NOTICE

The quality of this microform is heavily dependent upon the quality of the original thesis submitted for microfilming. Every effort has been made to ensure the highest quality of reproduction possible.

If pages are missing, contact the university which granted the degree.

Some pages may have indistinct print especially if the original pages were typed with a poor typewriter ribbon or if the university sent us an inferior photocopy.

Reproduction in full or in part of this microform is governed by the Canadian Copyright Act, R.S.C. 1970, c. C-30, and subsequent amendments.

AVIS

La qualité de cette microforme dépend grandement de la qualité de la thèse soumise au microfilmage. Nous avons tout fait pour assurer une qualité supérieure de reproduction.

S'il manque des pages, veuillez communiquer avec l'université qui a conféré le grade.

La qualité d'impression de certaines pages peut laisser à désirer, surtout si les pages originales ont été dactylographiées à l'aide d'un ruban usé ou si l'université nous a fait parvenir une photocopie de qualité inférieure.

La reproduction, même partielle, de cette microforme est soumise à la Loi canadienne sur le droit d'auteur, SRC 1970, c. C-30, et ses amendements subséquents.

Canada

Consensus: A Planning Protocol For Cooperating Expert Systems

Richard Clark

A Thesis

in

The Department

of

Computer Science

Presented in Partial Fulfillment of the Requirements
For the Degree of Master of Computer Science
Concordia University
Montréal, Québec, Canada

December 1992

ISBN 0-315-84649-6

Canada

Abstract

Consensus: A Planning Protocol
for Cooperating Expert Systems

Richard Clark

Planning is an important aspect of Distributed Problem Solving. Several approaches have been taken by researchers towards planning. In this thesis, we propose a distributed planning protocol titled *Consensus*. Consensus is useful in a situation where several expert systems cooperate to solve a problem. It is also applicable in solving ill-structured problems. The set of expert systems that plan using Consensus is called a *Consensus Group*. Each expert system in a Consensus Group makes a *Proposal*, from which the Final Plan is generated. The proposed protocol has a potential to minimize the cost of planning because negotiation is avoided. It is implemented and experimentally analyzed. For the purpose of analysis, four metrics were defined and a proposal generator was developed which simulates the expert systems creating their proposals. As a part of Consensus, three alternative heuristics were examined to overcome the computational complexity of a backtracking approach for the generation of the Final Plan. The experimental studies indicate the relative trade-off between the complexity of planning and the quality of the plan with respect to these heuristics. The experimental results and conclusions are presented in the thesis.

Thanks

There are many people I'd like to thank. My mother, whose contribution to my support enabled me to stay in school. My friends at the L-annexe who kept me sane. I'd like to thank Dr Radhakrishnan my thesis supervisor, for his financial support, and getting Victor Lesser to have his picture taken with me. I'd like to thank Cliff Grossner for his general help in getting my thesis off the ground, his ongoing help and support throughout my stay, and providing statistical help for chapter 5. Above of all though, I'd like to thank Annice, who was there when I needed her the most.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

"Where shall I begin, please, your Majesty?" he asked. "Begin at the beginning," the King said, gravely, "and go on till you come to the end: then stop." - Lewis Carroll, Alice in Wonderland

Expert systems are computer systems which are intended to solve problems in domains which were exclusively the domain of highly skilled human experts. Expert system development has been highly successful, with many systems now in commercial use.

Like many areas of computer science, the use of distributed computer systems in the artificial intelligence field, or in particular expert systems, offers a number of potential benefits. The low cost of individual computers and the relatively ease of interconnecting them makes the solution of problems by using numerous small computers attractively inexpensive compared to large mainframe systems. The geographical distribution of some problems makes a distributed approach a natural one. In other cases, the functional decomposition provided by a distributed approach can reduce the development effort needed as compared to a monolithic system. The potential for increased speed of computation is also a factor.

Of course, these benefits can be realized only if the individual computers (in our case, expert systems) cooperate effectively to solve problems. Having one computer reverse the decision of another, or attempt to solve the problem in a fashion incoherent with the rest of the problem solving group of computers could lead to non cooperation

Often, to get the expert systems to cooperate effectively, the diverse views of how problem solving should proceed must be resolved. To integrate the different views of the individual expert systems so that they may cooperate effectively, we have developed a protocol we call *Consensus*.

In this chapter we present an overview of distributed problem solving (DPS) and ill-structured problems. We describe the usefulness of solving ill-structured problems, and show how a blackboard architecture is well suited to solving them. We define Distributed Artificial Intelligence.

1.1 Ill-Structured Problems

The type of problems we address in this thesis, and the kind most often solved by expert systems, are known as *ill-structured problems* (ISPs). Many problems solved in the field of Artificial Intelligence can be categorized as ill-structured, the problems become well-structured only in the process of being prepared for the problem solvers [38]. Ill-structured and well-structured problems (WSPs) lie at opposite ends of a spectrum, most problems having characteristics of both, with no real boundary between the two. Simon states that well-structured problems have one or more of the following characteristics [38]:

- There is a definite criterion for testing any proposed solution, and a mechanizable process for applying the criterion.
- There is at least one problem space in which can be represented the initial problem state, the goal state, and all other states that may be reached, or considered, in the course of attempting a solution of the problem.
- For a given state, attainable state changes (legal moves) can be represented in a problem space, as transitions from that state to the states directly attainable from it. But "considerable" moves, whether legal or not, can also be represented - that is, all transitions from one considerable state to another.

- Any knowledge that the problem solver can acquire about the problem can be represented in one or more problem spaces.

- If the actual problem involves acting upon the external world, then the definition of state changes and of the effects upon the state of applying any operator reflect with complete accuracy in one or more problem spaces the laws (laws of nature) that govern the external world.

- All of these conditions hold in the strong sense that the basic processes postulated require only practicable amounts of computation, and the information postulated is effectively available to the processes — ie., available with the help of only practicable amounts of search.

Nii [33], referring to earlier work by Newell, summarizes the characteristics of an ill-structured problem as a) poorly defined goals and b) an absence of a predetermined decision path from the initial state to a goal. She states that even the criteria for determining whether a solution is acceptable is often not well-defined and the solution must be passed to human experts who debate the worthiness of the solution. A characteristic of ill-structured problems not mentioned by her, but which is important in the kind of problems we address, is that it is not possible to predict completely the result of all actions taken while solving a problem [5].

Numerous methods exist for solving ill-structured problems. There are the well known *weak methods*[36], problem dependent heuristics which attempt to bypass the combinatorial explosion which would be caused by an exhaustive search. There are also rule-based expert systems, neural nets, etc .... Simon argues in [38] that qualitatively different methods from the ones currently available are not required for the solution of ill-structured problems. In light of this, the next section presents a well-known method used in solving a number of ill-structured problems, called the blackboard model.

1.2 The blackboard model

The blackboard architecture describes the organization of knowledge and data, and the problem solving behaviour within an expert system[18, ch 1]. A basic description of the blackboard model is given in[18], from an analogy attributed to Newell:

> Metaphorically, we can think of a set of workers, all looking at the same blackboard: each is able to read everything that is on it, and to judge when he has something worthwhile to add to it ....

The blackboard model consists of three basic components:

knowledge sources The knowledge an expert systems needs to solve the problem is partitioned into a set of separate knowledge sources, each dealing with a particular aspect of the problem in which it specializes. Knowledge sources are usually represented as procedures, or sets of rules.

blackboard data structure The problem solving state data are kept in this data structure, which is a database global to all the knowledge sources in the expert system. Knowledge sources produce changes to the blackboard which lead incrementally to a solution to the problem. Communication and interaction among the knowledge sources take place solely through the blackboard. The objects on the blackboard are hierarchically organized into levels. Information associated with objects on one level serves as input to a set of knowledge sources which in turn place new information on the same or other levels.

control The type of control component is not specified in the blackboard architecture, it is determined by the system designer. The control component may be a part of the knowledge sources, or it may be be a separate module. It decides which knowledge source will be instantiated, and perhaps also the area of the blackboard in which it will work. The control component may cooperate with the knowledge sources to make these decisions.

The blackboard, also called the working memory, is divided into levels representing an application specific hierarchy, with possibly different data structures or

representational methods on each level. The data represented on a blackboard is uncertain, and consequently a single piece of data could eventually lead to a number of different interpretations. In the blackboard architecture, these potential interpretations are stored as *hypotheses*. Some hypotheses are more likely to be correct than others. This is represented by the knowledge source assigning a confidence factor or belief value to the interpretation suggested by the hypothesis. As other data and hypotheses are analyzed by the knowledge sources, these hypotheses may have their confidence levels raised or lowered.

A blackboard-based expert system offers a more flexible reasoning strategy than a traditional expert system. A blackboard-based expert system is not limited to using a single inference engine. Instead, the diverse knowledge in a traditional monolithic expert system can be segmented into knowledge sources where each one specializes in a particular aspect of the problem, possibly using a separate inference engine appropriate for it. In a blackboard model, opportunistic reasoning is used to produce results. Individual knowledge sources can use *forward reasoning* which proceeds from a given state to a goal state, *backward reasoning* to go from a goal state back to an initial state, or any other reasoning methods. Blackboard systems use *opportunistic reasoning*, which is defined as the system's ability to choose selectively its best data and most promising methods at any point during the problem solving process[19].

From the above discussion, it can be seen that blackboard systems are well suited to the solution of ill-structured problems. The representation of vague or uncertain data which characterize ill-structured problems is an integral part of the blackboard architecture. The incremental and opportunistic approach to problem solving activity, with no a priori determined reasoning path, performs well in a problem solving environment in which control uncertainty is a fundamental characteristic. In our work, an *expert system* consists of a number of knowledge sources with local memory. Shared among the expert systems is a central blackboard through which the expert systems communicate, subject to the "organization" described in chapter 3.

An example may be appropriate here to help make some of these concepts concrete. The Hearsay II project[20], the first successful modern blackboard systems

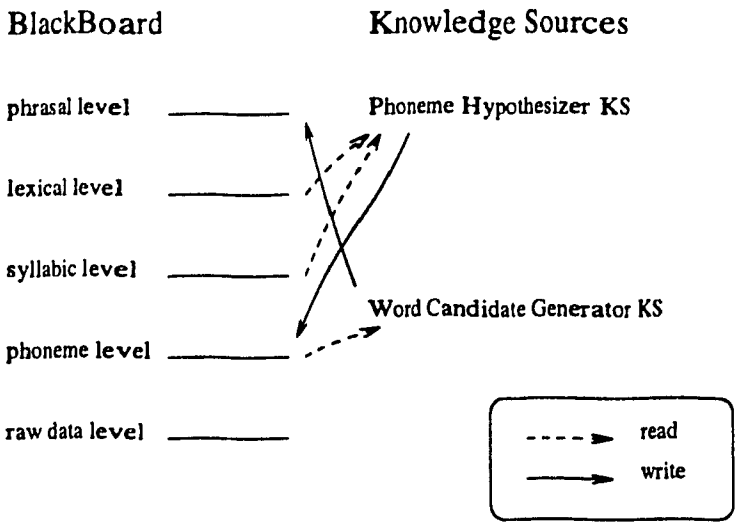


Figure 1.1: A simplified view of the Hearsay II blackboard structure

application, used numerous knowledge sources and a multi-level blackboard to perform the task of recognizing human speech (please see a simplified version of the Hearsay blackboard in Figure 1.1). The blackboard data structure was composed of seven levels, increasing in abstraction from the raw data input level, through the phoneme and lexical levels, and eventually to the phrasal level. There were eight different knowledge sources in the Hearsay II system. Some worked on a single level, combining the data into larger pieces. Others, like the phoneme hypothesizer, looked at the syllabic and lexical levels, and moved data lower in the abstraction hierarchy to constrain hypothesis creation (backward reasoning). Still others, like the word candidate generator, viewed lower levels of the abstraction hierarchy and created hypotheses at the lexical level in a data-directed or forward reasoning manner. The essence of a blackboard system has been demonstrated in this example, namely the abstraction hierarchy of the blackboard, and the opportunistic and multiple reasoning methods used.

1.3 Distributed Problem Solving

Cooperative Distributed Problem Solving (CDPS) and MultiAgent systems are fields of Distributed Artificial Intelligence (DAI), which is the field where artificial intelligence and distributed processing overlap [15]. CDPS considers how the work of solving a particular problem can be divided among a number of modules or individual expert systems that cooperate at the level of dividing and sharing knowledge about the problem and the developing solution [2]. This is different from what is known as "MultiAgent systems" in which multiple autonomous intelligent "agents" coordinate their behaviour, despite potentially having disparate goals.

There are numerous reasons for the growing interest in DAI [2, pg8]. In some domains, the knowledge or activity is inherently distributed because of geographic problem distribution coupled with processing or data bandwidth limitations, a good example of which is distributed sensing. Sending raw data to a single node for processing may exceed that node's *bounded rationality* - defined as the limited processing and control capabilities of a single computer[21]. Sending unprocessed data would also place heavy demands upon a communication network. The desire for fail-soft degradation of the system can also be a factor in selecting a distributed solution most systems can still function despite the loss of one or more participating expert systems. Another consideration is that the development of intelligent systems may be facilitated by building them as separate but interacting parts.

Cooperation among the expert systems in a CDPS is necessary because a single expert system does not have sufficient expertise, resources, and information to solve the problem completely, and the integration of the results of the individual expert systems is required[15]. The expert systems must coordinate their goals to avoid duplicating the work of other expert systems, or working at cross-purposes; this requires them to interact. The term *organization* is used to refer to a structure that is imposed on the expert systems of a CDPS (Cooperative Distributed Problem Solver) which determines the interactions among them. The interactions between expert systems involved in CDPS varies greatly among various protocols and has a profound effect on the functioning of the system because the ability to interact effectively

is a fundamental component of intelligence [15]. We discuss various strategies of interactions between expert systems in our analysis of DAI work in Chapter 3.

Cooperation and coordination, necessary for CDPS, are not easily achieved. No single expert system views the whole problem at any one time, and because of the bounded rationality of an expert system and limited communication bandwidth, they are forced to make decisions with incomplete knowledge. Expert systems may disagree on what goals to pursue, or when to pursue them. The sub-problems they solve are usually not independent, so the expert systems should avoid conflicts and where possible use knowledge from one sub-problem to aid in the solution of another. These issues must be addressed to have a system that works effectively.

1.4 Overview of this Thesis

In chapter 2, because the concepts of goals and planning are essential to our work, we give an overview of them relative to distributed problem solving. A literature survey of related work in the field of distributed artificial intelligence is presented in chapter 3, and motivation for our protocol is provided. In chapter 4, we then describe in detail the protocol we have developed, including some of the heuristics used, and show an example of its use. Chapter 5 analyzes and compares the performance of several key heuristics which Consensus can use. Chapter 6 provides conclusions and future work.

Chapter 2

Goals, Goal-Directed Control, and Planning

Linus : I guess it's wrong always to be worrying about tomorrow. Maybe we should think only about today.
Charlie Brown : No, that's giving up. I'm still hoping that yesterday will get better.
- Charles Schulz "Peanuts"

A *goal* represents either a particular problem state that a problem solver wishes to reach, or a constraint placed upon the method used to reach the desired state[2-1]. Goals are used to direct the sequence of actions leading to the solution of an ill-structured problem. This form of control of an expert system is called *Goal-directed* control. Goal-directed control is an alternative to *data-directed* control in which the problem solving process depends entirely upon the current state of the solution that is being constructed. Goal-directed control is a mechanism by which the problem solving process used by the expert system is focused onto areas that are determined to be important, not necessarily relying solely on the current state of the solution or the immediate effects of instantiating a knowledge source. Thus, goal directed control allows the expert system to reduce the cost of solving a problem by reducing the computational effort the expert system devotes to producing partial results that are never used as part of the final solution to the problem [7]. *Planning* is the act of selecting, before execution, a series of steps that an expert system will perform

to solve a given problem. If the various steps of the plan can not be undone, then the creation of a plan is more important [36, pg 249]. A strategy for planning in which the outcome of a step is uncertain, and hence may fail, is important for solving ill-structured problems [12].

2.1 Goals and Goal-Directed Control

A goal is the data structure used by an expert system to represent the intention to achieve a particular problem state. The representation used for goals is generally problem specific, and so no general method to describe goals exists. Usually, the data structure used to represent goals will contain many fields. We define two kinds of fields to be used in the Consensus protocol: specifications and attributes (see Figure 2.1). The specification fields describe the problem state to be reached by an expert system, as well as the expected quality or belief in the desired results. Attribute fields of the goal indicate the goal's rating, a unique name, and pointers to goals with which this goal has relationships. We will present more detailed examples of the fields used to represent goals for a specific problem in chapter 4 showing an example of the Consensus protocol.

The satisfaction of a single goal may require the work of several different knowledge sources or expert systems. To indicate the importance of data in the lower levels of a blackboard in achieving the original goal, and to permit multiple expert systems to participate in achieving a single goal, the goal can be decomposed into several *subgoals* by the planning component of an expert system. The goal decomposition requires domain-dependent knowledge. The root goal is referred to as a parent goal of the new subgoals.

The decomposition of the parent goal into subgoals can be used to distribute the workload to other expert systems in a CDPS. The decomposition by one expert also avoids requiring each knowledge source or expert system to possess the capability to know how it can contribute to the satisfaction of the parent goal. Subgoals each require the work of fewer knowledge sources, or less time than the original *parent goal*. The subgoals may in turn be subgoaled again, creating a *goal hierarchy* (see

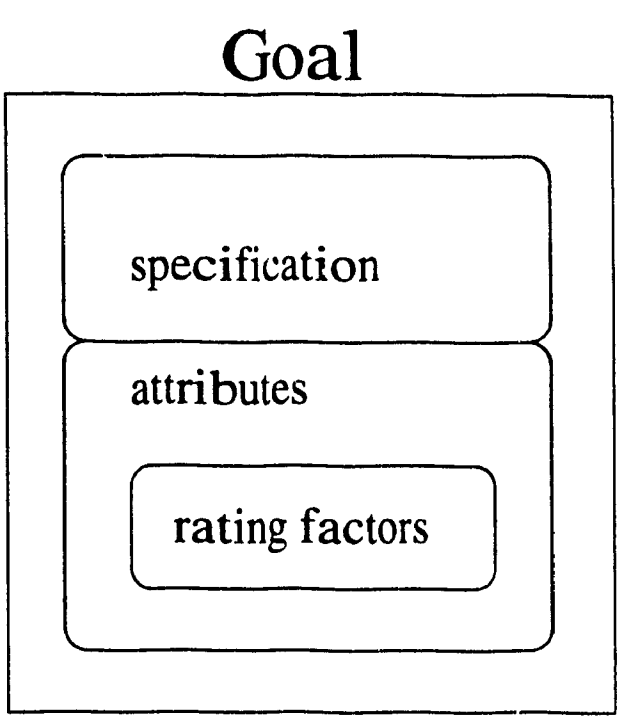


Figure 2.1: Goal fields

Figure 2.2). The satisfaction of the relevant subgoals ensures the satisfaction of the parent goal.

A *rating* of a goal is a numerical estimate made by an expert system and assigned to each field in a subset of the attributes of a goal we call *rating factors*. Rating factors are selected by the system designer and are meant to reflect the characteristics of the goal that the designer believes are important when the expert system must decide

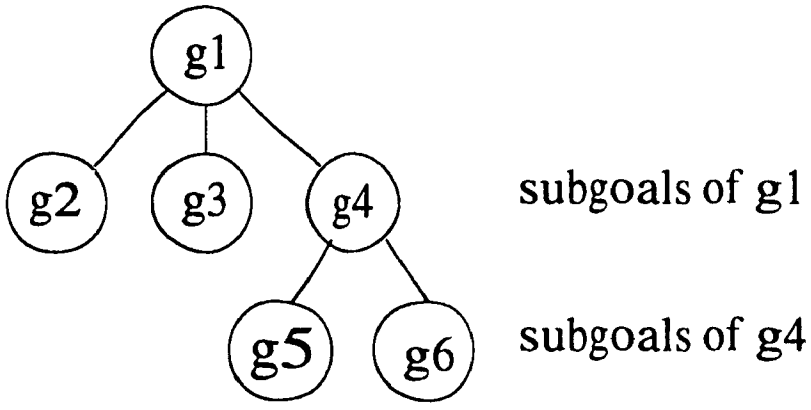


Figure 2.2: Goal decomposition through subgoaling

which goals it should pursue. Corkill [8, pg 174] mentions a number of rating factors: the goal's importance in solving the problem; the estimated cost of achieving the goal; and the probability of satisfying the goal. He also bases the goal rating on the rating of the hypothesis that stimulated the creation of the goal, and the rating of a parent goal. We know that in ill-structured problems, the number of goals which could be pursued at any one time is large. Thus, an expert system must select a subset of these goals to pursue. Generally, a numerical scale is used to represent the value assigned to each factor, and this is essential for the expert systems to compare different goals, although other attributes such as goal relationships (described below) can be used to aid in the selection process.

We have chosen two rating factors for goals in Consensus, called the the *Likelihood of Success* and *Desirability*. Desirability of a goal is an estimate of how useful pursuing the goal will be to the overall solution of the problem. The desirability of a goal is represented using a numerical scale ranging from zero to ten, where zero indicates low desirability and ten indicates a goal will be very important in achieving the overall solution to the problem. Likelihood of Success is similarly defined as a number between zero and ten, but it is an expert system's estimate of how likely the goal is to be achieved. The notions of Desirability and Likelihood of Success transcend all domains. Every problem domain is different and will require appropriate factors from which to determine the Desirability and Likelihood of Success of a goal.

Factors we have identified which indicate the Desirability of a goal are as follows: the area which the goal covers (for those domains in which the area of a goal is meaningful) where goals that cover larger areas given a higher rating; the level of the blackboard on which results will be stored (higher levels are more desirable); the potential to satisfy the goal with a minimum amount of resource consumption, and a much more vaguely defined notion of how well the goal actually addresses the overall problem. One factor which may affect the Likelihood of Success for a goal is the amount of supporting data that exists to help satisfy the goal. For a specific goal, as the supporting data which exists on the blackboard increases, so should its Likelihood of Success.

Formally, each goal g is described by specifications, a set of attributes A = $\{a_1, a_2, a_3, \ldots, a_n\}$, and a vector of rating factors RF = $< f_1, f_2, \ldots, f_n >$, $f_i \subset A$. Associated with each rating factor is a numeric value. The *rating* of goal g_i is $r(RF_i)$, where r() is a function chosen by the system designer which uses the numeric values associated with the rating factors. In our case, RF = {Desirability,Likelihood of Success}. A linear function of the form $r() = \sum_{i=1}^{n} w_i a_i$ is frequently used to determine the rating from the rating factors of a goal[22]. Often we simply say "the rating of g" rather than "the result of applying r() to the vector of numbers associated with the rating factors of a goal g."

In order to make decisions about which goals to select, the ratings of all goals must be *comparable*, that is, if G is the set of all goals, then $\forall g_a, g_b \in G, r(g_a) \leq r(g_b)$ or $r(g_b) \leq r(g_a)$.

The accuracy of an expert system's estimate of a goal's rating will vary with the nature of the goal. An expert system will be able to make a better estimate of the true Likelihood of Success of a given goal if the goal is at a low level in the goal hierarchy. Goals at a low level in the goal hierarchy are unlikely to be subgoaled further, or can be solved by a single knowledge source instantiation. Conversely, it is easier for an expert system to rate the Desirability of an abstract goal (one that can be subgoaled numerous times) than that of a low-level goal. The Desirability of a goal is a concept which predicts how useful pursuing a given goal would be in attaining an over-all solution to the problem.

The creation of a "good" plan (a good plan is explained in the next section) requires more than simply the ratings of goals, it also requires that the relationships among the goals are considered. Goal relationships are used to obtain *global coherence* in a network of cooperating expert systems. Global coherence is defined by Durfee [10] as "the activities of the nodes should make sense given overall network goals. Nodes should avoid unnecessarily duplicating the work of others, sitting idle while others are swamped with work, or transmitting information that will not improve overall network performance." In a complex environment, making appropriate control decisions is a difficult task and goal relationships can be used to guide planning.

This approach is explained in [31]. Wilensky also recognizes the importance of goal relationships [42, p14] :

> "My claim is that dealing with interactions should be moved from its secondary status to the primary framework around which the planner is designed, and that doing so will result in a substantially different and more advantageous planning control structure"

Several goal relationships were identified and discussed in [31] and are given below. We have included two goal relationships (repression and temporal) we developed in addition. We split the goal relationships into three broad categories : beneficial, neutral, and adverse. The beneficial relationships are those that make it useful to pursue both goals, such as the cooperation relationship. Adverse relationships are those like repression, in which pursuing both goals is not possible. Independence is an example of a neutral relationship.

assistance one goal, g1, is said to assist a second goal, g2, if the satisfaction of g1 implies satisfaction of g2. The assistance relationship identifies those goals that represent alternative approaches to generating a particular solution.

competition two goals are said to be competing if there is no possible partial solution that will contain both goals. Thus, if both goals are pursued the effort spent on one of them will eventually be discarded.

cooperation two goals are cooperating when it is possible for the goals to produce information that may be incorporated into a single result in the future.

independence two goals are independent if they are not competing and it is not possible for them to be incorporated into a single result at some point in the future.

repression two goals are said to have a repressive relationship if the acceptance of one implies the rejection of the other. The repressive relationship is especially

useful in situations where resource acquisition conflicts occur. Giving the resource temporarily to one expert system implies that other expert systems are not able to obtain it until it has been relinquished at some later time.

subsumption goal g1 subsumes a second goal g2 if the specifications of g2 are completely encompassed by the specifications of g1. Thus, if g1 is solved, g2 is solved.

temporal goal g1 has a temporal relationship with goal g2 if goal g1 should be attempted at a particular time with respect to goal g2. The temporal relationship is used to indicate those situations in which there is currently insufficient data for a goal g2 to be achieved, but the necessary data is expected to be provided by another goal g1. Thus goal g1 should be pursued before attempting g2. Similarly, if goal g1 fails, then g2 should also fail, even before it is attempted.

Goal-directed control of expert systems is the practice of using goals to define how the problem solving process should proceed to find a solution to the problem. In a goal-directed system, the blackboard is augmented to contain a goal blackboard which mirrors the data blackboard in dimensionality. The goals appearing on this blackboard represent a request to create a particular problem state on the data blackboard. The goals are used to develop into plans by a new component of the system called the planner. Goal-directed control is effective as a means of ensuring that an expert system will perform tasks in a coherent manner [7]. Goal-directed control can be seen as an improvement over early blackboard systems which were considered to be data driven. Data-directed control will often lead to unfocused activity in a complex domain as only the current state of the blackboard is used to control knowledge source scheduling. This can lead to the expert system pursuing numerous possible, but unlikely solution paths (ie., by instantiating knowledge sources that have good short term results, but poor long term ones). In a data directed approach, the scheduler does not record which information is missing in order to construct a solution to the problem, it is *assumed* that any data that is needed will eventually develop due to normal knowledge source scheduling, which is based strictly on the current state

of the blackboard. This leads to lower processing times because the attention of the problem solver is focused onto the long term solution of the problem [7].

2.2 Planning

Despite having an intuitive meaning, the formal definition of a plan is not unanimous among experts in the Artificial Intelligence field. Hayes-Roth describes a plan as "a temporally organized pattern of intended action descriptions" [41]. This is much broader than the definition of a plan in the STRIPS environment, described in [27] as "a sequence of operators". The information that would be contained in a typical plan is a list of actions, the order in which the actions are to be carried out, the objects that are required to perform the action, preconditions and postconditions associated with an action, and the goal to be achieved. As of yet, there is no general representation for a plan that allows goal to be described in domain-independent ways. The representation of a plan is important, because not only is it used to interact between the expert systems that create a plan and those that execute it, but in Consensus it is the primary object of communication among expert systems. Linden [41] proposed that a plan representation is in some ways similar to a programming language, however the expressiveness of a language for planning is much greater than that for describing execution. While a programming language is adequate for describing a set of actions to be performed, it does not contain a method to describe what or why the actions are to be performed. Describing the what and why of prescribed actions is an important component of the information communicated by a plan.

In [41], a planner is defined as a program that controls one or more devices capable of carrying out actions in the real world in order to achieve some definite purpose. We define a *planner* as a program, or a set of programs that choose a series of actions for an expert system to follow in order to achieve a goal. The planner will also be responsible for choosing the goals to be pursued.

There are two types of planners – those that work with complete information, called *strategic* planners, and those that decide what to do in situations in which the

information available to plan is limited, called *tactical* planners. Tactical planners must also deal with the fact that the results of actions are not always what was predicted. An example of a well known strategic planner is STRIPS, which generates plans for the blocks world problem. In this environment, the STRIPS planner has complete information about the current state of the problem and is certain of the outcome of each possible action. Strategic planners are not suited to solving ill-structured problems because of the uncertainty of the information. If a strategic planner was used in a domain in which the information available is incomplete, it would have to consider many options, most of which will not be used. A large number of options is a characteristic of an ill-structured problem, and a combinatorial explosion is likely to occur. Tactical planning works by focusing the system's energy onto "areas of the developing solution" which appear to be the most promising, thus avoiding the combinatorial explosion.

To deal with the problem of generating plans in problem domains where the information available is incomplete and the results of actions are uncertain, researchers have turned to systems that interleave execution with planning, called *incremental planning* [12]. Using incremental planning, it is possible to defer decisions until information required to evaluate the preconditions to some actions is available. When a system uses incremental planning, it is necessary for the incremental planner to create a plan, have the expert system attempt to follow the plan, and expand the plan if it was successfully carried out by the expert system, or repair the plan if it could not be carried out [41]. In incremental planning, the planner uses goal-directed reasoning to select long term goals. From these goals, subgoals are created that are more detailed and refer to actions to be taken in the near future. Hence, we call them short term goals. A detailed plan is created only for the short term goals so that the planner does not waste its time creating plans which may not be used if a plan fails [12]. When the plan is followed successfully, the next detailed short term goal is performed, if there is one. Otherwise, the planner develops another detailed sequence. If the plan is not followed successfully, the planner modifies the plan by introducing additional actions or choosing alternate goals. The planner thus gener-

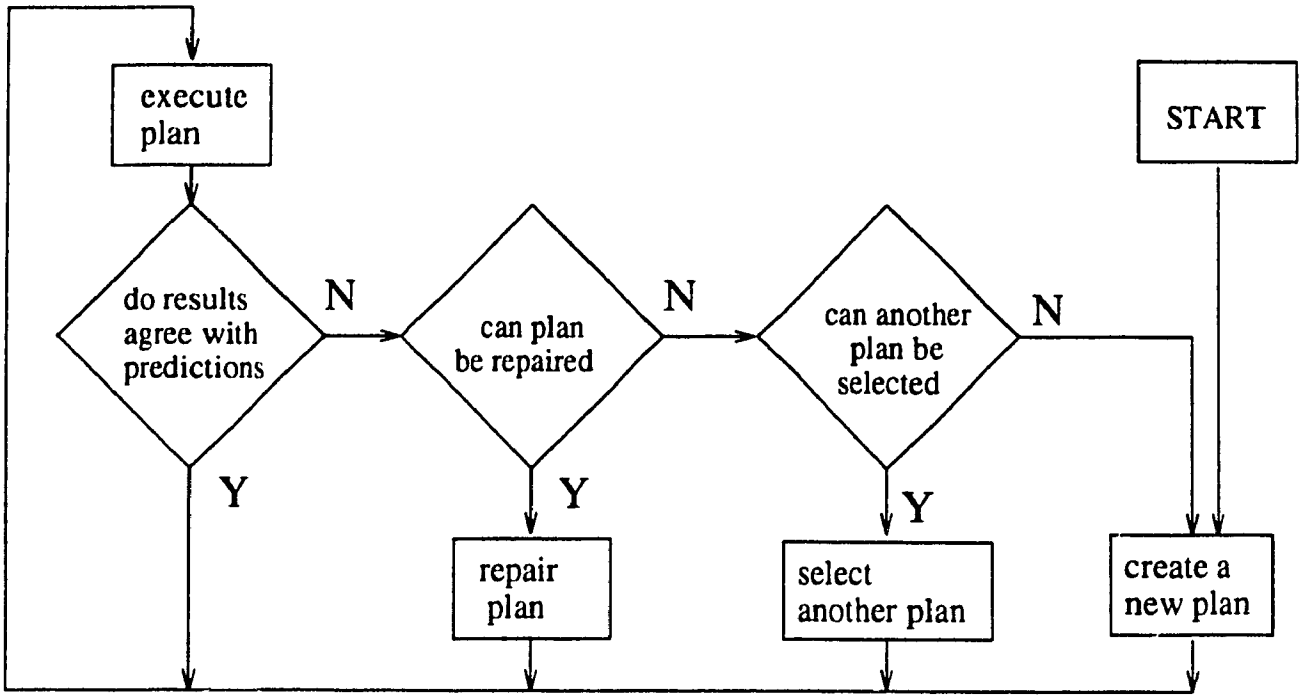


Figure 2.3: Incremental planning

ates, monitors, and repairs plans. Figure 2.3 illustrates the steps which occur during incremental planning.

The tactical planner reduces uncertainty and improves the efficiency of the problem solving process by ordering the actions taken by the expert system. Reducing uncertainty and minimizing the number of steps required to solve a problem is important in domains where the outcome of actions is uncertain.

To create what could loosely be referred to as a "good" plan, a planner must have domain-dependent knowledge, and also embody knowledge not specific to a particular domain, called a planning meta-theme [42, p31], examples of a planning meta-theme are as follows:

- don't waste resources
- achieve as many goals as possible
- maximize the value of the goals achieved
- avoid impossible goals

- create efficient plans

Evaluating a plan requires that there exists definite, measurable criteria to gauge its performance. Several criteria that can be used to evaluate resource consumption are the number of messages required to achieve the plan, the time required to execute the plan, or some other problem dependent resource. A common technique is to select a number of resources to measure, and use a weighted linear sum to derive a rating. The difficulty arises because a plan is created before it is executed - all or most criteria are simply *estimates* of what will actually occur. Additionally, the weight given for each criteria is unknown. Usually, the weights are determined experimentally through an adaptive training process during which the weights are altered each time the program is run [36, pg 187]. There is no guarantee that they will be applicable to other problems, or even other examples of the sample problem domain.

The criteria chosen by Durfee [17, pg 151] for rating a plan are as follows: the fraction of the plan completed, because a plan that is almost finished should be allowed to complete; the number of alternative (competitive) goals in the plan, because by working on several alternatives the planner delays committing to a particular plan; the rating of the next goal to be pursued in the plan because the rating is more likely to be accurate than for goals to be pursued further into the future; and the predicted result belief of the plan based upon the rating of the goals it contains. Durfee uses a normalized weighted sum to combine these factors. His experimental results indicate there is no set of weights which will allow the planner to rate plans optimally in all situations, even for the single problem domain of DVMT.

Incremental planning is well-suited to the solution of ill-structured problems. In incremental planning detailed planning is deferred until such time as the uncertainty about what action is to be performed next is reduced. The incremental planner reduces planning time because detailed plans for the near future are more likely to succeed than detailed actions planned long in advance. Incremental planning also preserves the advantage of a data directed approach, namely its ability to quickly react to change. In this way, the incremental planning technique meshes well with a prime characteristic of an ill-structured problem, namely the absence of a predefined

solution path. The advantages of planning are retained in incremental planning. By focusing on the long term while planning, it reduces the waste of resources on steps that might not be incorporated into the solution of the problem.

Chapter 3

Some Approaches to Distributed Problem Solving

The secret to creativity is knowing how to hide your sources. - Albert Einstein

A number of protocols for coordinating the actions of multiple expert systems have appeared in the DAI literature. The Contract Net protocol by Smith [39],[40],[9] was an early attempt at distributed problem solving, but is mainly a task allocation protocol. The Multistage Negotiation protocol by Conry et al. [6] is demonstrated by addressing the problem of restoring virtual circuits in a communication network, a problem we address ourselves in a later chapter. Partial Global Planning [11, 13, 14] by Durfee and Lesser is the closest to our work, we discuss why it is not entirely appropriate for the kind of interaction we require among expert systems. Finally, the framework for cooperation and coordination between expert systems proposed by Grossner et al. [26, 25] is discussed, as this is the environment in which the Consensus protocol is used.

3.1 Contract Net

The Contract Net protocol was one of the first protocols developed for distributed problem solving [39],[40],[9]. It was developed primarily for task allocation in those problem domains where the tasks are not assigned a priori to the expert systems in a network. An expert system may determine that it requires assistance with its

task because the task is too large for it to handle in a timely manner, or it lacks the expertise to solve the task. The expert system then decomposes the task into sub-tasks that are large enough to offset the cost of distribution, and independent enough that a large amount of communication is not required between the subtasks, which would offset any gain achieved by distributing the tasks.

Expert systems using the Contract Net Protocol proceed in the following manner: While decomposing a task, an expert system (the manager) uses either a broadcast or a multi-cast message to announce to potential bidders that a contract to perform the task is available. The contract announcement includes a specification of the task to be performed, required resources, a contract identifier, an expiration time for the contract and other information that may be relevant to the task. Expert systems submit bids for the contract to the manager, the manager selects the winning bidder and informs it, and the winning bidder becomes the contractor. Thus, establishment of a contract is by mutual selection, which may be superior to the simple master/slave arrangement of some earlier and simpler protocols [14].

The Contract Net is concerned primarily with task allocation and load balancing, neither of which are relevant to the Consensus protocol which is the focus of this thesis. Regardless, if task decomposition is difficult, or if there is interaction between subtasks, the Contract Net does not perform well [2] because the Contract Net does not specify how subtasks may communicate. The impact of actions taken by an expert system on the other expert systems is also not addressed. The protocol does not require contracts to be binding, leaving the possibility that contracts may remain unfulfilled. However, the Contract Net protocol is useful, as shown in the Multistage Negotiation protocol, as a base upon which one may build more advanced protocols. Used alone, it does not address some of these fundamental problems of DAI.

3.2 Multistage Negotiation

In the Multistage Negotiation(MSN) protocol [6], a set of expert systems is connected by a communications network. When MSN is applied to a problem that has a geographical distribution, each expert system has assigned to it a unique area of

responsibility for the solution of the problem. A goal which requires the cooperation of other expert systems to solve is broadcast by one or more expert systems. The expert systems which cooperate to solve the problem each create one or more local plans (known as a plan fragment) which may be a part of a potential solution to the problem. Negotiations ensue in which the expert systems choose a set of plan fragments such that the goal is satisfied, and the selection of one plan fragment does not violate a constraint of any other selected plan fragment. A global plan, which consists of the set of chosen plan fragments, is the result.

The example domain shown in [6] (please see Figure 4.14) is a set of expert systems wherein each have responsibility for maintaining virtual circuits in a predefined geographical area of a long-haul telecommunications network. The telecommunications network is different from the communications network that links the expert systems. The expert systems must cooperate to restore virtual circuits in the telecommunications network which have failed. The routing decisions made by a single expert system within it's area of responsibility is called a plan fragment. The plan fragment choices made by one expert system interacts with those of other expert systems because the routing choices must connect to become a single virtual circuit. Conflicts occur because of the limited bandwidth of a particular link in the network, requiring some circuits to use routes other than the optimal one. Sometimes, due to conflicts, the experts systems are unable to restore the virtual circuits.

The above example can be generalized as follows: The MSN protocol begins by broadcasting the goal to all expert systems. Each expert system knows if it might take part in the solution process and if so forms plan fragments for its potential part in the solution of the goal. Each expert system determines the subset of the expert systems with which its plan fragments may interact by using the Contract Net protocol. Plan fragments interact because the selection of a plan fragment made by one expert system limits the choices of another expert system. The individual expert systems then begin negotiating with the expert systems with whom their plan fragments will interact, to make a commitment consistent with its own plan fragment choice. Upon receipt of a message, an expert system can either tentatively commit to

making a plan fragment choice consistent with the choice of the other expert system, or reject the request. An expert system rejects a request by sending a message to the requesting expert system indicating a reason for the rejection. The rejection of a plan fragment, and the reason for its rejection are incorporated into the knowledge base of the expert systems involved in the negotiation, and lead to a continuation of the negotiation as the experts systems attempt to converge to a set of plan fragments acceptable to all of them.

The knowledge base of each expert system includes an AND-OR graph of the goals to be satisfied, called a "feasibility tree". The feasibility tree initially contains only the local alternative plan fragments and conflicts for solving goals. As information is exchanged, each expert system updates its feasibility tree reflecting the external constraints. Expert systems initially choose their highest rated plan fragment which satisfies the constraints in their feasibility tree, but select lower rated alternatives as external constraints indicating certain plan fragments are not viable options are imposed.

Termination of the protocol is guaranteed because an expert system is not allowed to make or withdraw a commitment it has made previously unless changes to its feasibility tree have occurred. The negotiation terminates when there is no pending activity and no incoming communications, or if an attempt is made to return to a previous commitment with no new knowledge from other experts.

MSN has some attractive properties. It is truly distributed in nature, no single expert system is necessarily aware of the entire global plan. The expert systems pass only the minimum amount of information to another expert system at one time, ie., the occurrence of a single constraint. The expert systems attempt to create plans using the highest rated plan fragments, although finding the optimal plan is not assured because an expert system commits to the individual plan fragments on a local rating basis.

MSN suffers from a number of problems and/or inefficiencies. Each reply to a commitment request requires the sending of at least one message. The non-binding nature of the protocol can cause an expert to back out of a commitment to a particular

plan fragment, creating a large number of messages to be sent as commitments roll back. As the number of conflicts between plan fragments increases, a high volume of message traffic is likely: at least one message is needed to report the conflict, and another is needed to request commitment to another plan fragment. Additionally, as the number of experts which must communicate to build the global plan increases, the number of stages in the creation of the plan increases. Each stage must wait for the previous one to finish to ensure that the plan is extended in a manner which is coherent in the global sense, although this isn't mentioned in the example in [6]. In order for the expert systems to detect the termination of the negotiation, it is necessary to either use some kind of token passing mechanism leading to increased overhead, or a binding commitment of resources. An expert system can commence plan execution after termination has been detected.

As we can see, the communication overhead for MSN is high because a large number of messages must be sent. For strategic planning, this makes sense because the high cost of planning is at least offset by guaranteed results, if a solution exists. For tactical planning, in which the outcome of actions planned for is uncertain, the high overhead is difficult to justify.

3.3 Partial Global Planning

Partial Global Planning (PGP) is a framework describing how expert systems can cooperatively interact over a communications network to solve a problem [13]. PGP describes how to coordinate the actions taken by the expert systems, but it does not specify the times that expert systems should interact, nor does it specify exactly what should be communicated between the expert systems. It adheres to the principles outlined in the Functionally Accurate/Cooperative approach described in [30].

Partial Global Planning has been extensively published in the literature [11, 13, 14], primarily in the context of the Distributed Vehicle Monitoring Testbed (DVMT) [30, 7, 8, 12, 10], [18, ch 18], the platform upon which the partial global planning experiments were performed. In brief, the DVMT simulation testbed is used to monitor vehicle movements using data obtained from acoustic data sensors. The

data obtained from the sensors is interpreted by nodes which are connected by a communications network. Each node contains an expert system and a blackboard from which the expert system interprets the data from the sensors and attempts to plot the path of vehicles. The nodes are assigned geographic regions and receive data only from sensors located in that region.

In Partial Global Planning, the expert systems in the network are assumed to be *semi-autonomous*, that is, they act asynchronously, are loosely coupled, operate in parallel, and have limited communication. The distribution of information among the nodes is such that the problem solving strategy is ill-suited to a functional decomposition among the nodes. Each node possesses the information necessary to perform only a portion of each function required to solve the problem because each node can perform problem solving functions only over the area in which sensors report back to it. To perform any one problem solving function completely would require a node to receive data from all sensors.

Each expert system has local plans constructed from its own knowledge and the information received from sensors in its area. Each expert system summarizes each of its local plans, building a node-plan that specifies the goals of the plan, the long-term order of the planned activities, and an estimate of how long each activity will take. Since node plans have much less detailed information than local plans, the expert systems can exchange them with less communication overhead than the local plans. The node plans contain enough information for an expert system to reason about the activity of other expert systems.

The expert system scans the node plans to recognize partial global goals, which is a goal that may encompass the local goals of several expert systems. The expert system then creates a *Partial Global Plan* (PGP) that represents the concurrent activities and intentions of all the expert systems that are working in parallel on different parts of the same sub-problem. The planner recognizes the relative timing of the expert systems activities to discover how the activities may be re-ordered to avoid harmful interactions, such as performing redundant activities, and to promote helpful interactions, such as sending predictive information to expert systems working

on the same sub-problem as soon as it is available. Partial Global Plans may also be exchanged when one expert system controls one or more other expert systems, allowing it to direct their activity. PGP's may also be exchanged between expert systems to negotiate and converge upon consistent views of the actions required to solve the problem.

This protocol is well-suited to its intended area of application, namely semi-autonomous nodes cooperating under real-time constraints. However, the protocol is not appropriate for non-commutative systems [34] in which the result of actions can not be undone. In Partial Global Planning, the individual expert systems are responsible for their own actions, and do not necessarily have consistent views of the entire problem. The expert systems exchange PGP's and node plans asynchronously, giving the possibility that work done by one expert system in its area of responsibility may not be what is most globally beneficial.

3.4 Cammarata's Strategies of Cooperation

The work by Cammarata et al. [3] discusses a number of different strategies for cooperation between expert systems in an air traffic control environment. The strategies are only used to select a single expert system to plan for the group of airplanes.

An expert system is located on each airplane whose function is to perform collision avoidance in air traffic control. The most important point in three of the four strategies is that a single expert system on board one airplane is selected to plan for all of them. The criteria for the selection of the expert system to do the planning is amount of information (position, heading, speed) the expert system aboard the airplane has about the others. The airplane with the most information on hand, or the plane which is least constrained can be selected to form the plan. Constraints an airplane can experience are things like low fuel and airplanes in close proximity). Raw data is sent from each of the airplanes to the single airplane which will be planning for all of them. In the fourth strategy, the planning is distributed among the expert systems in the airplanes by assigning an expert system to be responsible for a given area of the conflict resolution, however no detail is given to this Proposal. It is noted that

in highly constrained problems (more airplanes means more constraints), distributing the planning can lead to high communications costs because the expert systems have to communicate intimately to avoid potential negative interactions among the airplanes, ie., a mid-air crash.

There are hard real-time constraints in the problem domain addressed by this method. A negotiation strategy might be unsuccessful because of the time delays and numerous messages required by a negotiation. The alternative chosen by these designers, because a large number of airplanes are unlikely to occupy the same airspace at the same time, is to have a single expert system plan for all of them. This limitation on the number of participants makes the communication of raw data feasible, and less likely to exceed the bounded rationality of the single expert system that must create the plan. Many of the problems associated with a distributed solution are thus avoided. Attempting to scale this protocol to a larger number of expert systems however would lead to a bottleneck in the single expert system planner, as the designers admit.

The protocol is inadequate for our purposes because it does not conform to our belief that the plan should be jointly selected by the expert systems. Additionally, the sending of raw information to a single planner is likely to exceed the bounded rationality of the expert system which creates the plan for all the expert systems to follow.

3.5 Behaviour Hierarchies

The planning protocol using Behaviour Hierarchies for coordinating expert systems [16, 32] is one in which the participants are not known in advance. Each expert system determines the other expert systems that its plans may potentially interact with (conflict or beneficial relationships) by first broadcasting abstract "behavioural information" and listening for similar broadcasts from other expert systems. Those expert systems whose plans may interact then exchange increasingly detailed messages involving only the anticipated area of interaction in the plans. This technique reduces communication and processing costs because the communications take place

only between the expert systems which may interact and their area of potential interaction, rather than the entire set of expert systems and complete plans. This is well-suited to the example domain used in the literature which is a mobile robot system where the position of the robots, and hence their potential for plan interaction, is constantly changing.

The Behaviour Hierarchies approach assumes robots use the same language and can recognize potential interactions in their planned behaviour. A six dimensional space of behaviours is described, corresponding to who, what, where, when, why and how. The purpose of this is to provide more information for an expert system to reason about another's behaviour than the exchange of goals which simply indicate *what* the robot intends to do, or the exchange plans, which indicates *how* the intended behaviour will be carried out. If the planned behaviour will negatively impact upon another robot's plan, the robot can either change its plan so that the interaction will not occur, or it can negotiate at a more detailed level to try to resolve the conflict. By using abstraction, an expert can potentially reduce search from exponential to linear complexity. Using hierarchical problem solving, this can be further reduced to logarithmic complexity [32]. This protocol can be guaranteed to converge in a finite number of steps, a definite advantage in real time applications.

This protocol is concerned with semi-autonomous expert systems, much like Partial Global Planning in the DVMT example domain. Our work is concerned with CDPS [29] in which expert systems collectively decide what actions to perform.

3.6 Consensus and Decree Organizational Structure

The Consensus Protocol described in this thesis is part of a larger, ongoing work in distributed problem solving by Grossner and et al. [26, 25]. The portion of that work relevant to this thesis is the concept of an Organization of expert systems[1]. This section attempts to show the place of the Consensus protocol within the framework

[1]Information related to the development of human organizations, and subsequently organizations of expert systems, can be found in [26] and [21]

of an Organization, and the projected overall characteristics of these Organizations. A detailed discussion of the Consensus protocol alone will be presented in the next chapter.

A distributed system can be viewed as a particular *organization* - task decomposition and control regime - resulting from the distribution of a set of tasks over a set of logically or physically disjoint processing elements [21]. These organizations are important, because they enable computer systems to exceed the *bounded rationality* [38] of a single computer. An example of the power of organizational structuring is that of a colony of ants or bees. Despite the fact that each individual in the group is "unintelligent", the colony does exhibit overall intelligent behaviour [1]. This suggests a possibility for increased performance using multiple expert systems to solve a problem. However, Fox describes deciding how the task should be decomposed and the control regime to be used as the major problem with designing distributed systems [21].

An organization consists of two parts, a *coordination structure* used for planning, and an *organizational structure* used for execution [26]. Consensus and Decree define a coordination structure, which is a control regime for the distribution of the responsibilities for plan creation among the experts. Grossner defines Consensus and Decree in [26], based loosely on the modes of cooperation between expert systems using goal and data flow discussed by Benda in [1]. The "type B" interaction discussed by Benda describes expert systems negotiating goals to pursue, and the "type C" interaction describes an interaction in which one expert system controls another. Methods for generating more complex organizations based on the modes of interaction discussed in that paper are mentioned. A few informal definitions will suffice for our purposes:

Planning Group The set of expert systems involved in either the Consensus planning protocol, or the Decree planning protocol.

Combined Window The area of the blackboard viewed by the expert system(s) in a Planning Group during the Decree or Consensus planning protocols.

Decree A planning protocol for a set of experts in which only one of them directs the

others. The Combined Window during Decree is the window of the directing expert system.

Consensus A planning protocol for a set of expert systems in which the expert systems decide jointly, in a distributed manner, on a plan. The Combined Window during Consensus is the union of the windows of the expert systems in the planning group.

Experts are restricted to using a Window (part of the blackboard) in order to comply with the limits placed upon an expert by bounded rationality [26]. The assignment of a Window to an expert forces the expert to work with incomplete information. To help cope with this uncertainty, during Consensus an abstraction of the Windows of the experts in the Consensus Planning Group are visible to all members of the Planning Group.

3.6.1 Creation of Organizations

The coordination structure imposed upon a set of cooperating expert systems can be represented by a graph consisting of vertices (experts) connected by either a directed edge (Decree, the node upon which the edge is incident is the node being decreed to), an undirected edge (Consensus), or unconnected (no direct relationship). For the sake of simplicity, we assume that there is one kind of vertex (ie., all expert systems are the same), to eliminate permutations of expert systems being considered to be a different coordination structure, which decreases the number of organizations by $n!$. Organizations having the same structure are considered *equivalent* and were eliminated. An example of equivalent organizations is shown in Figure3.1 [4].

A few simple properties were devised to define the possible relationships among the nodes in an organization. They are:

- Only one of a single type of edge may exist between any two vertices. An expert system can not negotiate which goals to pursue, and dictate to the same expert system.

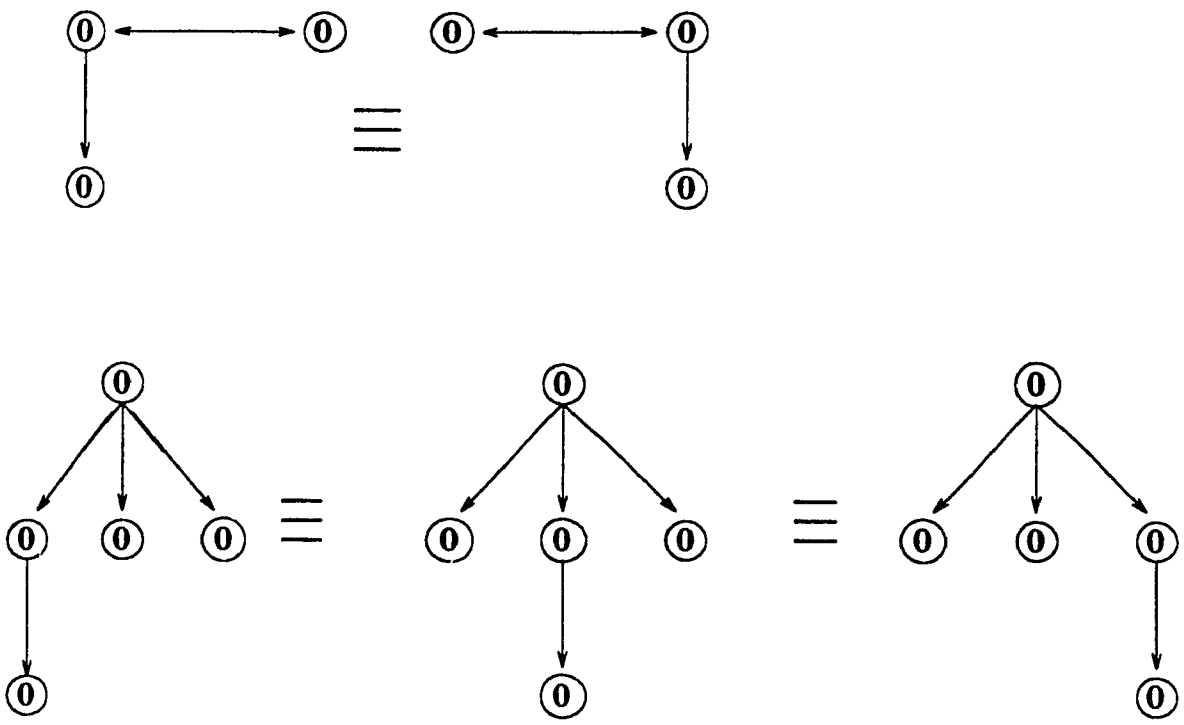


Figure 3.1: Equivalent organizations

- No cycles are permitted in the graph involving decree as an edge. A circular relationship would mean that an expert system is effectively dictating to itself
- Consensus is transitive. If an expert system A is in a Consensus group with expert system B, and B is in a Consensus group with C, then effectively A, B, and C are mutually choosing goals to pursue.
- Only one Decree edge may be incident upon a vertex. An expert system can not guarantee being able to respond to the goals imposed upon it by more than one other expert system.

Given these properties, we wished to derive the coordination structures that could be created for a given number of expert systems. Although we would have preferred to theoretically calculate the number of coordination structures possible for any number of expert systems, this proved to be very difficult, as no clear pattern developed in their generation. Consequently, we ch to empirically derive the organizations by writing a program which would generate all organizations for a given number of

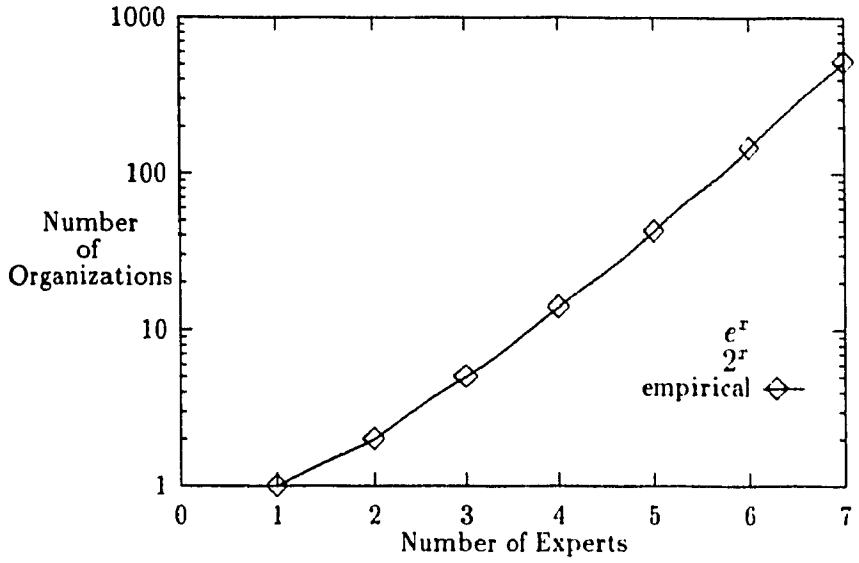


Figure 3.2: Number of Organizations vs. Number of Experts

experts [4].

A graph showing the number of possible coordination structures for a given number of experts is shown in Figure 3.2. We show the actual number of organizations generated as the empirical data, and compare it to two exponential functions. As can be seen from the figure, the number of organizations grows at least exponentially[2]. This large number of possible coordination structures for any significant number of expert systems has a serious implication. Our contention is that the organization of expert systems will have a large impact upon their problem solving performance. The hope of exhaustively testing the performance of all coordination structures of a given number of expert systems to experimentally determine which performed the best is futile. Most certainly when the number of possible organizations is combined with the possible Windows for the expert systems, the possible combinations are overwhelming.

The algorithm to generate these organizations starts with a list of organizations

[2]We tried to generate the number of organizations with more than seven vertices, but this proved unworkable. The processing time on an IBM RISC machine was already several hours for the generation of organizations of seven vertices, and we couldn't justify spending a great deal more time in rewriting the algorithms for a small projected gain in knowledge

with n experts and outputs a list of all organizations of $n+1$ experts. It obtains the new list of organizations by augmenting each existing organization in the given list of organizations with a single additional expert in a Consensus group and in a decree group. Organizations with duplicate structures are unfortunately created using these simple rules, and no easy way was found to avoid creating them, so they are created, compared with other newly created organizations, and then removed. Details of this program, the data structures and algorithms used, its results, and Common Lisp code can be found in [4]. This area was not pursued.

3.6.2 Characteristics of Different Organizations

The coordination structure inherently influences the cost associated with cooperative problem solving [1]. The characteristic of Organizations that use Decree operations only is that the planning time will be short, but it is expected that there will be many plan failures because of the limited Window of the decreeing expert system. In an organization that uses Consensus, the planning time is expected to be longer than that for organizations using Decree, but there will be fewer plan failures because the plan is formulated with information from the Windows of several expert systems. A well-designed organization will use both Consensus and Decree to provide the desirable characteristics of both, and solve the problem in a time shorter than each of the protocols used alone.

It is highly unlikely that any organization will always be the "best" for any problem. It is equally unlikely that even for a given problem domain, with different starting conditions, a single organization will always be optimal [13]. However, because of the characteristics of ill-structured problems, the a priori determination of the optimal organization for a given problem is an open research question. Dynamic reconfiguration of an organization is an option, but given the current limitations to accurately predict the impact of an organization on problem solving behaviour, this has not yet been adequately explored.

3.6.3 Analysis

A few fundamental design decisions form the basis around which Consensus and Decree are built, they are:

1. Each expert knows with which other experts it will interact, and this does not change throughout the problem solving process.

2. All jointly chosen plans are binding upon the experts.

3. Planning among the members of a planning group is a synchronous process.

The environment stated as suitable for Consensus and Decree is CDPS [5]. As discussed above, knowing a priori with whom an expert system may interact is not possible for some problem types. Static organizations using Consensus and Decree will not be used in that environment. Mobile autonomous expert systems, such as that described by Montgomery in [32] would be inappropriate for Consensus and Decree because in that environment expert systems are continually changing their relationships with others due to their changes in their geographical location. The organizational structure of Consensus and Decree would either have to be changed frequently, or the entire group of robots would have to be put into a Consensus Group so that they may plan all their actions together. The gain made by using a combination of Consensus and Decree would not be possible in this context.

Making planning decisions binding is another design decision that somewhat limits the domain of Consensus and Decree. According to Ginsberg [23], cooperation between experts necessitates that when they jointly choose goals to pursue, they will avoid those choices which are potentially bad for any of them. A binding protocol obviously can not guarantee that an expert system will not be required to attempt to carry out a goal that is potentially bad. In a problem such as DVMT, a binding protocol is not as suitable as the loosely coupled, semi-autonomous expert systems used. The expert systems can largely determine their own activities, and communicate to increase cooperation possibilities by reordering their local activities, and occasionally sending goals to one another. However, if the expectation is that a single expert can

be wrong in its interpretation of what it should do next, or a locally greedy approach would not provide the most efficient overall solution, the Consensus Group should be useful.

Synchronous planning, in which all expert systems plan at the same time, ensures that all expert systems in a Consensus group have input into the plan that they will all follow. It eliminates the possibility that one expert may be following one plan while others are changing it. This is consistent with other design decisions in which group problem solving is stressed.

Chapter 4

The Consensus Protocol

Never go to sea with two chronometers; take one or three. - Anonymous

The Consensus protocol has several properties that make it appropriate for use in the coordination structures of organizations for CDPS systems. Consensus is a distributed planning protocol. Consensus is considered to be distributed because each expert system in the Consensus Group independently uses the information available to the group to develop its own Joint Plan. Each Joint Plan will differ from the others because of the different *local context* of each expert system. Consensus is a *binding* protocol. A binding planning-protocol obligates the expert systems to carry out the plan that is created. This ensures that the expert systems in the Consensus Group will attempt to carry out the tasks in the Final Plan. The Consensus protocol will produce a Final Plan in a fixed number of stages. Thus, there is a bound placed upon the cost of creating the Final Plan.

The Consensus protocol assumes that the capabilities and Window of each expert system in the Consensus Group have been defined by the organization of the CDPS system. In addition, each expert system (e) possesses a function $Bel_e(EXP)$ that rates the credibility of another expert system EXP in its Consensus Group. An outline of the stages of the Consensus protocol is shown in Figure 4.1. Notice that the Consensus Group consists of expert systems A, B, and C. Expert systems D and E are part of the CDPS system, but are not part of the Consensus Group.The Consensus protocol consists of the following steps:

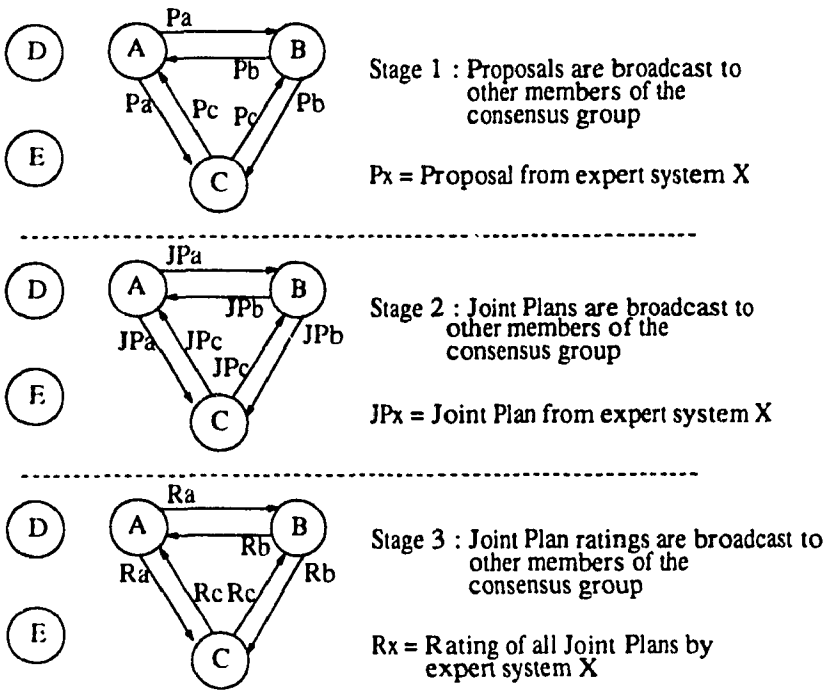


Figure 4.1: Stages of the Consensus Protocol

The Proposal Each expert system in the Consensus Group creates a Proposal. A Proposal contains the goals the expert system determines to be relevant in choosing the next Final Plan. The Proposal of each expert is then viewed by each member of the Consensus Group.

Joint Plan Construction A *Common Planning Structure* (CPS) is created by each expert system using the Proposals. The CPS contains all the goals from the different Proposals created by the experts in the Consensus Group, wherein goals that have the same specification are merged into a single goal. Each expert system then prunes the CPS it has created to produce its Joint Plan. The Joint Plans are then viewed by each member of the Consensus Group.

The Election An election is held to select among the Joint Plans created by the expert systems in the Consensus Group. The Joint Plan selected will become the Final Plan.

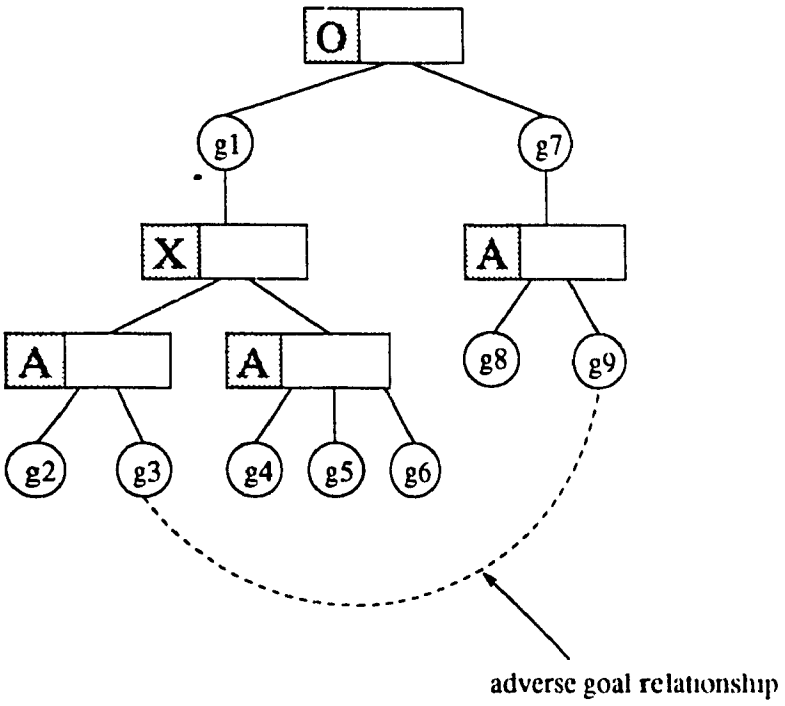


Figure 4.2: A Sample Proposal

4.1 The Proposal

A *Proposal* (see Figure 4.2) is a set of goals which are partially ordered by temporal precedence. An expert system uses goals to abstract the information in its Window to convey the information to other expert systems. By presenting the other expert systems in its Consensus Group with an abstraction of the information in its window, the expert system is less likely to exceed the bounded rationality of the expert systems to which it sends its Proposal. The goals in a Proposal are in hierarchies formed using the subgoal relationship. *Logical Lists* are used to connect the goals in the hierarchies to their subgoals. In this context, the Logical List is called the *subgoal list*. In Figure 4.2, the list immediately below goal g1 is the subgoal list of g1. The hierarchies in a Proposal are also connected by a Logical List. Goal relationships [31] other than the subgoal relationship are indicated by labelled links between the goals in the Proposal.

A Logical List is a labelled list of elements. The elements contained in a Logical List are either goals or Logical Lists. A Logical List may have one of the following

CPS Element	Rating
ANDList	minimum rating of its elements
XORList	maximum rating of its elements
ORList	maximum rating of its elements
Goal	assigned by expert system

Table 4.1: Summary of Ratings of CPS Elements

labels:

AND All elements on the list form an *Atomic Unit.* An Atomic Unit is indivisible. Thus, the expert systems must choose all or none of the elements in the Atomic Unit to be included into their Joint Plans.

OR Any elements on the list may be chosen by the expert systems for inclusion in their Joint Plans.

XOR Only one of the elements on the list is chosen for inclusion in the Joint Plan.

An ANDList is used to indicate that the achievement of some goals is useful only in the presence of other goals also being achieved – if one goal is not included in the Joint Plan, it isn't worth pursuing the other goals. For example, if we wanted to plan to wash something, we can set out three goals - find a tub, get some water, and get some soap. If one of those goals will not be pursued, it isn't worth pursuing any of them. Precedence relationships can not be used to indicate this dependency because that would unnecessarily serialize the intended actions. Goal ratings are not powerful enough to ensure the selection of goals as a unit, because ratings are given on an individual basis. Other than an ANDList, the only way to ensure that a set of goals are selected as a unit would be for the expert system to rate all the goals very highly; but this may not be true, nor what the expert system wanted to express. The current plan creation technique (described in detail later in this chapter) allows for the elimination of goals with some goal relationships between them, it does not allow for goals with certain relationships to be included in the plan. It works more by negative influences rather than positive ones. Thus, the use of the ANDList provides a reasonable way for the expert system to indicate that selection of a set of goals should occur.

The XORList allows an expert system which is creating a Proposal to indicate that a choice is to be made in selecting goals which represent alternate options for inclusion in the Joint Plan. If the choice is simply between two goals, an expert system would normally use a goal relationship (as discussed in Chapter 2) to indicate that only one should be chosen. However, when a choice between sets of goals must be made, the XORList provides a simple solution. The expert system includes different options in a Proposal because by doing so it can defer decisions about which option to select until a later point when it has received more information, ie., the Proposals from the other expert systems, which improves its decision making capabilities.

The OR list is used for those situations when selecting all or one of a group of goals may not be appropriate. The use of the OR list allows the expert system which will be choosing goals for the Joint Plan maximum flexibility. Examples of the use of AND/XOR/ORLists will be shown in the examples section later in this chapter where consensus is demonstrated.

The goals and Logical Lists in a Proposal are rated (see Table 4.1). Logical Lists have a rating derived from the ratings of their elements. The rating of an XORList is the maximum rating of any of its elements which are eligible for inclusion in the Joint Plan. This is because only one of its elements may be selected for inclusion in the Joint Plan, and the highest rated element will be chosen. The rating of an ANDList is the minimum rating of any of its elements, because all elements depend on each other for their satisfaction. Using the minimum rating of the elements of the ANDList for its rating reflects this. Each goal in the Proposal is given a rating by the expert system that created the Proposal. As discussed in chapter two, the rating for a goal is determined using two values, *Likelihood of Success* (LS) and *Desirability* (D). The Likelihood of Success of a goal is an estimate of whether the goal can be achieved given the current state of the solution to the problem being solved. The Desirability of a goal is an estimate of how important the goal is to the overall solution of the problem. We define both LS and D range in value between zero and ten. The *Risk of Failure* (R) of a goal is defined as

$$Risk = \begin{cases} 10 - LS & \text{if } LS < 9 \\ 1 & \text{otherwise} \end{cases}$$

The rating of a goal G is given by $r()$ = Desirability /Risk. The ratings of all Logical Lists ultimately depends only on the ratings of goals.

Each expert system creates its Proposal independently, the specific component of the expert system which does this is called the *planner*. The planner determines its view of the current state of the solution to the problem being solved from the information available in its window. Using the state of the solution and its knowledge base of goals that can be pursued for solving the problem, each expert system decides which goals should be included in its Proposal. The Proposals must include goals which will convey important information, because Proposals are the only means the expert systems use for communication while planning. The role of the Proposal is not simply to generate goals that are highly rated, but to convey information to other expert systems. An expert system can propose a goal with a poor rating, because knowing that a goal is poorly rated by an expert system is also important.

4.2 Joint Plan

A *Joint Plan* is a partially ordered set of goals where the goals are members of ANDLists (see Figure 4.3). The goals in a Joint Plan are in hierarchies formed using the subgoal relationship. Logical Lists labelled as AND are used to connect the goals in the hierarchy to their subgoals. The hierarchies in a Joint Plan are connected by an ANDList. All the goals are on ANDLists because the Joint Plan is binding.

We define a set of goal relationships GR = $(r_1, r_2, r_3, \ldots r_n)$, corresponding to those described in Chapter 2. Using these goal relationships, we can define a plan type P as allowing a subset of those goal relationships. The plan types we define are such that P1 $\subset$ P2 $\subset \ldots \subset$ GR.

Coherent Joint Plan only independent, temporal, and cooperative goal relationships are allowed

Competitive Joint Plan only independent, temporal, cooperative, and competitive goal relationships are allowed

Conflict Free Joint Plan only repressive relationships are disallowed

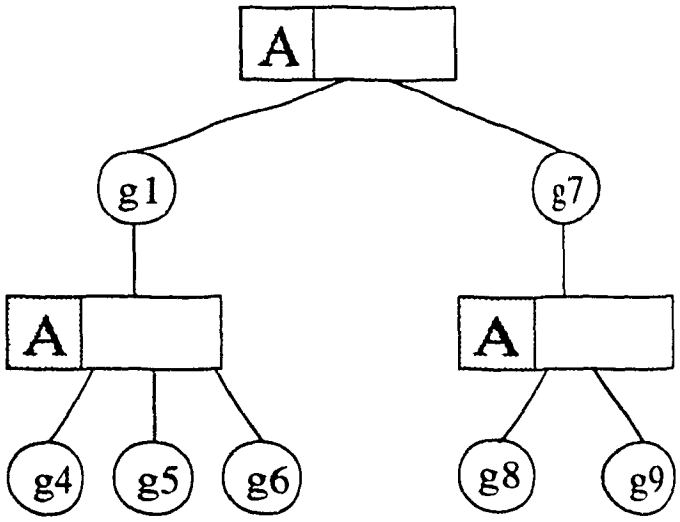


Figure 4.3: Example Joint Plan

As illustrated in Figure 4.4, the plan types are successively less demanding in the goal relationships allowed in the plan. We refer to any goal relationship not allowed in a given plan type as an *adverse* goal relationship. In a Coherent Joint Plan, only beneficial and neutral goal relationships (see Chapter 2) are allowed. In the Competitive Joint Plan, beneficial goal relationships are allowed, but competitive relationships (which may lead to performing unneeded work) are allowed. Finally, the Conflict-Free Joint Plan allows all goal relationships except those which are explicitly forbidden due to some constraint on the availability of a resource (ie., repression). Proposals have no restrictions on the type of goal relationship allowed in them.

Each of the plan types is expected to express a unique behaviour. The coherent plan does not permit any work (ie., knowledge source instantiations) which will possibly lead to duplication or potentially wasted effort. It assumes that decisions as to "which course of action should be taken" can be made using the current state of information on the blackboard. Of course, if that assumption is not valid, and in an ill-structured problem that is likely, there could very well be wasted effort because a solution path that looked promising could fail. A competitive plan allows competitive relationships which may help differentiate between different courses of action. Competitive goals can be pursued until one avenue of action is clearly superior, but also lead to performing some unneeded work. Another way of looking at this is that

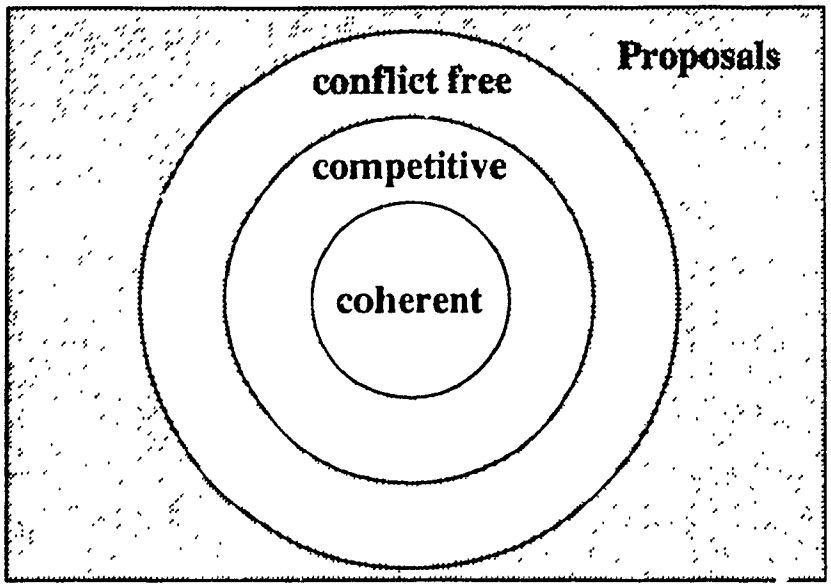


Figure 4.4: Joint Plan Types

the work is really needed, it was necessary to produce more information to make a decision as to the solution path to be followed. Conflict-free plans are plans in the more traditional sense of merely ensuring that the constraints of the problem are not violated. The conflict-free plan can be viewed as a "control" against which we can judge the effectiveness of the other plan types.

Once the Proposals are received, the goals they contain are rated by the expert system that received them. This rating is in addition to the rating the goals were assigned by the expert system that created them. Each expert system will rate each goal based upon the information visible in its window and its knowledge base. The rating assigned to each goal is combined with the goal's original ratings in the Proposals from which it was extracted. This rerating of goals occurs during the construction of the Common Planning Structure, discussed next.

4.2.1 CPS Creation

Using the Proposals created by the expert systems in its Consensus Group, each expert system will create the Common Planning Structure (CPS) as an intermediate step in producing a Joint Plan. The CPS contains all the hierarchies of goals found in the Proposals. When two or more expert systems have proposed *equivalent* goals,

the goals and their subgoals are merged. Goals are equivalent if they have the same specifications. Merging involves goal rerating, updating goal relationship fields so that resulting goal of merged goals has the sum of the relationships in the equivalent goals, and the merging of the subgoal lists of the equivalent goals. The creation of the CPS follows the algorithm shown in Figure 4.5.

The makeCPS procedure is initially called for each top-level goal in the hierarchies in the Proposals received by an expert system. The makeCPS() looks in the other hierarchies for goals equivalent to the one it has received as a parameter. If it finds equivalent goals, it combines them into a single goal and combines their rating and assigns it to the single goal. All those goals are marked so they are not seen by the makeCPS procedure again. The rating of the equivalent goals is combined by an expert system using a static, weighted belief system. Each expert system represents its belief in another expert system by using a number between 0 and 1. If it is desired that an expert system have the same belief in the ratings done by another expert system, it is required that its belief in that expert system is 1, and its belief in its own ratings is one. For example, if an expert system gave a goal a rating of 9, and another expert system gave the same goal a rating of 7, given equal beliefs of 1, the new rating would be 8. This is applied to each of the rating factors in a goal. To facilitate the understanding of how goals and Logical Lists are combined, please refer to the Figure 4.6. In the figure, for simplicity, goals with the same specification are given the same name, although in practice the names of the goals are unique. A few of the ways that the subgoal lists of equivalent goals are merged is shown. In Figure 4.6(a), there are two goals which are equivalent (g1), with ANDList subgoal lists. Because the goals on each ANDList are also equivalent, no new solution strategy is present, and the result of the goal merge is the same structure of one of the inputs.

In part (b) of the figure, the two top-level goals are equivalent, they both have ANDLists as subgoal lists, but the subgoals are not equivalent. In this case there are two distinct methods for solution of g1, and so an XORList is inserted into the hierarchy so that only one method will be selected.

Part (c) of the figure shows how equivalent goals whose subgoal lists are XORLists

```
procedure makeCPS (Goal G)
begin

if (the set of goals (E) equivalent to G is not empty){
   merge goals in E with G creating goal N;
   place N on CPS;

   if (none of E or G have subgoals)
       return;

   if (all subgoals of E and G are on ANDLists)
       remove all but one ANDList from groups of equivalent ANDLists;
       if (there is greater than one ANDList left) {
           create a new XORList;
           place remaining ANDLists on the XORList
           make the XORList a subgoal link of N
       }
       else
           place the ANDList as a subgoal list of N

   else if (there are subgoals of E or G on ORLists) {
       create a new ORList
       place non-equivalent members of ORLists on new ORList
       place ANDLists not equivalent to member of new ORList onto it
       place XORLists not equivalent to member of new ORList onto it
   }

   else {// there are subgoals on AND and XORLists
       create a new XORList
       place non-equivalent elements of G or E's XORList onto new XORList
       place non-equivalent ANDLists onto new XORList
   }
   for each subgoal (g) of N
       makeCPS(g);
}//if
else {
   place G on CPS
   for each subgoal (g) of G
       makeCPS(g);
}
end.
```

Figure 4.5: Outline of Algorithm for creating a Common Planning Structure

are merged. The goals g2 and g3 are alternatives to solving g1, and g2 and g4 are also alternatives. The result is simply an XORList connecting g1 with all non-equivalent alternatives on the list.

Part (d) of the figure shows how equivalent goals are merged when one has an ORList as a subgoal link. There are several alternatives for solution of g1, and so these alternatives are placed on an XORList.

4.2.2 Joint Plan Creation from the CPS

The next step in the creation of the Joint Plan is called *pruning*. Pruning is when each expert system selects a path in the CPS from the top-level goal in each hierarchy to the leaf goal(s) such that the elements along the selected path make the Joint Plan. The elements of the CPS are chosen such that if an ANDList is chosen, all of its elements are chosen, and if an XORList is chosen, only one of its elements is chosen. In the absence of goal relationships, the subgoals are chosen from the ANDLists and XORLists to maximize the ratings of the individual lists. This procedure is applied recursively until goals at the lowest level of each hierarchy are chosen. We call this set of goals, Logical Lists, and the links between them from the top of the goal hierarchy to the bottom the *principal path* in the hierarchy (see Figure 4.7).

When pruning the CPS, the expert system will use the goal relationships that exist among the goals in the CPS and the rating assigned to each goal to determine which portions of the CPS are to be included in the Joint Plan. Goals with adverse goal relationships between them can not co-exist in the Joint Plan. The procedure for pruning the CPS starts by examining the top goal of each hierarchy in the CPS. If the rating of the top-level goal is high enough to meet the rating threshold of the expert system, the top-level goal is chosen. The subgoals of the goals chosen to remain in the CPS are then selected to remain in the CPS according to the selection rules for the label of the Logical List on which they are found. If an XORList is chosen, the highest rated element from it is chosen. In the most straight-forward case, if a subgoal being chosen has an adverse relationship with another goal, the other goal is marked ineligible for inclusion in a Joint plan, and the selection of goals from the

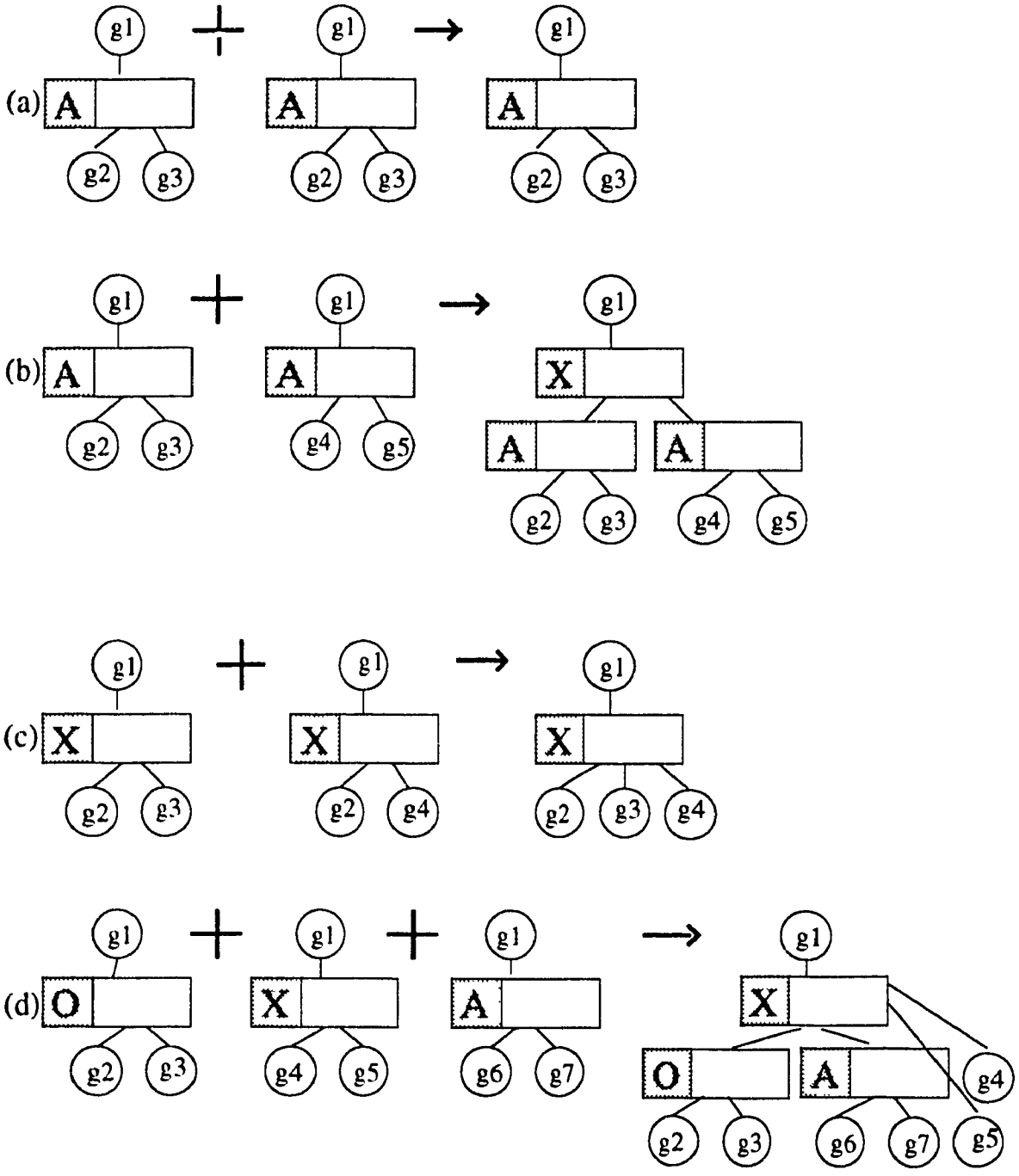


Figure 4.6: Combining Logical lists when creating CPS

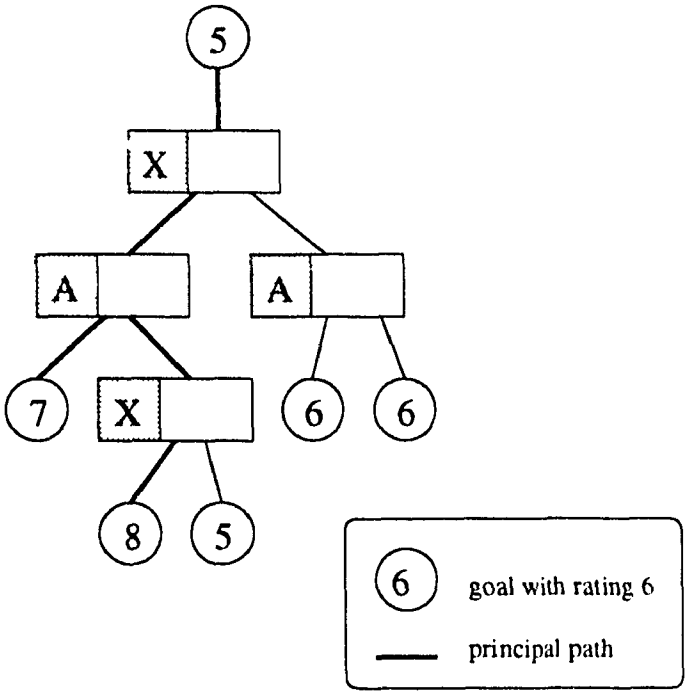


Figure 4.7: Principal Path in a Hierarchy

CPS to create the Joint Plan continues.

A few observations have led us to the belief that an alternate strategy to the straightforward process presented in the above two paragraphs is needed. We observe that the selection of goals in the CPS from among those in adverse relationships are not independent because of the AND/XORLists present. Figure 4.8 illustrates this problem. If a goal in the hierarchy we are currently selecting goals from has an adverse goal relationship with a goal in another hierarchy, the goal in the other hierarchy must be marked ineligible for inclusion in the Joint Plan. This can cause the other hierarchy to lose a highly rated goal, and when the selection process occurs on that hierarchy, the choice of a poorly rated goal may have to be made. We refer to this as the "rating loss problem". If there was another choice in the current hierarchy which had almost the same rating, it would be preferable to choose that path instead. In the figure shown, it would be preferable to select the goal rated 8 rather than the one with a rating of 9. If the one rated 9 was chosen, then the goal rated 8 would not be eligible for inclusion in the Joint Plan. The goal rated 4 would either be chosen, or the whole hierarchy it is a part of would be ineligible for inclusion in the Joint

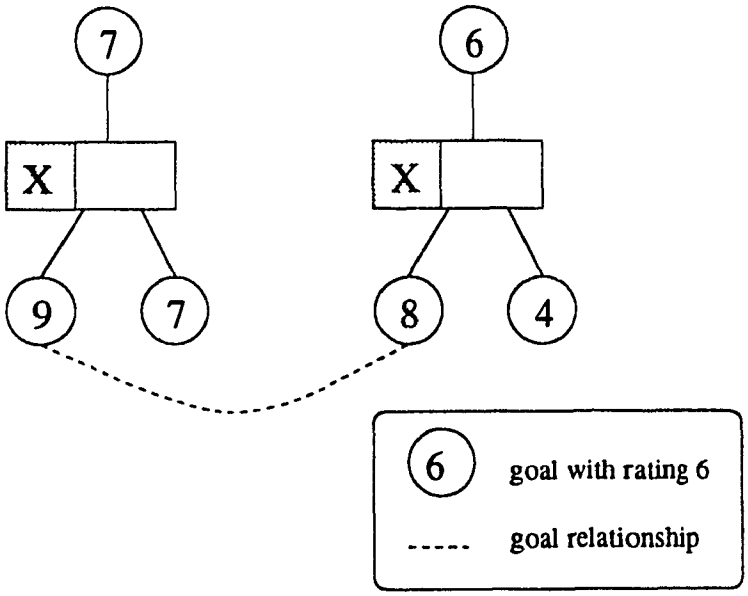


Figure 4.8: Simplified example of goal rating loss

Plan. It is clear that the order in which we choose goals for the Joint Plan can have a significant impact upon the goals chosen for inclusion in the Joint Plan.

The second observation is that under some conditions, whole hierarchies can be made ineligible for inclusion in the Joint Plan if goal relationships eliminate certain goals (called the "hierarchy loss problem"). For example, if a goal that is on an ANDList that is the subgoal list of a top level goal is eliminated, the whole hierarchy is not eligible for inclusion in the Joint Plan. We would prefer that as many goals as reasonably possible are attempted at one time, so as to minimize planning costs [42].

Heuristics for Joint Plan Creation from the CPS

The goal of pruning is to maximize the rating of the Joint Plan that will be created. Because of the goal relationships, the only guarantee of an optimal rating for the Joint Plan would be to exhaustively test all possible Joint Plans (ie., full backtracking) that could be created from the CPS. The resulting large computational cost can not be justified in a tactical planner, where the outcome of planned actions is uncertain. Instead, we don't attempt to achieve optimality. We have developed two alternate heuristics to attempt to maximize the Joint Plan rating while keeping the

computational cost within reasonable limits. These heuristics principally change the ordering in which an expert system chooses hierarchies from the CPS to create a Joint Plan. We compare the performance of these heuristics in an experiment described in chapter 5.

The first heuristic is called the *high-low heuristic*. In this heuristic, the expert system begins selecting goals (ie., pruning) from the highest valued hierarchies on the CPS first, and then progresses toward lower rated ones, hence the name. The value of the hierarchy is measured by adding the rating of the top level goal to the rating of each Logical List along the principal path. The rationale for this heuristic is that if the highest valued hierarchy is pruned first, it is not likely to lose its highly rated goals due to goal relationships. In this way, we attempt to create a highly rated Joint Plan by ensuring that highly valued hierarchies are preserved, at the expense of possibly losing lower valued hierarchies. It is thought that the ratings retained by choosing goals for the Joint Plan from the highest rated hierarchies first should outweigh the ratings loss of the low rated hierarchies.

The second hierarchy ordering heuristic is called the *compromise heuristic*. The heuristic consists of three parts:

1. hierarchy ordering

2. ANDList sorting

3. hierarchy preservation through limited backtracking

The hierarchy ordering is based on a metric we have developed called the *compromise value* which measures the sensitivity of a hierarchy (in terms of the rating loss by choosing an alternate goal) to the loss of a single goal through an adverse relationship. The structure (ie., shape, number of XORLists, number of elements on XORLists is small ...) and goal ratings in a hierarchy make some hierarchies more sensitive than others to being forced into selection of a lower rated goal if a goal relationship makes one of the goals along its principal path ineligible for inclusion in the Joint Plan. If there are few XORLists, or the branching factor of the XORLists is

small, the hierarchy would be more sensitive to the loss of a single goal. If the principal path has a goal that can not be selected due to an adverse goal relationship, the path leading from the next highest rated goal in the hierarchy must be chosen, known as the *compromise path*. The *compromise value of a hierarchy* is defined as the compromise value of the highest level logical list in the hierarchy. The higher the value, the more sensitive a hierarchy is to losing a highly rated goal due to a goal relationship. An infinite value implies that no compromise is possible while still satisfying the top level goal in the hierarchy. A brief example of compromise value would be an XORList with three goals as elements of it. If the first goal is rated 9, the second 7, and the third 6, the compromise value would be 2. The compromise value reflects the decrease in value that the XORList would have if the highest rated choice were not available due to it having a goal relationship with a goal that was chosen earlier.

A more realistic example of the computation of the compromise value can be seen in Figure 4.9. The computation of the compromise value of a hierarchy proceeds in a top-down manner. The top level goal requests the compromise value from its subgoal link list. The XORList X1 computes the ratings of its three elements, which are 7, 5 and 6, in left to right order. The highest rated subtree (the principal path) is on the left hand side of X1. If an adverse goal relationship lowered the ratings of the principal path below that of the second highest rated path (compromise path) on the right hand side, the compromise path would be chosen during Joint Plan creation. The difference in rating between the principal path and the compromise path is the compromise value. If however a goal in the principal path was eliminated due to an adverse goal relationship, but the rating of the principal path was still higher than the compromise path, the compromise value would be (principal path rating — the new path rating).

In the example shown (Figure 4.9), the top level XORList X1 asks its highest rated subtree for its compromise value. The ANDList A1 must compute this from its elements, so it requests the compromise value from its two elements X2 and X3. Goal g2 has no adverse goal relationships, so it returns a compromise value of 0. However,

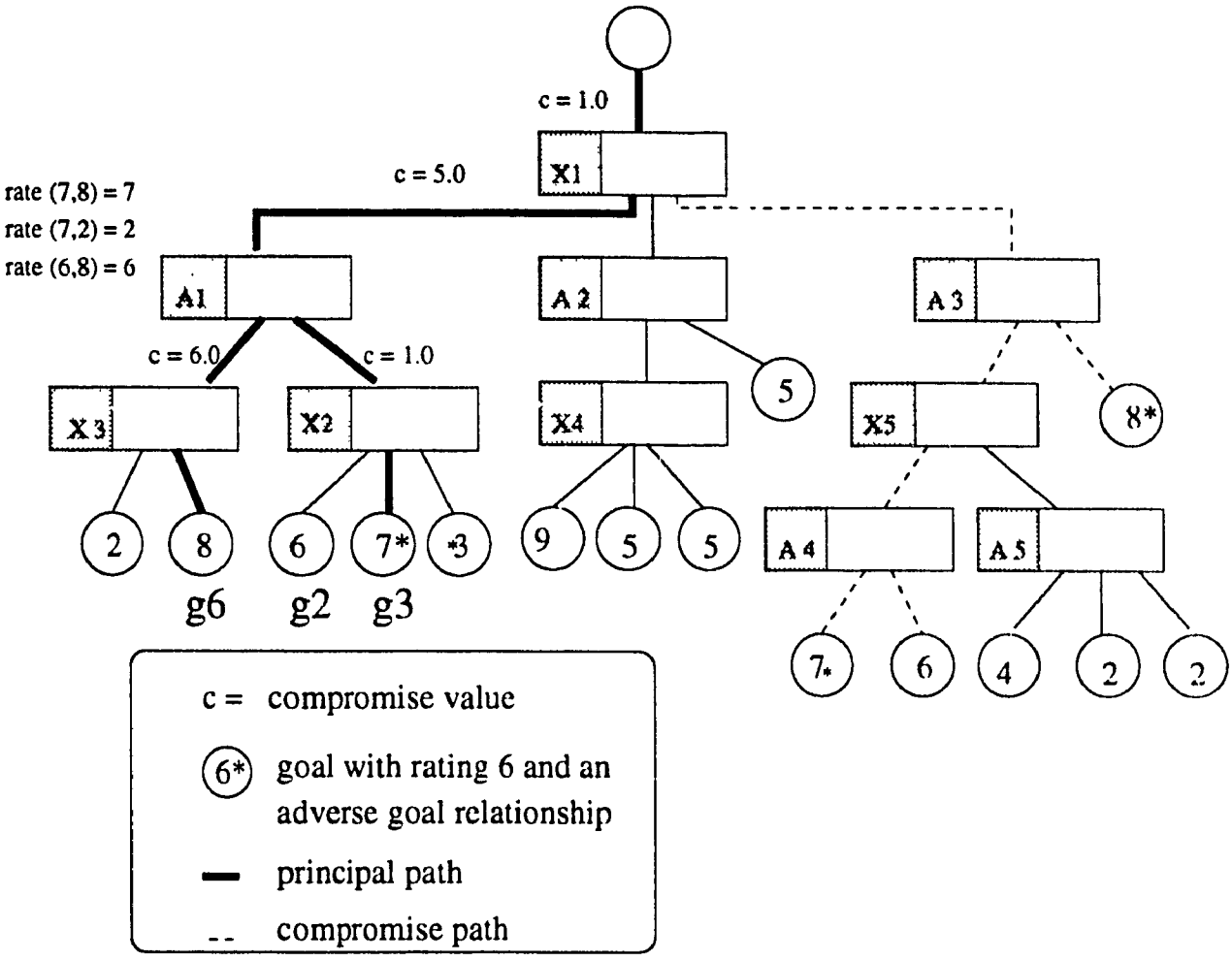


Figure 4.9: Compromise Values at Nodes in a Hierarchy

g3 has an adverse goal relationship, so it returns a compromise value of infinity (can not be selected if eliminated). The XORList X2 would select g2 with rating 6 if g3 was eliminated, so the compromise value of X2 is 1. In a similar way, X3 has a compromise value of 6. A1, which computes its rating as the minimum rating of its elements, computes its potential rating as the lesser of (7-1) & 8, and 7 & (8-6). The lesser value is 2. The original rating of A1 was 7, its potential rating if a single goal (in this case g6) is eliminated is 2, so it returns a compromise value of 5. This value is returned to X1. X1 determines that in the worst case for a single goal to be eliminate ' along the principal path it would choose another element (the compromise path) and the difference in X1's rating would be (7-6)=1, so its compromise value is 1. Notice that the compromise value was only computed using about one third of the hierarchy's elements. In general, for any XORList, the compromise value need only be computed for the highest rated element.

In an effort to alleviate the rating loss problem described above, the selection of top-level goals of hierarchies is done in the order of the hierarchy with the largest compromise value first. After the goals from that hierarchy have been selected, the compromise values of the remaining hierarchies are computed and then re-ordered again. The goal selection is done on the hierarchy with the largest compromise value, and so on until all hierarchies have had goals selected from them, if possible. The reason hierarchies with high compromise values are selected from first is so they will not encounter the compromise situations described earlier, which means that compromises made would be smaller.

A hierarchy has the maximum possible compromise value when the elimination of a single goal means that the whole hierarchy is made ineligible for inclusion in the Joint Plan. When multiple hierarchies had the highest possible compromise value, an earlier version of the compromise heuristic simply selected the first hierarchy seen with maximum compromise value from which to select. Thus, the order of arrival of the hierarchies at an expert system partly determined the order of the goal selection process was called upon them. A pilot study showed that this was often not an optimal solution, some hierarchies were made ineligible to be selected from when

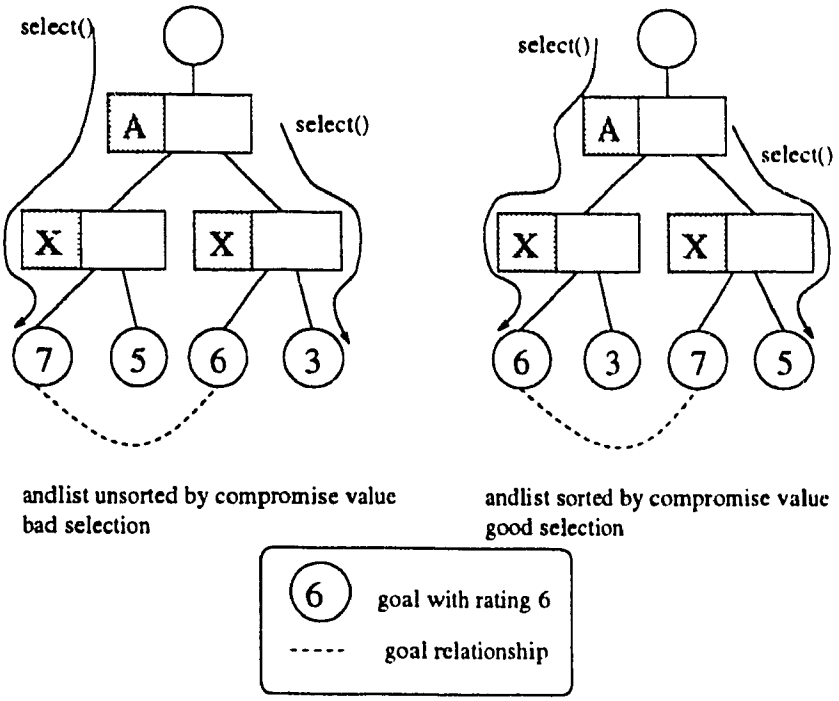


Figure 4.10: Sorted ANDLists vs. non-sorted ANDLists

other orderings of the hierarchies showed that more hierarchies could be used. In those situations, we felt the compromise heuristic could be improved. We observed that when multiple hierarchies have the maximum compromise value, the number of constraints (goal relationships) is likely high. To break the tie among the hierarchies, the hierarchy least likely to impact upon the other hierarchies was selected first. This selection was made by simply counting the number of goal relationships along the principal path of the hierarchy.

As a further addition to the compromise heuristic, the elements of each ANDList on the principal path are sorted by compromise value as the compromise value is computed. Doing this prevents the situation shown in Figure 4.10, where a goal relationship within a hierarchy can decrease the ratings loss due to a goal relationship.

A final component of the compromise heuristic is also used to prevent an entire hierarchy being made ineligible (the hierarchy loss problem). When a goal A is about to make another goal B ineligible for inclusion in a Joint Plan, goal A sends goal B the compromise value that would occur in A's hierarchy if B refuses to be made ineligible. Goal B then determines whether it will be made ineligible by comparing the

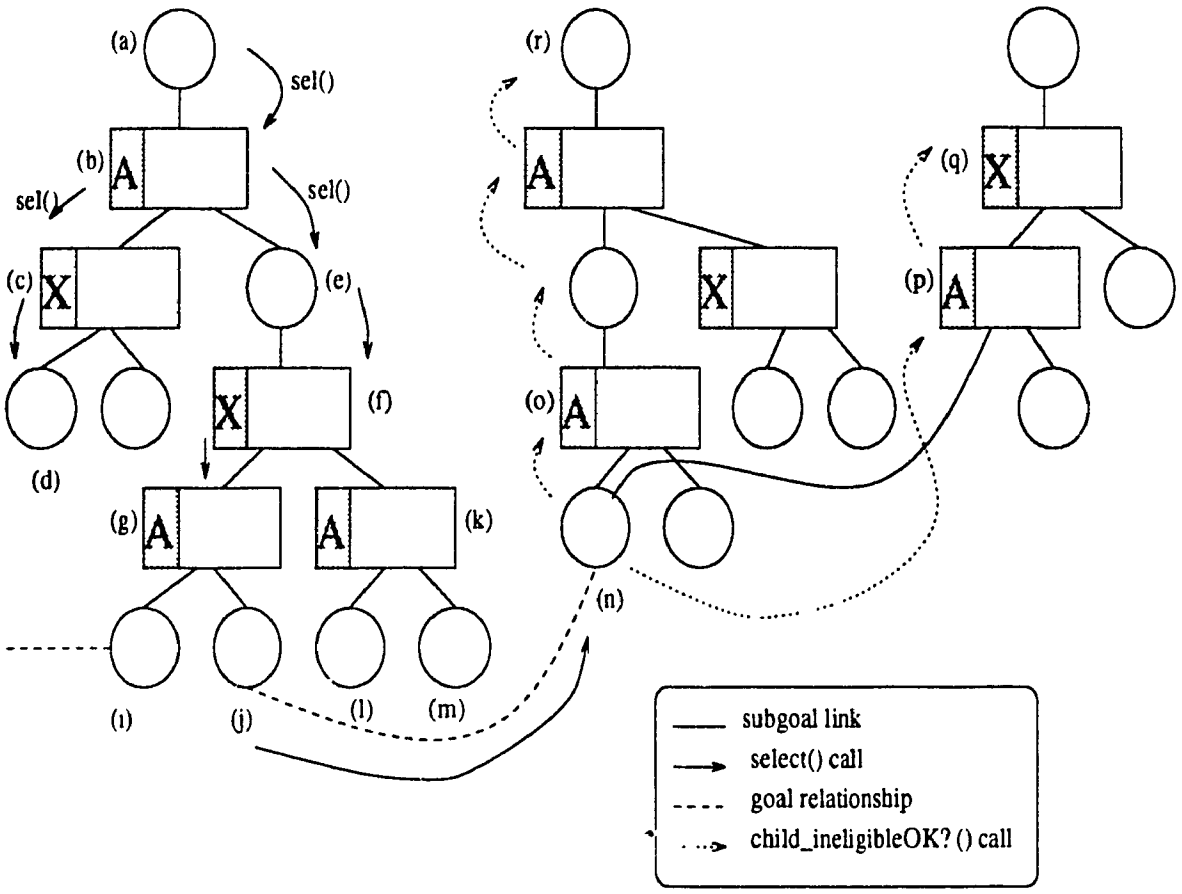


Figure 4.11: Creating a Joint Plan from the CPS

compromise value transmitted to the compromise that would occur in B's hierarchy. If goal B is already ineligible, it is made ineligible again and no compromise needs to be made in goal A's hierarchy. If goal B is eligible, the compromise value which would be made in its hierarchy must be determined from elements in the CPS higher than goal B in the hierarchy. The element above goal B in the hierarchy may also not be able to calculate the compromise that would occur. Eventually, if this message passes all the way to the top of the hierarchy, the cost is weighed against losing a hierarchy from the Joint Plan. We elaborate through an example using Figure 4.11 to describe how a Joint Plan is created from a CPS.

A program trace of the Joint Plan creation

We use the compromise heuristic in this trace, because it is the more complicated of the two hierarchy ordering heuristics described. A few simplifying assumptions,

such as not showing goal ratings, were made so as not to overly complicate the figure. They will be discussed as they are encountered.

The `select()` routine is the main routine used in creating the Joint Plan. `select()` is called on the top-level goal in a hierarchy, and it is called on successively lower elements in the hierarchy according to the selection rules for XORLists and ANDLists until either goals with no subgoal lists have been selected, or `select()` can not choose a goal. `select()` may fail due to goal relationships among the goals. The `select()` routine has two parameters: the first is the compromise value, which is the rating loss local to that area of the hierarchy that will occur if `select()` fails. The second is the plan type (either coherent, competitive, or conflict-free), and is used at each goal to determine what kind of relationships must be resolved during Joint Plan creation. This second parameter is not important to the discussion here, and will not be mentioned again. Similarly, the maintenance of counters in the various elements of the CPS as routines are called will not be mentioned. An example would probably best explain the procedure. In Figure 4.11, the goals and Logical Lists have been specially labelled for this discussion. To make the figure less cluttered, the return values for the function calls have not been indicated.

At point (a), the `select()` routine is called on the top level goal, which has no goal relationships. This goal is considered selected for the Joint Plan if `select()` called upon its subgoal list succeeds. The compromise value sent at this point is the maximum value, because if `select()` does not succeed at this goal, the hierarchy will not be selected. The top level goal then calls select on its subgoal list at point (b). The ANDList `select()` routine, requiring all of its elements to be selected for it to succeed, calls `select()` first on the XORList at point (c). XORList `select()` only requires one element to be selected from it for it to succeed, so it calls `select()` on its highest rated element which is the goal at point (d) (I have defined this element as the highest rated element, although ratings were left out of the figure for clarity). The compromise value sent in the `select()` routine has been changed by the XORList to the difference in ratings between the two goals on this XORList. This goal has no goal relationships, has no subgoal list, and so it returns SUCCESS. The XORList at point

(c), now returns SUCCESS as well, and the ANDList at point (b) calls `select()` on its other element, the one at point (e). The compromise value sent is the maximum value again. The goal at (e) has no goal relationships, but because it has a subgoal list it must call `select()` on it. `select()` is called on the XORList at (f), it calls `select()` on its highest rated element at (g). The compromise value passed is equal to the difference in rating of the two elements on the XORList. The goal at (i) has an adverse goal relationship with a goal not in our picture, the other goal is made ineligible, and so `select()` succeeds on the goal at (i). `select()` from (i) returns SUCCESS to the ANDList at (g), and `select()` is called on the goal at (j).

The goal at (j) has an adverse relationship with the goal at (n). The routine `mark_ineligible()` is called on the goal at (n), with the compromise value parameter the same as was passed in the `select()` routine. At the goal at (n) the consequences of it being marked ineligible are unknown, this can only be determined higher in the hierarchy. The goal at (n) calls the routine `child_ineligibleOK?()` on the elements immediately higher than it in the hierarchy, passing them the compromise value it was sent. Both of these elements must agree that it can be marked ineligible before that can occur. If one of them returns FAILURE, the goal returns FAILURE to `mark_ineligible()`. Note that one of these elements is in the second hierarchy, the other parent is in the third hierarchy.

In the third hierarchy at point (p), the ANDList there knows both of its elements must be selected and it can not lose one and still succeed in the future. However, there may be another point higher in the hierarchy where a compromise can be made. It calls `child_ineligibleOK?()` on the XORList at (q). The XORList has two elements, either one of which may be selected. One element can be lost and it could still potentially succeed if `select()` was called on it. The XORList compares the compromise value sent in `child_ineligibleOK?()` to the compromise it would have to make by choosing the lower rated of its two elements. In this case, the compromise at the XORList is smaller, so the XORList returns SUCCESS from `child_ineligibleOK?()`. This value returns through the ANDList at (p), and ends up at (n).

In the second hierarchy, where the other parent is, `child_ineligibleOK?()` is called on the ANDList at (o). Because no XORLists are between point (o) and the goal at the top of the hierarchy, `child_ineligibleOK?()` is called on each element above the ANDList at (o) until `child_ineligibleOK?()` is called on the goal at the top of the hierarchy at (r). Because the goal is at the top of the hierarchy, if `child_ineligibleOK?()` is called upon it, then if SUCCESS is returned it means that this hierarchy can not be selected in the future (because an element necessary for `select()` on its subgoal list to succeed will be ineligible for inclusion in the Joint Plan). The goal compares the compromise value sent in `child_ineligibleOK?()` to the value of its hierarchy. If the hierarchy has a rating less than the compromise value passed, it returns SUCCESS, otherwise it returns FAILURE. The details of the computation of the value of a hierarchy are not discussed here, but in this case the hierarchy is rated higher (again, ratings not shown, we define it that way) than the compromise value, and FAILURE is returned. This FAILURE result goes down the hierarchy back to point (n), where FAILURE is returned as the result of the goal at (j) calling `mark_ineligible()` on the goal at (n). The figure corresponds to the state of the function calls at this point.

The goal at (j) can not be selected, and it returns FAILURE as a result of the `select()` call on it from the ANDList at (g). An ANDList must have all its elements selected, and because one has failed, `unselect()` is called on the goal at (i) that was previously selected. Goal relationships that were previously resolved in favour of the goal at (i) are reversed. If there was a subgoal list on the goal at (i), `unselect()` would be called on that subgoal list as well.

The `select()` routine returns FAILURE from the ANDList at (g) to the XORList at (f). Because the XORList was not able to `select()` its highest rated element, there is only one element left, and the compromise value it was passed was the maximum value, it calls `select()` on the other element. This time, because it is the only element left, the compromise value it passes is the same one it received, namely, maximum compromise. In this case, `select()` is called on the ANDList at (k), it calls `select()` on the goals at (l) and (m), they succeed, and SUCCESS is eventually

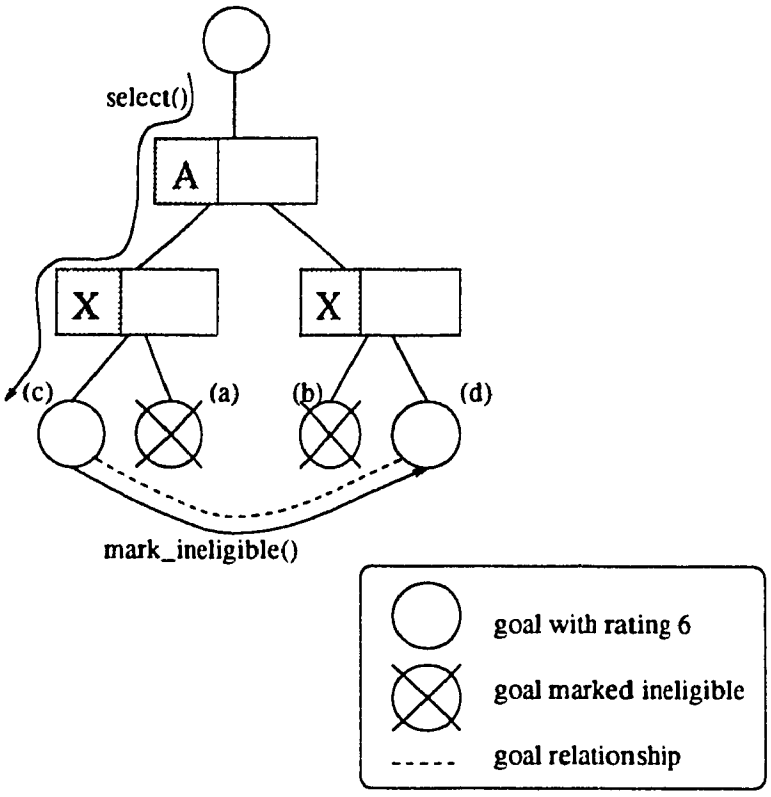


Figure 4.12: `select()` fails gracefully in an overconstrained problem

returned all the way up the hierarchy to eventually reach the goal at (a). The top level goal, having received the result SUCCESS to its `select()` call on its subgoal list, is included in the Joint Plan, along with any other goal on which `select()` was called and succeeded, provided `unselect()` was not called on the element after `select()`.

End of Program Trace

The set of routines written for `select()` to operate have been written so that `select()` will fail on a top level goal only if there are goal relationships within the hierarchy that can not be resolved without making the hierarchy ineligible, ie., the problem is over-constrained [1]. That `select()` will succeed otherwise is guaranteed by making the compromise value sent in `select()` from the top level goal equal to the

[1]The problem is over-constrained in the sense that nothing short of retracting all previous goal relationship resolutions involved in the given hierarchy could *potentially* prevent the hierarchy from being made ineligible. We have stated previously that a full backtracking-type algorithm is very costly, for little potential benefit.

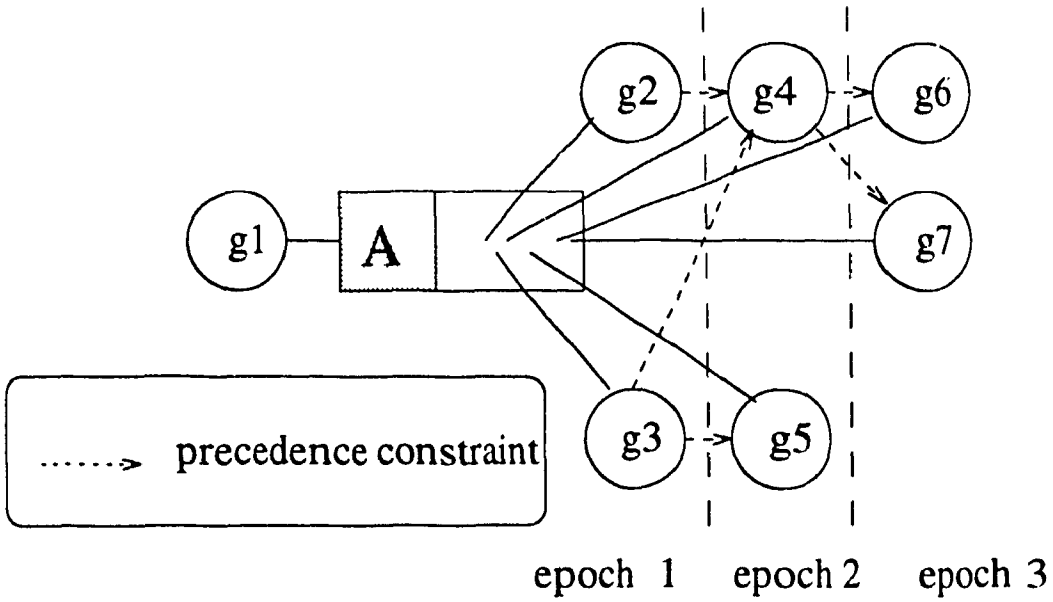


Figure 4.13: Joint Plan

highest possible compromise value, which is greater than the maximum rating for any hierarchy. A simple case where `select()` fails is shown in Figure 4.12. The goals at (a) and (b) have been marked ineligible by previous runs of `select()` on other hierarchies (not shown). `select()` is called on the top level goal, it in turn calls `select()` on the ANDList, which calls `select()` on the left hand XORList. The goal on that XORList calls `mark_ineligible()` on the goal at (d), and `child_ineligibleOK?()` moves up the hierarchy until it reaches the top level goal. In this situation, the top level goal recognizes that the problem is over-constrained because `select()` is still active and the compromise value passed in `child_ineligibleOK?()` is a maximum, so it returns FAILURE.

The goals that remain in the CPS after the completion of the pruning operation are grouped into *epochs* (see Figure 4.13). We define epochs using the chains of precedence constraints among the goals in the CPS: the goals in the CPS that have no precedence constraints are in epoch 1; goals that have precedence constraints with *only* the goals in epoch 1 are in epoch 2. In general, a goal is in epoch $n+1$ if from among those goals that precede it, the maximum epoch is n.

Epochs partition the set of goals in the CPS based upon the sequence in which they would be attempted if they are included in the Joint Plan. Epoch i containing

n goals will have a rating given by:

$$RE_i = \frac{\frac{\sum_1^n D_j}{n}}{(MAX[R_j] \times K_i)} \qquad j = 1, \ldots, n$$

The rating of each epoch is affected by its index in the sequence of epochs. The risk associated with each epoch is adjusted to reflect the fact that the success of the epoch is dependent upon the preceding epochs. The prediction made by an expert system for the likelihood of success of the goals in an epoch must be decreased as its position in the sequence of epochs increases. K_i is the adjustment factor applied to the risk of $epoch_i$ As K_i increases, the rating of $epoch_i$ (RE_i) decreases.

The value for K is defined recursively, we use

$$K_1 = 1$$

$$K_2 = 1 + \frac{Risk_{ep_1}}{10}$$

$$K_i = \left(1 - \frac{(Risk_{ep_{i-1}} - Risk_{ep_{i-2}})}{10}\right) K_{i-1} \quad for\ i = 3.4, 5 \ldots$$

where $Risk_{ep_i}$ is the maximum risk of the goals in epoch i.

K_1 is set to one because there are no previous epochs. K_2 is larger than K_1 to reflect our belief that the second epoch should be devalued compared to that of the first epoch, since it depends upon it. From K_3 onward, we use the differential of the maximum risk of the last two epochs to have K vary with the slope of the maximum risk differential. K_i depends on the maximum Risk of an epoch because as previous epochs become more risky, the likelihood of continuing to the next epoch diminishes. hence we do not wish to weigh the epochs uniformly.

The length of the Joint Plan is decided by using the rating of each epoch. First epoch 1 is included in the Joint Plan. Subsequent epochs in the sequence of epochs are included in the Joint Plan only if their ratings are greater than or equal to the rating of the previous epoch. The rating assigned to the Joint Plan with n epochs is

$$RATE_{JP} = \frac{NH_{JP}}{NH_{CPS}} \sum^n RE_i$$

where NH_{JP} is the number of goal hierarchies in the Joint Plan and NH_{CPS} is the number of hierarchies in the CPS. The multiplication factor NH_{JP}/NH_{CPS} decreases

the rating of those Joint Plans that have fewer goal hierarchies. This is meant to reflect our belief that a Joint Plan should try to achieve as many goals as possible.

4.3 Election

Each expert system calculates $RATE_{JP}$ for each of the Joint Plans created by the members of its Consensus Group. Then an election is held to determine which Joint Plan is to become the Final Plan. Election techniques can be optimized for different purposes; for example, the techniques reported in [35] minimize the number of messages exchanged between the expert systems during the election process. The Consensus protocol will use a simple election technique reported in [35] in which all expert systems rate all the Joint Plans and broadcast these ratings to all the other expert systems in the Consensus group. A cumulative rating for each Joint Plan is taken and the Joint Plan with the highest cumulative rating is selected as the Final Plan. Because the expert systems all use the same technique to select the Final Plan, they do not need to further communicate to know which Joint Plan has been selected as the Final Plan because they will all pick the same one. In the case of a tie between the cumulative ratings for two or more Joint Plans, as a convention, the Joint Plan created by the expert system with the lowest identifier is chosen.

In general, an election based on simple majority does not guarantee that the best option is chosen, but we believe such an election is adequate in the context of Consensus. In Consensus, an election is held on the set of Joint Plans and <u>not</u> on Proposals. Each Joint Plan has been created by each expert system using the information in its own window, and the information supplied to it in the Proposals created by the other expert systems. The Joint Plans are constructed using the information supplied by all the expert systems in the Consensus Group; and thus, Joint Plans reflect compromises proposed by the expert systems. In the context of Consensus, how often an election based on simple majority rule produces sub-optimal choices is left as future work.

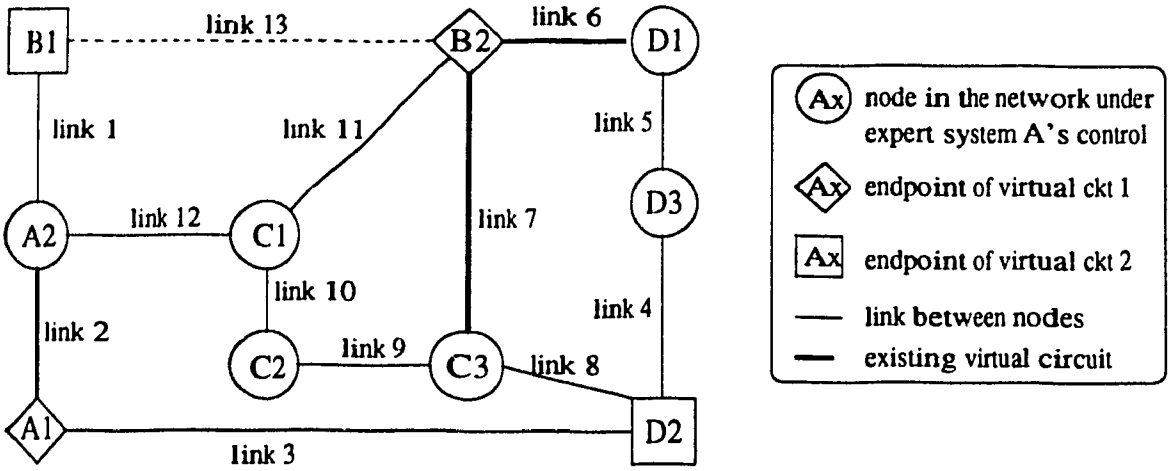


Figure 4.14: Virtual Circuit Routing

4.4 Example of Consensus

In this section, we demonstrate the use of the Consensus protocol when a CDPS is used to solve the problem of constructing virtual circuits in a telecommunications network. A simpler varia ion of this problem is described by Conry et al. [6]. The communications network shown in Figure 4.14 is partitioned into four geographical regions: A, B, C, and D. Each region consists of several switching sites called nodes, and is controlled by an expert system.

Each expert system knows only about the nodes in its region and about the nodes which have a direct connection to the nodes in its region. Problem-solving requires finding a path through the network for each virtual circuit that is requested, with the condition that a maximum of two virtual circuits may use any given link. We augmented the problem as stated in Conry et al. to include the notion that there is no guarantee that the goal 'allocate a link for a virtual circuit' will succeed. We use three levels of goals corresponding to different levels of abstraction. We use a "virtual circuit" level, an "abstract" level corresponding to connecting two nodes with possibly some restrictions on the links which can be used, and a "concrete" level in which the link to use is specified. An expert system is unaware of the details of the routing through another expert system's area of responsibility, so it may propose goals that are abstract, and left for another expert system to satisfy. These abstract

goals simply specify the endpoints that must be connected without detailing how this is to be done, except for perhaps some restrictions on links that may be used. These restrictions are used to ensure that an abstract goal is not subgoaled into concrete goals which specify a virtual circuit that backtracks over a link in the network which has already been allocated to the virtual circuit.

In our example, the Consensus is signalled because two virtual circuits have failed, due to a break in link 13 between B1 and B2. Virtual circuit 1 must connect node A1 to B2, and virtual circuit 2 must connect node B1 to D2.

The Proposals created by the expert systems for 'repairing' the virtual circuits are shown in Figure 4.15, and the goals are explained in Table 4.2.

In the Proposal produced by expert system A, the goal at the top of the first hierarchy (g1) has the specification 'restore virtual circuit 1'. Expert system A proposes three ways of restoring virtual circuit one, and so this goal has an XORList as a subgoal link. Each of the ways of restoring virtual circuit 1 is on an ANDList connected to the XORList. The reason each is on an ANDList is because it would not make sense to restore part of a virtual circuit using one routing, and restoring another part using another routing. The overall routing of the virtual circuit must be coordinated through multiple parts of the network so that those links going through multiple regions of the network connect together to form a continuous path. The subgoals of the left hand ANDList are abstract goals which specify that to satisfy the root goal of restoring virtual circuit 1 (which goes from node A1 to node B2), a connection should be made between nodes A1 and C1 (g2), and nodes C1 and B2 (g3). Because g2 lies within expert system A's area of responsibility, expert system A can subgoal it. It specifies that to establish a connection between nodes A1 and C1, a connection between A1 and A3 over link 2 (g4) and a connection between A2 and C1 over link 12 (g5) can be made. The requirement that g3 not use link 2 or link 12 is made so that an attempt is not made to reuse links already allocated to this virtual circuit. The second method of restoring virtual circuit 1 is to connect nodes A1 and D2 (g6) and D2 to B2 (g7). The third method is to connect nodes A1 to B1 (g8) and nodes B1 to B2 (g9). g8 can be subgoaled to g4 and g10. Note that

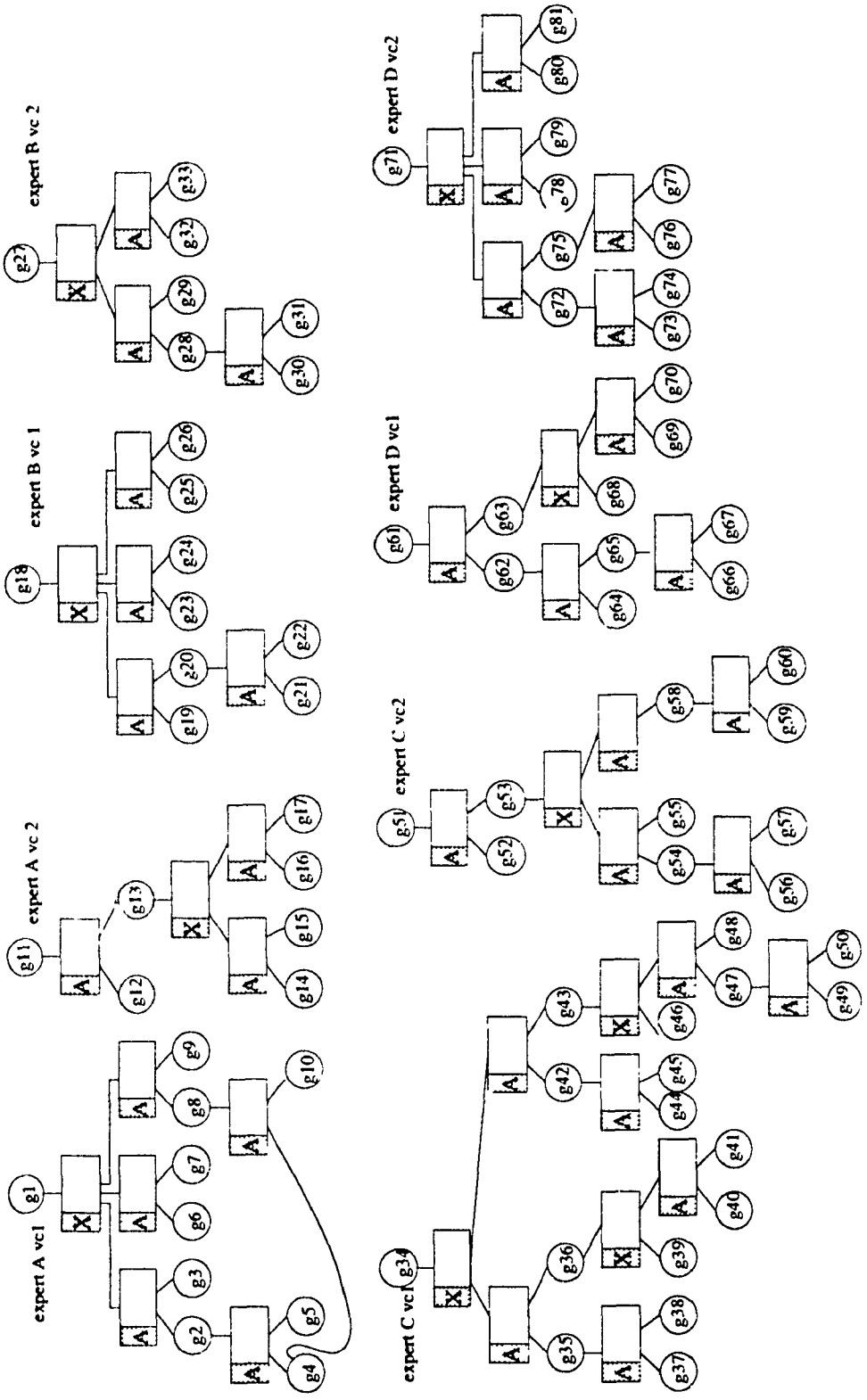


Figure 4.15: Proposals for virtual circuit restoral

at this point in time, expert system A is proposing connecting nodes B1 and B2, it is unaware that the link connecting those two nodes has failed.

The top level goal (g11) in the second hierarchy in expert A's Proposal is to restore virtual circuit 2, which connects node B1 to D2. A single way to restore the virtual circuit is in the Proposal, so there is no XORList subgoal link on g11. g11 is subgoaled into goal g12 which connects nodes B1 and A2 through link 1, and goal g13 which is an abstract goal to connect nodes A2 and D2. There are two ways of subgoaling g13, so there is a subgoal link from g13 to an XORList. On the XORList are two ANDLists. The left hand side ANDList has goal which connect nodes A2 and A1 through link 2 (g14), and connect nodes A1 and D2 through link 3 (g15). The other ANDList contains goals to connect nodes A2 and C1 through link 12 (g16), and an abstract goal to connect nodes C1 and D2 (without using links 1 or 12).

The Proposals produced by the other expert systems are different from A's Proposal. While the top goals of each Proposal are the same, the Proposals differ in the routes specified for restoring virtual circuit 1 and 2. Expert system C proposes routing virtual circuit 1 through its region because it has unused capacity available on its links. Expert system D proposes routing virtual circuit 1 through node D2. One interesting thing to note is that expert system B has a goal in its Proposal (g19) to connect nodes B1 and B2, and it has rated the *LS* of g19 very low because the link connecting those two nodes has failed. By including g19 in its Proposal and rating it poorly, expert system B indicates to the other expert systems in its Consensus Group that there is a problem with the B1 to B2 link. The CPS created by each of the experts is shown in Figure 4.16.

In constructing the CPS, the expert systems have merged the top goals of each of the Proposals because they have the same specification. Other goals have been merged as well, such as g5 and g38, which both specified connecting A2 and C1 through link 12. g2 and g35 both abstractly specified connecting A1 and C1. By merging these goals into g2, g2's subgoal link becomes an XORList onto which the two ANDLists are placed.

The subgoals of the top goals which specify 'restore circuit 1 and restore circuit 2'

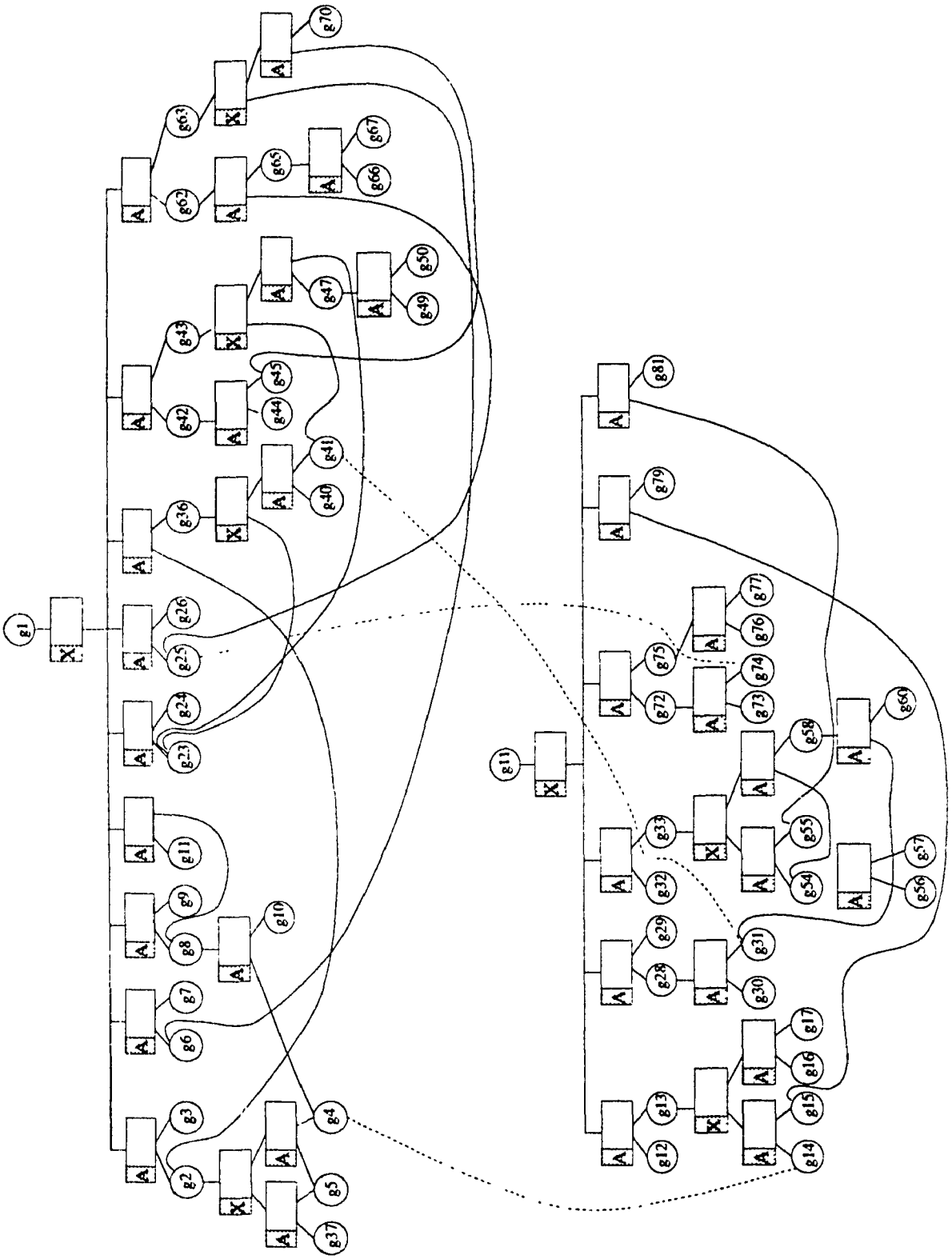


Figure 4.16: Common Planning Structure

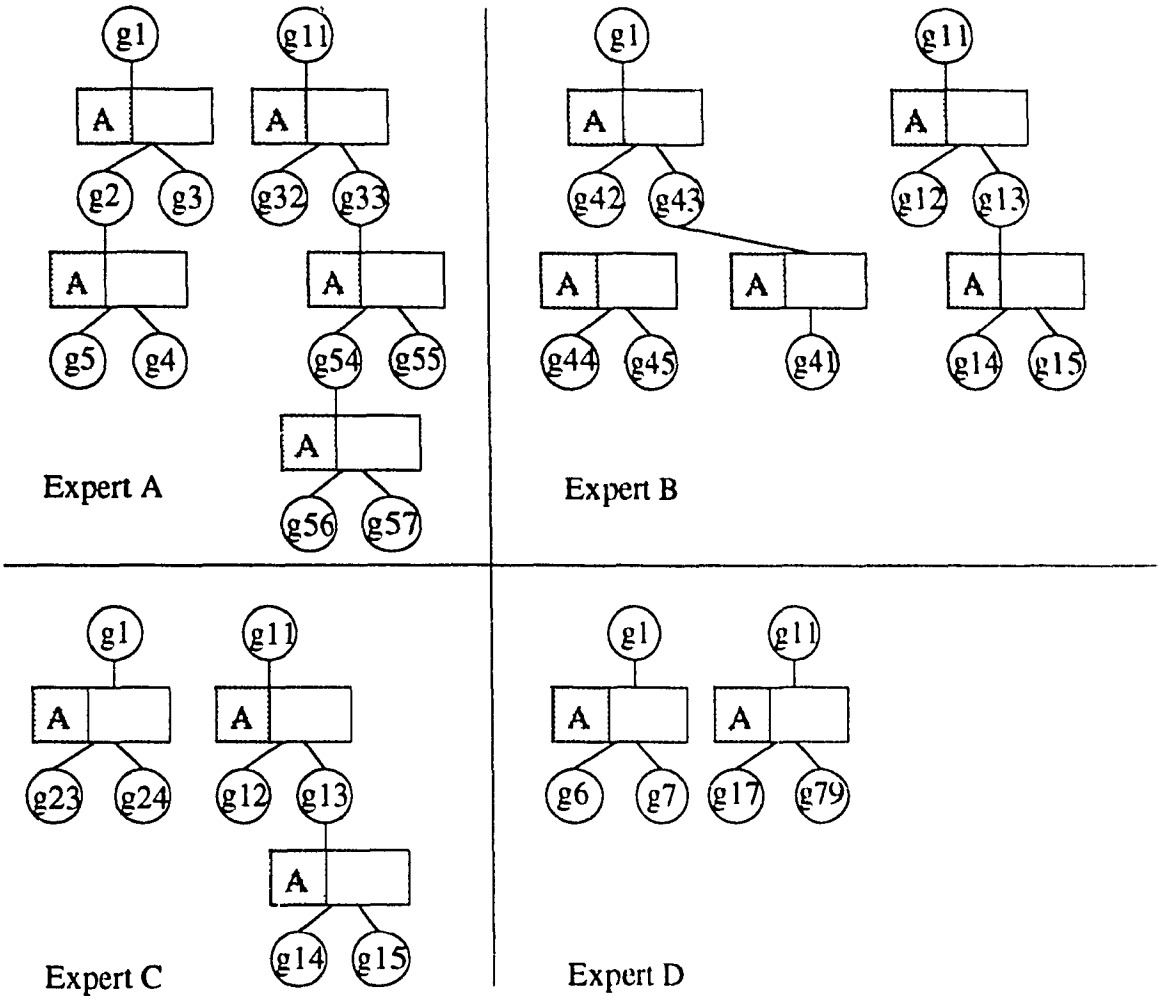


Figure 4.17: Joint Plans

were also present in all of the Proposals, and have been merged as well. The various alternatives for restoring virtual circuit 1 that are in the Proposals of the expert systems have been placed on an XOR list. The alternatives for restoring virtual circuit 2 have been treated in the same manner.

The expert systems will prune the CPS to create their Joint Plans as shown in Figure 4.17. Expert system A in pruning the CPS (see Figure 4.16) examined the options for restoring virtual circuit 1. The subgoals for restoring circuit 1 were attached to the XORList beneath goal 1. Expert system A sees the left-most ANDList as the highest rated, and so chooses goal g3 and goal g2. This corresponds to restoring virtual circuit 1 through expert system A and C's regions, and then connecting a node

in expert C's region to B2. g2 is further subgoaled with an XORList subgoal link, so expert system A must continue. It chooses the ANDList on the right hand side of the XORList which contains goals g4 and g5. Goal g4 is the connection of nodes A1 and A2 through link 2. Because this link has an existing circuit present, there is a resource conflict between the allocation of that link for restoring virtual circuit 1 and the allocation of the same link for restoring virtual circuit 2. Consequently, expert system A can not plan to allocate that link for virtual circuit 2. Expert system A, having completed its plans for restoring virtual circuit 1, now begins choosing a plan for restoring virtual circuit 2 by choosing goal g11 at the top of the other hierarchy in the CPS.

After choosing g11, expert system A chooses the XORList which in turn leads to choosing the ANDList containing goals g32 and g33. This routes the virtual circuit from B1 to A2 over link 1, and then routing from A2 to D2 respectively. Goal g33 is further subgoaled through an XORList. The left hand ANDList of that XORList is chosen, leading to goals g54, g55, g56, and g57 being chosen. Thus, the circuit is routed from A2, the endpoint of g32, to C1, then from C1 to C3, and finally from C3 to D2, the endpoint of the virtual circuit.

The Joint Plans created by the other expert systems will have differences because of the different local context of each expert system. Expert B creates a Joint Plan which routes virtual circuit 1 from A1 through D2 to C3, and then to B2. Virtual circuit 2 is potentially routed from B1 to A2, then A2 to A1, and then to D2. Expert system C creates a Joint Plan which routes virtual circuit 1 from A1 to C1, and then using link 11 to go from C1 to B2. For virtual circuit 2, expert system C has chosen the same path as expert system B. Finally, expert system D has chosen a path for virtual circuit 1 from A1 to D2 over link 3, and then abstractly from D2 to B?. For virtual circuit 2, it chooses a path from B1 to A1, and then from A1 to D2. Each of the experts systems now broadcasts its Joint Plan to the other members of the Consensus group, and an election to select the Final Plan is held.

This example illustrates several features of the Consensus protocol. The resource conflicts are explicitly represented allowing for their resolution when the Joint Plans

are created, and equivalent goals are merged. An expert system can propose one method for solving a problem but choose a method proposed by another expert in its Consensus Group when creating its Joint Plan.

4.5 Analysis

The primary strength of the Consensus protocol is the ability to rapidly propose and choose, in a distributed manner, a plan for all the experts to follow. All experts have input on both the goals which are proposed for the Joint Plan, and the selection of the Final Plan. The Consensus protocol produces a plan for all experts to follow in a fixed number of stages, as cpposed to a negotiation type strategy employed by other protocols [13, 6]. Although negotiation can potentially produce a better plan, its cost can be higher. In the context of incremental planning and ill-structured problems in which the outcome of actions is uncertain, a high cost of planning may not be justified.

The various plan types defined (ie., coherent, competitive, conflict-free) allow a static selection of the characteristics of the Final Plan that may prove useful. The plan types allow a trade-off between planning time and plan quality. However, we don't believe that any one plan type will necessarily prove to be consistently faster, nor execute faster than any other. The coherent plan type will result in the most number of adverse relationships, and so planning will take longer. This may be offset by an increased "quality" of Final Plan. On the other hand, the conflict-free Final Plan should take less time for Consensus to create because fewer goal relationships will be considered adverse, but a poorer quality of Final Plan may be produced. This is best determined by experimental verification.

The static beliefs that one expert system has of another are a simplification that may adversely impact the quality of the result of the Consensus Protocol. It seems clear to me that if a problem is divided into areas of responsibility for each expert system, the knowledge that one expert system has of another should be minimized in order to remain within the expert system's bounded rationality. A preferable method of taking into account other expert system's ratings would be to allow an

expert system to express a confidence in its rating. That way, the ratings given by an expert system in areas which it knows little about would not impact significantly on the overall rating. The ratings are important, because most of the major decisions in creating a plan come from them. It is the only way to express that one goal is preferable to another.

The technique used to develop a Joint Plan responds in some ways to beneficial goal relationships as well as adverse ones. Wilensky's [42] belief was that plan creation should come about as a result of beneficial relationships between goals, and in this respect Consensus achieved limited success with the use of ANDLists. However, ANDLists are used only for goals in the same Proposal. Unfortunately, other beneficial goal relationships are not used. Adverse goal relationships, where only one of a set of goals were handled both by using goal relationships, and in collectivities using XORLists. My impression however is that no domain-independent protocol will ever handle relationships perfectly.

A potential weakness of the Consensus protocol is that experts are unable, during an election, to vote on pieces of the plan. It is possible that several experts view a part of a plan as undesirable despite seeing the overall plan as of the highest ranking. This will result in selecting the "best" of the proposed plans, but one that is less than optimal. We don't know how often this situation will present itself. A simple solution would be not to include that part of the Final Plan which is not highly rated by all expert systems in the Consensus Group. However, this could lead to problems if the part of the Final Plan that was not accepted by all the expert systems was large with respect to the rest of the Final Plan.

The Consensus protocol, because it performs in a finite number of stages, can be scaled fairly well. Neglecting the reduced number of messages sent in the election stage, because of the techniques discussed for reducing the messages in an election, the message complexity is simply $n(n-1) + n(n-1) + n(n-1) = O(n^2)$, accurate to some constant depending on message size. Each of these factors is due to one stage of the protocol, where each of n expert systems sends a message to $(n-1)$ other expert systems. If broadcast facilities are available in the network, the message

complexity reduces to $O(n)$. Regardless of the scalability of the Consensus protocol, it is unlikely that a Consensus group would be large, because the organization of a large number of expert systems would be reconfigured into numerous Consensus groups. One disadvantage of the message sending in the Consensus protocol is that it occurs in bursts, resulting in non-uniform loading of the communication network.

name	connect	type	for	using	name	connect	type	for	using
g1	A1,B2	VC	ckt1		g42	A1,C3	ABS	ckt1	
g2	A1,C1	ABS	ckt1		g43	C3,B2	ABS	ckt1	$\neg l8$
g3	C1,B2	ABS	ckt1	$\neg l12, \neg l2$	g44	A1,D2	ABS	ckt1	
g4	A1,A2	CON	ckt1	$l2$	g45	D2,C3	CON	ckt1	$l8$
g5	A2,C1	CON	ckt1	$l12$	g46	C3,B2	CON	ckt1	$l7$
g6	A1,D2	CON	ckt1	$l3$	g47	C3,C1	ABS	ckt1	$\neg l8$
g7	D2,B2	ABS	ckt1	$\neg l3$	g48	C1,B2	CON	ckt1	$l11$
g8	A1,B1	ABS	ckt1		g49	C3,C2	CON	ckt1	$l9$
g9	B1,B2	ABS	ckt1	$\neg l1, \neg l2$	g50	C2,C1	CON	ckt1	$l10$
g10	A2,B1	CON	ckt1	$l1$	g51	B1,D2	VC	ckt2	
g11	B1,D2	VC	ckt2		g52	B1,A2	ABS	ckt2	
g12	B1,A2	CON	ckt2	$l1$	g53	A2,D2	ABS	ckt2	
g13	A2,D2	ABS	ckt2	$\neg l1$	g54	C1,C3	ABS	ckt2	
g14	A2,A1	CON	ckt2	$l2$	g55	C3,D2	CON	ckt2	$l8$
g15	A1,D2	CON	ckt2	$l3$	g56	C1,C2	CON	ckt2	$l10$
g16	A2,C1	CON	ckt2	$l12$	g57	C2,C3	CON	ckt2	$l9$
g17	C1,D2	ABS	ckt2	$\neg l1, \neg l12$	g58	C3,D2	ABS	ckt2	$\neg l9, \neg l10$
g18	A1,B2	VC	ckt1		g59	C3,B2	CON	ckt2	$l7$
g19	B2,B1	ABS	ckt1		g60	B2,D2	ABS	ckt2	$\neg l7$
g20	B1,A1	ABS	ckt1		g61	A1,B2	VC	ckt1	
g21	B1,A2	CON	ckt1	$l1$	g62	B2,D2	ABS	ckt1	
g22	A1,A2	CON	ckt1	$l2$	g63	D2,A1	ABS	ckt1	$\neg l6$
g23	B2,C1	CON	ckt1	$l11$	g64	B2,D1	CON	ckt1	$l6$
g24	C1,A1	ABS	ckt1	$\neg l11$	g65	D1,D2	ABS	ckt1	$\neg l6$
g25	B2,D1	CON	ckt1	$l6$	g66	D1,D3	CON	ckt1	$l5$
g26	D1,A1	ABS	ckt1	$\neg l6$	g67	D3,D2	CON	ckt1	$l4$
g27	B1,D2	VC	ckt2		g68	D2,A1	CON	ckt1	$l3$
g28	B1,C3	ABS	ckt2		g69	D2,C3	CON	ckt1	$l8$
g29	C3,D2	ABS	ckt2	$\neg l7, \neg l13$	g70	C3,A1	ABS	ckt1	$\neg l8, \neg l6$
g30	B1,B2	CON	ckt2	$l13$	g71	B1,D2	VC	ckt2	
g31	B2,C3	CON	ckt2	$l7$	g72	B1,D1	ABS	ckt2	
g32	B1,A2	CON	ckt2	$l1$	g73	B1,B2	ABS	ckt2	
g33	A2,D2	CON	ckt2	$\neg l1$	g74	B2,D1	CON	ckt2	$l6$
g34	A1,B2	VC	ckt1		g75	D1,D2	ABS	ckt2	$\neg l6$
g35	A1,C1	ABS	ckt1		g76	D1,D3	CON	ckt2	$l5$
g36	C1,B2	ABS	ckt1	$\neg l12$	g77	D3,D2	CON	ckt2	$l4$
g37	A1,A2	ABS	ckt1	$\neg l12$	g78	D2,A1	CON	ckt2	$l3$
g38	A2,C1	CON	ckt1	$l12$	g79	A1,B1	ABS	ckt2	$\neg l3$
g39	C1,B2	CON	ckt1	$l11$	g80	D2,C3	CON	ckt2	$l8$
g40	C1,C3	ABS	ckt1	$\neg l7, \neg l12$	g81	C3,B1	ABS	ckt2	$\neg l8$
g41	C3,B2	CON	ckt1	$l7$					

Table 4.2: Goal Specifications

Chapter 5

Hierarchy Ordering Heuristics Experiment

They said therefore unto him, What sign shewest thou then, that we may see, and believe thee? what dost thou work? John 6:30-31

The Consensus protocol consists of several stages: Proposals are exchanged by the expert systems in a Consensus Group, a Common Planning Structure (CPS) is created from the Proposals, the CPS is pruned to create Joint Plans, and then an election is held to select the Final Plan. When the expert systems are pruning the CPS, they are attempting to create Joint Plans that have the highest rating. While pruning the CPS, the expert systems must also resolve the *conflicts* that exist among the goals contained in the CPS. These conflicts are indicated by various goal relationships.

In Chapter 4, three alternative approaches were discussed for pruning the CPS: the full backtracking approach, and the two hierarchy ordering heuristics, namely High-Low and Compromise. It was believed that the full backtracking approach, while being the simplest and most exhaustive, would become intractable if there were many goal relationships in a CPS. The ordering heuristics were designed to produce better Joint Plans without incurring the high computational cost of the full backtracking approach.

The heuristics place an ordering upon the sequence in which goal relationships

are resolved and avoid the computational expense incurred with the backtracking approach. However, there is a cost for computing the ordering. The ordering heuristics will produce Joint Plans that are not as good as those that would be produced using the full backtracking approach. The assumption is that the ordering heuristics increase the quality of the Joint Plans produced when compared to the Joint Plans produced by resolving the goal relationships simply in the order they appear in the CPS. In addition, it has been assumed that the Compromise heuristic would preserve more hierarchies from the CPS while creating Joint Plans (and ultimately Final Plans), than the other heuristics.

In addition to the assumptions previously stated, it is believed that as the number of goal relationships in the CPS increases, the gap between the performance of the ordering heuristics and the effectiveness of resolving the goal relationships in the order in which they appear in the CPS will increase. Of course, the structure of the Proposals will also be a factor in the performance of the heuristics. The experiment described in this section will measure the effectiveness of the ordering heuristics as the number of goal relationships in the CPS increases using a set of randomly generated Proposals. The Proposals are generated randomly using a set of parameters describing the characteristics of their structure (a more complete description of the parameters for the generation of Proposals is provided later in this section).

The next section describes the design of the experiment, and what we hoped to learn from it. Following that, we present an experimental evaluation of the assumptions made about the effectiveness of the ordering heuristics, including a statistical analysis of the data. In the last section, we discuss the results of the experiment.

5.1 Experimental Design

The experiment is designed to investigate the cost and benefits obtained by using the ordering heuristics as compared to a random ordering for resolving goal relationships in the CPS. For this experiment, the number of goal relationships is set at five different levels. The experiment is conducted by randomly producing ten sets of sample Proposals according to a set of parameters describing their structure. The generated

Proposals are initially devoid of any goal relationships except for precedence constraints. The Proposals in each set are then augmented with goal relationships. Each set of Proposals is augmented five times producing Proposals with the same basic structure, but with a different number of goal relationships. The goal relationships are added randomly, but certain constraints are applied that eliminate the creation of goal relationships which are not *meaningful* (explained later). The fifty sets of Proposals are then processed twice as per the two ordering heuristics and finally resolving the goal relationships in a random ordering.

In this experiment, each randomly generated Proposal set is used to produce five Proposal sets that differ only in the goal relationships that they contain. Thus, the Proposal sets generated for this experiment are seen as having been drawn randomly from a population of Proposals defined by a specific set of parameters describing their structure. The Proposal sets have had two conditions applied to them during the experiment: the ordering heuristic used, and the level of goal relationships that was added. The design of this experiment is such that each randomly generated Proposal set is measured fifteen times as the different conditions of the experiment are applied to it. Thus, the data produced by this experiment is analysed using a Multivariate analysis of variance in order to determine if using different ordering heuristics and levels of goal relationships produces an effect. The goal of the experiment is to determine the following:

- Does the number of goal relationships in the Proposal set have an effect upon the benefit obtained when pruning the CPS?
- What is the overall trend in the benefit obtained as the number of goal relationships increase?
- Does the ordering heuristic used have an effect on the benefit obtained when pruning the CPS?
- Is there a difference in the benefit obtained using the random ordering as compared to the High-Low heuristic and the Compromise heuristic?

- Is there a difference in the benefit obtained using the High-Low heuristic as compared to the Compromise heuristic?
- Is the effect of the number of goal relationships on the benefit obtained different in the three ordering strategies?
- Is the effect of the number of goal relationships on the benefit obtained different for the random strategy as compared to the Compromise and the High-Low heuristic.
- Is the effect of the number of goal relationships on the benefit obtained different for the Compromise heuristic as compared to the High-Low heuristic.
- What is the trend for the benefit obtained when using each ordering strategy as the number of goal relationships increase.
- Does the number of goal relationships in the Proposal set have an effect upon the cost of pruning the CPS?
- What is the overall trend in the cost of pruning the CPS as the number of goal relationships increase?
- Does the ordering heuristic used have an effect on the cost of pruning the CPS?
- Is there a difference in the cost of pruning the CPS using the random ordering as compared to the High-Low heuristic and the Compromise heuristic?
- Is there a difference in the cost of pruning the CPS using the High-Low heuristic as compared to the Compromise heuristic?
- Is the effect of the number of goal relationships different on the cost of pruning the CPS in the three ordering strategies?
- Is the effect of the number of goal relationships on the cost of pruning the CPS different for the random strategy as compared to the Compromise and the High-Low heuristic.

- Is the effect of the number of goal relationships on the cost of pruning the CPS different for the Compromise heuristic as compared to the High-Low heuristic.
- What is the trend for the cost of pruning the CPS when using each ordering strategy as the number of goal relationships increase.

In order to investigate the effectiveness of the ordering heuristics, the costs associated with using them as well as the benefits obtained must be quantified. The measures used in this experiment are as follows:

Number of nodes visited in the CPS when pruning: A measure of the computation required for pruning the CPS.

Final Plan Rating: A measure of the quality of a Final Plan.

Total Compromise: Total Compromise is a measure of the value of the compromises that a given expert had to make when pruning the CPS. This measure is the sum of the compromise values associated with the Logical Lists where an alternative to the principle path had to be chosen when the CPS was pruned. In the case that the resolving of a goal relationship requires that a hierarchy is to be excluded from the Joint Plan the compromise value is taken as the *value* assigned for the hierarchy. The Total Compromise is also an indicator of the benefits obtained by the ordering heuristic. A smaller value for the Total Compromise that occurred when the CPS was pruned would indicate that more highly rated goals were retained in the Joint Plan.

Number of Hierarchies Preserved This metric is an indicator of the benefit obtained by using a particular ordering strategy as it indicates how successful the expert system was in including hierarchies in its Joint Plan.

The Proposal sets for this experiment are produced by a proposal generator that has been constructed for experimental research with the Consensus protocol. The

proposal generator constructs Proposal sets where the structure of the Proposals is controlled by a set of parameters. The parameters input to the proposal generator are as follows:

NUMEXPERTS Number of expert systems in the Consensus group.

NUMHIERARCH Number of hierarchies in the Proposal set.

ANDPROB, XORPROB, and GOALPROB The probability that the next object to be added to a Proposal is an ANDList, an XORList, or a Goal. These three probabilities must sum to one.

BRANCHFACT The typical number of subgoals attached to an XORList or an ANDList. This value is supplied as an upper and lower bound. The proposal generator will randomly assign a number of subgoals to a Logical List within these bounds.

RATE Upper and lower bounds for both the likelihood of success and desirability rating assigned to the goals in the Proposals.

DEPTH The maximum number of subgoal levels in the Proposals. This metric does not count Logical Lists.

MERGEFACTOR The ratio of equivalent goals to unique goals to be included in the Proposal set.

NUMSTAGES The number of epochs to be included in the Proposals.

STAGERATIOS The ratio of goals that will be in each stage.

MAXLEVEL The maximum number of levels permitted in a Proposal. This metric includes Logical Lists. It is used to prevent Proposals from attaining a depth which would become unwieldy.

REL_LEVELS The number of goal relationships in the Proposal set. This parameter can be a list indicating that Proposal Sets are to be generated with the same structure except for the goal relationships they contain. This level corresponds to the number of goal relationships in the Proposal / number of unique goals $\times$ 100.

The setting used for these parameters when generating the ten Proposal sets for this experiment are shown in Table 5.1. We selected NUMEXPERTS equal to four because it was thought that this provided a suitable number of expert systems for which to hold a Consensus. NUMHIERARCH was chosen to assure that a fair number of hierarchies were derived from each expert system. The probability of AND/XORLists and goals was chosen to reflect our beliefs in their likelihood. Certainly, Proposals without XORLists would not provide meaningful input to Consensus. The mean of ratings were chosen to have a uniform dispersion between 3 and 8. The number was then used as a mean to a normal distribution generator with a standard deviation of 1. This would ensure almost complete coverage of the rating range between 0 and 10. The branching factor was limited to a randomly chosen number between 2 and 3 because larger numbers created Proposals of enormous size. The DEPTH corresponds to the number of blackboard levels at which a goal can reside. A number randomly chosen between 3 and 5 seemed reasonable. The MERGEFACTOR, NUMSTAGES, STAGERATIOS were all set to what seemed to be "reasonable" levels so that the behaviour expected in the ordering heuristics could occur. MAXLEVEL was set to 4 to constrain the size of Proposals. Finally, the relationship levels were set so that they went from a small number (10%), all the way up to 50%. These goal relationship levels refer to the ratio of the number of goal relationships to the number of unique (ie., a goal that will appear in the CPS) goals in the Proposal. At the 50% level, on the average, every unique goal had a relationship (one relationship means two goals involved).

When the proposal generator creates Proposal sets, the rating that would be assigned to each goal by each expert system in the Consensus Group is in fact assigned by the proposal generator. At present, the proposal generator randomly assigns an initial rating for each goal using a uniform distribution. Using this initial rating as the mean for a normal distribution the rating of each expert system in the Consensus Group for the goal is generated by the proposal generator.

When the proposal generator adds goal relationships to the Proposal sets, it checks to ensure that the goal relationships are meaningful. The following relationships are

Parameter	Value
NUMEXPERTS	4
NUMHIERARCH	12
ANDPROB	0.1
XORPROB	0.2
GOALPROB	0.7
RATF	3 8 3 8
BRANCHFACT	RND 2 3
DEPTH	RND 3 5
MERGEFACTOR	50
NUMSTAGES	3
STAGERATIOS	0.3 0.2 0.1
MAXLEVEL	4
REL_LEVELS	10 20 30 40 50

Table 5.1: Parameters for the Proposal Generator

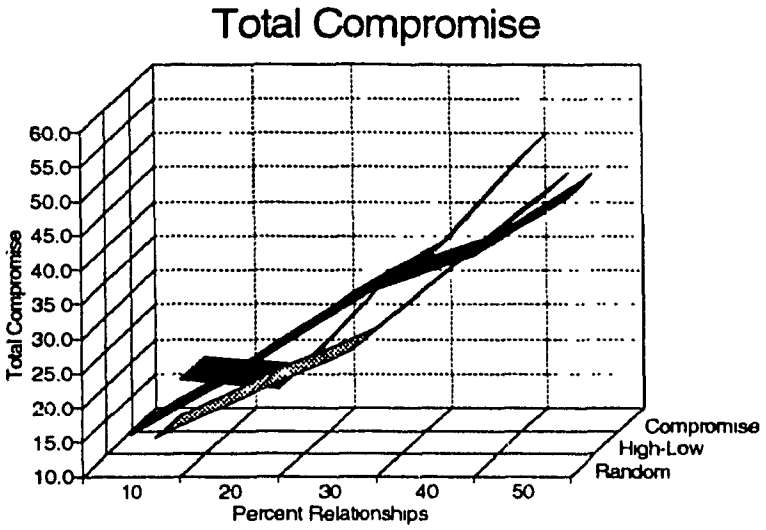
not permitted by the proposal generator, because the Proposal produced would be meaningless:

- A relationship between goals at different levels in the Proposals. Goals at different levels in the Proposal are a different levels of abstraction and in the sense of a goal relationship they are not comparable.
- A relationship between two goals that have the same specification. This relationship would be indicating that a goal is in conflict with itself.
- A relationship between two goals on an ANDList. This relationship would effectively mean the set of goals on the ANDList proposed by an expert system cannot be achieved. The expert system would be expected to detect this situation when creating its Proposal.

If a Proposal were to be created by the planner of an expert system with such a relationship this would be considered an error.

5.2 Experimental Results

The means and standard deviations for the Total Compromise for the three strategies are shown as a function of the number of goal relationships is shown in Figure 5.1.

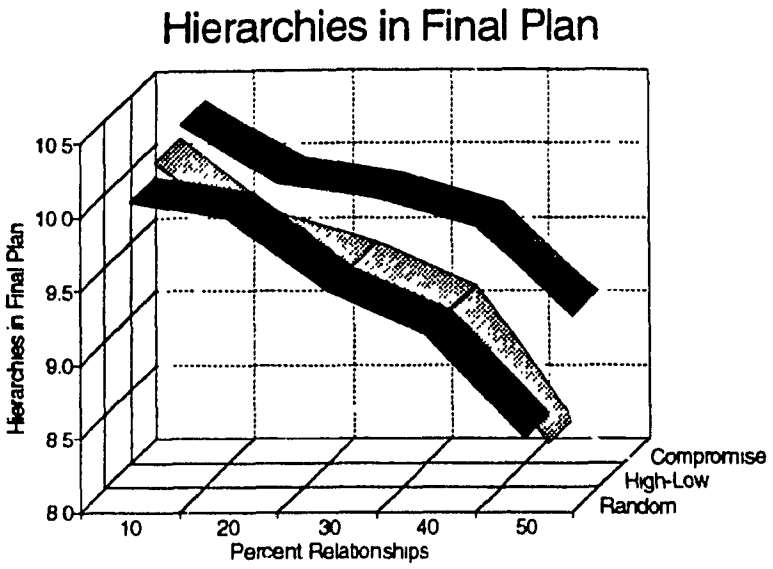


Goal Rel.	RANDOM	HIGH-LOW	COMPROMISE
10	15.98(16.96)	12.25(13.68)	17.42(19.60)
20	23.99(22.06)	18.93(21.76)	16.15(11.35)
30	33.17(22.45)	25 03(19.71)	30.52(28.14)
40	41.37(18.28)	36 64(18.25)	35.27(16.73)
50	56.24(16.88)	47.54(20.42)	44.01(21.23)

Figure 5.1: Means and Standard Deviations for Total Compromise

The Total Compromise increases as the number of goal relationships in the Proposals increases for both ordering heuristics as well as the random strategy. The Compromise heuristic outperformed the Random strategy in four of the five goal relationship levels used in the experiment, while the High-Low heuristic was better than the random strategy in all five levels. However, the Compromise heuristic did not consistently outperform the High-Low heuristic.

The means and standard deviations of the number of hierarchies preserved using each strategy on the Proposal sets with an increasing number of goal relationships is shown in Figure 5.2. The number of hierarchies preserved decrease as the number of goal relationships in the Proposals increase for both ordering heuristics as well as the random strategy. The Compromise heuristic outperformed the Random strategy in four out of the five goal relationship levels tested, and the High-Low heuristic was better than the random strategy only for Proposals with 10% goal relationships.



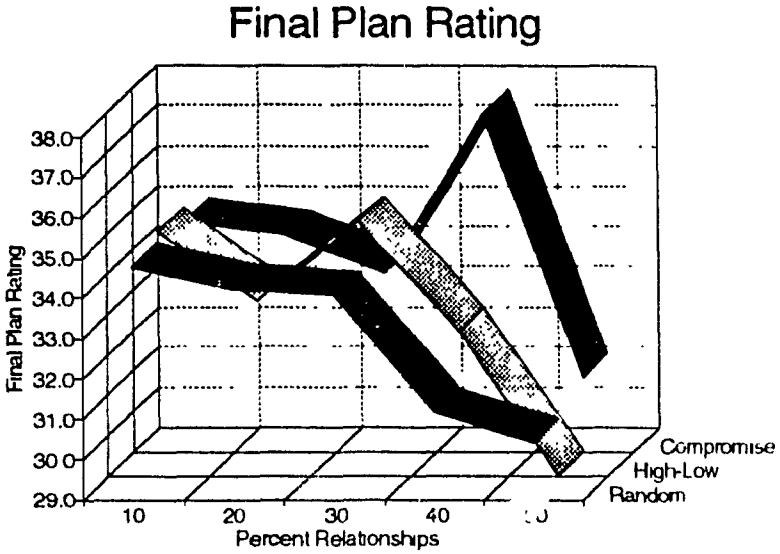
Goal Rel.	RANDOM	HIGH-LOW	COMPROMISE
10	10.10(0.74)	10.20(0.79)	10.30(0.68)
20	10.00(0.67)	9.70(0.68)	9.90(0.57)
30	9.50(1.08)	9.50(1.18)	9.80(0.92)
40	9.20(0.79)	9.20(0.92)	9.60(0.84)
50	8.50(1.18)	8.30(1.34)	9.00(1.49)

Figure 5.2: Means and Standard Deviations for Number of Hierarchies Preserved

Compromise outperformed the High-Low heuristic at all goal relationship levels.

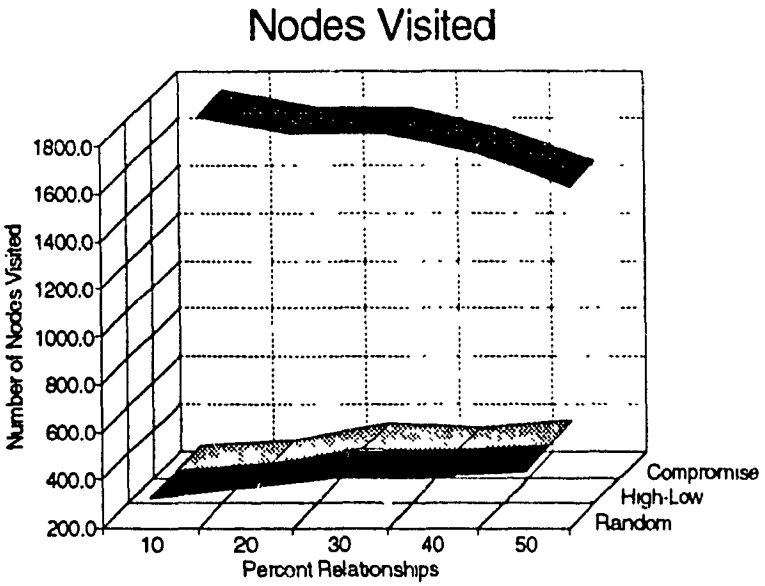
The means and standard deviations of the Final Plan rating obtained using each strategy on the Proposal sets with an increasing number of goal relationships is shown in Figure 5.3. The overall trend is downward for both ordering heuristics as well as the random strategy as the number of goal relationships in the Proposals increase. There is a peak that occurred between 30% and 40% goal relationships. The Compromise heuristic outperformed the Random strategy only at 40% goal relationships, and the High-Low heuristic was better than the random strategy at 10% and 40%. Compromise outperformed the High-Low heuristic only at 40% and 50% goal relationships.

The means and standard deviations for the number of nodes visited using each strategy on the Proposal sets with an increasing number of goal relationships is shown in Figure 5.4. The trend for the High-Low heuristic as well as the Random strategy is



Goal Rel.	RANDOM	HIGH-LOW	COMPROMISE
10	9.20(1.46)	9.48(1.88)	9.15(1.49)
20	9.19(1.88)	8.99(1.77)	8.99(1.67)
30	10.46(3.29)	10.44(3.30)	8.90(1.56)
40	9.20(2.53)	10.03(4.76)	11.05(5.32)
50	8.77(2.38)	8.10(1.22)	8.18(1.35)

Figure 5.3: Means and Standard Deviations for Final Plan Rating



Goal Rel	RANDOM	HIGH-LOW	COMPROMISE
10	327.30(39 08)	331 00(43.47)	1716 50(292 12)
20	366.70(43.99)	357 40(36 66)	1644 60(236 42)
30	412.00(53.67)	423 90(70 37)	1642 60(300.82)
40	417.40(54.84)	412.60(44 79)	1555 20(255 96)
50	437 20(55 72)	438 60(56 16)	1416 00(361 62)

Figure 5.4: Means and Standard Deviations for Number of Nodes Visited

upwards, while the trend for the Compromise heuristic is downwards as the number of goal relationships in the Proposals increase. The Compromise heuristic is more costly than the High-Low heuristic and random strategy at all levels of goal relationships. The High-Low heuristic is more costly in three of five levels.

The results of the analysis of variance performed for Total Compromise is shown in Table 5.2 and Table 5.3. The Multivariate test for the overall effect of strategy used to prune the CPS was significant at $\alpha = 0.006^a$ (Table 5.2). The individual contrasts performed indicated that the difference between the Compromise and the High-Low heuristic was not significant, but the High-Low and Compromise heuristic produce a different effect than the random strategy with $\alpha = 0.001^b$ (Table 5.2) The Multivariate test for the overall effect of the number of goal relationships was significant with $\alpha = 0.001^c$ (Table 5.2). The individual contrasts indicate that a significant linear trend is present in the Total Compromise as the number of goal

Multivariate Tests of Significance for Strategy			
Source of Variation	df	F	Sig of F
Between Group	2	10.59	0.006[a]
Within Group	8		

Individual Contrasts for Strategy						
	Source of Variation	Sum of Squares	df	Mean Squares	F	Sig. of F
CP vs HL	Between Group	8.99	1	8.99	0 13	0.725
	Within Group	616.07	9	68.45		
(CP+HL) vs RN	Between Group	1110.88	1	1110.87	23.44	0.001[b]
	Within Group	426.54	9	47.39		

Multivariate Tests of Significance for Goal Relationships			
Source of Variation	df	F	Sig. of F
Between Group	4	23.04	0 001[c]
Within Group	6		

Individual Contrasts for Goal Relationships						
	Source of Variation	Sum of Squares	df	Mean Squares	F	Sig. of F
LIN	Between Group	22271.15	1	22271.15	77.89	0.0[d]
	Within Group	2573.16	9	285.91		
QUAD	Between Group	327 59	1	327.59	0.43	0.529
	Within Group	6869.99	9	763 33		
CUB	Between Group	13.21	1	13 21	.03	0 863
	Within Group	3751.17	9	416.80		
FOUR	Between Group	62.95	1	62.95	0.091	0.769
	Within Group	6195.80	9	688.42		

Multivariate Tests of Significance for Strategy vs Goal Relationship			
Source of Variation	df	F	Sig of F
Between Group	8	1.24	0.521
Within Group	2		

Table 5.2: Analysis of Variance Total Compromise

relationships in the Proposals increase with $\alpha = 0.0^d$ (Table 5.2). The Multivariate and individual contrasts testing the trend for total compromise with each heuristic and the random strategy individually indicates a significant linear trend present in all three cases (Table 5.3).

The results of the analysis of variance performed for the Number of Hierarchies preserved is shown in Table 5.4 and Table 5.5. The Multivariate test for the overall effect of strategy used to prune the CPS was not statistically significant. The Multivariate test for the overall effect of the number of goal relationships was significant with $\alpha = 0.011^a$ (Table 5.4). The individual contrasts indicate that a significant

Multivariate Tests of Significance for Random Strategy			
Source of Variation	df	F	Sig of F
Between Group	4	27.98	0.001
Within Group	6		

Individual Contrasts for Random Strategy						
	Source of Variation	Sum of Squares	df	Mean Squares	F	Sig of F
LIN	Between Group	9579 91	1	9579.90	62 86	0.0
	Within Group	1371.5	9	152.39		
QUAD	Between Group	115.92	1	115.92	0 36	0.561
	Within Group	2862.21	9	318 02		
CUB	Between Group	30.26	1	30.26	0.14	0 715
	Within Group	1914.15	9	212.68		
FOUR	Between Group	13.69	1	13.69	0 06	0 811
	Within Group	2021 79	9	224 64		

Multivariate Tests of Significance for High-Low Strategy			
Source of Variation	df	F	Sig. of F
Between Group	4	27.98	0 001
Within Group	6		

Individual Contrasts for High-Low Strategy						
	Source of Variation	Sum of Squares	df	Mean Squares	F	Sig of F
LIN	Between Group	7797.60	1	7797 60	99 86	0
	Within Group	702 75	9	78 08		
QUAD	Between Group	138.96	1	138 96	0 54	0 479
	Within Group	2296.56	9	255 17		
CUB	Between Group	0.02	1	0.02	0	0 99
	Within Group	1631.08	9	181.23		
FOUR	Between Group	21 70	1	21 70	0 099	0.759
	Within Group	1954 93	9	217 21		

Multivariate Tests of Significance for COMPROMISE Strategy			
Source of Variation	df	F	Sig. of F
Between Group	4	6.03	0.027
Within Group	6		

Individual Contrasts for COMPROMISE Strategy						
	Source of Variation	Sum of Squares	df	Mean Squares	F	Sig of F
LIN	Between Group	5227.58	1	5227.58	21 64	0 001
	Within Group	2174 27	9	241.58		
QUAD	Between Group	77 35	1	77.35	0.32	0.586
	Within Group	2181.33	9	242.37		
CUB	Between Group	135.82	1	135.82	1.11	0 319
	Within Group	1096.80	9	121.87		
FOUR	Between Group	216.11	1	216.11	0.61	0 456
	Within Group	3204.57	9	356.06		

Table 5.3: Analysis of Variance Total Compromise Within Each Strategy

Multivariate Tests of Significance for Strategy			
Source of Variation	df	F	Sig of F
Between Group	2	4.06	0 061
Within Group	8		

Multivariate Tests of Significance for Goal Relationships			
Source of Variation	df	F	Sig of F
Between Group	4	8.73	0.011[a]
Within Group	6		

Individual Contrasts for Goal Relationships						
	Source of Variation	Sum of Squares	df	Mean Squares	F	Sig. of F
LIN	Between Group	41.81	1	41.81	31.75	0.0[b]
	Within Group	11.85	9	1.32		
QUAD	Between Group	1.37	1	1.37	0.85	0.380
	Within Group	14.48	9	1.61		
QUB	Between Group	0.85	1	0 85	1.03	0.337
	Within Group	7.48	9	0.83		
FOUR	Between Group	.07	1	.07	0.03	0 857
	Within Group	17.94	9	1.99		

Multivariate Tests of Significance for Strategy vs Goal Relationships			
Source of Variation	df	F	Sig of F
Between Group	8	1.0	0.590
Within Group	2		

Table 5.4: Analysis of Variance Number of Hierarchies Preserved

linear trend is present in the number of hierarchies preserved as the number of goal relationships in the Proposals increase with $\alpha = 0.0^b$ (Table 5.4). The Multivariate and individual contrasts testing the trend for number of hierarchies preserved with each heuristic and the random strategy individually indicates a significant linear trend present in the High-Low heuristic and the random strategy, but not in the Compromise Strategy (Table 5.5).

The results of the analysis of variance performed for the Final Plan rating is shown in Table 5.6. The Multivariate test for the overall effect of strategy used to prune the CPS was not significant. The Multivariate test for the overall effect of the number of goal relationships in the Proposals on the Final Plan rating was not statistically significant. The Multivariate test for the trend in the Final Plan rating as the number of goal relationships on the Proposals increases for each heuristic was

Multivariate Tests of Significance for Random Strategy			
Source of Variation	df	F	Sig of F
Between Group	4	10.0	0.008
Within Group	6		

Individual Contrasts for Random Strategy						
	Source of Variation	Sum of Squares	df	Mean Squares	F	Sig of F
LIN	Between Group	16.0	1	16.0	34.28	0 0
	Within Group	4.2	9	0.47		
QUAD	Between Group	0.71	1	0 71	1 5	0 25
	Within Group	4.28	9	0 47		
CUB	Between Group	0 0	1	0.0		
	Within Group	3 8	9	0.42	0 0	1 0
FOUR	Between Group	0.20	1	0.20	.31	0 592
	Within Group	5.99	9	0 67		

Multivariate Tests of Significance for HIGH-LOW Strategy			
Source of Variation	df	F	Sig. of F
Between Group	4	5.61	0.032
Within Group	6		

Individual Contrasts for HIGH-LOW Strategy						
	Source of Variation	Sum of Squares	df	Mean Squares	F	Sig of F
LIN	Between Group	18.49	1	18 49	27 69	0 001
	Within Group	6.01	9	0 67		
QUAD	Between Group	0.57	1	0 57	0.58	0 464
	Within Group	8.92	9	0 99		
CUB	Between Group	0 81	1	0 81	1.97	0 193
	Within Group	3.69	9	0 41		
FOUR	Between Group	0.00	1	0 00	0 00	0 964
	Within Group	5.90	9	0.66		

Multivariate Tests of Significance for COMPROMISE Strategy			
Source of Variation	df	F	Sig of F
Between Group	4	3.75	0.073
Within Group	6		

Table 5.5: Analysis of Variance Number of Hierarchies Preserved within Each Strategy

Multivariate Tests of Significance for Strategy			
Source of Variation	df	F	Sig of F
Between Group	2	0 12	0.892
Within Group	8		

Multivariate Tests of Significance for Goal Relationship			
Source of Variation	df	F	Sig. of F
Between Group	4	3.59	0.080
Within Group	6		

Multivariate Tests of Significance for Strategy vs Goal Relationship			
Source of Variation	df	F	Sig. of F
Between Group	8	0.27	0.928
Within Group	2		

Multivariate Tests of Significance for Random Strategy			
Source of Variation	df	F	Sig. of F
Between Group	4	1.69	0.268
Within Group	6		

Multivariate Tests of Significance for High-Low Strategy			
Source of Variation	df	F	Sig of F
Between Group	4	3 11	0.104
Within Croup	6		

Multivariate Tests of Significance for COMPROMISE strategy			
Source of Variation	df	F	Sig. of F
Between Group	4	1.3	0.368
Within Group	6		

Table 5.6: Analysis of Variance Final Plan Rating

not statistically significant.

The results of the analysis of variance performed for the Number of Nodes visited is shown in Table 5.7 and Table 5.8. The Multivariate test for the overall effect of strategy used to prune the CPS was significant at $\alpha = 0.0^{a}$ (Table 5.7). The individual contrasts performed indicated that the difference between the Compromise and the High-Low heuristic was significant at $\alpha = 0.0^{b}$ (Table 5.7), and the High-Low and Compromise heuristic produce a different effect than the random strategy with $\alpha = 0.0^{c}$ (Table 5.7). The Multivariate test for the overall effect of the number of goal relationships was not significant. The Multivariate and individual contrasts testing

Multivariate Tests of Significance for Strategy			
Source of Variation	df	F	Sig of F
Between Group	2	189.87	0.0[a]
Within Group	8		

Individual Contrasts for Strategy						
	Source of Variation	Sum of Squares	df	Mean Squares	F	Sig of F
CP vs HL	Between Group	36136929.96	1	36136929.96	391.50	0.0[b]
	Within Group	830722.64	9	92302.51		
(CP + HL) vs RN	Between Group	12068898.61	1	12068898.61	416.01	0.0[c]
	Within Group	261097.05	9	29010.78		

Multivariate Tests of Significance for Goal Relationships			
Source of Variation	df	F	Sig. of F
Between Group	4	1.30	0.367
Within Group	6		

Multivariate Tests of Significance for Strategy vs Goal Relationships			
Source of Variation	df	F	Sig. of F
Between Group	8	0.88	0.634
Within Group	2		

Table 5.7: Analysis of Variance Number of Nodes Visited within Each Strategy

the trend for the number of nodes visited with each heuristic and the random strategy individually indicates a significant linear trend present in the High-Low heuristic and the random strategy (Table 5.8).

5.3 Discussion of Results

Discussion of Total Compromise Metric

The assumptions made about the benefits obtained by using the hierarchy ordering heuristics are statistically supported by the Total Compromise metric. The Compromise and High-Low heuristic performed better than the random strategy. The performance of the Compromise heuristic and the High-Low heuristic are close. The value of the compromises that must be made by the hierarchy ordering heuristics or the random strategy tend to increase linearly with the number of goal relationships. An interesting observation is that the Compromise heuristic performs at a level roughly equivalent to the high-low heuristic, but chooses hierarchy orderings

Multivariate Tests of Significance for Random Strategy			
Source of Variation	df	F	Sig. of F
Between Group	4	41.14	0.0
Within Group	6		

Individual Contrasts for Random Strategy						
	Source of Variation	Sum of Squares	df	Mean Squares	F	Sig. of F
LIN	Between Group	73170.25		73170.25	84.03	0.0
	Within Group	7836.45		870.72		
QUAD	Between Group	4469.15		4469.15	5.1	0.050
	Within Group	7889.06		876.56		
CUB	Between Group	72.25		72.25	0.08	0.783
	Within Group	8061 05		895.67		
FOUR	Between Group	1431.43		1431 43	0.65	0.441
	Within Group	19789.96		2198.88		

Multivariate Tests of Significance for HIGH-LOW Strategy			
Source of Variation	df	F	Sig. of F
Between Group	4	15 13	0.003
Within Group	6		

Individual Contrasts for HIGH-LOW Strategy						
	Source of Variation	Sum of Squares	df	Mean Squares	F	Sig. of F
LIN	Between Group	73116.16		73116.16	71.09	0.0
	Within Group	9256 84		1028.54		
QUAD	Between Group	4412.83		4412.83	2.55	0.145
	Within Group	15575.86		1730.65		
CUB	Between Group	7.84		7.84	0.0	0.941
	Within Group	12015.16		1335.02	0	
FOUR	Between Group	7755.57		7755.57	3.22	0.106
	Within Group	21657.31		2406.37		

Multivariate Tests of Significance for Compromise Strategy			
Source of Variation	df	F	Sig. of F
Between Group	4	1.96	0.220
Within Group	6		

Table 5.8: Analysis of Variance Number of Nodes Visited within Each Strategy

independent of the hierarchy's actual rating.

In the current implementation, the compromise at a given XORList is compared to the value of another hierarchy when `mark_ineligible()` is called. Thus, multiple compromises in the same hierarchy do not impact upon the Joint Plan creation (ie., the path `select()` takes), only the immediate impact of the current possible compromise is used. We conjecture that it may prove better to keep a running total of the compromises made at the XORLists as `select()` moves down the hierarchy. Once the Total Compromise for a given hierarchy has exceeded some bound, it may be preferable to backtrack and attempt to run `select()` down a different path in the hierarchy. The exact bound would best be determined by further experimentation, but may be related to the value of the hierarchy. A pilot study examined a modification to the `XORList::select()` routine so that `XORList::select()` would backtrack if the compromise it would make was greater than the compromise that was passed to it from a parent. The results of the modification did not indicate a significant improvement to the heuristic.

Discussion of Hierarchies Preserved Metric

The assumptions made about the benefits obtained by using the hierarchy ordering heuristics are somewhat apparent in the number of hierarchies preserved. While examining the raw data, it is found that the Compromise heuristic is a step in the right direction. The superiority of the Compromise heuristic in preserving hierarchies was not confirmed at a 95% confidence level, but we could confirm it at a 0.93 confidence level. For the High-Low heuristic and Random strategy, a linearly decreasing trend was observed in the number of hierarchies preserved as the number of goal relationships increased. However the same trend was not evident in the Compromise heuristic.

A detailed analysis of the execution flow of Consensus with respect to the results of this metric showed that the Compromise heuristic successfully retained as many hierarchies as the other metrics (or better) in every instance except one. As stated in chapter 4, the Compromise heuristic used the tie breaking mechanism of counting the

number of goal relationships along the principal path in the hierarchy for hierarchies with the maximum compromise. This tie-breaking heuristic seems to be a reasonable one, but it does not always result in the best solution because a single goal can knock out several other hierarchies. More important than just counting the relationships is the "position" of the relationship, resulting in its actual potential to knock out another hierarchy. To actually use the position of the goal to determine hierarchy ordering would require following the relationship to the other hierarchy. This would be complicated and was not pursued. However, the heuristic should (on the average) be better than a simple random selection. In the cases when it failed, it was partly because all experts made the same choices. These same choices were made because the number of goal relationships seen by all experts is the same. This results in the same, possibly erroneous, choices in a few cases. Other techniques, although individually inferior to the heuristic used, occasionally worked better with multiple experts because of the diversity of opinion. The Final plan was selected from a more diverse group of Joint Plans, and this would help to weed out bad Joint Plans. Note that this did not occur often. There were only a couple of cases where the other hierarchies retained more hierarchies, and in those specific cases this lack of diversity of opinion was found to be the cause of Consensus failing to select a Final Plan with more hierarchies.

Discussion of Final Plan Rating metric

The assumptions made about the benefits obtained by using the hierarchy ordering heuristics are not evident in the Final Plan rating. While the raw data indicated that the ordering heuristics performed slightly better than the random strategy in some cases there was no statistical confirmation of this effect. The raw data also seemed to indicate a decrease in the Final Plan rating as the number of goal relationships increased, but again this effect was not confirmed. It would seem that the benefits obtained as measured by the Total Compromise and the number of hierarchies preserved did not translate into Final Plans with better overall ratings.

In all experiments, the appropriateness of the metrics chosen should be carefully

examined. The Final Plan Rating metric is unlike the other metrics used, in that it is a *derived* quantity. It is a formula created by the system designer, based upon other criteria or metrics, to produce a number expected to reflect the "worth" of the Final Plan. Finding appropriate metrics upon which to base the Final Plan rating is an experimental process, but realistically, no set of metrics and weights will be suitable for all problems [17].

Another reason for the lack of differentiation between the Final Plan ratings for the various hierarchy ordering techniques may be in the way ratings were assigned to the goals. In an effort to reduce experimenter bias, the proposal generator program randomly assigned rating to the goals, something that an expert system is unlikely to do. The distribution of ratings among goals in a Proposal is not seen as directly related to the work in this thesis, and so will be pursued in future work.

Discussion of Number of Nodes Visited metric

The assumptions made about the relative cost of the Compromise heuristic compare to the Random and High-Low were apparent in the results of number of nodes visited metric. The Compromise heuristic was clearly more computationally expensive than the other two heuristics. This can be attributed to the overhead for both the computation of the compromise value, and the determination if a hierarchy would be eliminated by the resolution of a goal relationship.

The general trend for the compromise heuristic in this cost metric was to *decrease* as the number of goal relationships increased. This can be attributed to two effects: The declining number of hierarchies as the number of goal relationships increased, and the larger number of XORLists made ineligible allowed the resolution of goal relationships at a lower level in the hierarchy. The general trend for the other two heuristics was to increase the number of nodes visited as the number of goal relationships increased. This can solely be attributed to the number of times the routine to mark another goal ineligible was called, as neither of these heuristics used backtracking.

The original assumptions that were made concerning the benefits of using the hi-

erarchy ordering heuristics are supported by the Total Compromise metric, somewhat apparent in the number of hierarchies preserved, but not evident in the Final Plan rating. As expected, the cost of the Compromise heuristic is greater than the cost of either the High-Low heuristic or the random strategy. However, the cost of the High-Low strategy was close to the cost associated with the random strategy. The trends observed as the number of goal relationships increase, indicate that the number of highly rated elements in the CPS that are discarded when using the hierarchy ordering heuristics or the random Strategy tend to increase linearly with the number of goal relationships. However, the Compromise heuristic proved to be more robust at preserving hierarchies as the number of goal relationships increased.

Thus, the High-Low heuristic presents a low cost alternative to the Compromise heuristic when preserving hierarchies is not a critical issue.

Our goal in this research is to study "real-world" problems and determine the important factors, metrics, and heuristics in distributed problem solving systems. We defined the metrics first and then applied them to the simulated problems of the real world. There seems to be no definitive way of coming up with these metrics, and defining them would be an iterative and learning process. It is likely that these metrics will change as we discover more from further experiments.

Chapter 6

Conclusions

If I had it to do over again, I would rather have learned to play the piano.
- Carol DeKoven, after her Master of Computer Science thesis defense

6.1 Conclusions

To cooperate successfully, expert systems must exchange information to plan their actions. In this thesis, we have proposed a protocol called Consensus, which is suitable for distributed planning. A number of heuristics have been used to tackle this complex problem. In a step-by-step manner, the Consensus protocol produces a plan which the cooperating expert systems will follow. The protocol is considered distributed because the input of all the expert systems is used to construct a plan, and they jointly select it.

A prototype implementation of the Consensus protocol consisting of about 4000 lines of C++ code was created, and it provides a basis upon which future CDPS experiments can be performed. A parameterizable Proposal generator, consisting of about 2000 lines of C++ code, was used to provide input to the Consensus program. Using the implementation of Consensus we developed, and the input provided by the proposal generator, an experimental analysis was carried out to study the performance of the planning protocol.

For the experimental analysis, we have proposed four metrics: the Number of

Nodes Visited in the Common Planning Structure which indicates the computational complexity; the Number of Goal Hierarchies in the Final Plan, which is an indirect measure of the quality of the plan; Total Compromise value which is a measure of the inability to make the best choice because of interrelationships among goals; and the rating of the Final Plan.

The three heuristics presented in the thesis, namely Compromise, High-Low, and Random, have been studied with respect to the above metrics and detailed results are presented in chapter 5. The compromise heuristic performed best in 4 out of 5 goal relationship levels with regard to the preservation of goal hierarchies in the plan, although statistical significance at only the 93% confidence level was achieved. The High-Low heuristic performed similar to the Compromise heuristic with respect to the Total Compromise metric. However, the computational cost as reflected by the Number of Nodes Visited metric was much greater for the Compromise heuristic. Thus, if the number of goal hierarchies preserved is important, the compromise heuristic should be used, otherwise, the high-low heuristic provides a lower cost alternative. The overall conclusion is that the consensus protocol is a viable protocol for distributed planning.

The experiment revealed a counter-example to the common belief that reaching an acceptable solution is easy if all expert systems agree on the solution. We showed that inferior techniques, if diverse among the expert systems, may provide results comparable to an individually superior technique. After completing this thesis, in the last couple of weeks, we have found that an observation similar to the above effect was made by another researcher [28].

6.2 Future Work

The size and scope of the Consensus protocol were such that it was impossible to implement every idea. The nature of this work was experimental, leading to iterative refinement of the program. It is hoped that this will continue in the future, with the possible implementation of the following refinements:

The ratings of goals is currently static – the ratings are assigned by expert systems,

and merged by the CPS creation routines. Instead, a system whereby the ratings of goals are updated as the pruning of the CPS progresses would be a useful addition, albeit at a greater computational cost. A technique whereby the desirability of goals moves down the hierarchy, and the likelihood of success moves up the hierarchy is envisioned. A static version of this has already been implemented to provide goal ratings in the proposal generator.

Static beliefs are currently used between the expert systems. Two methods would be acceptable replacements for it: In the first, an expert system would send its own belief in its rating along with the rating, in essence a confidence factor similar to that used by Shortliffe in MYCIN [37]. A second alternative would have each expert system familiar with the areas of expertise of other expert systems in its Consensus group, and weight the goal ratings it receives from it accordingly. The rating would thus depend both on the goal and the expert whereas the current system depends only on the expert. Either of these techniques would allow us to relax the constraint that all experts in a Consensus group are able to rate arbitrary goals.

Consensus assumes that an expert system knows which information to send in a Proposal. The information that should be in a Proposal should be further investigated, but this is beyond the scope of the current work.

While creating a Joint Plan, Consensus does not look at the epoch in which a goal to be chosen appears. The current Joint Plan rating formula makes it advantageous to select goals in the earliest possible epoch.

Bibliography

[1] M. Benda, V. Jagannathan, and R. Dodhiawala. On optimal cooperation of knowledge sources - an empirical investigation. Technical report, Boeing Advanced Technology Center, Boeing Computing Services, Seattle, Washington, July 1986.

[2] Alan H. Bond and Les Gasser. An analysis of problems and research in DAI. In *Readings in Distributed Artificial Intelligence*, chapter 1, pages 3–35. Morgan Kaufman Publishers, Inc., San Mateo, California, 1988.

[3] Stephanie Cammarata, David McArthur, and Randall Steeb. Strategies of cooperation in distributed problem solving. *Proceedings of the 8th International Joint Conference on Artificial Intelligence*, pages 767–770, 1983. also in Alan H. Bond and Les Gasser, editors, *Readings in Distributed Artificial Intelligence*, pages 102–105. Morgan Kaufman Publishers, Inc., San Mateo, California, 1988.

[4] R. Clark and C. Grossner. Generation of organizations for cooperating expert systems, DAI technical report DAI-0290-0001. Technical report, Concordia University, Montreal, Quebec, February 1990.

[5] R. Clark, C. Grossner, and T. Radhakrishnan. Consensus : A planning protocol for cooperating expert systems. In *Proceedings of the Eleventh International Workshop on Distributed Artificial Intelligence*, pages 43–58, Glen Arbor, Michigan, February 1992.

[6] Susan E. Conry, Robert A. Meyer, and Victor R. Lesser. Multistage negotiation in distributed planning. In *COINS TR86-87*, pages 1–17, 1986. also in Alan H.

Bond and Les Gasser, editors. *Readings in Distributed Artificial Intelligence*, pages 367–384. Morgan Kaufman Publishers, Inc., San Mateo, California, 1988.

[7] Daniel D. Corkill and Victor R. Lesser. Unifying data-directed and goal-directed control : An example and experiments. In *AAAI*, pages 143-147, 1982.

[8] Daniel David Corkill. *A Framework for Organizational Self-Design in Distributed Problem Solving Networks.* PhD thesis, University of Massachussetts, February 1983.

[9] Randall Davis and Reid G. Smith. Negotiation as a metaphor for distributed problem solving. *Artificial Intelligence*, 20:63–109, 1983. also in Alan H. Bond and Les Gasser, editors, *Readings in Distributed Artificial Intelligence*, pages 333–356. Morgan Kaufman Publishers, Inc., San Mateo, California, 1988.

[10] Edmund H. Durfee, Victor Lesser, and Daniel D. Corkill. Coherent cooperation among communicating problem solvers. *IEEE Transactions on Computers*, C-36(11):1275–1291, November 1987. also in Alan H. Bond and Les Gasser, editors, *Readings in Distributed Artificial Intelligence*, pages 268–284. Morgan Kaufman Publishers, Inc., San Mateo, California, 1988.

[11] Edmund H. Durfee and Victor R. Lesser. Increasing coherence in a distributed problem solving network. In *IJCAI*, pages 1025–1030, 1985.

[12] Edmund H. Durfee and Victor R. Lesser. Incremental planning to control a blackboard based problem solver. In *AAAI*, pages 58-64, 1986.

[13] Edmund H. Durfee and Victor R. Lesser. Using partial global plans to coordinate distributed problem solvers. In *IJCAI*, pages 875–883, 1987. also in Alan H. Bond and Les Gasser, editors, *Readings in Distributed Artificial Intelligence*, pages 285–293. Morgan Kaufman Publishers, Inc., San Mateo, California, 1988.

[14] Edmund H. Durfee and Victor R. Lesser. Partial global planning : A coordination framework for distributed hypothesis formation. *IEEE Transactions on Systems, Man, and Cybernetics*, 21(5):1167–1183, September 1991.

[15] Edmund H. Durfee, Victor R. Lesser, and Daniel D. Corkill. Trends in cooperative distributed problem solving. *Transactions on Knowledge and Data Engineering*, 1(1):63-83, March 1989.

[16] Edmund H. Durfee and Thomas A Montgomery. A hierarchical protocol for coordinating multiagent behaviors. In *AAAI*, pages 86–93, 1990.

[17] Edmund Howell Durfee. *A Unified Approach to Dynamic Coordination : Planning Actions and Interactions in a Distributed Problem Solving Network.* PhD thesis, University of Massachussetts, September 1987.

[18] Robert Engelmore and Tony Morgan. *Blackboard Systems.* Addison Wesley, Reading Mass., 1988.

[19] Lee D. Erman, Frederick Hayes-Roth, Victor R. Lesser, and D. Raj Reddy. *The Hearsay II Speech-Understanding System: Integrating Knowledge to Resolve Uncertainty*, chapter 3. Addison-Wesley, 1988. appears in *Blackboard Systems* by Engelmore and Morgan.

[20] Richard D. Fennell and Victor R. Lesser. Parallelism in artificial intelligence problem solving : A case study of hearsay II. *IEEE Transactions on Computers*, pages 98–111, February 1977. also in Alan H. Bond and Les Gasser, editors, *Readings in Distributed Artificial Intelligence*, pages 106–109. Morgan Kaufman Publishers, Inc., San Mateo, California, 1988.

[21] Mark S. Fox. An organizational view of distributed systems. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-11(1):70–80, January 1981. also in Alan H. Bond and Les Gasser, editors, *Readings in Distributed Artificial Intelligence*, pages 140–150. Morgan Kaufman Publishers, Inc., San Mateo, California, 1988.

[22] Simon French. *Decision Theory: An Introduction to the Mathematics of Rationality.* Ellis Horwood, Chichester, England, 1986.

[23] M.L. Ginsberg. Decision procedures. In M.N. Huhns, editor, *Distributed Artificial Intelligence*. Morgan Kaufmann, 1987.

[24] C. Grossner. Blackbox : An ill-structured problem. Technical report, Concordia University, Montreal, Quebec, June 1990.

[25] C. Grossner. Information deficit : A metric for the data distribution of an organization. Technical report, Concordia University, Montreal, Quebec, August 1990.

[26] C. Grossner and T. Radhakrishnan. Organizations for cooperating expert systems, DAI technical report DAI 0790-0002. Technical report, Concordia University, Montreal, Quebec, July 1990.

[27] James Hendler, Austin Tate, and Mark Drummond. AI planning : Systems and techniques. *AI Magazine*, 11(2):61–77, Summer 1990.

[28] T. Hogg and B.A. Huberman. Controlling chaos in distributed systems. *Systems, Man and Cybernetics*, 21(6):1325–1332, November 1991.

[29] DAI Group discussion by email, 1991. moderated by Michael Huhns.

[30] Victor R. Lesser and Daniel D. Corkill. Functionally accurate, cooperative distributed systems. *IEEE Transactions on Systems, Man and Cybernetics*, 11(1):81–96, 1981. also in Alan H. Bond and Les Gasser, editors, *Readings in Distributed Artificial Intelligence*, pages 295–310. Morgan Kaufman Publishers, Inc., San Mateo, California, 1988.

[31] Victor R. Lesser, Daniel D. Corkill, Robert C. Whitehair, and Joseph A. Hernandez. Focus of control through goal relationships. In *IJCAI*, pages 497–503, 1989.

[32] Thomas A. Montgomery and Edmund H. Durfee. Search reduction in hierarchical distributed problem solving. In *Proceedings of the Eleventh International Workshop on Distributed Artificial Intelligence*, pages 247–260, February 1992.

[33] H. Penny Nii. Blackboard application systems and a knowledge engineering perspective (part 2). *AI Magazine*, August 1986.

[34] Kristina Pitula. Cooperative problem solving through the game of distributed blackbox. Master's thesis, Concordia University, Montreal, Quebec, June 1990.

[35] Malcolm Richard Railey. *Dynamic Control Structures For Cooperating Processes.* PhD thesis, Univ. of Illinois at Urbana-Champaign, January 1986.

[36] Elaine Rich. *Artificial Intelligence.* McGraw Hill, New York, New York, 1983.

[37] E.H. Shortliffe and B.G. Buchanan. *Rule-Based Expert Systems: The MYCIN Experiments of the Stanford Heuristic Programming Project.* Addison-Wesley, 1986.

[38] Herbert A. Simon. The structure of ill-structured problems. *Artificial Intelligence*, 4:181–201, 1973.

[39] Reid G. Smith. The contract net protocol: High level communication and control in a distributed problem solver. *IEEE Transactions on Computers*, C-29(12):1104–1113, December 1980. also in Alan H. Bond and Les Gasser, editors, *Readings in Distributed Artificial Intelligence*, pages 357–366. Morgan Kaufman Publishers, Inc., San Mateo, California, 1988.

[40] Reid G. Smith and Randall Davis. Frameworks for cooperation in distributed problem solving. *IEEE Transaction of Systems, Man, and Cybernetics*, SMC 11(1):61–70. January 1981. also in Alan H. Bond and Les Gasser, editors, *Readings in Distributed Artificial Intelligence*, pages 61–70. Morgan Kaufman Publishers, Inc., San Mateo, California, 1988.

[41] William Swartout, Thomas Dean, Theodore Linden, Richard Fikes, Peter Cheeseman, and Drew McDermott. DARPA Santa Cruz workshop on planning. *AI Magazine*, pages 115–131, Summer 1988.

[42] Robert Wilensky. *Planning and Understanding : A Computational Approach to Human Reasoning.* Addison-Wesley, Reading, Mass., 1983.