

A Decision Making Procedure for Collaborative Planning

Timothy W. Rauenbusch
Division of Engineering and Applied Sciences
Harvard University
Cambridge, MA 02138 USA
tim@eecs.harvard.edu

ABSTRACT

A team of agents planning to perform a complex task make a number of interrelated decisions as they determine precisely how that complex task will be performed. The decision set includes the choice of recipes and parameters, and the determination of responsibilities for each agent. The paper formally defines a search problem with search operators that correspond to the team planning decisions. It defines rationality for a collaborative team in a way that ensures that a team will abandon performance of a complex task if it is ultimately in its economic best interest to do so. Algorithms for making group decisions by solving the search problem support interleaving planning and acting. The algorithm respects the constraints on mental states specified by the SharedPlans theory of collaboration and links the solution to the search problem to the mental states of agents.

1. INTRODUCTION

Agents working together as a team can accomplish tasks that no agent on the team is able to perform independently. The act of forming a team to accomplish a task is necessary but not sufficient to ensure success. Not all teams are equally effective. For instance, consider two teams of volunteers building a wall of sand bags to fortify a sea wall before a flood. One team can unload 30 bags of sand per minute when working together by passing bags from hand to hand. The other team has each volunteer running back and forth from the truck to the shore unloading a bag at a time, and unloads only five bags per minute. Each team has the same goal of averting a flood, but the first team is more likely to achieve its goal because it has been smarter about planning how its members will work together.

The problem faced by a team G that has agreed to perform action¹ A is distributed and complex. It is distributed

¹Extensions to allow the theory to support teams that wish to perform one of a set of actions are supported by minor modifications to the model and are omitted to simplify the

because the information relevant to making an effective decision is not possessed by any individual, and it is not practical to transfer all relevant information to one central point.

It is complex because there is a set of of three broad types of interrelated decisions that grows exponentially with the complexity of A . The three types of decisions are: (a) recipe selection, (b) parameter binding and (c) assignment. Recipe selection decisions relate to choosing one of the possible ways of performing A . Parameter binding is the process of further refining an act type taken from a recipe into a concrete action to be performed (e.g., specifying when action execution will begin). Assignment requires a subset of the agents on the team to be assigned responsibility for each action. Complexity results because the decisions are interrelated and because each subact of the recipe selected for A may be further decomposed, opening up new sets of decisions that need to be made by its responsible (sub)team. Furthermore, these new sets of decisions for subacts are interrelated with the decisions for the parent action A .

This paper provides an algorithm for making the three types of interrelated decisions by recasting the problem as a search problem. To do so, it first defines proposal trees, a new data structure that is similar to Plan Trees [4], but specifically built to represent the team decision making problem. It then defines a search space and provides a precise definition for three search operators: selecting a recipe, binding parameters, and assigning responsible agents. The definition of *execution* provides the link between proposal trees and the mental states of collaborative agents [2].

The objective function for the search is based on an extension of economic individual rationality to team rationality. One major benefit of team rationality is that it requires that a team abandon its plan to perform an action if the costs of performing that action prove to be too high. Planning may be interleaved with acting and the costs of actions that have been performed are treated in a rational way when planning present and future actions, based on the economic notion of sunk costs.

The next section provides an example of a team that is planning to perform a task in the victim rescue domain. Section 3 defines a proposal tree in terms of the relevant terminology from the multi-agent planning literature. Section 4 casts multi-agent decision making for performing an action

presentation.

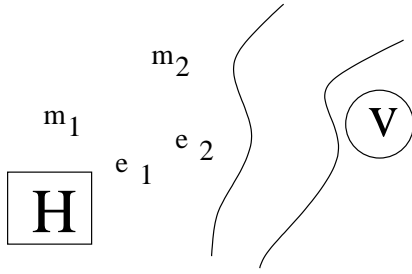


Figure 1: Four-agent rescue domain

Recipe Library

```

rescue-victim(ST,ET,east-bnk) {
  move-to-loc(ST1,ET1,east-bnk)
  assess-victim(ST2,ET2)
  transport-victim(ST3,ET3,
    east-bnk,hospital)
  Constraints: (ST1>=ST,ST2>=ET1,
    ST3>=ET2,ET>=ET3)
}
...

```

Skills

```

swim-across-river
take-vital-signs
...

```

Figure 2: Selected recipes and skills of Agent m_1

as a search problem over proposal trees. Section 5 defines team rationality and the objective of the search problem. Section 6 situates our work in relationship to the relevant literature.

2. VICTIM RESCUE

The rescue domain is illustrated in Figure 1. Four agents (two medics and two engineers) labeled e_1, e_2, m_1 and m_2 and a hospital (marked ‘H’) are located on the west bank of a river. A victim (labeled ‘v’) lies on the east bank and awaits medical attention and transportation to the hospital. The agents form a team with the goal of rescuing the victim and transporting her to the hospital.

Each of the agents has a different recipe library and set of skills. Figure 2 shows some recipe and skill information for one of the agents. ST and ET are parameters that refer to the start and end times of the `rescue-victim` action. No single agent in the group may possess all of the skills and recipes necessary to achieve the goal. For instance, the `transport-victim` action may require knowledge and skills of an engineer agent. Even if there was a single agent that could accomplish the goal on its own, it may be more efficient to work as a team in which each agent performs the tasks to which it is best suited.

3. PROPOSAL TREES

This section first introduces necessary terminology then defines proposal trees. Section 3.6 provides a link between

group decision making and the mental states of collaborative agents.

3.1 Actions, Action Types, and Parameters

An action can be either basic or complex. A *basic action* is performed by a single agent and is treated as atomic. It is assumed that an agent can perform a basic action at will, under certain conditions [2]. A *complex action* is performed by one or more agents and is decomposable. The predicate $Basic(A)$ holds if and only if A is a basic action.

An action is defined by its action type and parameters. The action A may be written as $\bar{A}(p_1^A, \dots, p_k^A)$ where \bar{A} denotes the action type and the p_i^A are parameters to the action. The superscript $(^A)$ indicates that the parameter is one of the action A ; this superscript may be omitted when the action to which the parameter refers is clear.

The *action type* is a property of an action that is used as a way to group relevant information about related actions. $ActionType(A)$ denotes the action type of action A . *Parameters* are variables or constants that provide further detail about the action that is to be performed. $Params(A)$ denotes the set of parameters of action A . The predicate $Constant(p)$ holds if and only if parameter p is a constant. An action in which all parameters are constants is said to be *fully specified*.

3.2 Recipe

A recipe encodes the decomposition of an action² into a set of constituent actions and the constraints among them. A *recipe* for a complex action A is a set of actions, $\{B_1, \dots, B_n\}$, and constraints, $\{\rho_1, \dots, \rho_m\}$, such that the doing of the actions under the constraints constitutes the doing of A . $Recipes(A)$ denotes the set of recipes for action A .

The following axiom states that every parameter of action A is found in at least one constituent action in all of its recipes.

$$\forall p \in Params(A), R \in Recipes(A) \\ \implies \exists B \in Actions(R) \text{ s.t. } p \in Params(B)$$

3.3 Plan Trees

Plan Trees [4] explicitly represent the choices that a group of agents have made when working on a SharedPlan. They are used to summarize the mental states (e.g., goals and intentions) of each agent in a group that is planning to perform a complex action.

Each node of a Plan Tree corresponds to an action. Nodes are categorized based on the type of action they represent (basic or complex) and whether a group of agent(s) has been assigned responsibility for performance of the action (resolved or unresolved). An action that has a group of one

²In the implementation, we make use of action types to encode recipe information for a large number of actions (that differ in the settings of their parameters but have the same action type) compactly.

or more agents that has been assigned to perform it is said to be *resolved*; other actions are *unresolved*. The four node categories are summarized below:

Node Type	Action Characteristics	
β	basic	resolved
κ	complex	resolved
ϵ	basic	unresolved
μ	complex	unresolved

The root node of the plan tree corresponds to the highest-level action that group is planning to perform. The goal of the group of agents in planning is to expand the Plan Tree into a Full Plan Tree, which contains only β and κ nodes, that correspond to resolved actions.

A Plan Tree also contains nodes that correspond to planning actions. Planning actions are actions that specify how a group will make the decisions necessary for expanding a Plan Tree into a Full Plan Tree.

3.4 Proposal Tree

Proposal trees have a similar structure to Plan Trees but do not contain planning actions.

Each node n of a proposal tree contains a triple $\langle A, C, G \rangle$ where A is an action, C is a set of constraints on the parameters of A , and G is a set of one or more agents. Each node has zero or more child nodes; the children of a node correspond to the actions in a recipe for A . $Parent(n)$ denotes the parent node of node n . $Children(n)$ denotes the set of child nodes of n .

A *fully specified basic node* in a proposal tree is a node $\langle A, C, G \rangle$ such that A is an action that is basic and fully specified and G is a singleton that corresponds to a single agent.

A *proposal tree for A*, denoted PT_A , is a proposal tree whose root node contains the action A . A *full proposal tree* (FPT) is a proposal tree in which all of its leaf nodes are fully specified basic nodes. A *partial proposal tree* (PPT) is a proposal tree that is not a FPT. A *minimal proposal tree for A* is a proposal tree for A that contains a single node.

3.5 Relationship between Plans and Proposals

Work on SharedPlans has focused on the mental state of agents as they incrementally expand a partial plan into a full plan. Our discussion of Proposal Trees focuses on the specification of a search problem and is not directly concerned with mental states of agents.

There are several important parallels between this and prior work, summarized in Table 1. Hunsberger [4] separates nodes of a Plan Tree into four categories. Proposal trees do not require such fine categorization. Basic and unresolved actions can be accommodated in the general representation of actions. β nodes of a Plan Tree correspond to fully specified basic nodes in a proposal tree.

Plans	Proposals
Plan Tree	Proposal Tree
κ, ϵ, μ nodes	nodes of proposal tree
β node	fully specified basic node
Full Plan Tree	Full Proposal Tree
Partial Plan Tree	Partial Proposal Tree

Table 1: Overview of plans and proposals

3.6 Execution

A team is said to *execute A according to PT_A* when it adopts the mental states required for performing A and chooses sub-actions, constraints and agent assignments that correspond to those elements of the proposal tree PT_A .

4. SEARCH SPACE

In this section, we define the search problem faced by the group of planning agents as they search for the set of basic actions they will take in order to perform a group action. We first provide a representation of the problem. Then, we will evaluate the effectiveness of standard search algorithms for it, and discuss the need for domain-specific extensions to the standard algorithms. We illustrate the use of proposals and responses as a language for agent communication.

4.1 Problem Definition

A multi-agent planning problem is defined by a triple

$$\langle A, C, G \rangle$$

where A is an action to be performed, C is a set of constraints among those actions, and G is the set of agents that form a group that is responsible for performing the action.

4.2 Representation

The search space is represented as a tree. Each node s of the search tree contains a proposal tree. The root node of the search tree contains a minimal proposal tree. The children of each node s of the search tree contain the proposal tree attained by applying an operator to the proposal tree contained by s . Each leaf node of the search tree contains an FPT.

4.2.1 Initial State

The initial node of the search tree contains the minimal proposal tree for A , which contains the single node $\langle A, C, G \rangle$.

4.2.2 Operators

The operators SELECTREC, ADDAGENT, and BINDPARAMETER transform proposal trees. Successors of each search state correspond to search nodes containing those transformed proposal trees.

SELECTREC (Figure 3) expands a complex action A in a proposal tree into a set of constituent actions. A recipe is chosen that provides a set of constituent actions for a leaf of the proposal tree that contains a complex action. A node is created for each of these constituent actions and placed into the proposal tree as children of A .

SELECTREC(PT)
 where PT is a PPT
 returns a Proposal Tree

- Let $n = \langle A, C, G \rangle$ be a leaf of PT such that $\neg Basic(A)$ holds
- Choose recipe R from $Recipes(A)$
- For each $B \in Actions(R)$
 - $C_m \leftarrow Constraints(R) \cup PROPAGATECONSTRAINTS(C, B)$
 - Construct node $m = \langle B, C_m, \emptyset \rangle$
 - Add node m to PT as a child of n
- Return PT

Figure 3: Procedure for recipe selection

PROPAGATECONSTRAINTS(C, B)
 where C is a set of constraints and B is an action
 returns a set of constraints

- Remove all constraints of C that do not contain a member of the set $Params(B)$
- Return C

Figure 4: Procedure for constraint propagation

Constraints from the parent action are propagated down into these new nodes. From Axiom [1], we know that every parameter of the parent action will appear in at least one constituent action. The PROPAGATECONSTRAINTS procedure (Figure 4) ensures that all constraints involving each child's parameters will appear as constraints in the child. Thus, all relevant constraint information will be passed to the child nodes. SELECTREC specifies that the empty set of agents will be associated with the constituent actions.

ADDAGENT (Figure 5) adds an agent, taken from the group assigned to a node's parent, to the group that is assigned to perform an action. Either a basic or complex node is chosen, denoted by m . If m is basic then there must be no agent currently assigned to it since only one agent can be assigned to a basic action. If m is complex, there must be at least one agent assigned to the node's parent n that is not assigned m itself. Then, one agent is chosen from n (that is not currently assigned to m), and is assigned to m .

BINDPARAMETER (Figure 6) selects a variable parameter and binds it to a constant. A parameter p which is a variable (not a constant) of an action is chosen. That variable is replaced by a constant, respecting the constraints contained in the node corresponding to that parameter.

4.3 Search in the Rescue Domain

The search for a plan for rescuing the victim begins with its root node which contains the minimal proposal tree, which contains the single action `rescue-victim`, the group of four agents, and perhaps a constraint (ET<4pm) which says the action must be completed by 4pm.

ADDAGENT(PT)
 where PT is a PPT with more than one node
 returns a Proposal Tree

- Let $m = \langle B, C_m, G_m \rangle$ be a non-root node of PT and let $n = \langle A, C_n, G_n \rangle$ be the parent of m , such that:
 - If $Basic(B)$ holds, then G_m is \emptyset
 - Else $G_m \neq G_n$
- $g \leftarrow$ an element of $G_n \setminus G_m$
- Let $G_m \leftarrow G_m \cup g$
- return PT

Figure 5: Procedure for adding an agent

BINDPARAMETER(PT)
 where PT is a PPT
 returns a Proposal Tree

- Let $n = \langle A, C, G \rangle$ be a node of PT such that $Params(A)$ contains at least one variable
- Choose $p \leftarrow$ an element of $\{q | q \in Params(A) \wedge \neg Constant(q)\}$
- Bind p in A to a constant, respecting constraints in C
- return PT

Figure 6: Procedure for binding a parameter

The SELECTREC operator is chosen to expand the search state. It would choose the one node of the proposal tree `rescue-victim`³ which is the only leaf. Three nodes (corresponding to the actions from m_1 's recipe for `rescue-victim`) `move-to-loc`, `assess-victim`, and `transport-victim` would be created under the `rescue-victim` node. The proposal tree of these four nodes would form a child node in the search tree of the root node. Other agents may have recipes for `rescue-victim`; proposal trees corresponding to the expansion of their recipes would also appear in search nodes that are siblings of this new node. ADDAGENT and BINDPARAMETER are not applicable to the initial search state.

Construction of the search tree continues in this way. Any of the three operators may expand the search tree at the second level. Even in a small problem like the rescue domain that the search space can be large because of the large number of decisions that must be made. The problem of selecting the leaves of the search tree that are most desirable is discussed in the next section.

5. NET VALUE OF PERFORMANCE

A team of rational agent should perform action A if and only if the benefits of performing A exceed the costs of performing it. $V(A)$ denotes the benefit, or intrinsic value of performance of A . The cost of performing an action is a function of the resources consumed in performing it. The

³nodes are identified by their corresponding action for ease of illustration.

GETCOST(PT)
 where PT is a PT
 returns real value

- Let $n = \langle A, C, G \rangle$ be the root of PT
- If $Basic((A))$ then return $C_G(A)$
- let $c \leftarrow 0$
- For each $m \in Children(n)$ do
 - $c \leftarrow c + GETCOST(m)$
- return c

Figure 7: Cost of performing actions in a Proposal Tree

cost to agent G of performing basic action B is given by $C_G(B)$. The function GETCOST shown in Figure 7 returns the cost to a group of agents of performing the tasks specified in a proposal tree.

The *Net Value of Performance* (NVP) quantifies the net benefit a team that executes a proposal according to PT_A .

$$NVP(PT_A) = V(A) - GETCOST(PT)$$

The following theorem defines economic rationality for a team of agents. A rational team of agents will not perform an action if its costs outweigh its benefits.

THEOREM 1 (TEAM RATIONALITY). *A team should not perform A if $NVP(PT_A) < 0$.*

5.1 Interleaving Planning and Acting

The solution to the team decision problem is equivalent to finding the full proposal tree PT_A that maximizes the $NVP(PT_A)$, according to the cost function GETCOST. In problems in which A is sufficiently complex, it will be impossible to find the optimal plan. In such cases, it will be desirable to use a search technique that finds plan trees within optimality bounds.

The problem representation allows for planning and acting to be interleaved. Furthermore, by exposing parameters such as start times and end times of actions to a search procedure, the decision of the best time to begin to execute an action can be made by the search procedure itself. Thus, it is not necessary to have found a leaf of the search tree (which contains an FPT) before actions are taken.

However, the model must take into account that the cost of performing those earlier tasks has already been occurred. These costs are termed *sunk costs* because they have been incurred in the past. Sunk costs should have no influence on current and future decision making [8].

Let $performed(A)$ be a predicate that holds if an action has already been performed. Figure 8 illustrates a revised GETCOST function that correctly ignores the sunk costs associated with actions that have been performed.

GETCOST(PT)
 where PT is a PT
 returns real value

- Let $n = \langle A, C, G \rangle$ be the root of PT
- If $performed(A)$ then return 0
- If $Basic(A)$ then return $C_G(A)$
- let $c \leftarrow 0$
- For each $m \in Children(n)$ do
 - $c \leftarrow c + GETCOST(m)$
- return c

Figure 8: Cost of performing actions in a Proposal Tree, ignoring sunk costs

6. RELATED WORK

A proposal tree is a concept that is not previously defined in the multi-agent planning literature. By contrast, the concepts of plan and recipe [9] are well defined; explanations of each are given in this section. An agent that *knows* a recipe for an action simply has an understanding of how that action is performed. An agent that *has* a plan to do an action has a mental state that incorporates a commitment that that action is performed. A proposal tree encodes more specific information than a recipe and relates through the notion of *execution* to the mental states of agents involved in a collaboration.

The nomenclature used for the definitions in Section 3 is adapted from work by Hunsberger [4] and Grosz and Kraus [2, 3] to maintain consistency with their work in Shared-Plans. The group decisions faced by teams is independent of a particular mental model of agents and applies to other relevant models of teamwork [1, 5].

Much research in the intersection of AI and economics has related to strategic interactions among agents [6; 12, inter alia]. Such work makes the assumption of individual rationality of agents — that an agent will make the decision that is best for itself. Team rationality does not explicitly model an agent’s individual decision of whether to join a team. The model assumes that agents in a team are committed to performance of the group action, unless it turns out that it is in the team’s best interest to abandon performance of the action.

In social psychology and experimental economics literature, decision making by a group of (typically human) agents is commonly termed *negotiation* [11, 10]. Negotiation typically involves three interrelated tasks: definition of possible outcomes, argumentation, and the search for a solution. Argumentation [7] refers to the process of performing communicative acts with the goal of changing the mental states of other agents. Our work assumes that relevant knowledge is either private (e.g., cost of performance, recipe libraries) or common (e.g., value of complex tasks). Extensions to our work that allow arguments to change common or private beliefs are an interesting area of inquiry.

7. CONCLUSION AND FUTURE WORK

The complex problem of decision making in collaborative teams was cast as a search problem by defining an initial state, a set of operators, and an objective function. Specification of the initial state and operators required the new concept of proposal trees to be defined. A team is said to *execute* an action according to a proposal tree when it adopts the appropriate mental states that correspond to the actions represented by the decisions represented in the plan tree.

The new concept of net value of performance was defined as the objective function for the search that enables the search to focus on the most effective team decisions. Team rationality was defined as a concept that ensures that a team of agents will execute a complex action only when doing so is in its best interest.

For future work, we plan to use the definitions and concepts defined in this paper to provide support for replanning than may be required due to dynamic changes in the environment (e.g., failure). The theory will be extended to illustrate which subset of group decisions is affected by these dynamic changes and needs to be reconsidered.

We plan to test various search techniques to discover which are most appropriate under various domain assumptions. We expect that standard informed, uninformed, and local search methods will each be appropriate given different domain characteristics. Search techniques tailored specifically for the planning problem that can take advantage of action timing information to make decisions of when acting should be begin while search for decisions continues is a promising new research area. Real world search techniques will need to incorporate the cost of time spent searching that could be spent on performance of domain actions.

8. REFERENCES

- [1] Phil Cohen and Hector Levesque. Teamwork. *Nous*, 25:11–24, 1991.
- [2] Barbara Grosz and Sarit Kraus. Collaborative plans for complex group activities. *Artificial Intelligence Journal*, 86(2):269–357, 1996.
- [3] Barbara Grosz and Sarit Kraus. The evolution of SharedPlans. In Michael Wooldridge and Anand Rao, editors, *Foundations and Theories of Rational Agency*, number 14 in Applied Logic Series, pages 227–262. Kluwer Academic Publishers, The Netherlands, 1999.
- [4] Luke Hunsberger. Making shared plans more concise and easier to reason about. In Jörg Müller, Munindar P. Singh, and Anand S. Rao, editors, *Proceedings of the 5th International Workshop on Intelligent Agents V: Agent Theories, Architectures, and Languages (ATAL-98)*, volume 1555, pages 81–98. Springer-Verlag: Heidelberg, Germany, 1999.
- [5] David Kinny, Magnus Ljungberg, Anand S. Rao, Liz Sonenberg, Gil Tidhar, and Eric Werner. Planned team activity. In *Artificial Social Systems, volume 830 of Lecture Notes in Computer Science*, pages 227–256, 1994.
- [6] Sarit Kraus. *Strategic Negotiation in Multi-Agent Environments*. MIT Press, Cambridge, MA, 2001.
- [7] Sarit Kraus, Katia Sycara, and Amir Evenchik. Reaching agreements through argumentation: a logical model and implementation. In *Artificial Intelligence*, volume 104, pages 1–69, 1998.
- [8] Richard Lipsey, Peter Steiner, Douglas Purvis, and Paul Courant. *Economics*. Harper Row, 1990.
- [9] Martha Pollack. Plans as complex mental attitudes. In Philip R. Cohen, Jerry Morgan, and Martha Pollack, editors, *Intentions in Communication*, pages 77–103. MIT Press, Cambridge, Massachusetts, 1990.
- [10] H. Raiffa. *Lectures on Negotiation Analysis*. PON Books, Cambridge, Mass., 1996.
- [11] Alvin E. Roth. Bargaining experiments. In John Kagel and Avlin E. Roth, editors, *Handbook of Experimental Economics*, pages 253–348. Princeton University Press, 1995.
- [12] Tuomas W. Sandholm. Distributed rational decision making. In Weiss, editor, *Multiagent Systems*, chapter 5. MIT Press, 1999.