Decision Network Semantics of Branching Constraint Satisfaction Problems*

Ken Brown[1], Peter Lucas[2], and David Fowler[3]

[1]Cork Constraint Computation Centre, Department of Computer Science
University College Cork, Cork, Ireland
Email: k.brown@cs.ucc.ie

[2]Institute for Computing and Information Sciences
University of Nijmegen, Toernooiveld 1
6525 ED Nijmegen, The Netherlands
Email: peterl@cs.kun.nl

[3]Department of Computing Science, University of Aberdeen
AB24 3EU, Aberdeen, UK
Email: dfowler@csd.abdn.ac.uk

Abstract

Branching Constraint Satisfaction Problems (BCSPs) have been introduced to model dynamic resource allocation subject to constraints and uncertainty. We give BCSPs a formal probability semantics by showing how they can be mapped to a certain class of Bayesian decision networks. This allows us to describe logical and probabilistic constraints in a uniform fashion. We also discuss extensions to BCSPs and decision networks suggested by the relationship between the two formalisms.

1 Introduction

Resource allocation is the problem of assigning resources to tasks subject to constraints, and has been studied in operations research and computer science for many years [1, 2, 9]. Recently, the problem has been investigated using constraint satisfaction methods [17], which allow arbitrary combinatorial constraints to be placed on the problem. In its simplest form, tasks can be represented by variables, and resources by values to be assigned to the variables, while constraints restrict the values that can be assigned simultaneously. A solution to a problem is then an assignment of values such that all constraints are satisfied. Initially, such approaches were restricted to deterministic, static problems; more recently it has been extended to problems that change over time, and for which there is some uncertainty about what the changes will be. *Branching Constraint Satisfaction* [6, 7] has been proposed to model problems where new variables (or tasks) are added to the problem after some decisions have been made. The uncertainty in the sequence of additions is modelled by a transition

*Published in: Th.D. Nielsen and N.L. Zhang (eds.), *Symbolic and Quantitative Approaches to Reasoning with Uncertainty*: Proc ECSQARU 2003, LNAI2711. Springer, Berlin, 2003, pp. 230–242.

tree with arcs labelled with probabilities. Branching CSPs are known to be NP hard [8]. Complete and incomplete optimising algorithms have been developed, using a combination of constraint-based tree search and decision-theoretic computation, and the methods have been compared to those used in Markov Decision Problems [8]. However, the probability semantics of BCSPs were presented only informally.

Bayesian networks have been introduced as formalisms to represent and reason with joint probability distributions, taking into account conditional independence statements [15]. Given a Bayesian network and a (possibly empty) set of evidence concerning the variables included in the network, the probability distribution on any subset of variables can be computed. For this, efficient algorithms exist with fast, well-engineered computer implementations [10, 13], even though the problem of probabilistic reasoning in Bayesian networks is known to be NP hard in general [4]. However, reasoning with Bayesian networks for real-world problems is normally feasible. Formally, Bayesian networks can only be used for probabilistic reasoning, but recent work [14, 12] shows how some logical consistencies can be modelled and solved. Finally, we can augment a network with decision theory, to obtain *influence diagrams* or *decision networks*, which can be used for decision-making under uncertainty [10, 16].

In this paper, we study the relationship between branching CSPs, Bayesian networks and decision networks. Our aim is to establish the probability semantics by mapping BCSPs to decision networks, providing a uniform representation for probabilistic and logical constraints. We introduce Branching CSPs, giving a precise, formal definition, and we summarise Bayesian networks and decision networks. We then show how BCSPs can be mapped to decision networks, and in particular we show how to represent combinatorial constraints. We prove that optimal solutions to problems in the two different formalisms are equivalent. Finally we consider how the techniques of decision networks may be used to generalise BCSPs, and, similarly, how BCSP methods might allow us to make explicit use of constraints in decision networks.

2 Branching constraint satisfaction problems

2.1 Preliminary definitions

In the following, we borrow the terminology for graphs from [18]; if $S = (V, A)$ is a directed tree with set of vertices V and set of directed arcs $A \subseteq V \times V$, then the set of children of a vertex $v \in V$ is denoted by $\sigma(v)$; the unique parent of a vertex $v \in V$ is represented by $\pi(v)$. Furthermore, the *level* of a vertex v is defined as the length of the path from the root to v [11]. The set of all vertices in the tree at the same level $n \in \mathbb{N}$ is denoted by $\lambda(n)$. The terminology will be generalised for acyclic directed graphs. Sets of elements will be represented by bold face letters, e.g. $\mathbf{V}$, if confusion may arise otherwise.

2.2 A motivating example

We first present a simple motivating example. A company has three workers, x, y and z, and five possible tasks, A, B, C, D and E that it may be asked to carry out. Each worker is qualified to do some of the tasks, as shown in Fig. 1; each task is associated with a utility, representing the profit resulting from completing the task successfully. No worker can do more than one task. The company has some uncertain knowledge about the sequence of tasks it will be asked to perform, sketched as a probabilistic state transition tree in Fig. 1. There

	Tasks				
Staff	A	B	C	D	E
x	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$
y	$\surd$	$-$	$-$	$-$	$\surd$
z	$\surd$	$-$	$\surd$	$-$	$-$
Utilities	3	6	10	6	6

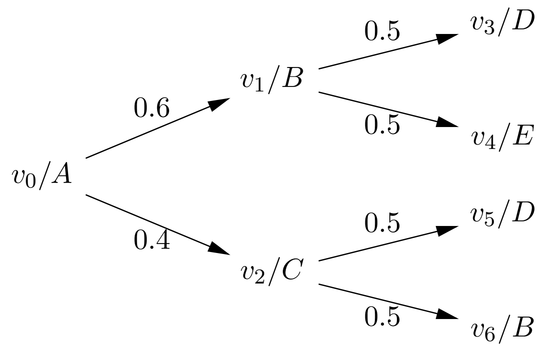

Figure 1: An example BCSP. *Left:* table of staff skills and tasks, with associated utilities for individual tasks; $\surd$ means that the task is suitable for the worker, and '$-$' that it is unsuitable. *Right:* probabilistic state transition tree. An entry v/X indicates that variable X arrives in vertex v; numeric labels on the arcs indicate transition probabilities.

will definitely be three tasks, and the first task to arrive will be A. Subsequently, either task B or C will arrive, with probabilities 0.6 and 0.4 respectively. If the second task is B, then the last task will be either D (with probability 0.5) or E (probability 0.5). If the second task is C, then the last task will either be D (0.5) or B (0.5). Some sequences of tasks may not be feasible for the company to do, and so it may choose to reject some tasks. The aim is to assign workers to tasks as soon as the tasks arrive, maximising the expected utility, while ensuring all constraints are satisfied.

2.3 Formal definition

We give the formal definition of branching CSPs below. $\top$, or *null*, is a special value used to represent an explicit decision not to assign a value to a variable. An assignment of $\top$ to a variable will mean that any constraint on that variable will be satisfied by default.

Definition 1 *A* binary branching CSP *is a tuple* BCSP $= (X, D, \delta, C, U, S, \tau)$*:*

- *X is a finite set of* variables*;*
- *D is a finite set of* values*, with function $\delta : X \to \wp(D \cup \{\top\})$ associating a* domain *of possible values to each variable $x \in X$, such that $\top \in \delta(x)$ for each $x \in X$;*
- *C is a finite set of binary* constraints*, where each $c \in C$ is a set of triples (x, y, R), $x, y \in X$, and $R \subseteq \delta(x) \times \delta(y)$ such that $\forall a \in \delta(x) \forall b \in \delta(y) : (\top, b) \in R$ and $(a, \top) \in R$;*
- *$U : X \times (D \cup \{\top\}) \to \mathbb{R}$ associates a* utility *to each value $w \in D \cup \{\top\}$ assigned to a variable $x \in X$, with $U(x, \top) = 0$ for each $x \in X$;*
- *$S = (V, A, \gamma)$ is a probabilistic state transition tree with vertices V and arcs A; there is a distinguished vertex $v_0 \in V$ called the* root*, which has no parent; the function $\gamma : V \times V \to [1, 0]$ is defined such that $\gamma(v, v') = 0$ if $(v, v') \notin A$, and if $\sigma(v) \neq \varnothing$, $\sum_{v' \in \sigma(v)} \gamma(v, v') = 1$, for each $v \in V$; γ represents the conditional probability that vertex v' is the next to become active, given that the previous active vertex was v;*

- *$\tau : V \to X$ is a surjective function such that for any two vertices v, v' on the same path p in S, $v = v'$ if $\tau(v) = \tau(v')$. τ assigns a variable to each vertex, ensuring that no variable appears twice on a path from root to leaf.*

The probabilistic transitions are defined in terms of the vertices of the tree, and not directly in terms of the variables. Each vertex represents an *event*, and multiple different events may cause the same variable to become active. The probability of an event depends only on its immediate predecessor, and thus the problem obeys the Markov property.

Definition 2 *An* assignment *to a BCSP is a function $\varphi : V \to D \cup \{\top\}$ which assigns to each vertex either a value from the domain of its associated variable or the null value $\top$.*

Definition 3 *A* solution *to a BCSP is an assignment φ such that if v and w are vertices on a path in $S = (V, A, \gamma)$ and $(\tau(v), \tau(w), R) \in C$, then $(\varphi(v), \varphi(w)) \in R$, i.e. φ satisfies all constraints appearing on a path.*

Definition 4 *The* expected utility *of a vertex v in a solution φ to a BCSP, denoted by $\hat{U}_\varphi(v)$, is defined as*

$$\hat{U}_\varphi(v) = U(\tau(v), \varphi(v)) + \sum_{v' \in \sigma(v)} \gamma(v, v')\hat{U}_\varphi(v')$$

The expected utility of a solution to a BCSP is the expected utility of the root vertex in the solution, i.e. $\hat{U}_\varphi(v_0)$.

Note that a solution φ is a contingent solution, specifying an assignment to a variable dependent on the sequence of arrivals. In fact, the assignments are defined in terms of events (i.e. vertices of the tree), and not directly in terms of the variables. Further, the solution can be executed as the problem unfolds; the assignments are not dependent on subsequent developments of the problem. Thus the solution is a *policy.*

Definition 5 *Let v_i be a vertex at level i in the tree $S = (V, A, \gamma)$, and let $\mathbf{h} = \{(v_0, x_0), (v_1, x_1), \ldots, (v_{i-1}, x_{i-1})\}$ be the* history *of assignments made at vertices in the path from v_0 to v_i, with $v_{j+1} \in \sigma(v_j)$, in some solution φ. We say that the pair (v_i, x_i) is* consistent *with $\mathbf{h}$, written $(v_i, x_i) \propto \mathbf{h}$, if it satisfies all constraints between v_i and assignments in $\mathbf{h}$.*

Definition 6 *The* maximum expected utility *at a vertex v given its history $\mathbf{h}$ is the maximum, over consistent assignments, of the utility of an assignment plus the weighted sum of the maximum expected utility of the child vertices, given the history $\mathbf{h}$ extended with the new assignment:*

$$\hat{U}(v \mid \mathbf{h}) = \max_{x \in \delta(\tau(v)) : (v,x) \propto \mathbf{h}} \left[U(\tau(v), x) + \sum_{v' \in \sigma(v)} \gamma(v, v')\hat{U}(v' \mid \mathbf{h} \cup \{(v, x)\}) \right]$$

The goal of a BCSP is to find a solution with maximal expected utility. The maximal expected utility is thus $\hat{U}(v_0 \mid \varnothing)$. A BCSP is essentially a decision tree, but which separates out the probabilities from the logical constraints on the decisions. It is possible to combine the constraints into the tree, but at the cost of a (worst-case) exponential explosion in the tree size [8].

Reconsider the example introduced above. Formulated as a branching CSP it holds that $X = \{A, B, C, D, E\}$, $D = \{x, y, z\}$, $\delta(A) = \cdots = \delta(E) = \{x, y, z, \top\}$, $U(x, \top) = 0$ for each $x \in X$, and for each $w \in D$: $U(A, w) = 3$, $U(B, w) = 6$, $U(C, w) = 10$, $U(D, w) = 6$, $U(E, w) = 5$, and the constraint set C consists of the following elements:

- $\langle A, B, \{(x, \top), (y, x), (y, \top), (z, x), (z, \top), (\top, x), (\top, \top)\}\rangle$
- $\langle A, C, \{(x, z), (x, \top), (y, x), (y, z), (y, \top), (z, x), (z, \top), (\top, x), (\top, z), (\top, \top)\}\rangle$
- $\langle A, D, \{(x, \top), (y, x), (y, \top), (z, x), (z, \top), (\top, x), (\top, \top)\}\rangle$
- $\langle A, E, \{(x, y), (x, \top), (y, x), (y, \top), (z, x), (z, y), (z, \top), (\top, x), (\top, y), (\top, \top)\}\rangle$
- $\langle B, C, \{(x, z), (x, \top), (\top, x), (\top, z), (\top, \top)\}\rangle$
- $\langle B, D, \{(x, \top), (\top, x), (\top, \top)\}\rangle$
- $\langle B, E, \{(x, y), (x, \top), (\top, x), (\top, y), (\top, \top)\}\rangle$
- $\langle C, D, \{(x, \top), (z, x), (z, \top), (\top, x), (\top, \top)\}\rangle$
- $\langle C, E, \{(x, y), (x, \top), (z, x), (z, y), (z, \top), (\top, x), (\top, y), (\top, \top)\}\rangle$
- $\langle D, E, \{(x, y), (x, \top), (\top, x), (\top, y), (\top, \top)\}\rangle$

The probabilistic state transition tree $S = (V, A, \gamma)$ with the definition of the function τ is according to Fig. 1. The optimal solution is $\varphi(v_0) = y$, $\varphi(v_1) = x$, $\varphi(v_2) = z$, $\varphi(v_3) = \top$, $\varphi(v_4) = \top$, $\varphi(v_5) = x$, $\varphi(v_6) = x$, with expected utility $\hat{U}_\varphi(v_0) = 13$. Note that the task D is given a different allocation depending on the arrival sequence: it is rejected if it arrives in event v_3 (after B in v_1), but it is allocated worker x if it arrives in event v_5 (after C in v_2).

The definition above is a slightly modified form of the one given in [7]. There it was assumed that the the utility function U did not distinguish between different values for a given variable (with the exception of $\top$); i.e. $U(x, v) = U(x, v')$ for each $v', v \in \delta(x)\backslash\{\top\}$. Also, in the probabilistic state transition tree, the sum of the transition probabilities for the children of a vertex was allowed to be less than 1. The missing probability represented the case where the parent event had no successor. In the definition given here, we could represent this by having a special variable whose domain is restricted to $\top$, and ensuring any vertex which activates this variable has no children.

3 Bayesian networks and decision networks

A *Bayesian network* $\mathcal{B}$ is a pair $\mathcal{B} = (G, P)$, where $G = (\mathbf{N}, A)$ is an acyclic directed graph with set of chance nodes $\mathbf{N}$, representing random variables, and set of arcs $A \subseteq \mathbf{N} \times \mathbf{N}$, representing statistical independence relationships among the variables [15]. Here we assume all random variables to be discrete. A joint probability distribution P is defined on the set of variables as follows:

$$P(N) = \prod_{X \in \mathbf{N}} P(X \mid \pi(X))$$

A Bayesian network allows for computing any a posteriori probability distribution of interest after entering evidence $\mathbf{e}$ into the network. In Bayesian network software packages, a posteriori probability distributions are computed from the marginal probability distribution of an updated probability distribution $P^{\mathbf{e}}$; for every (free) variable $X \in \mathbf{N}$, it holds that

$$P^{\mathbf{e}}(X) = P(X \mid \mathbf{e})$$

A *decision network* $\mathcal{D} = (G, P, \mathbf{N}, \mathbf{D}, \mathbf{W}, \mathbf{u})$, or *influence diagram*, is a Bayesian network with the addition of decision nodes $\mathbf{D}$ and utility nodes $\mathbf{W}$, standing for decision and utility variables, respectively. There is always a unique directed path in a decision network, on which every decision node D in $\mathbf{D}$ occurs, i.e. decision nodes are linearly ordered. Each utility variable $W \in \mathbf{W}$ stands for a utility function $u_W : \delta(\mathbf{Z}) \to \mathbb{R}$, where $\delta(\mathbf{Z})$ is the Cartesian product of the domains of variables in $\mathbf{Z}$, and $\mathbf{Z} = \pi(W)$. The collection of utility functions is indicated by $\mathbf{u}$.

Initial proposals of decision networks only included a single utility node. In more recent descriptions, such as in the book by Jensen [10], a decision network may incorporate more than one utility node W_i, $i = 1, \ldots, n$, and it is assumed that the resulting multi-attribute utility function $u_{\mathbf{W}}$ is additive, i.e. the resulting utility $u_{\mathbf{W}}$ is defined as follows:

$$u_{\mathbf{W}}(\mathbf{Z}) = \sum_{i=1}^{n} u_{W_i}(\mathbf{Z}_i)$$

where $\mathbf{Z}_i = \pi(W_i)$, $\mathbf{W} = \bigcup_{i=1}^{n} \{W_i\}$, $\mathbf{Z} = \bigcup_{i=1}^{n} \mathbf{Z}_i = \pi(\mathbf{W})$. Clearly, defining a utility function in this fashion reduces the amount of utility information that has to be specified; the space-complexity reduction can be as drastic as from exponential to linear.

The aim of evaluating a decision network is to determine the optimal expected utility $\hat{u}$ for each decision d at a given decision node D, given the available evidence $\mathbf{e}$, which includes all previously made decisions. We assume a topological order $\prec$ of the nodes in the network, in which we have combined consecutive nodes of the same type, and we place the utility nodes last. Thus we have $Y_0 \prec D_0 \prec Y_1 \prec D_1 \prec \cdots \prec D_{n-1} \prec Y_n \prec W$. We then define the maximum expected utility at a decision node D_i given some evidence $\mathbf{e}$ to be

$$\hat{u}_{D_i}(\mathbf{e}) = \max_{d_i \in D_i} \hat{u}_{Y_{i+1}}(\mathbf{e} \cup \{D_i = d_i\})$$

and at a chance node Y_i the expected utility is

$$\hat{u}_{Y_i}(\mathbf{e}) = \sum_{y_i \in Y_i} P(Y_i = y_i \mid \mathbf{e}) \hat{u}_{D_i}(\mathbf{e} \cup \{Y_i = y_i\})$$

In particular, we have the expected utility over the whole network:

$$\hat{u}_{Y_0}(\varnothing) = \sum_{y_0 \in Y_0} P(Y_0 = y_0) \hat{u}_{D_0}(\{Y_0 = y_0\})$$

and for the terminating case we have:

$$\hat{u}_{Y_n}(\mathbf{e}) = \sum_{y_n \in Y_n} P(Y_n = y_n \mid \mathbf{e}) u_W(\mathbf{e} \cup \{Y_n = y_n\})$$

In diagrams of Bayesian networks and decision networks, chance nodes are indicated by circles or ellipses, decision nodes by boxes and utility nodes by diamonds.

4 Relationship of branching CSPs to decision networks

4.1 Mapping branching CSPs to decision networks

Let BCSP $= (X, D, \delta, C, U, S, \tau)$. Below, we define the steps that make up the mapping from this representation to a decision network $\mathcal{D} = (G, P, \mathbf{N}, \mathbf{D}, \mathbf{W}, \mathbf{u})$.

- For each set of vertices $\lambda(n)$ at level $n \in \mathbb{N}$ of the tree S, there is a chance node Y_n. The domain of the associated random variable Y_n is $\delta(Y_n) = \{v \mid v \in \lambda(n)\}$. The associated probability distribution P is defined by: $P(Y_n = u \mid Y_{n-1} = v) = \gamma(v, u)$ for $n > 0$, and $P(Y_0 = v_0) = 1$. Note that for two vertices $v \in \lambda(n-1)$ and $u \in \lambda(n)$ with $(v, u) \notin A_S$ we have that $P(Y_n = u \mid Y_{n-1} = v) = 0$, indicating that this transition cannot take place.

- Corresponding to each random variable Y_n with domain $\delta(Y_n)$, there is a decision node D_n, with domain equal to $\delta(D_n) = \{v.x \mid v \in \delta(Y_n), x \in \delta(\tau(v))\}$. There exists an incoming arc to each decision node from its associated chance node. In addition, the decision nodes are linked in a chain in an order reflecting the order of their associated chance nodes. The nodes will be used to assign values to their associated decision variables, which corresponds to assigning values to variables in the BCSP.

- For each chance node Y_n there is a corresponding utility node U_n. The parents of U_n are Y_n and the decision node D_n. If Y_n takes value v, and the decision node D_n takes any value $v.x$, $x \neq \top$, then the utility value is $U(\tau(v), x)$; otherwise it is 0. The utility nodes give their reward if a vertex (and hence a variable) has become active, and we have assigned a non-null value to that instance of the variable.

- For each pair of chance nodes (Y_i, Y_j) such that there are vertices $v \in \delta(Y_i)$ and $v' \in \delta(Y_j)$ with a constraint $(\tau(v), \tau(v'), R) \in C$, there exists a chance node $C_{i,j}$ with domain $\{t, f\}$ to represent the constraints on the corresponding decisions. The parents of $C_{i,j}$ are the decision nodes D_i and D_j. The probability distribution is defined as follows:

$$P(C_{i,j} = t \mid D_i = v.x, D_j = w.y) = \begin{cases} 0 & \text{if } (\tau(v), \tau(w), R) \in C, (x, y) \notin R \\ 1 & \text{otherwise} \end{cases}$$

- There is one distinguished utility node U_C, whose parents are all the constraint chance nodes, with utility value 0 if all parents have value t, and utility value equal to $-M$ otherwise, where M is a penalty value larger than the sum of all utilities in the BCSP. This node ensures that the constraints are satisfied.

- Finally, for any given history in the execution of a BCSP solution, there is a corresponding evidence set for the network, defined by the function β below. Let $\mathbf{H}$ be the set of all possible history sets, and $\mathbf{E}$ be the set of all possible evidence sets. Then

$$\beta : \mathbf{H} \to \mathbf{E} : \mathbf{h} \mapsto \{Y_i = v, D_i = v.x : (v, x) \in \mathbf{h}, v \in \lambda(i)\}$$

Note that there are particular features of the mapping above, which can be exploited to simplify the utility calculations:

(1) Each variable Y_j is conditionally independent of each variable Y_k, $k = 0, \ldots, j-2$, and of each decision variable D_l given variable Y_{j-1}.

(2) The utility function u defined above for the utility nodes U_j is additive:

$$u(\tau(y_0), d_0, \ldots, \tau(y_m), d_m) = \sum_{i=0}^{m} U(\tau(y_j), d_j)$$

where y_j is a possible value of random variable Y_j and d_j is a possible value of decision variable D_j.

(3) We can create a topological order $Y_0 \prec D_0 \prec Y_1 \prec D_1 \prec \cdots \prec D_n \prec \mathbf{C} \prec \mathbf{W}$ where $\mathbf{C}$ represents the constraint chance nodes, and $\mathbf{W}$ represents the utility nodes. The initial node Y_0 has domain $\{v_0\}$, so the maximum expected utility of the network $\hat{u}_{Y_0}(\varnothing) = \hat{u}_{D_0}(Y_0 = v_0)$, and the utility function $u_W(\mathbf{e}) = \sum_{i=0}^{n} U(\tau(y_i), d_i) + u_C(\mathbf{e})$.

We can now simplify the utility definitions as follows:

$$\hat{u}_{D_i}(\mathbf{e}) = \max_{d_i \in D_i} \left[U(\tau(y_i), d_i) + \hat{u}_{Y_{i+1}}(\mathbf{e} \cup \{D_i = d_i\}) \right]$$

$$\hat{u}_{D_n}(\mathbf{e}) = \max_{d_n \in D_n} \left[U(\tau(y_n), d_n) + \hat{u}_C(\mathbf{e} \cup \{D_n = d_n\}) \right]$$

The $\hat{u}_C$ term in the second equation is simply the maximum expected utility from the constraint nodes. If any of the constraints evaluate to false, then the utility is $-M$. Otherwise, it is 0.

The highest expected utility at the first decision node is equal to the optimal expected utility of the BCSP, and the optimal decisions of the decision nodes correspond to the optimal plan of the BCSP. We prove this in the next section.

The result of mapping the example BCSP discussed in Section 2 is shown in Fig. 2. From the mapping designed above, it follows that the domain of the variable Y_0 is equal to $\{v_0\}$, for Y_1 it is equal to $\{v_1, v_2\}$; the domain of the decision variable D_0 is $\{v_0.x, v_0.y, v_0.z, v_0.\top\}$, and for D_1 it is equal to $\{v_1.x, v_1.\top, v_2.x, v_2.z, v_2.\top\}$.

4.2 Proof that the mapping is correct

We need to show that the optimal solution to the BCSP (i.e. the maximum expected utility at the root node) has the same value as the maximum expected utility of the first decision node in the network.

We will show that the maximum expected utility from any node in the tree given some history is the same as the maximum expected utility from the corresponding decision node in the decision network, given the corresponding evidence.

Theorem 1 *Let $\mathcal{D} = (G, P, \mathbf{N}, \mathbf{D}, \mathbf{W}, \mathbf{u})$ be the decision network corresponding to the* BCSP $=$ $(X, D, \delta, C, U, S, \tau)$ *obtained by the mapping defined in Section 4.1, then for each node at level k it holds that:*

$$\hat{U}(v_k \mid \mathbf{h}) = \hat{u}_D(\beta(\mathbf{h}) \cup \{Y_k = v_k\})$$

Proof: (By backwards induction on the level of the node in the tree.)

Basis Suppose v is a vertex in $\lambda(n)$, with n maximal level. Then v must be a leaf vertex. It holds that

$$\hat{U}(v \mid \mathbf{h}) = \max_{x \in \delta(\tau(v)) : (v,x) \propto \mathbf{h}} U(\tau(v), x)$$

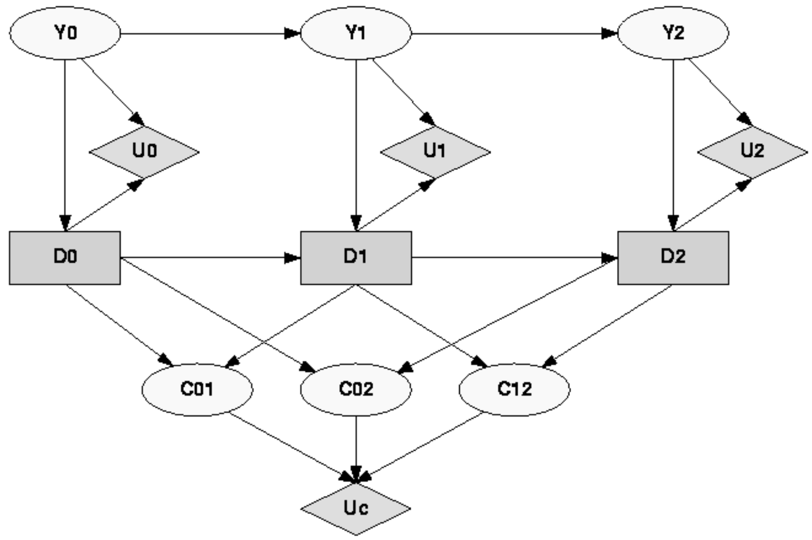

Figure 2: Decision network resulting from the mapping of the example BCSP.

and Y_n and D_n are its corresponding chance and decision nodes. For the decision network, v must be the value observed at chance node Y_n, so we have:

$$\hat{u}_{D_n}(\beta(\mathbf{h}) \cup \{Y_n = v\}) = \max_{d_n \in D_n} [U(\tau(v), d_n) + \hat{u}_C(\beta(\mathbf{h}) \cup \{Y_n = v, D_n = d_n\}]$$

By the definition of the penalty value, we only need to consider those d_n which do not violate the constraints. There will always be at least one, namely $v.\top$, and so the $\hat{u}_C$ term will be 0. Thus we have as required

$$\hat{u}_{D_n}(\beta(\mathbf{h}) \cup \{Y_n = v) = \max_{x \in \delta(\tau(v)):(v,x) \propto \mathbf{h}} U(\tau(v), x)$$

Induction hypothesis Suppose that

$$\hat{U}(v_j \mid \mathbf{h}) = \hat{u}_{D_j}(\beta(\mathbf{h}) \cup \{Y_j = v_j\})$$

holds for all vertices in the BCSP at levels $j = n, n-1, \ldots, i+1$.

Induction step Now consider a vertex v in the BCSP at level i. If v is a leaf, then the result is true by the basis argument. Now, suppose v is not a leaf, then it holds that:

$$\hat{U}(v \mid \mathbf{h}) = \max_{x \in \delta(\tau(v)):(v,x) \propto \mathbf{h}} \left[U(\tau(v), x) + \sum_{v' \in \sigma(v)} \gamma(v, v') \hat{U}(v' \mid \mathbf{h} \cup \{(v, x)\}) \right]$$

but v' must be a node at level $i+1$, so by the induction hypothesis

$$= \max_{x \in \delta(\tau(v)):(v,x) \propto \mathbf{h}} \left[U(\tau(v), x) + \sum_{v' \in \sigma(v)} \gamma(v, v') \hat{u}_{D_{i+1}}(\beta(\mathbf{h}) \cup \{Y_i = v, D_i = v.x, Y_{i+1} = v'\}) \right]$$

but since all $v_{i+1} \in Y_{i+1}$ with $v_{i+1} \notin \sigma(v)$ give a zero probability, and the decisions in D_i which give a non-negative utility are exactly those in $\delta(\tau(v))$ which satisfy the constraints in $\mathbf{h}$

$$= \max_{d_i \in D_i} \left[U(\tau(v), d_i) + \sum_{v' \in Y_{i+1}} P(Y_{i+1} = v' \mid Y_i = v) \hat{u}_{D_{i+1}}(\beta(\mathbf{h}) \cup \{Y_i = v, D_i = d_i, Y_{i+1} = v'\}) \right]$$

$$= \max_{d_i \in D_i} \left[U(\tau(v), d_i) + \hat{u}_{Y_{i+1}}(\beta(\mathbf{h}) \cup \{Y_i = v, D_i = d_i\})\right]$$

$$= \hat{u}_{D_i}(\beta(\mathbf{h}) \cup \{Y_i = v\})$$

and thus we have proved the result by induction. □

As a corollary, we obtain that if the root node of the BCSP has an empty history, we can write $\hat{U}(v_0) = \hat{u}_{D_0}(Y_0 = v_0)$. Thus, we have proved the equivalence of the two representations of the problem.

5 Future work: generalised branching CSPs

The mapping designed in Section 4.1 not only offers a decision-theoretic description of a BCSP's components, but also indicates how these components interact. The decision networks that are produced are of a restricted form. Studying these restrictions suggests ways in which BCSPs might be generalised to handle a wider range of problems.

The Bayesian network component of the resulting decision network is a linear chain $Y_0 \rightarrow Y_1 \rightarrow \cdots \rightarrow Y_n$ with a completely certain initial event, whereas general Bayesian networks are directed acyclic graphs. This restriction arises from the BCSP state transition tree, and the fact that the root node of the tree is the known arrival of the first variable. The certain initial event can easily be relaxed by having an empty root vertex with a number of possible children, but relaxing the cause of the linear chain would require replacing the tree with a directed acyclic graph more similar in style to a Bayesian network. This would allow us to represent events which have multiple conditionally independent successors, while maintaining a temporal interpretation of the arcs, instead of mutually exclusive children as at present. Similarly, we could represent mutually independent parents of an event, instead of single parents. We would also be able to make a distinction between temporal (state-transition) and atemporal arcs, thus giving us a structure similar to a dynamic Bayesian network [5].

BCSPs currently assume that the arrival of variables is governed by uncertainty, but actual decisions to assign a value to a variable do not influence the uncertainty. By adding observation nodes to the uncertainty structure, linking these to explicit decision nodes, and taking these observations into account when assessing utilities, we could model situations where decisions may have an effect on the future distribution of tasks.

If we introduce both non-temporal arcs and explicit decision nodes, then we can represent problems where instant decisions are not necessary. Solutions to the problem could wait until more evidence had been received before making a decision (a restricted form of this was proposed in [7]), or the solution method would be required to decide upon the best sequence of decisions.

It should be noted that new algorithms would be required for the BCSP generalisations discussed above, and that these algorithms might be neither easy to develop nor efficient. Further study will be aimed at determining which of the generalisations still allow us to solve BCSPs in reasonable time (in the average case). A different approach might be to add explicit logical constraints into a decision network, and then attempt to produce BCSP-style algorithms for these extended decision networks.

Finally, although we have presented a mapping from BCSPs to decision networks, we have said nothing about the complexity of the mapping, or the ease of constructing representations of problems. One of the advantages of Constraint Programming in general is the ease of modelling, and the simplicity with which complex combinatorial constraints can be expressed. Similarly, constraint algorithms are design to take advantage of the structure of the constraints. We need to establish the complexity of the transformation in Section 4.1, and determine what effect the extensional representation of the constraints has on the running time of the decision network algorithms. Results here would help indicate which of our plans for future work would be most profitable.

6 Conclusions

In this paper, we have given a decision-theoretic interpretation to a particular class of constraint-satisfaction problems with uncertainty, viz. branching constraint satisfaction problems. We have done this using decision networks as a representation formalism to which decision-theoretic, probabilistic and logical constraints were mapped, giving rise to a uniform representation. The biggest advantage of this approach is that it allows us to study the interactions between the various components of a BCSP more clearly. In addition, the insight gained this way acted as a suitable foundation for the design of extensions to the original BCSP formalism.

Acknowledgements

This work was carried out while Ken Brown was at the University of Aberdeen.

References

[1] R.E. Bellman. Dynamic Programming. Princeton University Press, Princeton, 1957.

[2] R.W. Conway, W.L. Maxwell and L.W. Miller. Theory of Scheduling. Addison-Wesley, Reading, Massachusetts, 1967.

[3] G.F. Cooper. A method for using belief networks as influence diagrams. In: Proceedings of the 4th Workshop on Uncertainty in Artificial Intelligence 1988: 55–63.

[4] G.F. Cooper. The computational complexity of probabilistic inference using Bayesian belief networks. Artificial Intelligence 1990; 42(2-3): 393–348.

[5] P. Dagum, A. Galper and E. Horvitz. Dynamic network models for forecasting. In: Proceedings of UAI92, 1992, pp. 41–48.

[6] D.W. Fowler and K.N. Brown. Branching constraint satisfaction problems for solutions robust under likely changes. Proceedings CP2000, Springer Verlag, Berlin, 2000, pp. 500–504.

[7] D. W. Fowler. Branching Constraint Satisfaction Problems. PhD Thesis, Department of Computing Science, University of Aberdeen, 2002.

[8] D.W. Fowler and K.N. Brown. Branching constraint satisfaction problems and Markov decision problems compared. Annals of Operations Research 2003; 118: 85–100.

[9] E. Ignall and L. Schrage. Applications of the branch and bound technique to some flowshop scheduling problems. Operations Research 1965; 13(3): 400–412.

[10] F.V. Jensen. Bayesian Networks and Decision Graphs. Springer, New York, 2001.

[11] D.E. Knuth. The Art of Computer Programming, Vol. 1: Fundamental Algorithms, 3rd Ed. Addison-Wesley, Reading, MA, 1997.

[12] D. Larkin and R. Dechter. Bayesian inference in the presence of determinism. In: C.M. Bishop and B.J. Frey (eds), Proceedings of the 9th International Workshop on Artificial Intelligence and Statistics, Jan 3-6, 2003, Key West, FL.

[13] S.L. Lauritzen, D.J. Spiegelhalter. Local computations with probabilities on graphical structures and their application to expert systems. Journal of the Royal Statistical Society (Series B) 1987; 50: 157–224.

[14] P.J.F. Lucas. Bayesian model-based diagnosis. International Journal of Approximate Reasoning 2001; 27: 99–119.

[15] J. Pearl. Probabilistic Reasoning in Intelligent Systems. Morgan Kaufman, San Mateo, California, 1988.

[16] R.D. Shachter. Evaluating influence diagrams. Operation Research 1986; 34(6): 871–882.

[17] E. Tsang. Foundations of Constraint Satisfaction. Academic Press, London, 1993.

[18] R.J. Wilson. Introduction to Graph Theory. Longman, Burnt Mill, 1979.