Appeared in: Theory And Decision, Vol. 44, No. 2, April 1998, pp. 149--171.

GAME TREES FOR DECISION ANALYSIS

Prakash P. Shenoy
School of Business, University of Kansas, Summerfield Hall, Lawrence, KS 66045-2003, USA.

ABSTRACT

Game trees (or extensive-form games) were first defined by von Neumann and Morgenstern in 1944. In this paper, we examine the use of game trees for representing Bayesian decision problems. We propose a method for solving game trees using local computation. This method is a special case of a method due to Wilson for computing equilibria in 2-person games. Game trees differ from decision trees in the representations of information constraints and uncertainty. We compare the game tree representation and solution technique with other techniques for decision analysis such as decision trees, influence diagrams, and valuation networks.

Key Words: Game trees, decision trees, influence diagrams, valuation networks, roll-back method

---

1. INTRODUCTION

Game trees were defined first by von Neumann and Morgenstern [1944]. In 1953, Kuhn proposed a more general definition. Game trees were extensively studied in the 1950s and had a strong influence on the decision tree technique for representing and solving decision problems proposed by Raiffa and Schlaifer [1961]. Since a decision problem can be viewed as a game in which there is only one player, the use of game trees for decision theory is natural.

Although there are many features in common between (one-person) game trees and decision trees, there is a key difference in how information constraints are encoded. In decision trees, information constraints are encoded by the sequencing of the decision and chance nodes in the directed tree. Thus if the true value of a chance variable R is known at the time a value of a decision variable T is chosen, then node R must precede node T in the decision tree. The converse is also true. If the true value of a chance variable R is not known at the time a value of a decision variable T is chosen, then node R must follow node T in the decision tree. In game trees, information constraints are encoded partially by sequencing of chance and decision nodes, and partially by information sets. Thus, as in decision trees, if the true value of a chance variable R is known at the time a value of a decision variable T is chosen, then node R must precede node T in the game tree. The converse is not true. If the true value of a chance variable R is not known at the time a value of a decision variable T is chosen, then node R does not have to follow node T in the game tree. We

could have R precede T and use information sets instead to encode the lack of knowledge of R when a value of T has to be chosen.

The added flexibility of game trees in sequencing decision and chance nodes has two important consequences. First, in game trees, we can use the sequence of variables to represent time or causation instead of information. In decision trees, the sequence of nodes must represent information and nothing else. Thus, for example, if variable D (disease present or not) is the cause of variable S (symptom exhibited or not), and the decision maker has to make a decision T (to treat or not) after observing the true value of S (but not D), then in a decision tree, the sequence must be STD. In a game tree, however, we could, for example, choose sequence DST that represents time (disease comes first, symptom next, and treatment last).

Second, assuming we have a causal probability model for the chance variables, if we choose to sequence the chance variables to represent causation, then there is no need for Bayesian revision of probabilities for representing a problem as a game tree. The need for Bayesian revision of probabilities in decision tree representation is a major drawback from the viewpoint of artificial intelligence for two important reasons. First, since typically representation is done by humans, and solution is done by machine, the representation technique should facilitate representation of a problem. Ideally, no preprocessing should be required before we can represent a problem in machine solvable form. Decision trees violate this maxim whereas game trees do not. We note that influence diagrams were invented by Howard and Matheson [1981] as a “front-end” for decision trees to avoid the problem of pre-processing the probabilities during the representation phase. (To solve an influence diagram representation, Howard and Matheson described a method that consisted of first converting the influence diagram representation to a decision tree representation and then solving the decision tree representation.) Second, the Bayesian revision of probabilities required in decision trees can be computationally intractable if the number of chance variables is large. Elsewhere we have shown that most of the computation involved in Bayesian revision of probabilities is unnecessary for solving the problem [Shenoy 1994].

We describe a roll-back method (backward recursion, or dynamic programming) to solve game trees using local computation. This method is a special case of a method due to Wilson [1972] for computing equilibria of 2-person games (see also [Kreps and Wilson 1982]). The Bayesian revision of probabilities is part of the solution phase. Also, we can use local computation to some extent to minimize the computational burden of computing probabilities. The roll-back method for solving game trees described in this paper generalizes the roll-back method of decision trees.

Influence diagrams and valuation networks are more appropriate representations than decision trees and game trees for symmetric or almost symmetric decision problems involving many variables since it is practically intractable to either represent or solve such problems using either decision or game trees. However if we have a small (i.e., few variables) highly asymmetric decision problem, then a decision or game tree representation may be more appropriate than

influence diagrams and valuation networks since the latter are unable to represent asymmetry as efficiently as decision or game trees. And if we have a small highly asymmetric problem involving Bayesian revision of probabilities, then the game tree representation and solution technique may be more appropriate than decision trees since the former involves no pre-processing of probabilities.

An outline of this paper is as follows. In Section 2, we describe the oil wildcatter's problem [Raiffa 1967]. This is a small slightly asymmetric problem involving Bayesian revision of probabilities. We describe the traditional decision tree representation and solution of this problem. In Section 3, we describe a game tree representation and illustrate it using the oil wildcatter's problem. In Section 4, we describe a roll-back method for solving game trees using local computation. This method generalizes the roll-back method of decision trees. In Section 5, we compare briefly game trees to decision trees, influence diagrams, and valuation networks. Finally, in Section 6, we conclude with a summary.

2. THE OIL WILDCATTER'S PROBLEM

The oil wildcatter's problem is reproduced with minor modifications from Raiffa [1968].

An oil wildcatter must decide either to drill (*d*) or not to drill (~*d*). He is uncertain whether the hole is dry (*dr*), wet (*we*) or soaking (*so*). Table I shows his monetary payoffs and his subjective probabilities of the various states. The cost of drilling is $70,000. The net return associated with the *d-we* pair is $50,000 that is interpreted as a return of $120,000 less the $70,000 cost of drilling. Similarly the $200,000 associated with the *d-so* pair is a net return (a return of $270,000 less the $70,000 cost of drilling). We assume that the wildcatter's utility function is a linear function of the monetary profits, i.e., the wildcatter is risk neutral.

Table I.
The Payoff Matrix for the Oil Wildcatter's Problem

Wildcatter's profit, $ (π)			Act: Drill (*d*)	Act: Not drill (~*d*)	Probability of state
State	Dry	(*dr*)	–70,000	0	0.500
	Wet	(*we*)	50,000	0	0.300
	Soaking	(*so*)	200,000	0	0.200

At a cost of $10,000, the wildcatter could take seismic soundings that will help determine the geological structure at the site. The soundings will disclose whether the terrain below has no structure (*ns*)—that's bad, or open structure (*os*)—that's so-so, or closed structure (*cs*)—that's really hopeful. The experts have provided us with Table II that shows the probabilities of seismic test results conditional on the amount of oil.

Table II.
Probabilities of Seismic Test Results Conditional on the Amount of Oil

P(R | O)			Seismic Test Results (R) No structure (*ns*)	Open structure (*os*)	Closed structure (*cs*)
Amount	Dry	(*dr*)	0.600	0.300	0.100
of	Wet	(*we*)	0.300	0.400	0.300
Oil (O)	Soaking	(*so*)	0.100	0.400	0.500

Figures 1 and 2 show a decision tree representation and solution of this problem. Notice that even before we can completely specify the decision tree, we have to compute the conditional probabilities required by the decision tree (since they are not given in the statement of the problem). This is done in Figure 1. In Figure 1, the probability tree on the left computes the joint probabilities, and the probability tree on the right computes the marginals of test results and the conditional probabilities of amount of oil given test results. The conditional probabilities for O shown in the tree on the right are approximate (rounded-off to three decimal places).

Figure 1. The preprocessing of probabilities in the oil wildcatter's problem.

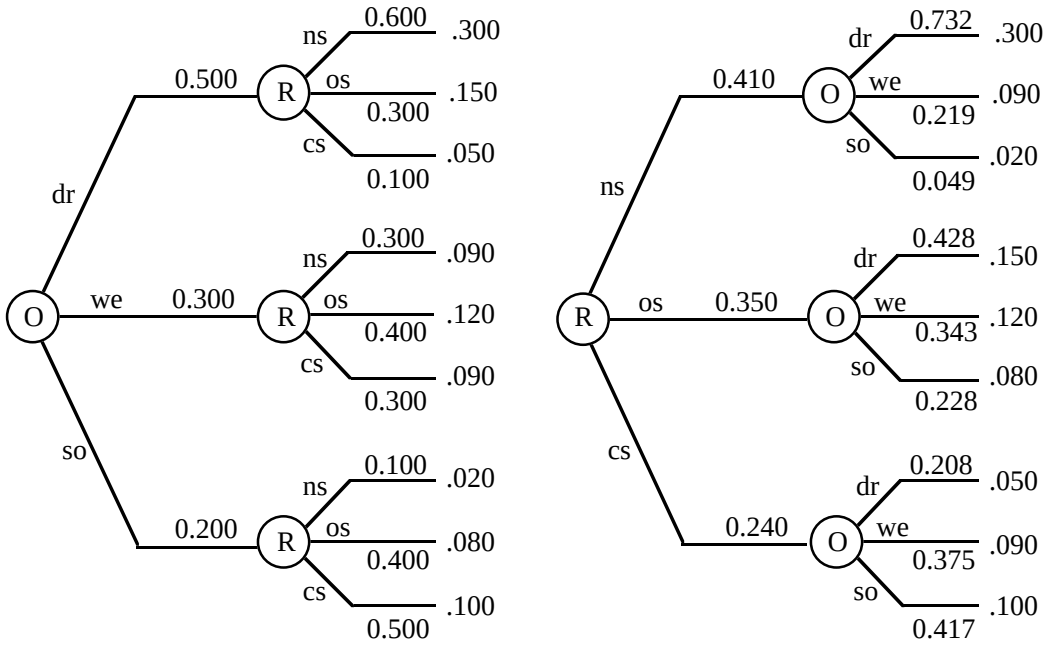


Figure 2. A decision tree representation and solution of the oil wildcatter's problem.

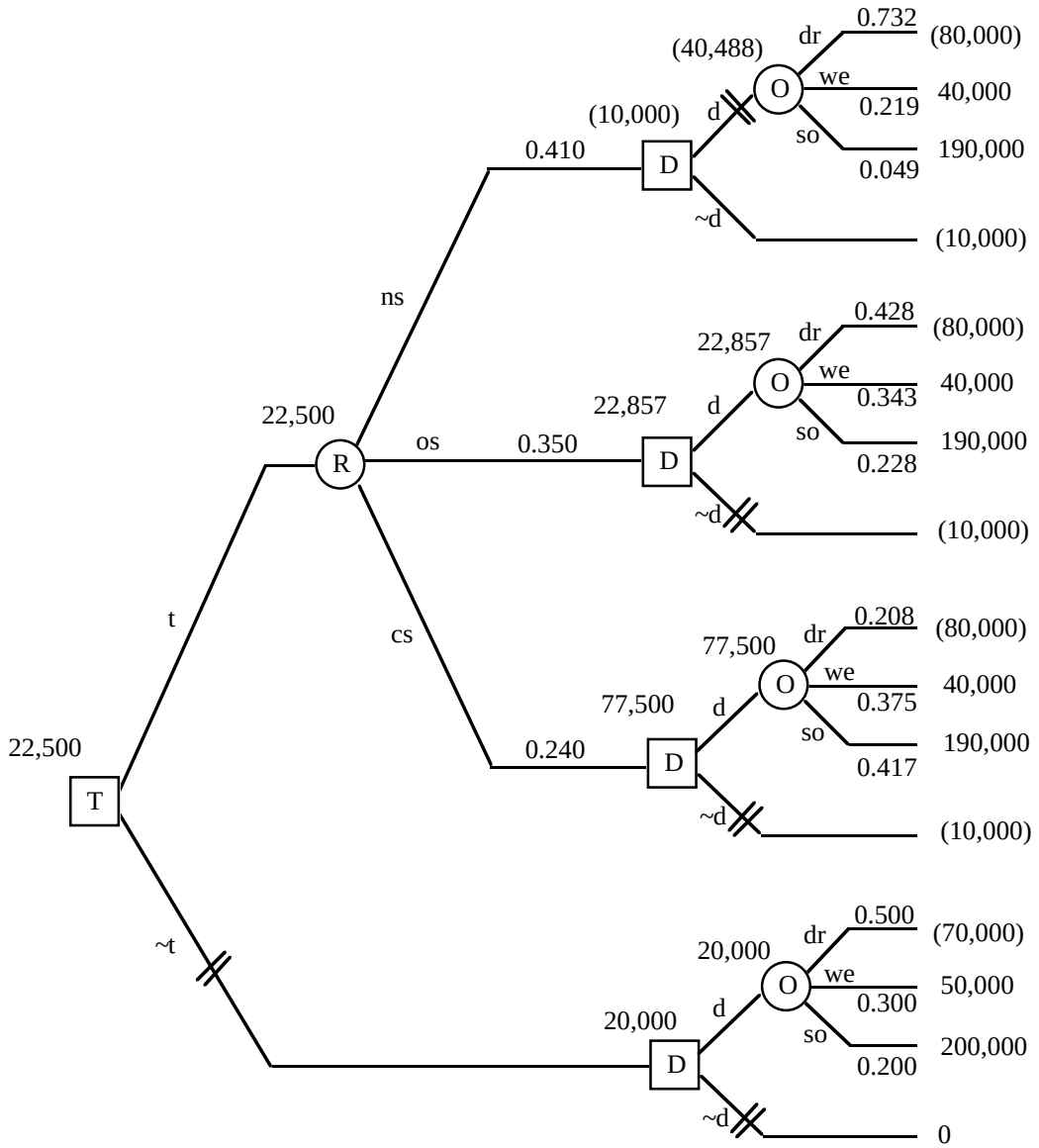


Figure 2 shows the solution of the oil wildcatter's problem using the roll-back method. In the rollback method, we recursively prune chance and decision nodes whose outward neighbors (away from the root) are all utilities. A chance node is pruned by averaging the utilities using the probability distribution on its edges. A decision node is pruned by maximizing the utilities associated with its edges. The optimal strategy is to do a seismic test; not drill if seismic test reveals no structure, and drill if the seismic test reveals either open or closed structure. The expected profit associated with this strategy is $22,500.

A count of algebraic binary operations (additions, multiplications, divisions, or comparisons) required to represent and solve the oil wildcatter's problem reveals that 24 operations are required

for preprocessing the probabilities and 30 operations are required to prune the decision tree for a total of 54 operations. This total count is an imperfect measure of the computational effort required to solve the decision tree representation as it does not account for other computer operations that are required to solve the problem.

3. GAME TREE REPRESENTATION

In this section, we describe a game tree representation of a decision problem. We use Kuhn's [1953] definition of an extensive form game appropriately modified. In particular, we assume only one player (the decision maker), and we assume perfect recall.

A game tree representation of a decision problem consists of the following:

(i). A rooted tree consisting of nodes and edges.

(ii). The leaf nodes of the rooted tree are called *utility nodes*. To each utility node, there is assigned a *utility value* describing the outcome (in von Neumann-Morgenstern utilities [von Neumann and Morgenstern 1947]) to the decision maker when the problem ends there.

(iii). A partition of the set of non-utility nodes of T into 2 subsets denoted by Γ and Δ. The elements of Γ are called *chance nodes* and correspond to chance variables. Each edge leading out of a chance node corresponds to a possible *value* of the random variable. The elements of Δ are called *decision nodes* and correspond to decision variables. Each edge leading out of a decision node corresponds to a possible *value* of the decision variable, i.e., an act available to the decision maker if that point in the tree is reached.

(iv). Each chance node has a probability distribution over its edges (values).

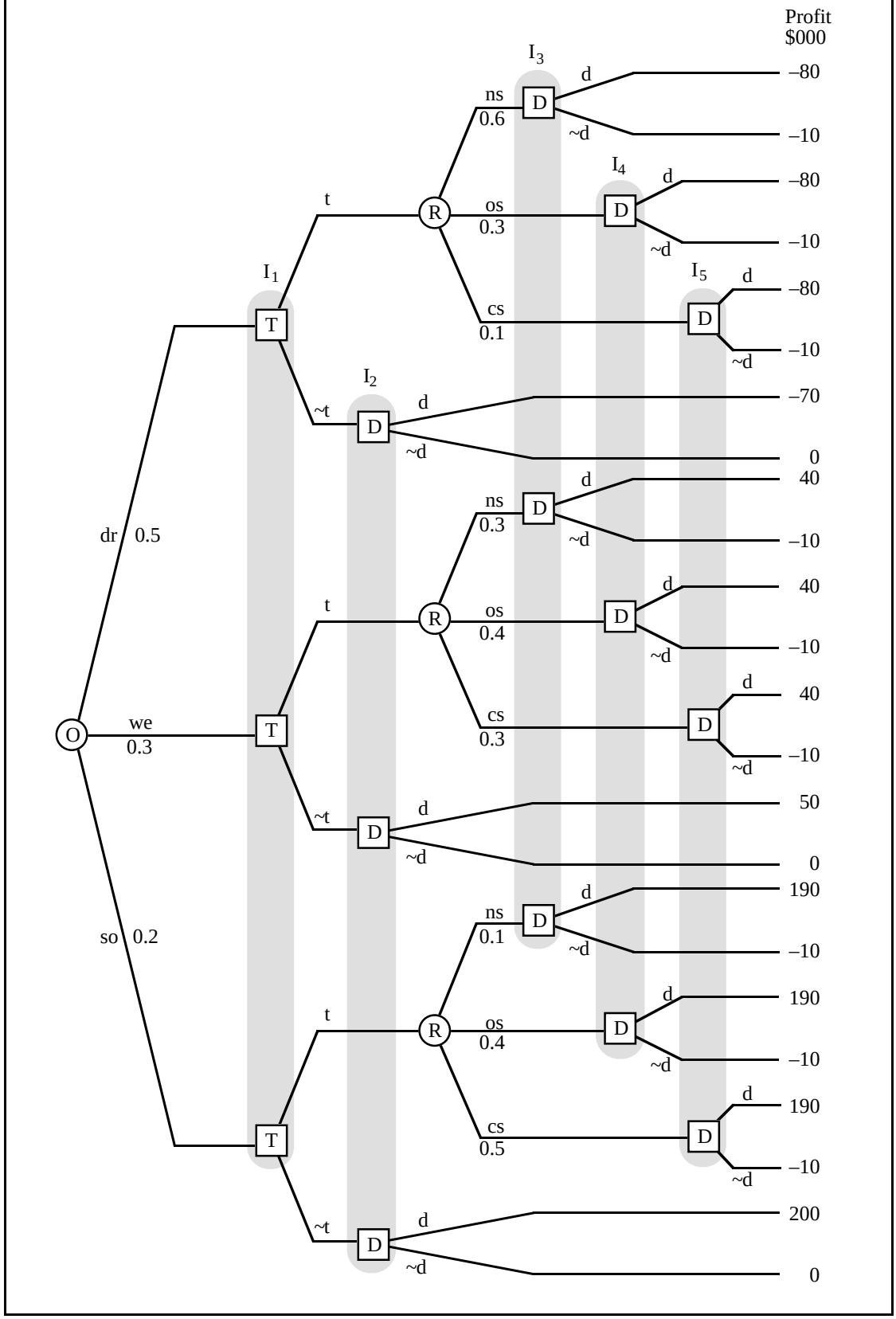
(v). Δ is partitioned into k *information sets*, $I_1, ..., I_k$, such that for each $j = 1, ..., k$, the following three conditions are satisfied:

(a) all decision nodes in I_j have the same number of values, and there is a one-to-one correspondence between the sets of values of nodes in I_j;

(b) every path from the root to a leaf node intersects each I_j at most once (a path may not intersect an information set at all), and

(c) if a path from the root to a node in I_i intersects another information set, say I_j, then every path from the root to a node in I_i must intersect the information set I_j and contain the corresponding edge (value) of the decision node in I_j.

Figure 3. A game tree representation of the oil wildcatter's problem.



The semantics of (i)–(iv) are similar to decision trees and need no further explanation. The semantics of information sets described in (v) are as follows. When it is the decision-maker's turn to choose an act, she is aware of the information set she is in, but does not know the precise vertex within this set. All nodes in an information set correspond to the same decision variable. Consequently, condition (a) requires all decision nodes in an information set to have the same number of edges, and each edge corresponds to a value of the decision variable. Condition (b) states that a node in an information set cannot come after another one in the same information set. Condition (c) states that at each information set, the decision-maker remembers past decisions (her choices at earlier decision nodes). Condition (c) is called "perfect recall" in game theory [Kuhn 1953, Hart 1992].

Figure 3 shows a game tree representation of the oil wildcatter's problem. Notice that in the game tree, random variable O (amount of oil) comes before R (test results). Also the decision nodes are partitioned into 5 information sets, I_1, ..., I_5. I_1, for example, encodes the fact that the oil wildcatter does not know the amount of oil at the time he decides whether to conduct a seismic test (*t*) or not (~*t*). I_2 encodes the fact that the oil wildcatter knows the decision made at node T (no seismic test), but not the amount of oil. At I_3, I_4, and I_5, the oil wildcatter knows the decision at node T (conduct a seismic test), and the test result, but not the amount of oil. Notice that no preprocessing is required to represent the problem as a game tree. Finally, the nodes in the information set I_1 precede nodes in information sets I_2, ..., I_4. This reflects the fact that the decision whether to conduct a seismic test or not precedes the decision whether to drill or not.

Information sets give us flexibility to represent a problem as a game tree with chance variables in any sequence. The only constraint we have in sequencing chance and decision nodes is that if a random variable R is observed when a decision D has to be made, then the chance node R must precede decision node D in the game tree. In decision trees, we do not have information sets. Instead, the sequence of chance variables represents information constraints and thus we have less flexibility than in game trees. It is this flexibility that allows us to represent any decision problem with no preprocessing.

4. A ROLL-BACK METHOD FOR SOLVING GAME TREES

In this section, we describe a roll-back method for solving a decision problem using local computation starting from its game tree representation. This roll-back method is a special case of a more general method due to Wilson [1972] for computing equilibria of 2-person games.

In game theory, an optimal strategy of a game tree is found by first reducing the game tree to an equivalent strategic form by (1) enumerating all strategies available to the decision maker, (2) enumerating all possible events (configuration of values of all random variables), (3) computing the utility for each strategy-event pair, and (4) computing the joint distribution of all random

Table III
A Strategic Form Representation of the Oil Wildcatter's Problem

Profit, $000	(*dr*, *ns*)	(*dr*, *os*)	(*dr*, *cs*)	(*we*, *ns*)	(*we*, *os*)	(*we*, *cs*)	(*so*, *ns*)	(*so*, *os*)	(*so*, *cs*)	EMV
(*t*, –, *d*, *d*, *d*)	–80	–80	–80	40	40	40	190	190	190	10
(*t*, –, *d*, *d*, ~*d*)	–80	–80	–10	40	40	–10	190	190	–10	–11
(*t*, –, *d*, ~*d*, *d*)	–80	–10	–80	40	–10	40	190	–10	190	–1.50
(*t*, –, *d*, ~*d*, ~*d*)	–80	–10	–10	40	–10	–10	190	–10	–10	–22.50
(*t*, –, ~*d*, *d*, *d*)	–10	–80	–80	–10	40	40	–10	190	190	22.50
(*t*, –, ~*d*, *d*, ~*d*)	–10	–80	–10	–10	40	–10	–10	190	–10	1.50
(*t*, –, ~*d*, ~*d*, *d*)	–10	–10	–80	–10	–10	40	–10	–10	190	11
(*t*, –, ~*d*, ~*d*, ~*d*)	–10	–10	–10	–10	–10	–10	–10	–10	–10	–10
(~*t*, *d*, –, –, –)	–70	–70	–70	50	50	50	200	200	200	20
(~*t*, ~*d*, –, –, –)	0	0	0	0	0	0	0	0	0	0
Probability	.300	.150	.050	.090	.120	.090	.020	.080	.100	

variables. Once we have a strategic form, computing an optimal strategy is easy. We simply compute the expected utility of each strategy, and choose an optimal strategy that has the highest expected utility.

A strategic form for the oil wildcatter's problem is shown in Table III. Each row corresponds to a distinct strategy. A strategy is a 5-tuple that specifies an action in each of the 5 information sets I_1, ..., I_5. Each column corresponds to a configuration of the joint frame of all chance variables.

There are two computational problems associated with the strategic form technique. First, the number of strategies is an exponential function of the number of information sets. Second, the number of events is an exponential function of the number of random variables. Thus computing the strategic form of a game tree is computationally intractable when we have many information sets or if we have many random variables. We will now describe a method for solving a game tree representation of a decision problem using local computation (or dynamic programming).

If each information set is a singleton, then we solve the game tree using the backward recursion method of dynamic programming [Zermelo 1913, Kuhn 1953]. This technique is also called "roll-back method" in decision tree literature. A decision tree can be thought of as a game tree in which each information set is a singleton subset.

The backward recursion method of dynamic programming can be generalized to include non-singleton information sets. We start from the neighbors of utility nodes and go toward the root until all decision nodes have been pruned. The rule for pruning chance nodes is exactly the same as in the roll-back method of decision trees. The rule for pruning decision nodes is slightly different from the roll-back method of decision trees.

Pruning Chance Nodes. Suppose that we have a chance node all of whose edges lead to utility nodes. We prune a chance node in two steps. First we compute the average of the utilities at the ends of its edges using the probability distribution on its edges. Second we replace the sub-tree consisting of the chance node, its edges, and the corresponding utility nodes, by a utility node whose utility value is the average computed in the first step.

Pruning Decision Nodes. The rule for pruning decision nodes in singleton information sets is exactly the same as in decision tree. Let us review this rule. Suppose we have a decision node in a singleton information set all of whose edges lead to utility nodes. First, we compute the maximum of the utilities at the ends of all edges leading out of the decision node. Second, we identify an edge that leads to the maximum computed in the first step. If we have alternative optimal edges, then any one of the optimal edges can be identified. Third, we replace the sub-tree consisting of the decision node, its edges, and the corresponding utility nodes by a utility node whose value equals the maximum utility identified in the first step. Notice that the second step in the pruning rule is necessary to identify an optimal strategy at the end of the roll-back method.

Before we explain how to prune a decision node in a non-singleton information set, we make the following observations. Although we can prune each decision node individually, it will be efficient to prune all decision nodes in an information set before we prune other nodes. Since all decision nodes in an information set correspond to the same decision variable, we need to identify one value of the decision variable as an optimal act that the decision maker must choose if she finds herself in that information set.

Suppose we have an information set such that the edges leading out of the decision nodes end at utility nodes. First, we compute the conditional probability distribution on the decision nodes of the information set conditioned on the event that we have reached the information set (the details of this step are explained in the following paragraph). Second, for each value of the decision variable associated with the information set, we compute the expected utility using the utilities at the end of corresponding edges and using the conditional distribution computed in the first step. Third, we identify a value of the decision variable (and the corresponding edges of each decision node) associated with the maximum expected utility. Fourth, we prune each decision node by replacing the corresponding subtree by a utility node whose utility is equal to the utility at the end of its edge identified in step 3. We call this rule (for pruning decision nodes in an information set) pruning by *maximization of conditional expectation*. Pruning by maximization of conditional expectation generalizes the rule for pruning a singleton information set by maximization.

Computing the conditional distribution for an information set is easy. For each decision node in the information set, we simply multiply all probabilities on the edges in the path from the root to the decision node. If the path involves acts, then we assume the probabilities of such edges to be 1. This gives an unconditional distribution on the nodes in the information set. The sum of these probabilities gives us the probability of reaching the information set assuming prior decisions that

allow us to get there. To compute the conditional distribution, we normalize this unconditional distribution by dividing by the sum of the probabilities. As we will see shortly, from a computational perspective, the normalization step is unnecessary and can be dispensed with. It is only required for semantical reasons.

We will illustrate pruning decision nodes by conditional expectation for the game tree representation of the oil wildcatter's problem. Consider information set I_5. Notice that all three decision nodes in I_5 have edges leading to utility nodes. The unconditional distribution for the three nodes in I_5 (from top to bottom) is given by the probability vector (0.5*0.1, 0.3*0.3, 0.2*0.5) = (0.05, 0.09, 0.10). The sum of these probabilities is 0.24. The conditional distribution is given by $(\frac{0.05}{0.24}, \frac{0.09}{0.24}, \frac{0.10}{0.24}) \cong$ (0.208, 0.375, 0.417). Thus for D = *d*, the expected utility is (0.208 *–80) + (0.375*40) + (0.417*190) = 77.5, and for D = ~*d*, the expected utility is (0.208 *–10) + (0.375*–10) + (0.417*–10) = –10. Since 77.5 > –10, we identify edge D = *d* with each node in I_5. The resulting pruned game tree is shown in Figure 4.

Notice that for the purposes of pruning decision nodes in an information set, there is no need to normalize the unconditional distribution. We can compute the "expected utility" using the unconditional probability distribution since the same normalization constant (that is always positive) can be factored out of all expected values being compared. Thus, solving game trees involves no divisions.

A comment on computing the unconditional probability distribution of an information set. If we have a game tree with many variables, then computing the conditional distribution as described above can be computationally intensive. However, note that we can simplify the computation as follows. If the paths from the root to the decision nodes in an information set share a common sub-path, then the probabilities on the common sub-path can be ignored in computing the conditional distribution. This is because these probabilities get canceled in the normalization process. Furthermore, if the paths from the root to the decision nodes include edges having the same probability, then again, we can ignore these probabilities for the same reason as above. Thus to a limited extent, we can exploit conditional independencies in the joint probability distribution.

Figure 5 shows the result after we prune the entire game tree. The optimal strategy for the oil wildcatter is to conduct a seismic test; if the test result is no structure, then do not drill; and if the test result is either open structure or closed structure, then drill. The expected profit associated with the optimal strategy is $22,500.

A count of algebraic binary operations required to solve the game tree representation of the oil wildcatter's problems using the roll-back method reveals that 9 operations are required to compute the unconditional distributions at the 5 information sets, 11 operations are required to prune each of the five information sets, and 5 operations are required to prune each of the four chance nodes, for a total of 84 operations.

Figure 4. The game tree of Figure 3 after pruning decision nodes in I_5.

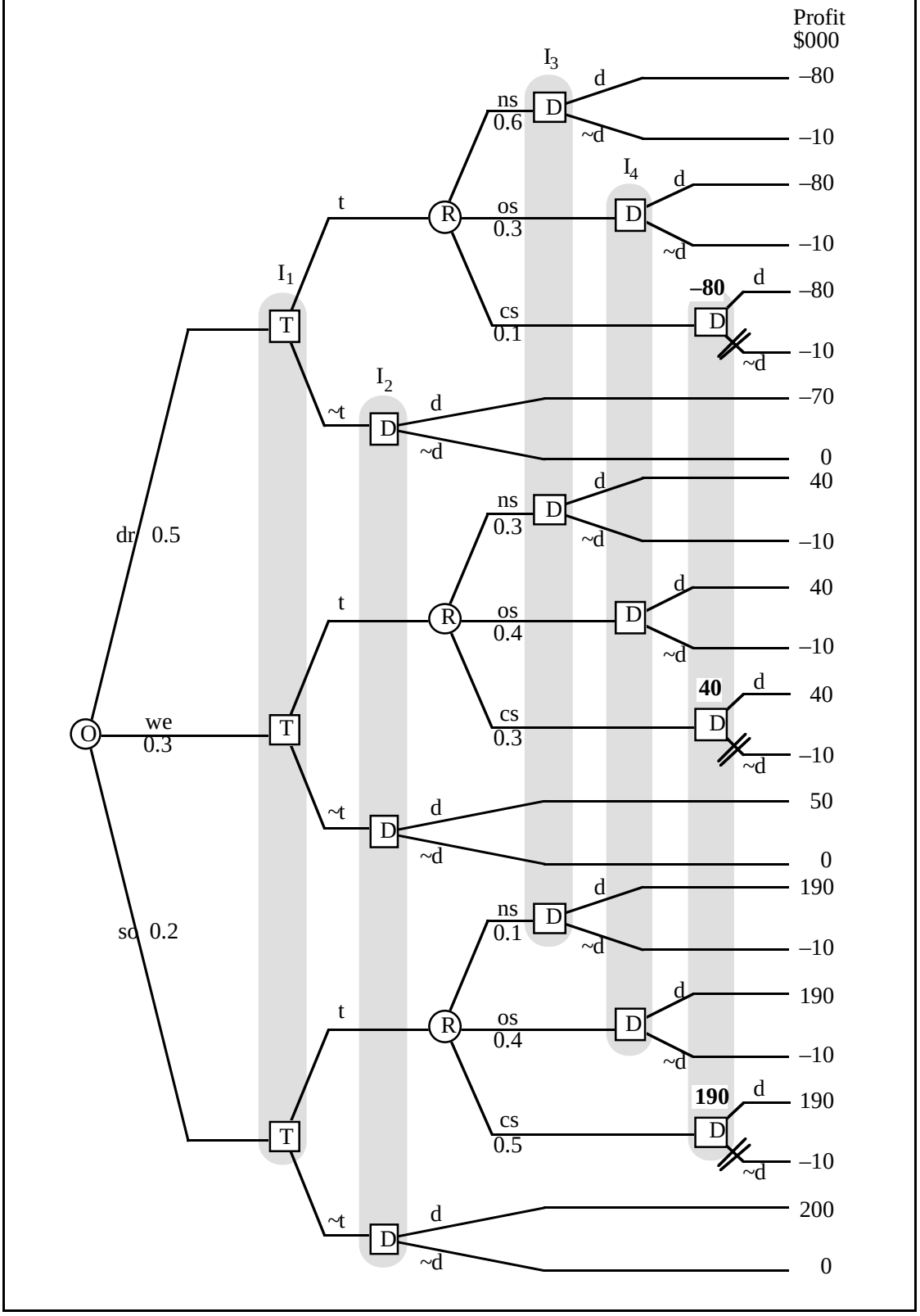
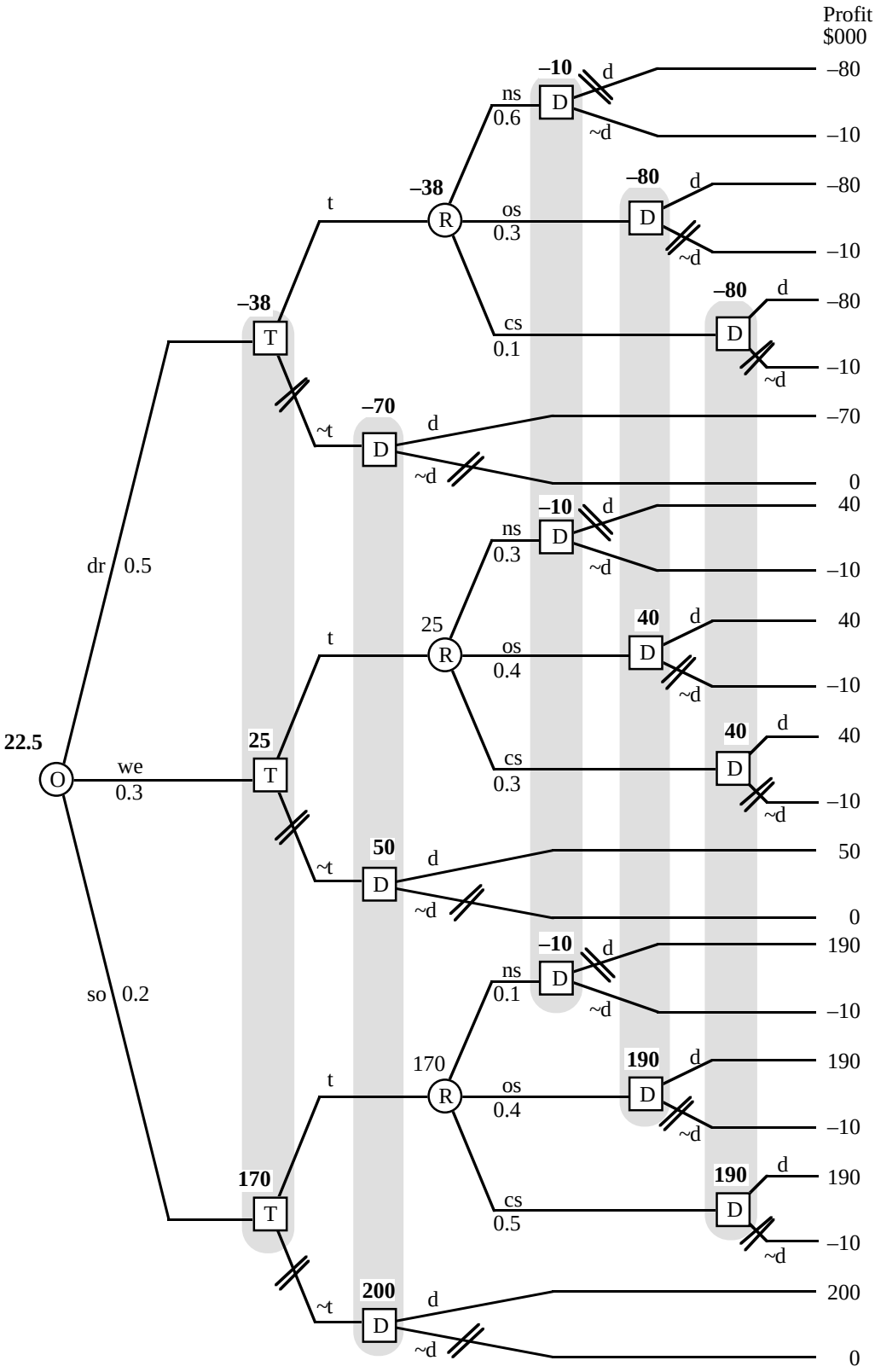


Figure 5. The game tree of Figure 3 after complete pruning.



Comparing with the corresponding decision tree representation, the roll-back method involves 30 more operations. Some of the savings in decision tree representation comes from not having chance variable O after the decision to not drill (since it is not relevant). In other words, 20 additional operations in game tree solution comes from a failure to recognize that in 4 of the 5 information sets, the decision to not drill results in a constant utility (independent of the value of amount of oil), but the game tree representation is unable to represent this fact explicitly, and therefore fails to take advantage of it. Furthermore, the roll-back method does not take advantage of the conditional expected utility computed at each information set. The conditional expected utility is used only to prune decision nodes in each information set, and then it is discarded. In [Shenoy 1995], we describe an alternative method for solving game trees that is sometimes more efficient than the roll-back method just described.

The correctness of the roll-back method described here is easy to show using dynamic programming techniques (by correctness, we mean that the rollback method will always identify an optimal strategy as defined by the strategic form of the decision problem). A sketch of a proof is as follows. The roll-back method described here uses the backward recursion technique of dynamic programming to compute an optimal behavioral strategy. Since the decision maker has perfect recall, it follows from Kuhn [1953] that behavioral strategies suffice, i.e., an optimal behavioral strategy is also an optimal strategy. Therefore the optimal behavioral strategy identified by the roll-back method is an optimal strategy.

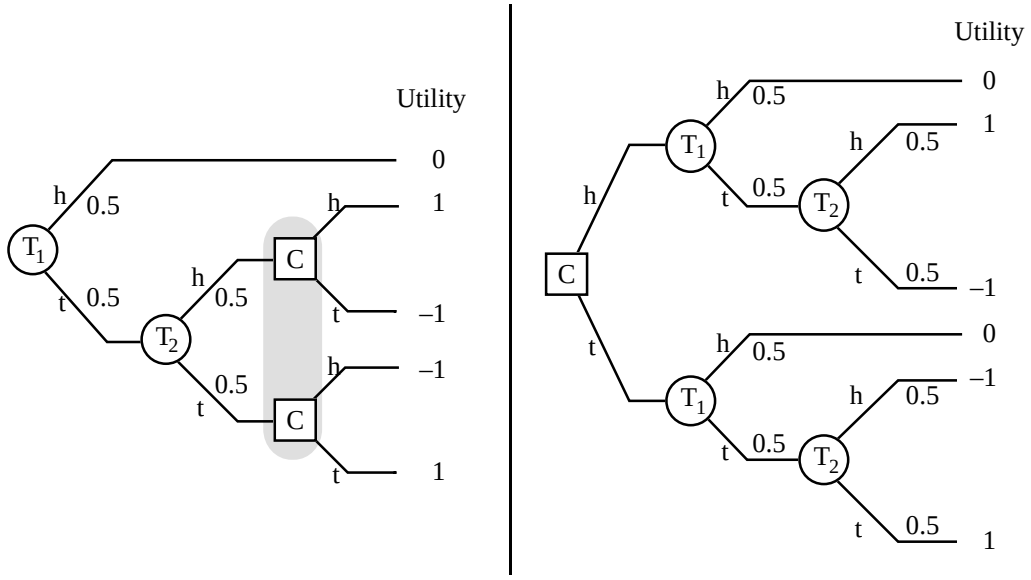
5. COMPARISON

In this section, we compare briefly game tree representation and solution method with other representation and solution methods, namely decision trees, influence diagrams, and valuation networks.

Comparison with Decision Trees. There are many similarities between decision trees and game trees. Game trees generalize decision trees. A decision tree can be considered a game tree in which all information sets are singletons. Given any game tree representation of a decision problem, we can always find a decision tree representation of the same problem (where all information sets are singletons) and we call such a decision tree representation a *corresponding* decision tree.

What makes game trees different from decision trees are information sets. Information sets give game trees added flexibility over decision trees. The sequence of variables in a game tree can represent time, causation, etc. In a decision tree, the sequence of variables must represent information. One advantage of this added flexibility is that we can structure a decision problem so that no preprocessing of probabilities is required before representing a problem as a game tree. The preprocessing of probabilities is a major drawback of decision trees.

Figure 6. Left: A game tree. Right: A corresponding decision tree.



The roll-back method in game trees is conceptually more complex than the roll-back method in decision trees since the maximization rule for pruning decision nodes in the latter is replaced by maximization of conditional expectation rule for pruning decision nodes in non-singleton information sets in the former.

We do not have any general results on the computational efficiency of the game tree roll-back method compared to decision tree roll-back method. As we saw for the oil wildcatter's problem, the game tree roll-back method is less efficient than the decision tree roll-back method. However, in the decision problem shown in Figure 6, solving the game tree representation shown using roll-back method requires only 7 algebraic binary operations whereas solving the corresponding decision tree representation shown requires 13 operations.

For the oil wildcatter's problem, the game tree representation shown in Figure 3 is bushier than the decision tree representation shown in Figure 2. The decision tree representation has only 16 leaves whereas the game tree has 24 leaves. Is this true in general? The answer is no. It is easy to construct asymmetric decision problems such that a game tree representation has fewer leaves than a corresponding decision tree. Figure 6 shows an example where this is true. In symmetric decision problems, both game trees and decision trees will have the same number of leaves (equal to the product of the sizes of the frames of all variables in the problem).

In general, in game trees, there will always be as many information sets as decision nodes in a corresponding decision tree. Since information sets may contain more than one decision node, game trees will have at least as many decision nodes as in corresponding decision trees. Thus in the oil wildcatter's problem, the game tree representation shown in Figure 3 has 15 decision nodes,

whereas the decision tree representation shown in Figure 2 has only 5 decision nodes. Typically, in decision trees, decision nodes occur closer to the root whereas in game trees, decision nodes are closer to the leaves. Thus, typically, there are more decision nodes in game trees than in corresponding decision trees.

Typically, in game trees, chance nodes occur closer to the root than in corresponding decision trees. Thus, game trees typically have less chance nodes than corresponding decision trees. For the oil wildcatter's problem, the game tree representation shown in Figure 3 has one occurrence of O and three occurrences of R, whereas the decision tree representation shown in Figure 2 has one occurrence of R and 4 occurrences of O. In asymmetric problems, game trees may have more chance nodes than in corresponding decision trees.

A major disadvantage of decision tree representation is its combinatorial explosion when there are many variables. Game trees also suffer from this disadvantage. Thus, like decision trees, game trees are practical only for small problems (i.e., with few variables).

Comparison with Influence Diagrams and Valuation Networks. The influence diagram representation was proposed by Howard and Matheson [1981]. A technique for solving influence diagram directly (without first converting them to decision trees as was described by Howard and Matheson) was proposed by Olmsted [1983] and Shachter [1986]. The valuation network representation and solution method was proposed by Shenoy [1992, 1993a, 1993b]. Both influence diagrams and valuation networks are useful representations for symmetric (or almost symmetric) problems. Figure 7 shows an influence diagram representation of the oil wildcatter's problem. The quantitative information is omitted. Figure 8 shows a valuation network representation of the oil wildcatter's problem. Again, the details of the probability functions, o and ρ, and the details of the profit functions, π and κ, are omitted.

For small highly asymmetric problems, these techniques are inefficient when compared to decision trees and game trees. Call and Miller [1990], Fung and Shachter [1990], Smith, Holtzman and Matheson [1993], Kirkwood [1993], and Covaliu and Oliver [1995] examine hybrid representation methods that combine influence diagrams and decision trees. Shenoy [1993b] describes an efficient generalization of the valuation network technique for asymmetric decision problems.

The solution techniques of influence diagrams and valuation networks use local computation not only for identifying an optimal strategy (as in game trees), but also for computing probabilities (unlike game trees). These techniques exploit conditional independencies in the joint probability distribution [Shenoy 1994]. Since conditional independence is not explicitly represented in game trees, the roll-back method of game trees is unable to take complete advantage of conditional independencies.

Figure 7. An influence diagram representation of the oil wildcatter's problem.

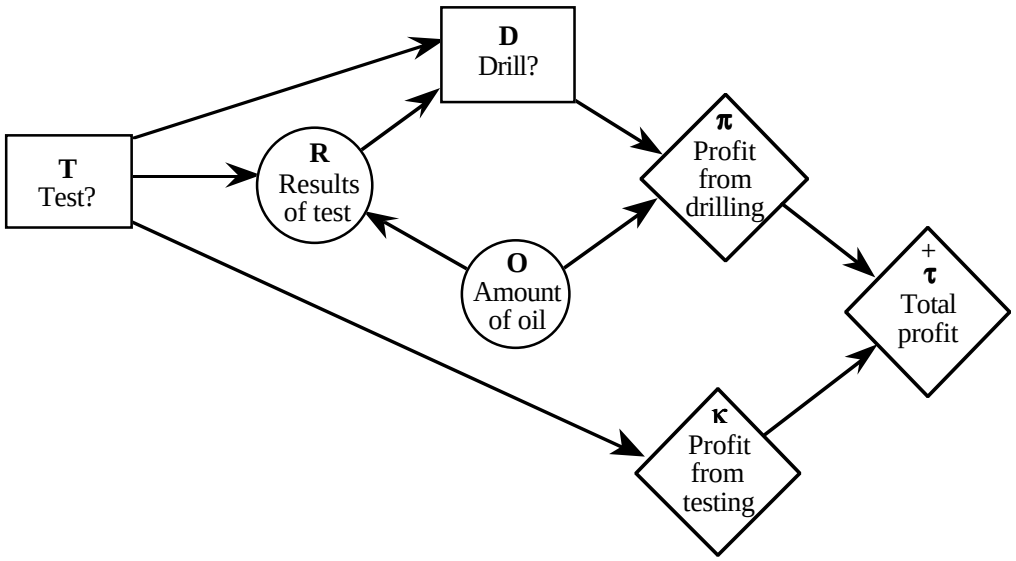
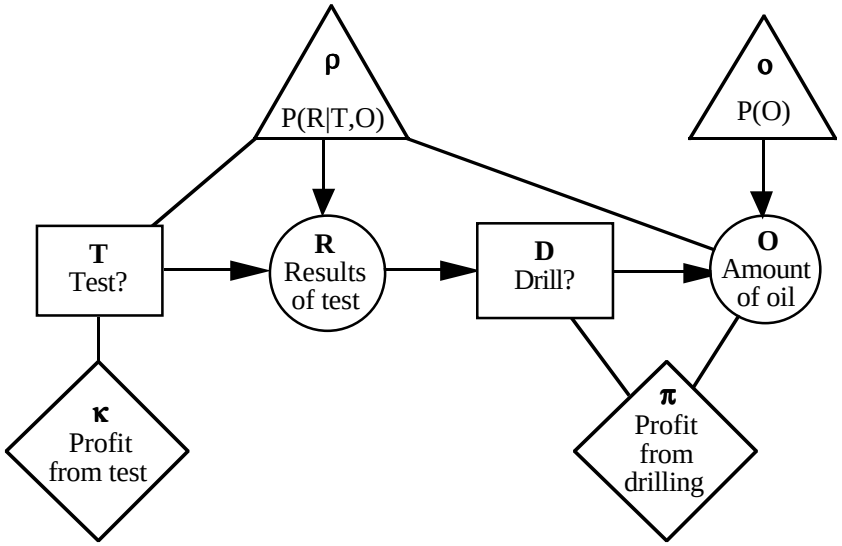


Figure 8. A valuation network representation of the oil wildcatter's problem.



6. CONCLUSION

Game trees are similar in many respects to the decision trees. Game trees were proposed first by von Neumann and Morgenstern [1944] and later generalized by Kuhn [1953] in the context of games. Game trees pre-date decision trees. Decision trees were invented by Raiffa and Schlaifer in the late 1950s. In this paper, we have simply restated Kuhn's definition of game trees for the special case of one-person games with perfect recall, i.e., for decision problems. Game trees are

more general than decision trees. A decision tree can be thought of as a special case of a game tree in which all information sets are singletons.

The main contribution of this paper is to re-introduce the concept of information sets from game theory to representation information constraints in one-person decision problems. A distinct advantage of using information sets is not having to pre-process the probabilities that is required in a decision tree representation of some decision problems. Based on this idea, Charnes and Shenoy [1996] propose a Monte Carlo method for solving decision problems that uses the conditionals stated in an influence diagram representation directly as opposed to explicitly converting them using arc reversals before doing the Monte Carlo sampling.

ACKNOWLEDGMENTS

This work is based upon work supported in part by the National Science Foundation under Grant No. SES-9213558. I am grateful to the participants of the Artificial Intelligence Seminar in general and Rui Guo in particular for their comments on an earlier draft. Comments from two anonymous referees has improved the exposition. I am grateful to Daphne Koller and an anonymous referee for pointing out the work of Wilson [1972] and Kreps and Wilson [1982].

REFERENCES

Call, H. J. and W. A. Miller (1990), "A comparison of approaches and implementations for automating decision analysis," *Reliability Engineering and System Safety*, **30**, 115–162.

Charnes, J. M. and P. P. Shenoy (1996), "A forward Monte Carlo method for solving influence diagrams using local computation," Working Paper No. 273, School of Business, University of Kansas, Lawrence, KS.

Covaliu, Z. and R. M. Oliver (1995), "Representation and solution of decision problems using sequential decision diagrams," *Management Science*, **41**(12), 1860–1881.

Fung, R. M. and R. D. Shachter (1990), "Contingent influence diagrams," Working Paper, Advanced Decision Systems, Mountain View, CA.

Hart, S. (1992), "Games in extensive and strategic forms," in R. J. Aumann and S. Hart (eds.), *Handbook of Game Theory with Economic Applications*, **1**, 19–40, North-Holland, Amsterdam.

Howard, R. A. and J. E. Matheson (1981), "Influence diagrams," reprinted in R. A. Howard and J. E. Matheson, eds. (1984), *The Principles and Applications of Decision Analysis*, **2**, 719-762, Strategic Decisions Group, Menlo Park, CA.

Kirkwood, C. W. (1993), "An algebraic approach to formulating and solving large models for sequential decisions under uncertainty," *Management Science*, **39**(7), 900–913.

Kreps, D. and R. Wilson (1982), "Sequential equilibria," *Econometrica*, **50**, 863-894.

Kuhn, H. W. (1953), "Extensive games and the problem of information," in H. W. Kuhn and A. W. Tucker (eds.), *Contributions to the Theory of Games*, **2**, 193-216, Princeton University Press, Princeton, NJ.

Olmsted, S. M. (1983), "On representing and solving decision problems," Ph.D. thesis, Department of Engineering-Economic Systems, Stanford University, Stanford, CA.

Raiffa, H. and R. O. Schlaifer (1961), *Applied Statistical Decision Theory*, Harvard Business School, Cambridge, MA.

Raiffa, H. (1968), *Decision Analysis: Introductory Lectures on Choices Under Uncertainty*, Addison-Wesley, Reading, MA.

Shachter, R. D. (1986), "Evaluating influence diagrams," *Operations Research*, **34**(6), 871-882.

Shenoy, P. P. (1992), "Valuation-based systems for Bayesian decision analysis," *Operations Research*, **40**(3), 463–484.

Shenoy, P. P. (1993a), "A new method for representing and solving Bayesian decision problems," in D. J. Hand (ed.), *Artificial Intelligence Frontiers in Statistics: AI and Statistics III*, 119–138, Chapman & Hall, London.

Shenoy, P. P. (1993b), "Valuation network representation and solution of asymmetric decision problems," Working Paper No. 246, School of Business, University of Kansas, Lawrence, KS.

Shenoy, P. P. (1994), "A comparison of graphical techniques for decision analysis," *European Journal of Operational Research*, **78**(1), 1–21.

Shenoy, P. P. (1995), "A new pruning method for solving decision trees and game trees," in P. Besnard and S. Hanks (eds.), *Uncertainty in Artificial Intelligence: Proceedings of the Eleventh Conference*, 482–490, Morgan Kaufmann, San Francisco, CA.

Smith, J. E., S. Holtzman and J. E. Matheson (1993), "Structuring conditional relationships in influence diagrams," *Operations Research*, **41**(2), 280–297.

von Neumann, J. and O. Morgenstern (1944), *Theory of Games and Economic Behavior*, 1st edition (2nd edition 1947, 3rd edition 1953), John Wiley & Sons, New York, NY.

Wilson, R. (1972), "Computing equilibria of two-person games from the extensive form," *Management Science*, **18**(7), 448–460.

Zermelo, E. (1913), "Uber Eine Anwendung Der Mengenlehre Auf Die Theorie Des Schachspiels," *Proceedings of the Fifth International Congress of Mathematics*, Cambridge, UK, **2**, 501-504.

SELECTED WORKING PAPERS

Unpublished working papers are available via ftp from
Host: ftp://ftp.bschool.ukans.edu
User ID:
Password:
Directory: /data/pub/pshenoy/
File: wpxxx.ps (put appropriate Working Paper # in place of xxx)

No. 184. "Propagating Belief Functions with Local Computations," Prakash P. Shenoy and Glenn Shafer, February 1986. Appeared in *IEEE Expert*, **1**(3), 1986, 43–52.

No. 190. "Propagating Belief Functions in Qualitative Markov Trees," Glenn Shafer, Prakash P. Shenoy, and Khaled Mellouli, June 1987. Appeared in *International Journal of Approximate Reasoning*, **1**(4), 1987, 349–400.

No. 197. "AUDITOR'S ASSISTANT: A Knowledge Engineering Tool for Audit Decisions," Glenn Shafer, Prakash P. Shenoy, and Rajendra Srivastava, April 1988. Appeared in *Auditing Symposium IX: Proceedings of 1988 Touche Ors/University of Kansas Symposium on Auditing Problems*, 61–84, School of Business, University of Kansas, Lawrence, KS.

No. 200. "Probability Propagation," Glenn Shafer and Prakash P. Shenoy, August 1988. Appeared in *Annals of Mathematics and Artificial Intelligence*, **2**(1–4), 1990, 327–352.

No. 203. "A Valuation-Based Language for Expert Systems," Prakash P. Shenoy, August 1988. Appeared in *International Journal of Approximate Reasoning*, **3**(5), 1989, 383-411.

No. 208. "Constraint Propagation," Prakash P. Shenoy and Glenn Shafer, November 1988.

No. 209. "Axioms for Probability and Belief-Function Propagation," Prakash P. Shenoy and Glenn Shafer, November 1988. Appeared in Shachter, R. D., M. Henrion, L. N. Kanal, and J. F. Lemmer (eds.), *Uncertainty in Artificial Intelligence*, **4**, 1990, 169–198. Reprinted in Shafer, G. and J. Pearl (eds.), *Readings in Uncertain Reasoning*, 1990, 575–610, Morgan Kaufmann, San Mateo, CA.

No. 211. "MacEvidence: A Visual Evidential Language for Knowledge-Based Systems," Yen-Teh Hsia and Prakash P. Shenoy, March 1989. An 8-page summary of this paper appeared as "An evidential language for expert systems," in Ras, Z. W. (ed.), *Methodologies for Intelligent Systems*, **4**, 1989, 9–16, North-Holland, Amsterdam.

No. 213. "On Spohn's Rule for Revision of Beliefs," Prakash P. Shenoy, July 1989. Appeared in *International Journal of Approximate Reasoning*, **5**(2), 1991, 149–181.

No. 216. "Consistency in Valuation-Based Systems," Prakash P. Shenoy, February 1990, revised May 1991. Appeared in *ORSA Journal on Computing*, Vol. 6, No. 3, 1994, 281–291.

No. 220. "Valuation-Based Systems for Bayesian Decision Analysis," Prakash P. Shenoy, April 1990, revised May 1991. Appeared in *Operations Research*, **40**(3), 1992, 463–484.

No. 221. "Valuation-Based Systems for Discrete Optimization," Prakash P. Shenoy, June 1990. Appeared in Bonissone, P. P., M. Henrion, L. N. Kanal, and J. F. Lemmer, eds., *Uncertainty in Artificial Intelligence*, **6**, 1991, 385–400, North-Holland, Amsterdam.

No. 223. "A New Method for Representing and Solving Bayesian Decision Problems," Prakash P. Shenoy, September 1990. Appeared in: Hand, D. J. (ed.), *Artificial Intelligence Frontiers in Statistics: AI and Statistics III*, 1993, 119–138, Chapman & Hall, London, England.

No. 226. "Valuation-Based Systems: A Framework for Managing Uncertainty in Expert Systems," Prakash P. Shenoy, March, 1991. Appeared in: Zadeh, L. A. and J. Kacprzyk (eds.), *Fuzzy Logic for the Management of Uncertainty*, 1992, 83–104, John Wiley and Sons, New York, NY.

No. 227. "Valuation Networks, Decision Trees, and Influence Diagrams: A Comparison," Prakash P. Shenoy, June 1991. Appeared as: "A Comparison of Graphical Techniques for Decision

Analysis" in *European Journal of Operational Research*, Vol. 78, No. 1, 1994, 1–21.

No. 233. "Using Possibility Theory in Expert Systems," Prakash P. Shenoy, September 1991. Appeared in *Fuzzy Sets and Systems*, **52**(2), 1992, 129–142.

No. 234. "Using Dempster-Shafer's Belief-Function Theory in Expert Systems," Prakash P. Shenoy, September 1991. Appeared in: Yager, R. R., M. Federizzi, and J. Kacprzyk (eds.), *Advances in the Dempster-Shafer Theory of Evidence*, 1994, 395–414, John Wiley & Sons, New York, NY.

No. 236. "Conditional Independence in Valuation-Based Systems," Prakash P. Shenoy, September 1991. Appeared in *International Journal of Approximate Reasoning*, **10**(3), 1994, 203–234.

No. 238. "Valuation Networks and Conditional Independence," Prakash P. Shenoy, September 1992. Appeared as "Representing Conditional Independence Relations by Valuation Networks" in *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, **2**(2), 1994, 143–165.

No. 239. "Game Trees for Decision Analysis," Prakash P. Shenoy, February 1993. Revised February 1994. An 8-page summary titled "Information Sets in Decision Theory" appeared in Clarke, M., R. Kruse and S. Moral (eds.), *Symbolic and Quantitative Approaches to Reasoning and Uncertainty*, Lecture Notes in Computer Science No. 747, 1993, 318–325, Springer-Verlag, Berlin.

No. 242. "A Theory of Coarse Utility," Liping Liu and Prakash P. Shenoy, February 1993. Revised September 1993. Appeared in *Journal of Risk and Uncertainty*, Vol. 11, 1995, pp. 17–49.

No. 245. "Modeling Ignorance in Uncertainty Theories," Prakash P. Shenoy, April 1993. Appeared in Gammerman, A. (ed.), *Probabilistic Reasoning and Bayesian Belief Networks*, 1995, 71–96, Alfred Waller, Henley-on-Thames, UK.

No. 246. "Valuation Network Representation and Solution of Asymmetric Decision Problems," Prakash P. Shenoy, April 1993. Revised September 1995. A 10-page summary of this paper appeared as "Representing and Solving Asymmetric Decision problems Using Valuation Networks" in Fisher, D. and H.-J. Lenz (eds.), *Artificial Intelligence and Statistics V*, Lecture Notes in Statistics, **112**, 99–108, Springer-Verlag, New York, 1996.

No. 247. "Inducing Attitude Formation Models Using TETRAD," Sanjay Mishra and Prakash P. Shenoy, May 1993. Revised October 1993. Appeared as "Attitude Formation Models: Insights from TETRAD" in Cheeseman, P. and R. W. Oldford (eds.), *Selecting Models from Data: Artificial Intelligence and Statistics IV*, Lecture Notes in Statistics No. 89, 1994, 223–232, Springer-Verlag, Berlin.

No. 258. "A Note on Kirkwood's Algebraic Method for Decision Problems," Rui Guo and Prakash P. Shenoy, November 1993. Revised May 1994. To appear in *European Journal of Operational Research*, 1996.

No. 261. "A New Pruning Method for Solving Decision Trees and Game Trees," Prakash P. Shenoy, March 1994. Appeared in: Besnard, P. And S. Hanks (eds.), *Uncertainty in Artificial Intelligence: Proceedings of the Eleventh Conference*, 1995, 482–490, Morgan Kaufmann, San Francisco, CA.

No. 267. "Computing Marginals Using Local Computation," Steffen L. Lauritzen and Prakash P. Shenoy, July 1995, revised May 1996.

No. 270. "Binary Join Trees," Prakash P. Shenoy, December 1995. To appear in: Horvitz, E. and F. V. Jensen (eds.), *Uncertainty in Artificial Intelligence: Proceedings of the Twelfth Conference*, 1996, Morgan Kaufmann, San Francisco, CA.

No. 271. "A Comparison of Graphical Techniques for Asymmetric Decision Problems," Concha Bielza and Prakash P. Shenoy, February 1996, revised June 1996.

No. 273. "A Forward Monte Carlo Method for Solving Influence Diagrams Using Local Computation," John M. Charnes and Prakash P. Shenoy, February 1996.