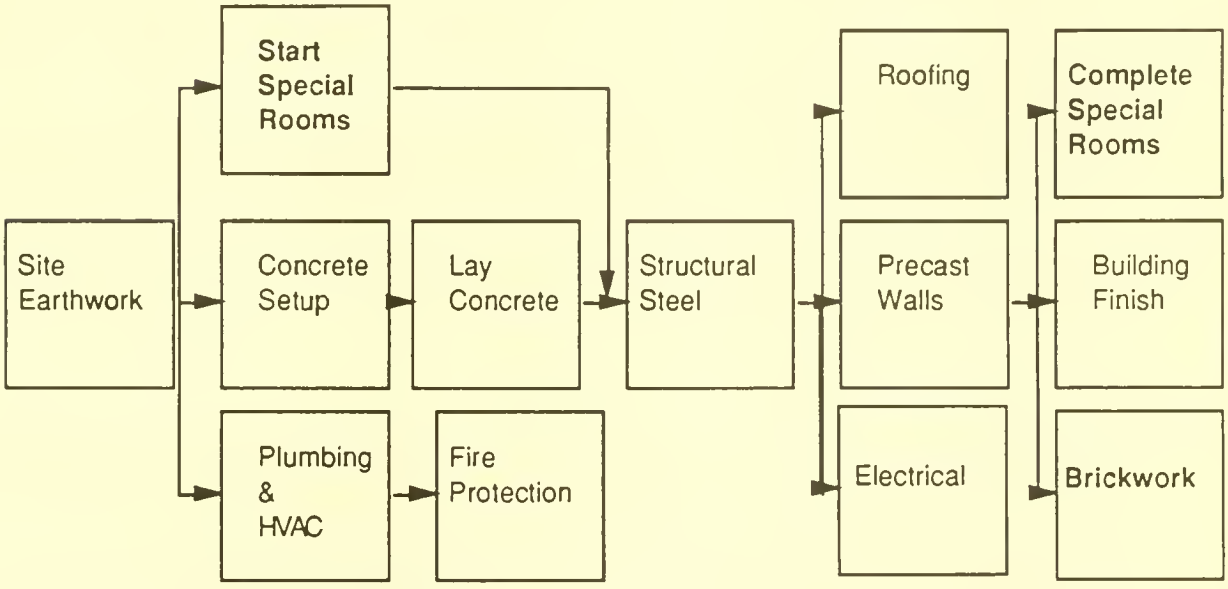


FIGURE 2: The Warehouse and School Plans

3.3. Two Approaches to Plan Integration

This section describes two approaches to plan integration: one provably correct, and the other heuristic. Nonlinear plan integration is useful in supporting collaborative planning for two reasons: 1) the steps in a nonlinear plan are partially ordered (rather than totally ordered, as in a linear plan), thus allowing a natural representation of the parallelism

inherent in most collaborative plans, and 2) nonlinear planning is more efficient than linear planning (McAllester, 1991).

3.3.1. A Provably Correct Plan Integrator

The plan integrator is identical to the provably correct nonlinear planner described in (Rosenblitt, 1991), except that the only steps that may be added to a plan are those that correspond to some node in the original plans (i.e. any step that corresponds to a node in the set $\{NODES_{AB}\}$). (The nonlinear planner, in contrast, can add any step in the domain.) The nonlinear planner is guaranteed to find the globally minimal length solution, whereas the integrator is not, but will instead find the "locally" minimal length solution with respect to the steps represented in $\{NODES_{AB}\}$. If the number of nodes in this solution is less than the number of nodes in $\{NODES_{AB}\}$, then one or more of the nodes in $\{NODES_{AB}\}$ were *redundant*. The integrator will miss a globally minimal solution P_{MIN} if P_{MIN} requires one or more steps that do not correspond to any node in $\{NODES_{AB}\}$. For example, consider the problem of integrating two plans for producing different types of car bodies, and assume that both plans build the car body out of steel. It may be the case, however, that in the globally minimal plan, the car bodies must be built from plastic, using some steps that are not present in either of the original plans. The integrator will fail to find this globally minimal plan, since it only considers steps that are represented in the original plans. In contrast, the nonlinear planner would find this global minimum, since it considers all possible steps in the domain.

The integrator is incomplete, because it may fail to find a solution even if the integration problem (i.e. the planning problem $<I_{int}, G_{int}, \{S_{int}\}>$) is solvable. This is because an integration problem may be solvable, but not satisfiable,[5] and the integrator only finds a solution if the problem is satisfiable. We conjecture that in order for an integrator to be complete (i.e. find a solution whenever the integration problem is solvable), the integrator must embed the full capabilities of a planner.

Plan integration is appropriate if there is some nontrivial investment in the original plans, and thus the only desirable changes are those that exploit synergies and avoid conflicts between the plans (by deleting redundant nodes, reordering, and rescaling nodes). In contrast, planning "from scratch" is appropriate if the best plan for the combined problem is sought, even if that requires significant modifications (e.g., substitution of nodes) to the original plans.

3.3.2. A Heuristic Plan Integrator

Another approach to plan integration is to start out with both of the original plans placed in parallel, P_A and P_B, and to explore ways of deleting, reordering, or increasing the production of nodes until no conflicts or redundant nodes remain.

Definition. The *parallel combination* P_{AB} of two plans P_A and P_B, is a plan consisting of the nodes, precedence constraints, and causal links of plans P_A and P_B.

An integrator could proceed as follows:

---

[5] It may be the case that the integration problem can only be solved by the addition of some node that does not correspond to any node in the original plans.

1) Compute P_{MIN}, a subplan of P_{AB}, by successively deleting redundant nodes[6] from P_{AB} until no redundant nodes remain

2) Invoke the planner to explore all ways of extending P_{MIN} into a solution, with the restriction that no new nodes be added.

I conjecture that this approach will yield a solution if exploiting a redundancy does not affect the resolvability of a conflict.[7] In some cases, this heuristic integrator may be more efficient than the provably correct integrator.

4. A Collaborative Planning Example: Checking and Integrating Plans to Construct a Warehouse and a School

This section illustrates the conflicts and redundancies that may arise when two plans (shown previously in Figure 2) are checked, and how these conflicts and synergies may be integrated. A collaborative construction planning scenario is used here for illustrative purposes. Synapse (running on a Macintosh II with 8 Megabytes of memory) required 2.95 seconds of CPU time to check the two plans, and 37.75 seconds to integrate the two plans. Note that plan checking and integration may be performed centrally, by an agent with access to all of the collaborating agents' plans, or in a decentralized fashion, where agents exchange their plans as needed. The decentralized approach is more flexible, while the centralized approach can detect some conflicts that may go otherwise unnoticed (e.g., a precedence loop spanning several plans).

4.1. Checking the Plans

This section discusses the conflicts and redundancies between two plans: a plan for building a warehouse, and a plan for building a school, shown previously in Figure 2. The tasks that are specific to either the warehouse or school project are in boldface (the other tasks are generic construction tasks).

All of the conflicts are resource deficits, and fall into four categories:

1) The roofing, precast walls, and structural steel tasks in both plans compete for the 2 cranes allocated in the initial state (the aggregate demand is for 5 cranes).

2) The structural steel tasks in both plans compete for the 600 tons of steel allocated in the initial state (the aggregate demand is for 748 tons of steel).

3) The lay concrete and concrete setup tasks in both plans compete for the cement mixer allocated in the initial state (the aggregate demand is for 2 cement mixers).

4) The lay concrete tasks in both plans compete for the 10000 tons of concrete allocated in the initial state (the aggregate demand is for 11600 tons of concrete).

---

[6] This choice could be informed by some heuristic that evaluates the cost of the node, and estimates the distance to the goal (e.g., how the duration will increase, if time is a priority, what the impact of any foreseeable rescales might be, etc.).

[7] I conjecture that this condition would be satisfied in many domains, but this is an empirical question. If this condition is not satisfied, an integrator that explores all possible ways of deleting *non-essential* nodes (i.e. nodes that do not appear in every integration) from P_{AB} could be used.

The redundancies are due to pairs of tasks that appear in both plans and may potentially be *mergeable* (i.e. one task in the pair may be deleted, while the other may be rescaled to achieve the desired effect of both tasks). The potentially mergeable tasks are concrete setup, roofing, fire protection, electrical, building finish, precast walls, structural steel, plumbing & hvac, site earthwork, and lay concrete.

4.2. Integrating the Plans

The integrated plan (computed by Synapse) is shown below in Figure 3. The rescaled nodes are indicated in boldface.

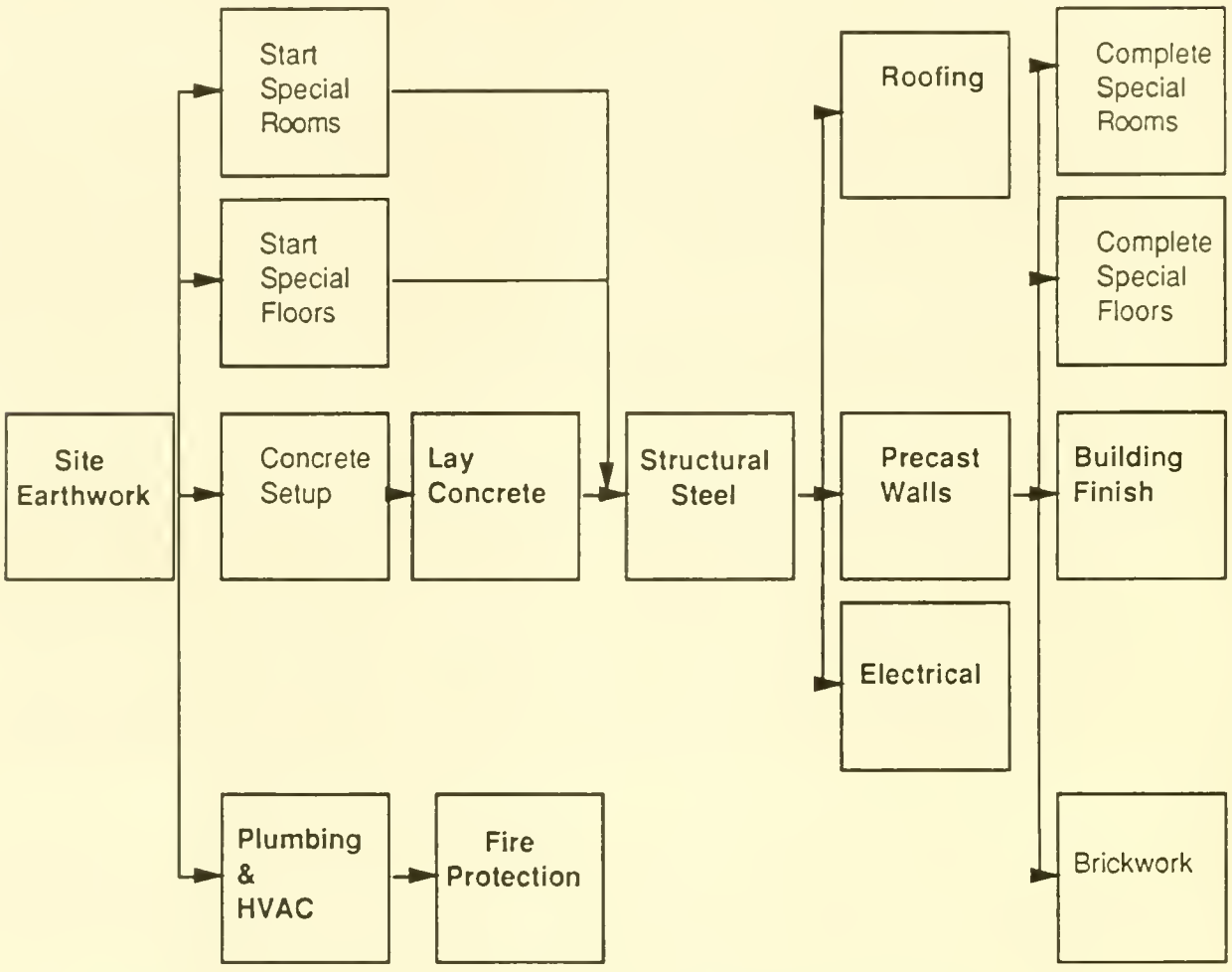


FIGURE 3: Integrating the Warehouse and School Plans

The conflicts that arose in Figure 2 have been eliminated in the integrated plan, since there are fewer tasks competing for the reusable resources (i.e. the cranes and cement mixer).

The initial resource allocations were increased by the *problem reformulation mechanism* (for details see (Rosenblitt, 1991)). It turns out that the two plans were *unintegrable*, since the steel and concrete deficits were *unproducible resource deficits*, which could only be resolved by increasing the initial allocations of steel and concrete from 10000 tons of concrete and 600 tons of steel to 11600 tons of concrete and 748 tons of steel.

5. Future Extensions to Synapse

Synapse is implemented in MacIntosh Allegro Common Lisp, and runs as an Object Lens application (Lai, Malone, and Yu, 1988). Object Lens provides a sophisticated user interface, including a powerful object-oriented template editor and hypertext links between objects.

This section briefly discusses several possible extensions to Synapse that may be useful in practice.

5.1. Reasoning About Time

Various temporal reasoning and scheduling capabilities could be added to Synapse (e.g., Vere, 1985, Sathi, Morton, and Roth, 1986, Miller, Firby, and Dean, 1987, von Martial, 1990). For example, Synapse currently assumes that a task needs to reserve each utilized resource for its entire duration. However, it may be the case that a task can it relinquish its reservation before the task completes, or acquire its reservation after the task has begun. The task representation could be extended to allow the specification of a reservation interval, whose size could be related to the levels of other resources by consumption constraints.

5.2. Replanning

A practical planning system would also need some sort of *replanning* capability to adapt a plan to a changing environment. For example, if the current state or goal is modified, some tasks may be redundant, since their effects are no longer required, while new tasks may be required to achieve new preconditions or goals. Synapse could be extended to incorporate techniques used in PRIAR (Kambhampati, 1989), which adapts an existing plan to a new problem, as well as in SIPE (Wilkins, 1988), where a plan is adapted to changes in the execution environment.

5.3. Hierarchical Planning

Organizations can be said to plan in a hierarchy of abstraction spaces (Sacerdoti, 1974), where progressively finer levels of detail are introduced at progressively lower levels of the organizational hierarchy. A task at one level of abstraction may be viewed as a planning problem at a more detailed level of abstraction, which may be solved by a plan consisting of a set of subtasks. For example, a car production task may be decomposed into a plan consisting of subtasks to produce and assemble various car components (e.g., engine, chassis, transmission). Conversely, a planning problem may be collapsed into a task whose prerequisites are specified in the initial state, and whose results are specified in the goal state.

A major challenge in such a scenario is to support *hierarchical integration*, by maintaining a consistent task and plan hierarchy in the presence of change. Tasks will inevitably be modified, causing *hierarchical conflicts*:

1) A plan containing a modified subtask no longer solves its planning problem

2) A subplan no longer solves a modified planning problem (due to a modified task)

Changing a single task may cause conflicts to propagate up and down the task/plan hierarchy. Similarly, the following *hierarchical synergies* are possible:

1) A modified task now has a redundant subtask.

2) A modified task now has a subtask that produces excess quantities of resources

It should be relatively straightforward to extend Synapse to detect and integrate such hierarchical conflicts and synergies. Kambhampati's plan reuse framework (Kambhampati, 1989), where an existing plan is adapted to a new problem, and SIPE's replanning capability (Wilkins, 1988), where a plan is adapted to changes in the execution environment, would also be relevant to hierarchical integration in the presence of changing tasks.

5.4. Synapse Agents

Object Lens agents (Lai, Malone, and Yu, 1988) that invoked the plan checker or integrator could be triggered by certain events. One possibility is that when a user obtains a new plan (e.g., upon receipt of an electronic mail message from a colleague containing a hypertext link to a new plan) the plan checker could check the incoming plan with respect to a pre-specified existing plan. For example, the mail message might describe the warehouse construction plan, and the recipient of the message might be the developer of the school construction plan. Synapse agents could also be triggered by changes to a plan, or changes to a task contained by a plan.

Conclusion

I described a plan integrator that can reason about conflicting and synergistic interactions between plans. Synapse is the first planner to provide a general, provably correct mechanism for detecting and integrating conflicting and synergistic interactions involving 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 (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; 4) there may be conflicts (e.g., a resource's aggregate demand exceeds its aggregate supply) and synergies (e.g., mergeable and redundant steps), resulting from resource interactions between plans. Plan integration is appropriate if there is some nontrivial investment in the original plans, and thus the only desirable changes are those that exploit synergies and avoid conflicts between the plans. In contrast, planning "from scratch" is appropriate if the best plan for the combined problem is sought, even if that requires significant modifications (e.g., substitution of nodes) to the original plans.

Bibliography

Agre, P.E. and Chapman, D., Pengi: An implementation of a theory of activity, *Proceedings AAAI-87*, Seattle, WA, 268-272.

Agre, P.E. and Chapman, D., What are plans for?, MIT AI Lab Memo 1050, September, 1988.

Barrie, D.S. and Paulson, B.C., *Professional Construction Management*, McGraw-Hill, 1984.

Chapman, D., Planning for conjunctive goals, *Artificial Intelligence* **32** (1987) 333-377.

Corkill, D.D., Hierarchical planning in a distributed environment, *Proceedings IJCAI-79*, Cambridge, MA, 168-175.

Croft, W.B., Using a planner to support office work, *Proceedings of the Conference on Office Information Systems*, Palo Alto, CA, 1988, 55-62.

Durfee, E.H. and Lesser, V.R., Using partial global plans to coordinate distributed problem solvers, *Proceedings IJCAI-87*, Milan, Italy, 875-883.

Fikes, R.E., and Nilsson, N.J., STRIPS: A new approach to the application of theorem proving to problem solving, *Artificial Intelligence* **2** (1971) 198-208.

Fox, M., Constraint-directed search: A case study of job-shop scheduling, (Ph.D. thesis), Tech. Report, Computer Science Dept., Carnegie-Mellon University, Pittsburgh, PA, 1983.

Georgeff, M.P., Planning, *Annual Review of Computer Science*, Volume 2: 359-400, 1987.

Georgeff, M.P. and Lansky, A.L., Reactive reasoning and planning, *Proceedings of Sixth National Conference on Artificial Intelligence, 1987*, 677-682.

Johnston, M.D., SPIKE: AI Scheduling for NASA's Hubble Space Telescope, *Proceedings of the IEEE Conference on AI Applications*, 1990.

Kambhampati, S., Flexible reuse and modification in hierarchical planning: A validation structure based approach (Ph.D. thesis), Center for Automation Research, University of Maryland, College Park, MD, 1989.

Lai, K.Y., Malone, T.W., Yu, K.C., Object Lens: A "spreadsheet" for cooperative work, *ACM Transactions on Office Information Systems*, Volume 6, Number 4, October 1988, 332-353.

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

Miller, D., Firby, J., and Dean, T., Deadlines, travel time, and robot problem-solving, *Proceedings IJCAI-85*, Los Angeles, CA.

Rosenblitt, D.A., Supporting collaborative planning: The plan integration problem (Ph.D. thesis), Department of Electrical Engineering and Computer Science, M.I.T., Cambridge, MA, January, 1991.

Rosenschein, S.J., Synchronization of multiagent plans, *Proceedings AAAI-82*, Stanford, CA, 115-119.

Sacerdoti, E.D., Planning in a hierarchy of abstraction spaces, *Artificial Intelligence* **5** (1974) 115-135.

Sacerdoti, E.D., *A Structure for Plans and Behavior* (American Elsevier, New York, 1977), also SRI AI Tech. Note 109, Menlo Park, CA, 1975.

Sathi, A., Morton, T.E., and Roth, S.F., Callisto: An intelligent project management system, *AI Magazine*, Winter, 1986, 34-52.

Tate, A., Generating project networks, *Proceedings IJCAI-77*, Cambridge, MA, 888-893.

Vere, S.A., Planning in time: Windows and durations for activities and goals, *IEEE Trans. Pattern Anal. Mach. Intell.* **5** (3), 1983, 246-267.

Vere, S.A., Splicing plans to achieve misordered goals, *Proceedings IJCAI-85*, Cambridge, MA, 1016-1021.

von Martial, F., A conversational model for resolving conflicts among distributed office activities, *Proceedings of the Conference on Office Information Systems*, Cambridge, MA, 1990, 99-108.

Wilkins, D.E., *Practical Planning: Extending the Classical AI Planning Paradigm*, Morgan Kaufmann, San Mateo, CA, 1988.

Wilkins, D.E., Can AI planners solve practical problems?, SRI Tech. Note 468R, Menlo Park, CA, 1989.