Information-Driven Search Strategies in the Board Game of CLUE®

Silvia Ferrari, *Senior Member, IEEE*, and Chenghui Cai, *Member, IEEE*

***Abstract*—This paper presents an information-driven sensor management problem, referred to as treasure hunt, which is relevant to mobile-sensor applications such as mine hunting, monitoring, and surveillance. The objective is to infer a hidden variable or treasure by selecting a sequence of measurements associated with multiple fixed targets distributed in the sensor workspace. The workspace is represented by a connectivity graph, where each node represents a possible sensor deployment, and the arcs represent possible sensor movements. An additive conditional entropy reduction function is presented to efficiently compute the expected benefit of a measurement sequence over time. Then, the optimal treasure hunt strategy is determined by a novel label-correcting algorithm operating on the connectivity graph. The methodology is illustrated through the board game of CLUE®, which is shown to be a benchmark example of the treasure hunt problem. The game results show that a computer player implementing the strategies developed in this paper outperforms players implementing Bayesian networks, *Q*-learning, or constraint satisfaction, as well as human players.**

***Index Terms*—Bayesian networks (BNs), computer game playing, influence diagrams (IDs), label-correcting algorithms, mine hunting, path planning, search theory, sensor planning, value of information.**

I. Introduction

In THIS paper, the basic treasure hunt problem and algorithms are developed and illustrated through the board game of CLUE®. The same algorithms are applied in [1] to manage a robotic sensor in a mine-hunting application. As was recently pointed out in [2], games are ideal benchmarks for testing computational intelligence theories and algorithms because they provide challenging dynamic environments with rules and objectives that are easily understood. CLUE® constitutes an excellent benchmark for the treasure hunt problem because the information obtained during the game depends on the position of the pawn. The objective is to infer hidden cards from suggestions made upon entering the rooms of the CLUE® mansion. However, in order to make a suggestion involving a particular room card, the pawn must occupy the corresponding room in the mansion. Therefore, a game strategy must plan the suggestions' sequence and enabling pawn motions based on the evidence that becomes available over time. By viewing the suggestions as measurements and the rooms as targets, an approach is developed for computing optimal strategies from an influence diagram (ID) representation of the game.

As shown in [1] and [3], the treasure hunt is a basic information-driven sensor management problem that is relevant to several mobile-sensor applications, such as robotic mine hunting [4], cleaning [5], and monitoring of urban environments [3], manufacturing plants [6], and endangered species [7]. Recently, several authors have demonstrated that information-driven sensor management is a general and effective framework for planning a measurement sequence based on a probability mass function (PMF) that is referred to as sensor model [8]–[13]. Relative entropy was used in [8] to solve a multisensor–multitarget assignment problem and also in [9] and [13] to manage agile sensors with Gaussian models for target detection and classification. Entropy and the Mahalanobis distance measure were used in [10] for sensor selection in *ad hoc* sensor networks. The theory of optimal experiments was implemented in [11] to develop optimal adaptive-search strategies for the detection and classification of buried targets. In [12], the Rényi information divergence applied to the prior and posterior joint multitarget probability density functions was used to manage the trajectories of multiple aerial vehicles.

The objective of information-driven sensor management is to decrease the uncertainty or entropy of one or more hypothesis variables that are not observable or *hidden*. A basic difficulty associated with the use of entropy-based functions is that they are typically nonadditive and myopic, in that they do not consider the effects of prior measurements on those that are performed subsequently [8]–[10], [13]–[15]. In this paper, the effect of prior measurements is taken into account by defining the measurement benefit as the expected entropy reduction (EER) conditioned on prior measurements, which is shown to be additive over time in Section IV.

The sensor workspace is represented by a novel connectivity graph with a subset of nodes that are referred to as observation cells. Each observation cell represents a deployment or configuration that enables one sensor measurement. The sensor can move between adjacent nodes by paying a penalty or cost that is attached to the arc between them. The treasure hunt can be viewed as an extension of the satisficing-search problem of digging buried treasure chests [16]–[18] in which the connectivity graph specifies the ordering constraints to be satisfied by the search. As in [16]–[18], the measurements' outcomes are unknown *a priori* and cannot be factored into the

Manuscript received February 18, 2008; revised June 16, 2008, August 28, 2008, and August 29, 2008. This work was supported by the National Science Foundation under CAREER Award ECS 0448906. This paper was recommended by Associate Editor R. Lynch.

S. Ferrari is with the Department of Mechanical Engineering and Materials Science, Duke University, Durham, NC 27708 USA (e-mail: sferrari@duke.edu).

C. Cai is with the Department of Electrical and Computer Engineering, Duke University, Durham, NC 27708 USA (e-mail: cc88@duke.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TSMCB.2008.2007629

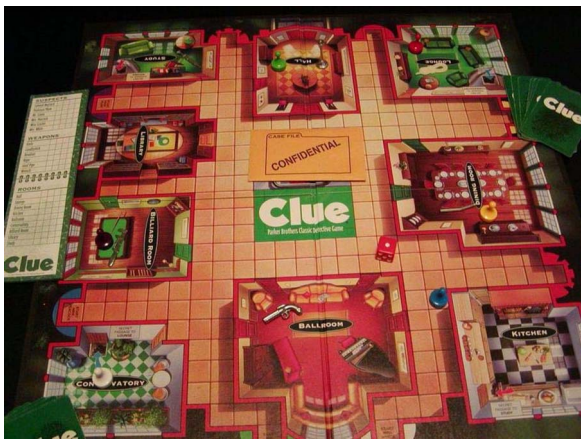

Fig. 1. CLUE® mansion and game pieces. CLUE® & 2006 Hasbro, Inc. Used with permission.

decision problem. However, unlike satisficing searches, in the treasure hunt, measurements are not necessarily binary and are used to infer a hidden hypothesis variable. Existing dynamic programming algorithms based on label-correcting [19] and Viterbi searches [20] are not directly applicable to the connectivity graph. Thus, a novel label-correcting pruning algorithm is presented in Section VI-A to reduce the number of feasible solutions based on distance while retaining the strategy with maximum measurement profit. This label-correcting algorithm generates a pruned connectivity tree that is folded into an ID to obtain a compact representation of the treasure hunt problem (Section VI-B).

The board game of CLUE®, described in Section II, is used to illustrate the treasure hunt problem formulated in Section III. The EER measurement-benefit function and its properties are presented in Section IV. The ID representation of the treasure hunt problem is obtained in Section VI and is demonstrated through the board game of CLUE® in Section VII. The game results presented in Section VIII show that a computer player implementing the strategies obtained from the ID outperforms computer players implementing Bayesian networks (BNs) [21], Q-learning [22], or constraint satisfaction (CS), as well as human players.

II. Board Game of CLUE®

The board game of CLUE® is chosen as a benchmark example because its rules and objectives are easily understood, and it exhibits all the basic challenges of the treasure hunt problem. The game's objective is to determine the guilty suspect, weapon, and room of an imaginary murder. However, in order to gather evidence about the room of the murder, the player's pawn must be inside the room. Hence, the type of evidence that may be gathered by the player depends on the position of the pawn in the CLUE® mansion. The mansion has nine rooms and is shown on the game board in Fig. 1. The six suspects are represented by the pawns and may use any one of six toy weapons. Each item in the game is also represented by an illustrated card in the deck, for a total of 21 cards.

At the onset of the game, one card from each item category (room, suspect, and weapon) is randomly selected, removed from the deck, and hidden in an envelope for the remainder of the game. The remaining cards are dealt to the players who subsequently eliminate their own cards from their list of possible hidden items. The players move their pawns about the mansion by rolling the die, and when they enter a particular room, they can make a suggestion that must include that room, and any item from the suspect and weapon categories. The following are the three ways for a pawn to enter a room: 1) a door illustrated on the game board (Fig. 1); 2) a secret passage connecting opposite corners (Fig. 1); or 3) being suggested as the murder suspect by one of the other players. After a player makes a suggestion involving the potential room, suspect, and murder weapon, the next player must refute it by showing one of these cards from his/her own deck, if possible. Otherwise, the player must state that he/she has none of the suggested cards, and other players must attempt to refute the suggestion. When none of the players is able to refute it, it can be inferred that the suggested cards are the hidden ones or that they belong to the player who made the suggestion. By entering rooms and making suggestions, players gather evidence about the adversaries' cards and infer the values of the hidden cards.

The problem of making decisions on the pawn movements and suggestions that optimize the value of the information gathered during the game can be viewed as an example of the treasure hunt problem formulated in the next section.

III. Treasure Hunt Problem

The objective of the treasure hunt problem is to infer a hidden variable or treasure y, referred to as *hypothesis* variable, from the outcomes of multiple measurements. In this paper, y is a discrete random variable with a finite range $\mathcal{Y} = \{y^1, \ldots, y^p\}$, where y^ℓ represents the ℓth possible value of y. A *measurement* m_i is a discrete random variable that represents a clue or feature that is observable from a corresponding target. Every measurement variable m_i has a finite range $\mathcal{M}_i = \{m_i^1, \ldots, m_i^N\}$, where m_i^ℓ denotes the ℓth possible value of m_i. Although y is hidden, it can be inferred from one or more measurements in an available set $M = \{m_1, \ldots, m_r\}$ by means of a known joint PMF $P(y, M) = P(y, m_1, \ldots, m_r)$ that admits a BN factorization (Section IV).

The set M represents all possible measurements that can be obtained from r fixed targets located in the sensor workspace. The sensor workspace is represented by a finite set of discrete cells, $\mathcal{K} = \{\kappa_1, \ldots, \kappa_q\}$, where each cell represents a possible sensor deployment or configuration. The sensor may visit only one cell at a time and obtain at most one measurement $z(t_k) = m_i \in M$ in each cell. An *observation cell*, denoted by $(\bar{\cdot})$, is defined as a deployment that enables one sensor measurement, while a deployment in a *void cell* does not enable any measurements. The set of all cells that enable a measurement m_i is denoted by $\bar{\mathcal{K}}_i$. After visiting a cell κ_i, the sensor can move to an adjacent cell κ_j by incurring a cost $d_{ij} = d_{ji}$, referred to as *distance*. In this paper, it is assumed that the distance and adjacency relationships between cells are provided by a graph:

Definition 3.1 (Connectivity Graph With Observations): A connectivity graph with observations, $\mathcal{G}$, is a nondirected graph where each node represents either an observation cell or a void

cell, and two nodes, κ_i and κ_j, are connected by an arc (κ_i, κ_j), with the distance d_{ij} attached, if and only if the corresponding cells are adjacent.

A systematic cell decomposition approach for computing $\mathcal{G}$ in robotic-sensor applications is provided in [1]. Also, an approach for obtaining the BN factorization of $P(y, M)$ for a variety of sensor types is presented in [23] and [24]. If measurements are required from all r targets, then the search for the minimum-cost complete-coverage path [25] can be reduced to a traveling salesman problem in $\mathcal{G}$ [26]. However, in many applications, complete coverage is infeasible, and only a subset of the targets can be visited due to energy and time considerations. Let the sensor movements between cells be indexed by a discrete time t_k. Then, if the sensor is in κ_i at time t_k, the cells that can be visited at a time t_{k+1} are all the cells adjacent to κ_i in $\mathcal{G}$. Thus, at every time t_k, the sensor makes an action decision, $a(t_k)$, on which cell to move to at time t_{k+1}, and a test decision, $u(t_k)$, on whether to make an available measurement when in an observation cell (see Section V for a review of test and action decisions). The sensor reward at t_k is the measurement profit defined as the measurement benefit, B, minus the cost of the measurements, J, and of the sensor movement or distance, D, i.e.,

$$R(t_k) = w_B \cdot B(t_k) - w_J \cdot J(t_k) - w_D \cdot D(t_k) \tag{1}$$

where w_B, w_J, and w_D are user-defined weights that are chosen based on the units and relative importance of the respective objective functions. B is shown to be an additive function of $P(y, M)$ in Section IV, and J and D are additive by definition.

Then, an optimal sensor decision strategy can be obtained by solving the following problem.

Problem 3.2 (Treasure Hunt Problem): Given a connectivity graph with observations, $\mathcal{G}$, and a joint PMF, $P(y, m_1, \ldots, m_r)$, of a hypothesis variable y and r measurements, $m_1, \ldots, m_r$, find the strategy $\sigma^* = \{u(t_k), a(t_k) | \, k = 0, \ldots, f\}$ that maximizes the total measurement profit

$$V = \sum_{k=0}^{f} R(t_k) \tag{2}$$

for a sensor that moves from a cell κ_0 to a cell κ_f in $\mathcal{G}$.

If the targets could be observed in any order, then the optimal strategy would be to select the measurements in decreasing order of their benefit-to-cost ratios [17]. However, in the treasure hunt, measurements are obtained over time by visiting observation cells according to the adjacency relationships in $\mathcal{G}$, which constitute a new class of ordering constraints [17]. Thus, the treasure hunt problem can be viewed as an extension of the satisficing-search problem of digging treasure chests buried at multiple sites, which must be searched in a specified sequential order [17]. As in [17], the measurements' outcomes are unknown *a priori* and cannot be therefore factored into the decision problem. Decisions on which cells to visit are made based on the expected benefit of the measurements. However, unlike the satisficing-search problem, the treasure hunt is not limited to binary measurement variables or to AND–OR-tree representations of the problem. By introducing a hidden hypothesis variable and a nonmyopic additive measurement-benefit function (Section IV), the treasure hunt problem can be extended to mobile-sensor applications, as shown in [1].

The board game of CLUE® constitutes an excellent benchmark for the treasure hunt problem because the pawn's position determines the allowable suggestion (or measurement) about the hidden room card, which is viewed as the hidden hypothesis variable. Decisions on the pawn's movements must be made before the suggestion's outcome becomes available. Therefore, the game strategy σ^* must optimize a tradeoff between the expected benefit of visiting a room and the cost, which consists of the distance traveled and of the turn used to make a suggestion. As shown in the next section, the measurement benefit B can be formulated as an additive function of the joint PMF $P(y, M)$ that estimates the reduction in uncertainty on y, conditioned on the evidence that may be obtained from one or more of the available measurements.

IV. Measurement-Benefit Function

A basic difficulty in the treasure hunt and other sensor management problems consists of assessing the benefit of performing one or more measurements $m_i, \ldots, m_j \in M$ prior to obtaining them from the targets. The objective is to decrease the uncertainty of the hypothesis variable y, which can be represented by entropy. However, entropy-based functions are typically nonadditive and myopic, in that they do not consider the effect of prior measurements on those that are performed subsequently [8]–[10], [13]–[15]. The effect of a prior measurement, z, can be taken into account by considering the conditional entropy function defined in terms of the variable's conditional PMF, $P(y|z)$, i.e.,

$$\begin{aligned} H(y|z) &= -\sum_{z^j \in \mathcal{Z}} \sum_{y^\ell \in \mathcal{Y}} P(y = y^\ell, z = z^j) \log_2 P(y = y^\ell | z = z^j) \\ &= -\mathbb{E}_{y,z} \{\log_2 P(y|z)\} \end{aligned} \tag{3}$$

where, y^ℓ and z^j denote values in the variables' ranges, $\mathcal{Y}$ and $\mathcal{Z}$, respectively, and $\mathbb{E}$ denotes expectation with respect to its subscript [27]. However, conditional entropy is not additive because $H(y|z_i, z_j) \neq H(y|z_i) + H(y|z_j)$. Consider instead the *conditional mutual information* of three discrete and random variables y, z_1, and z_2, defined as

$$I(y; z_2|z_1) = H(y|z_1) - H(y|z_1, z_2) \tag{4}$$

$$= \mathbb{E}_{y,z_1,z_2} \left\{ \log_2 \frac{P(y, z_2|z_1)}{P(y|z_1)P(z_2|z_1)} \right\}. \tag{5}$$

For simplicity, the shorthand notation $z_k = z(t_k)$ is adopted in the remainder of this section. The following result can be used to define a nonmyopic and additive measurement-benefit function.

Theorem 4.1: Let $Z_{t_f} = \{z_1, \ldots, z_f\}$ be a sequence of measurements about a hypothesis variable y that is performed through a discrete-time Markov process defined over a set of decision epochs $\{t_1, \ldots, t_f\}$. Then, the entropy reduction

$$\begin{aligned} \Delta H(t_k) &\equiv H(t_{k-1}) - H(t_k) \\ &= H(y|z_{k-1}, \ldots, z_1) - H(y|z_k, \ldots, z_1) \\ &= I(y; z_k|z_{k-1}, \ldots, z_1) = I(y; z_k|Z_{t_{k-1}}) \end{aligned} \tag{6}$$

is a conditional mutual information, and represents the reduction in uncertainty in y that is incurred at time t_k, when an additional measurement z_k is performed. The entropy reduction is a reward function that is additive over time. Hence, the total measurement benefit

$$B_{\text{tot}} = \sum_{k=1}^{f} \Delta H(t_k) = I(y; z_1, \ldots, z_f) \tag{7}$$

is the reduction in uncertainty brought about by all measurements in Z_{t_f}. See Appendix I for the proof.

The following result is useful in obtaining a decision policy that selects an optimal measurement sequence $Z_{t_f} = \{z_1, \ldots, z_f\}$ from a set of r possible measurements, with $f < r$.

Remark 4.2: The measurement benefit at any time $t_k \in (t_0, t_f]$

$$B(t_k) = \sum_{i=1}^{k} \Delta H(t_i) \tag{8}$$

obtained from a sequence of measurements whose individual outcomes are independent of time, is independent of the order in which the measurements are performed.

A proof is provided in Appendix II.

As shown in Section VI-A, using the aforementioned results, the connectivity graph can be pruned based solely on distance, without eliminating strategies that optimize the total measurement profit (2). As shown in (8), at any time t_k, the measurement-benefit function must account for the entropy reduction of all combinations of measurement outcomes that may be obtained prior to t_k. The results in the following section are derived to demonstrate that for certain forms of the joint PMF, $P(y, M)$, the entropy reduction can be obtained efficiently using a computation that is recursive with respect to time.

A. Efficient Computation of EER Over Time

BNs are a convenient paradigm for representing multivariate joint PMFs obtained from data or expert domain knowledge, such as sensor models [23], [24], [28]. Every random variable in the BN universe $X = \{x_1, \ldots, x_n\}$ is assumed to have a finite range and is represented by a node in the graph. Arcs between the nodes represent conditional probability relationships between the variables and determine the form of the recursive factorization

$$P(X) \equiv P(x_1, \ldots, x_n) = \prod_{x_l \in X} P\left(x_l | \pi(x_l)\right) \tag{9}$$

where $\pi(x_l)$ denotes the set of parents of node x_l in X. Factors in (9) are conditional PMFs given by the BN conditional probability tables (CPTs). Assume that the joint PMF in the treasure hunt problem, $P(y, M)$, admits the BN factorization in (9). From (6), the measurement benefit of an admissible measurement $z_k = m_j \in M$ at time t_k is

$$I(y; z_k = m_j | Z_{t_{k-1}}) = \mathbb{E}_{y, m_j, z_{k-1}, \ldots, z_1} \left\{ \log_2 \frac{P(y, z_k = m_j | Z_{t_{k-1}})}{P(y | Z_{t_{k-1}}) P(z_k = m_j | Z_{t_{k-1}})} \right\} \tag{10}$$

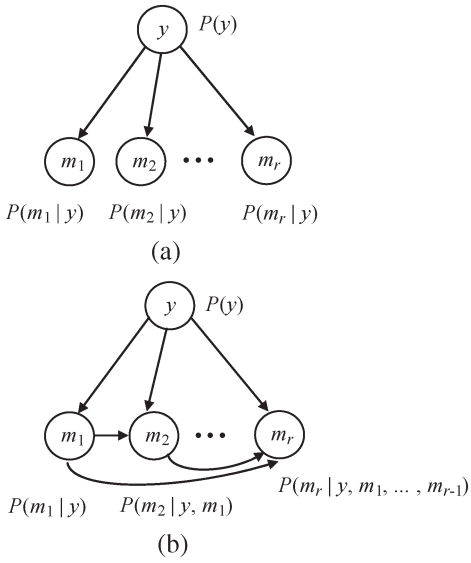

Fig. 2. Examples of (a) divergent and (b) feedforward BN factorizations of the joint PMF $P(y, M)$.

and must be computed for every admissible sequence of measurements $Z_{t_{k-1}} \equiv \{z_1, z_2, \ldots, z_{k-1}\}$, obtained before t_k. Then, the joint PMF $P(y, z_k = m_j, Z_{t_{k-1}})$ is obtained by marginalizing the remaining measurements out of the joint PMF $P(y, M)$, and the posterior PMFs in (10) are obtained from $P(y, z_k = m_j, Z_{t_{k-1}})$ by a straightforward application of the factorization in (9) and Bayes' rule. If the joint PMF $P(y, M)$ admits a divergent or a feedforward factorization, as shown in Fig. 2, the measurement-benefit function (8) can be computed recursively over time, as shown in the remainder of this section.

Suppose that the joint PMF $P(y, M)$ can be represented by the divergent BN structure in Fig. 2(a), which represents the factorization

$$P(y, M) = P(y) \prod_{m_i \in M} P(m_i | y) \tag{11}$$

where all factors are known from the BN CPTs [Fig. 2(a)]. At time t_0, the entropy of the hypothesis variable is $H(t_0) = H(y)$ and can be obtained from the CPT $P(y)$ in Fig. 2(a). If a sequence Z_{t_k} contains the subset of measurements $M_k \subset M$, and none of these measurements is ever repeated over the time interval $(t_0, t_f]$, then $H(t_k) = H(y | Z_{t_k})$ can be obtained from (3) using the posterior PMF

$$P(y | Z_{t_k}) = \frac{P(y, Z_{t_k})}{P(Z_{t_k})} = \frac{\sum_{m_i^\ell \in \mathcal{M}_i, m_i \notin M_k} P(y, m_1, \ldots, m_i = m_i^\ell, \ldots, m_r)}{P(Z_{t_k})}.$$

For an admissible measurement $z_k = m_j \in M$ at time t_k, the aforesaid posterior PMF can be written as (12), shown at the bottom of the next page, and the joint PMF is given by

$$P(Z_{t_k}) = \sum_{y^\ell \in \mathcal{Y}} P\left(y = y^\ell, Z_{t_k}\right) = \sum_{y^\ell \in \mathcal{Y}} P\left(z_k = m_j | y = y^\ell\right) P\left(y = y^\ell, Z_{t_{k-1}}\right) \tag{13}$$

where $P(z_k = m_j|y)$ is available from the BN CPT associated with node m_j. The aforementioned PMFs are both formulated with respect to $P(y, Z_{t_{k-1}})$, which is available from the previous time step t_{k-1}. This recursive entropy computation exploits the BN factorization to gain computational efficiency compared to a brute-force marginalization of $P(y, M)$. A similar approach is also used in the arc-reversal method and in junction-tree propagation algorithms [29].

Another case in which the entropy reduction can be computed recursively is that of a BN structure with feedforward connections among the measurement nodes, as shown in Fig. 2(b). The measurement nodes are ordered such that if $i < j$, then there is an arc from m_i to m_j, and the joint PMF exhibits the following factorization:

$$P(y, M^o) = P(y) \prod_{j=1}^{r} P(m_j|y, m_{j-1}, \ldots, m_1) \tag{14}$$

where the measurement set is now totally ordered $M^0 \equiv \{m_1, \ldots, m_r\}$. This type of BN structure may be useful in representing the probabilistic relationships between measurements with the same range that are interchangeable and can thus be ordered according to the time at which they are performed, such that $M_k^o = \{m_1, \ldots, m_k\} = Z_{t_k}$ and $M_k^o \subset M^o$. The entropy reduction is still computed from (6), but the posterior PMF needed to compute $H(t_k)$ is now obtained by applying the probability law $P(y|z_1, z_2) = P(z_2|y, z_1)P(y|z_1)/P(z_2|z_1)$ as follows:

$$\begin{aligned} P(y|Z_{t_k}) &= P(y|z_k = m_k, Z_{t_{k-1}}) \\ &= \frac{P(z_k = m_k|y, Z_{t_{k-1}})P(y|Z_{t_{k-1}})}{P(z_k = m_k|Z_{t_{k-1}})}. \end{aligned} \tag{15}$$

Then, the new BN factorization (14) is exploited to compute the posterior PMF at the denominator recursively

$$\begin{aligned} &P(z_k = m_k|Z_{t_{k-1}}) \\ &\quad = \sum_{y^\ell \in \mathcal{Y}} P\left(y = y^\ell, z_k = m_k|Z_{t_{k-1}}\right) \\ &\quad = \sum_{y^\ell \in \mathcal{Y}} P\left(z_k = m_k|y = y^\ell, Z_{t_{k-1}}\right) P\left(y = y^\ell|Z_{t_{k-1}}\right). \end{aligned} \tag{16}$$

The joint PMF needed for computing the expectation in (10) is also computed recursively as follows:

$$\begin{aligned} P(y, Z_{t_k}) &= P(y, z_k = m_k, Z_{t_{k-1}}) \\ &= \sum_{m_i^\ell \in \mathcal{M}_i, m_i \notin M_k^o} P\left(y, m_1, \ldots, m_i = m_i^\ell, \ldots, m_r\right) \\ &= \sum_{m_i^\ell \in \mathcal{M}_i, m_i \notin M_k^o} P(y) \\ &\quad \times \prod_{j=1}^{r} P\left(m_j|y, m_{j-1}, \ldots, m_i = m_i^\ell, \ldots, m_1\right) \\ &= P(y) \prod_{i=1}^{k} P(m_i|y, m_{i-1}, \ldots, m_1) \\ &\quad \times \sum_{m_j^\ell \in \mathcal{M}_j} \prod_{j=k+1}^{r} P\left(m_j = m_j^\ell|y, m_{j-1}, \ldots, m_1\right) \\ &= P(y) \prod_{i=1}^{k} P(m_i|y, m_{i-1}, \ldots, m_1) \\ &= P(z_k = m_k|y, Z_{t_{k-1}})P(y, Z_{t_{k-1}}). \end{aligned} \tag{17}$$

Equations (12) and (15) are used to efficiently compute the nonmyopic additive measurement-benefit function in (8) from the PMFs obtained at the previous time step, $P(y|Z_{t_{k-1}})$ and $P(y, Z_{t_{k-1}})$, and from the available BN CPTs (Fig. 2). Also, by arc reversal [28], the joint PMF $P(y, Z_{t_k})$ obtained from (17) is the CPT of the hypothesis variable y in the ID representation of the treasure hunt problem derived in Section VI. The definitions of ID and related background material are reviewed in the next section.

V. Background on IDs

IDs can be used to represent discrete-time decision processes with limited information involving both action and test decisions and are reviewed comprehensively in [28]. Although other paradigms, such as decision trees [30] and independent choice logic [31], are also applicable, IDs are used in this paper because they provide a compact representation of the decision problem underlying the treasure hunt and illustrate graphically the domain structure [32]. The PMF P along with a chosen decision policy determines the evolution of the system's state, x. Let t_k index the time slices, and let $x(t_k)$ and $a(t_k)$ denote a chance and an action decision variable at time t_k, respectively. Then, the decision nodes in the ID have the temporal order $a(t_1) \prec a(t_2) \prec \cdots \prec a(t_f)$, and each chance node preceding

$$\begin{aligned} P(y|Z_{t_k}) &= P(y|z_k = m_j, Z_{t_{k-1}}) \\ &= \frac{P(y)\left[\prod_{m_i \in M_k} P(m_i|y)\right]\left[\sum_{m_i^\ell \in \mathcal{M}_i} \prod_{m_i \notin M_k} P\left(m_i = m_i^\ell|y\right)\right]}{P(Z_{t_k})} \\ &= \frac{P(y)\prod_{m_i \in M_k} P(m_i|y)}{P(Z_{t_k})} = \frac{P(z_k = m_j|y)P(y, Z_{t_{k-1}})}{P(Z_{t_k})} \end{aligned} \tag{12}$$

an action node, e.g., $x(t_{i-1}) \prec a(t_i)$, is known by the time that action is taken.

Definition 5.1: An *ID* is a tuple $\{U, A, \Omega, P, R\}$ representing a directed acyclic graph (DAG) over a set of nodes, U, with the following properties.

1) U is a set of nodes that can be partitioned into chance nodes U_C, decision nodes U_D, and utility nodes U_V. The members of U are also referred to as variables of the ID.
2) A is a set of arcs defined over U such that $\{U, A\}$ forms a DAG. If the DAG contains an arc $(x_i, x_j) \in A$, then the node x_i is a parent of x_j, and x_j is a child of x_i. Utility nodes have no children.
3) Ω is the range of the chance and decision nodes, where each node has a finite set of mutually exclusive states. The utility nodes have no states.
4) P is a conditional PMF defined over the chance nodes, given their parents. The CPT $P(x|\pi(x))$ is attached to every chance node x in the DAG, and $P(x = x^{\ell}|\pi(x) = w)$ is a positive number such that $P(\Omega(x)|\pi(x) = w) = 1\ \forall w$.
5) $R : \Omega(\pi(v)) \rightarrow \mathbb{R}$ is the utility or reward function defined over the parents of the utility node, v, and R is said to be attached to v.

In an ID, a measurement variable can be represented by a chance or test node, z. The decision on whether to perform an admissible measurement is referred to as a test decision and is denoted by u. While an action decision changes the state of the system, x, a test decision affects the evidence that may become available about the hidden variables [28, Ch. 4]. The type of evidence obtained for the hidden variables may, in turn, influence the action decisions, but it does not change the state of the system directly, only our knowledge of it. From the ID representation of a process, it is possible to compute an optimal strategy for the sequence of action and test decisions between an initial and a final time, t_0 and t_f, respectively.

Definition 5.2: A *strategy* is a class of admissible policies that consists of a sequence of functions

$$\sigma = \{c_0, c_1, \ldots, c_f\}$$

where c_k maps the state and test variables $\{x(t_k), z(t_k)\}$ into the admissible action and test decisions

$$\{a(t_k), u(t_k)\} = c_k\left[x(t_0), a(t_1), u(t_1), x(t_1), \ldots, a(t_{k-1}), u(t_{k-1}), x(t_{k-1})\right] \quad (18)$$

such that $c_k[\cdot] \in \Omega(U_D(t_k))$, for all $x(t_k)$ and $z(t_k)$.

Based on the results in Section IV, the treasure hunt reward R can be represented as a measurement-profit function (1) that is additive over time. Thus, given an ID representation of the treasure hunt problem, an optimal strategy σ^* that maximizes the total reward (2) over the time interval $(t_0, t_f]$ can be obtained using variable-elimination and junction-tree algorithms reviewed in [28, Ch. 7] and [33]. One difficulty associated with the use of IDs is determining the structure and CPTs, which are not easily learned from data due to the time-varying nature of the problems they describe. Another difficulty is obtaining tractable ID structures that capture the problem domain by a minimal number of nodes and arcs, without giving rise to complexity problems [28]. A systematic and effective approach for obtaining an ID representation of the treasure hunt problem is presented in the next section.

VI. ID Representation of the Treasure Hunt Problem

The treasure hunt problem addresses the coupled processes of sensor motion and inference that arise when planning the sensing strategy of a mobile sensor. The treasure hunt reward, defined in (1), includes the measurement benefit (8) to be optimized with respect to the measurement sequence Z_{t_k}. However, since each measurement is only enabled by a subset of observation cells, the sequence Z_{t_k} depends on the sensor's motion in the connectivity graph $\mathcal{G}$, which represents the search constraints. In the next sections, an approach is developed for obtaining an ID representation of $\mathcal{G}$ that includes a pruned subset of admissible decisions. Then, the ID is augmented to include y and the posterior PMF $P(y|Z_{t_k})$ derived in Theorem 4.1, in order to model the inference process, and compute the measurement benefit over time. In the next section, a novel label-correcting algorithm is presented to transform $\mathcal{G}$ into a pruned connectivity tree that contains a subset of admissible paths, including the one associated with the optimal strategy. It is shown that by exploiting the definitions of void and observation cells, pruning can be conducted based solely on distance while still guaranteeing optimality with respect to the total reward (2).

A. Pruned Connectivity Tree

A tree representation of all admissible sensor paths from κ_0 to κ_f in $\mathcal{G}$ can be obtained by an exhaustive search approach, such as breadth-first search [14, pp. 73–74], which takes into account the adjacency relationships in $\mathcal{G}$. By this approach, the number of cells that can be visited at a time t_k grows exponentially with k. Label-correcting algorithms employ the principle of optimality [34] to reduce the amount of computation required to search for the shortest path in a graph or network [19]. Since the measurement profit (1) depends not only on distance but also on the measurement sequence, existing graph-searching algorithms are not applicable to the treasure hunt problem. The novel pruning algorithm presented in this section takes into account the objective of minimizing distance in order to eliminate a significant number of suboptimal paths at every time step t_k based on the principle of optimality [34]. For convenience, the following terminology is introduced.

Definition 6.1 (Connectivity Tree): The connectivity tree T_r associated with a connectivity graph with observations, $\mathcal{G}$, and two nodes $\kappa_0, \kappa_f \in \mathcal{G}$, is a tree graph with κ_0 as the root and κ_f as the leaf. The tree nodes represent void or observation cells, and the additive distance metric d_{ij} is attached to the arc (κ_i, κ_j). A *tree branch* from the root to a leaf represents a sequence of cells or *channel* joining κ_0 to κ_f.Then, the pruning algorithm shown in Appendix III transforms $\mathcal{G}$ into a *pruned* connectivity tree with the following properties.

1) Two branches are said to be *information equivalent* if they connect the same cells κ_i and κ_l and contain the same set of observation cells, regardless of the order.

2) Two branches that are information equivalent can coexist in T_r if and only if they represent the same channel.
3) A branch in T_r connecting any two cells κ_i and κ_l has the shortest total distance of any other information-equivalent branch in $\mathcal{G}$.

Assume that the nodes in $\mathcal{G}$ are provided as a list of integers $L = \{0, \pm 1, \ldots, \pm q\}$ that are ordered by their absolute value and represent the list of cells $\mathcal{K} = \{\kappa_0, \kappa_1, \ldots, \kappa_q\}$, such that an integer $n = \pm i$ in L represents the cell κ_i, and $n \geq 0$ if κ_i is a void cell, or $n < 0$ if κ_i is an observation cell. The arcs in $\mathcal{G}$ are represented by a $q \times q$ symmetric matrix $A = \{a_{ij}\}$, where $a_{ij} = 1$ if κ_i and κ_j are connected in $\mathcal{G}$, $a_{ij} = 0$ if they are disconnected, and $a_{ii} = 0$ to ensure tree growth. Another $q \times q$ symmetric matrix $D = \{d_{ij}\}$ contains the distance attached to every arc (κ_i, κ_j) in $\mathcal{G}$. The pruning algorithm in Appendix III takes L, A, D, κ_0, and κ_f as the inputs and returns a pruned connectivity tree T_r in the form of two 2-D arrays, TREE and DIST. The element in the jth row and tth column of TREE, denoted by TREE(j, t), is an integer $n = \pm i$, representing the cell κ_i that is encountered at time index t, along the jth branch in T_r. The array DIST contains the distance associated with each arc in TREE such that DIST$(j, t) = d_{ki}$ if TREE$(j, t) = \pm i$ and TREE$(j, t-1) = \pm k$.

The pruning algorithm compiles the arrays TREE and DIST incrementally, beginning at the root κ_0 and growing branches spatially (columnwise) and temporally (row-wise) one node at a time. Similar to other label-correcting algorithms [19], [35], pruning also takes place incrementally, eliminating branches that are information equivalent and suboptimal with respect to distance at every time index. For this purpose, the array DIST_{tot} is computed such that $\text{DIST}_{\text{tot}}(j, t)$ is the total distance between κ_0 and the cell in TREE(j, t). At every time index $t \in (t_0, t_f]$, the pruning algorithm updates the array VISITED, which contains a list of integers representing the nodes that have already been visited by the algorithm, and the array $\text{DIST}_{\text{short}}$, which contains the corresponding distance from κ_0. As shown in Appendix IV, by applying the principle of optimality, $\text{DIST}_{\text{short}}$ is guaranteed to contain the shortest distance of all information-equivalent branches connecting κ_0 to every cell in VISITED. Since T_r contains all branches that are not information equivalent, the same cells and channels can appear in multiple branches that represent alternate paths for obtaining subsets of measurements in M. However, as shown in Appendix IV, T_r only contains the shortest of all information-equivalent paths from κ_0 to κ_f in $\mathcal{G}$, including the shortest path from κ_0 to κ_f. Each of these paths enables a different subset of measurements by means of the minimum distance. By Remark 4.2, the measurement benefit does not depend on the order of the measurements, but only on the chosen subset. Thus, by retaining all information-equivalent paths in T_r, the measurement benefit can be excluded by the transformation $\mathcal{G} \Rightarrow T_r$, without eliminating any candidate solutions to the optimal strategy σ^*. A detailed proof of the properties of T_r is provided in Appendix IV.

A simple example of connectivity tree obtained from $\mathcal{G}$ in Fig. 3 is shown in Fig. 4. Another example, obtained from the CLUE® connectivity graph (Fig. 10), is shown in Appendix VI and discussed in Section VII-B. As shown in Tables I and II, pruning brings about considerable computational savings and significantly reduces the size of the connectivity tree. The size of the connectivity graph that can be transformed into a connectivity tree in approximately the same amount time is considerably larger for the pruning algorithm than for an exhaustive search (Table I). By pruning information-equivalent branches of longer distance, the branches of T_r display a higher density of observation cells and may thus lead to a higher measurement benefit in later steps. For example, T_r obtained by the pruning algorithm in Table I contains a 1:1 ratio of void to observation cells, while T_r obtained by exhaustive search (which does not consider cell labels) contains a 3:1 ratio of void to observation cells.

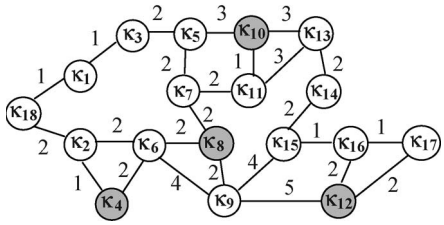

Fig. 3. Example of connectivity graph $\mathcal{G}$ with observation cells labeled in gray, and distances attached to the arcs.

As shown in Appendix III, the computation of T_r can be further simplified by means of a user-specified parameter d_M that represents the maximum allowable distance from the shortest path between κ_0 and κ_f. When $\mathcal{G}$ is very large, the user can use d_M to limit the size of T_r at the expense of eliminating candidate solutions to σ^*. A comparison of the computation times required by the pruning algorithm and by exhaustive search is shown in Table II for the same connectivity graph with 20 nodes, 45 arcs, and 12 observation cells, and different values of d_M, using a Pentium 4 computer with a 3.06-GHz processor. Our numerical simulations also show that the pruning algorithm requires only a few hours, or even minutes, to generate T_r from graphs with thousands of nodes, arcs, and observation cells, which cannot be reasonably handled by exhaustive search.

As reviewed in [28, p. 228], it is always possible to transform a tree into an ID and vice versa. After applying the pruning algorithm to $\mathcal{G}$, an ID representation of the pruned connectivity tree T_r can be obtained by means of the following assignments.

1) Let the chance node $x(t_k)$ denote the set of all cells that the sensor can visit at time t_k. For example, $\Omega(x(t_2)) = \{\kappa_1, \kappa_2\}$ in Fig. 4.
2) Let the action decision node $a(t_k)$ denote the set of admissible action decisions at time t_k, where μ_ℓ denotes the action decision to move to cell $\kappa_\ell \in \mathcal{K}$. For example, $\Omega(a(t_2)) = \{\mu_3, \mu_4, \mu_6\}$ in Fig. 4, and $\Omega(a(t_f)) = \{\mu_f\}$.
3) Let the reward function be the negative distance $R(t_k) = -D[x(t_k), a(t_k)] = -d_{i\ell}$ for $\forall \kappa_i \in \Omega(x(t_k))$ and $\forall \mu_\ell \in \Omega(a(t_k))$.

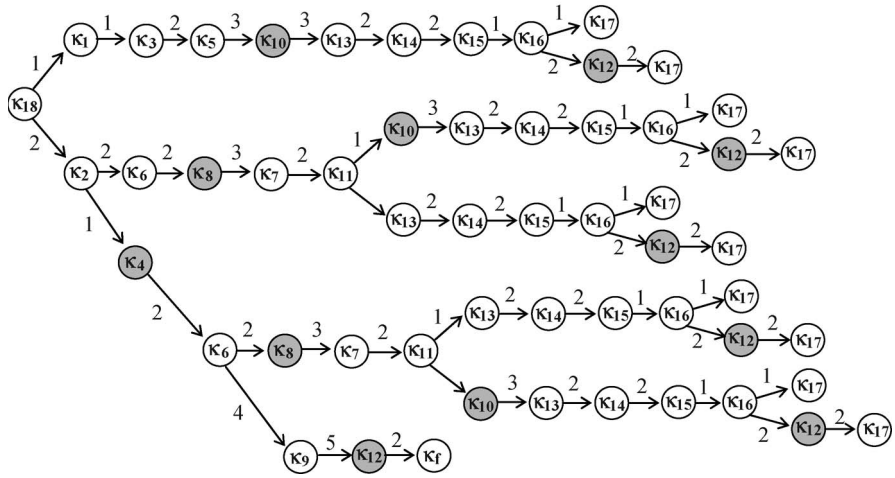

Fig. 4. Connectivity tree T_r obtained by the pruning algorithm from $\mathcal{G}$ in Fig. 3 for $\kappa_0 = \kappa_{18}$ and $\kappa_f = \kappa_{17}$ (observation cells are labeled in gray, and distances are attached to the arcs).

TABLE I
PRUNING ALGORITHM PERFORMANCE

Method	Nodes | Arcs in $\mathcal{G}$	Observation cells in $\mathcal{G}$	Branches in T_r	Computation time
Pruning Algorithm	531 | 686	270	48	79 s
Exhaustive Search	51 | 106	43	51	87 s

TABLE II
COMPUTATION TIME COMPARISON

Method	d_M	Branches in T_r	Time slices in T_r	Computation time
Pruning Algorithm	None	16	17	7 s
	10	8	15	3 s
	5	4	9	1 s
Exhaustive Search	None	34835	19	1570 s
	10	27452	19	1064 s
	5	13693	17	378 s

Then, the ID in Fig. 5 represents the connectivity tree T_r, provided that the CPT

$$P\left(x(t_{k+1}) = \kappa_j | x(t_k) = \kappa_i, a(t_k) = \mu_\ell\right) = \begin{cases} 1, & \text{if } (\kappa_i, \kappa_j) \in T_r \text{ and } j = \ell \\ 0, & \text{if } (\kappa_i, \kappa_j) \in T_r \text{ and } j \neq \ell \\ 0, & \text{if } (\kappa_i, \kappa_j) \notin T_r \quad \forall j, \ell \end{cases} \tag{19}$$

is attached to node $x(t_k)$, where $\kappa_i \in \Omega(x(t_k))$, $\kappa_j \in \Omega(x(t_{k+1}))$, and $\mu_\ell \in \Omega(a(t_k))$ if and only if $(\kappa_i, \kappa_\ell) \in T_r$ and $\kappa_\ell \in \Omega(x(t_{k+1}))$. The cells in the chance nodes' ranges determine the admissible measurements and test decisions at every time slice. The ID representation of the treasure hunt problem is derived in the next section, by combining the ID representation of the sensor motion constraints (Fig. 5) with that of the inference process, which includes test variables and decisions, and utility nodes representing the measurement profit.

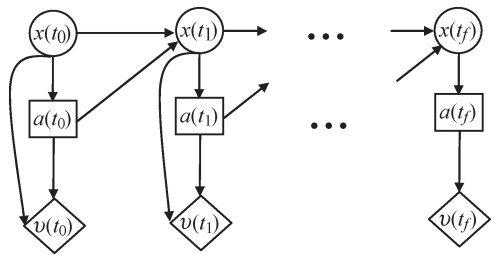

Fig. 5. ID representation of T_r.

B. ID With Test Variables and Decisions

In the treasure hunt problem, the measurement variables are distributed throughout the sensor workspace and are obtained through the Markov process in Fig. 5, representing the sensor motion. The sensor position $x(t_k)$ determines the subset of admissible measurements at time $t_k \in (t_0, t_f]$, and the outcome of a measurement $m_i \in M$ is known only after the sensor visits an observation cell $\bar{\kappa}_j \in \bar{\mathcal{K}}_i$. It also follows that the cells that can be visited by the sensor at t_k (based on Fig. 5) specify the range of the test variable z at t_k. As reviewed in [28, Ch. 7], a test variable in an ID can be represented by a chance node. Let the chance node $z(t_k)$ denote the set of all measurements that can be performed by the sensor at time t_k.

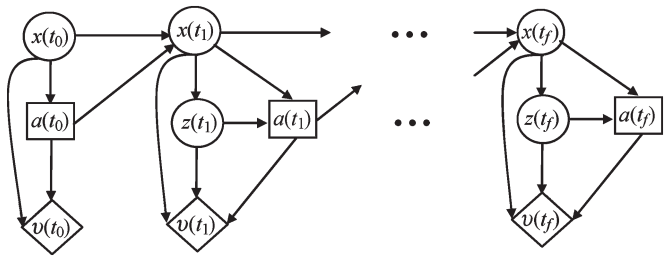

Fig. 6. ID representation of sensor motion and measurements with action decisions.

Then, its range is specified by the observation cells in the range of $x(t_k)$

$$m_i \in \Omega\left(z(t_k)\right), \quad \text{iff } \Omega\left(x(t_k)\right) \ni \bar{\kappa}_j \in \bar{\mathcal{K}}_i. \tag{20}$$

With the aforementioned assignment, the ID in Fig. 6 represents the sensor motion and measurement process, provided that the CPT

$$P\left(z(t_k) = m_i | x(t_k) = \bar{\kappa}_j\right) = \begin{cases} P(m_i), & \text{if } \bar{\kappa}_j \in \Omega\left(x(t_k)\right) \text{ and } \bar{\kappa}_j \in \bar{\mathcal{K}}_i \\ 0, & \text{otherwise} \end{cases} \tag{21}$$

is attached to node $z(t_k)$, and the utility node $v(t_k)$ represents the measurement benefit minus the distance traveled at time t_k.

Test decision nodes $u(t_k)$ represent the option of performing one or none of the available measurements. In addition to the distance traveled, the measurement process may involve a cost $J(t_k)$ representing energy or computational power consumption. Then, through the test decision u, a sensor in an observation cell may decide to perform the corresponding measurement only if its benefit exceeds its cost. Following an approach presented in [28, Ch. 7], u is included in the ID in Fig. 6 by including the value *unobserved*, denoted by m_{un}, in the range of the chance nodes z's and by using the following assignments.

1) Let the chance node $z'(t_k)$ denote the set of all measurements that can be performed by the sensor at time t_k, plus the value m_{un}: $\Omega(z'(t_k)) = \{\Omega(z(t_k)), m_{\text{un}}\}$.
2) Let the decision node $u(t_k)$ denote the set of admissible test decisions at time t_k, where ϑ_i denotes the test decision to make measurement $m_i \in \Omega(z'(t_k))$ at time t_k, and ϑ_{un} denotes the decision of not performing any measurements.

Then, the following CPT is attached to node $z'(t_k)$

$$P\left(z'(t_k) = m_i | x(t_k) = \bar{\kappa}_j, u(t_k) = \vartheta_l\right) = \begin{cases} P(m_i), & \text{if } \bar{\kappa}_j \in \bar{\mathcal{K}}_i \text{ and } i = l \\ 0, & \text{if } \bar{\kappa}_j \notin \bar{\mathcal{K}}_i, \quad i \neq l, \text{ or } l = un \end{cases}$$

$$P\left(z'(t_k) = m_{\text{un}} | x(t_k) = \bar{\kappa}_j, u(t_k) = \vartheta_l\right) = \begin{cases} 1, & \text{if } l = un \quad \forall j \\ 0, & \text{if } l \neq un \quad \forall j. \end{cases} \tag{22}$$

In Fig. 7 and in the remainder of this paper, the chance node $z'(t_k)$ is denoted simply by $z(t_k)$.

The reward function R attached to the utility node $v(t_k)$ in Fig. 7 is the measurement profit (1), which includes the measurement-benefit function (8). Since (8) is a function of the PMF $P(y, M)$, y must be included in the ID. First, the ID is augmented with a hidden node $y(t_k)$ that has the same range as y, and the parent set $\pi(y(t_k)) = \{z(t_1), \ldots, z(t_k)\}$, as shown in Fig. 7. Then, based on arc reversal [28], at every time slice t_k, the CPT attached to $y(t_k)$ is $P(y|Z_{t_k})$, and may be computed by one of the recursive formulas derived in Section IV-A [(12) or (15)], depending on the BN factorization of $P(y, M)$. Since $y(t_k)$ is parent only to the utility node, the ID in Fig. 7 does not have a complexity problem, provided that $x(t_k)$ is observable. If $x(t_k)$ is hidden, or if the set $\mathcal{M}_1 \cup \cdots \cup \mathcal{M}_r$ is very large, the solution of this ID may require information blocking or the use of a limited memory ID (LIMID) [33]. The ID representation (Fig. 7) and solution are demonstrated through the game of CLUE®, as explained in the next section. The game results obtained in Section VIII show that a computer player implementing the optimal strategies obtained from the ID solution outperform previous computer players designed, for example, via Q-learning and CS, as well as human players.

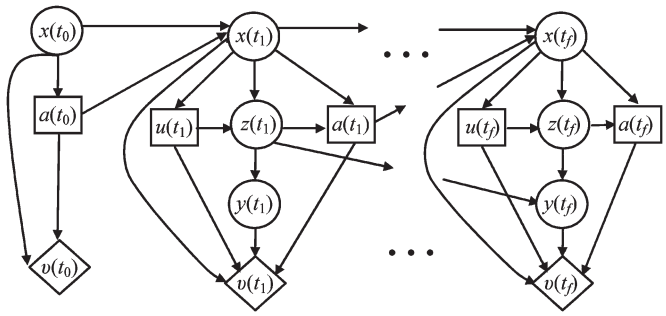

Fig. 7. ID representation of the treasure hunt problem.

VII. Implementation

Although there exist several computerized CLUE® games, so far, the mathematical development of game strategies has been limited to the works in [21] and [22]. In this section, the methodology presented in Section VI is implemented to obtain an ID representation of CLUE® and to compute optimal game strategies for moving the pawn and making suggestions based on the information that becomes available during the game. The suggestions are viewed as measurements that depend on the room occupied by the pawn, and the hidden cards are the hypothesis variables. The BN factorization of the PMF $P(y, M)$ is obtained in the next section, and the connectivity graph is derived in Section VII-B using an exact cell decomposition approach.

A. CLUE® BN

This section reviews the BN model of the CLUE® cards, which was first presented in [21] to automate inference of the hidden cards. Assume that there are three players in the game, namely, p_1, p_2, and p_3. After the three hidden cards are drawn, the remaining 18 cards are randomly distributed into groups of six cards per player. Every card in the deck is represented by one node in the CLUE® BN and has a

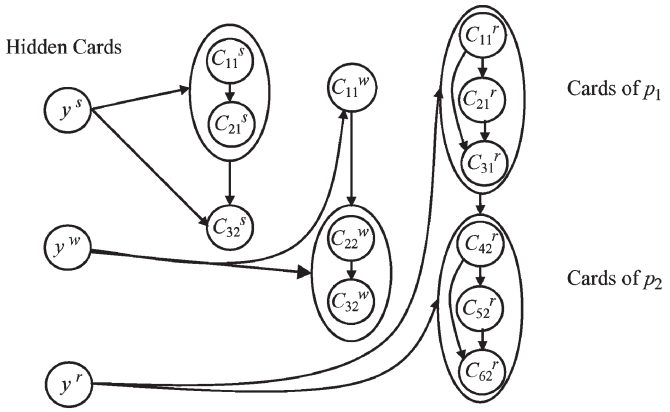

Fig. 8. Structure of approximate BN model for the CLUE® cards (taken from [21]).

range that depends on its category. A room card C^r can take one of nine values, $\Omega(C^r) = \{r_d, r_l, r_b, r_h, r_k, r_g, r_a, r_s, r_c\}$, representing the dining room (r_d), library (r_l), billiard room (r_b), hall (r_h), kitchen (r_k), lounge (r_g), ballroom (r_a), study (r_s), and the conservatory (r_c). A weapon card C^w can take one of six values, $\Omega(C^w) = \{w_k, w_r, w_c, w_p, w_v, w_h\}$, representing the knife (w_k), rope (w_r), candlestick (w_c), lead pipe (w_p), revolver (w_v), and the wrench (w_h). A suspect card C^s can take one of six values, $\Omega(C^s) = \{s_m, s_t, s_p, s_g, s_w, s_k\}$, representing Col. Mustard (s_m), Ms. Scarlett (s_t), Prof. Plum (s_p), Mr. Green (s_g), Mrs. White (s_w), and Mrs. Peacock (s_k). In [21], the exact BN model of the game cards was shown to have a complexity problem and CPTs of dimensions $\mathcal{O}(10^{25})$. A tractable CLUE® BN structure was obtained by introducing an assumption on the categories of the cards that are dealt to each player. The cards are dealt uniformly among the three players such that the following are observed: 1) p_1 has two suspect cards, one weapon card, and three room cards; 2) p_2 has one suspect card, two weapon cards, and three room cards; and 3) p_3 has two suspect cards, two weapon cards, and two room cards.

Let C^s_{ji}, C^w_{ji}, and C^r_{ji} denote the jth suspect, weapon, and room cards, respectively, which belong to the ith player, with $i = 1, 2$. The ID player is always chosen to be p_3, who has the disadvantage of having less room cards than p_1 and p_2. After the cards are dealt by the simulation, the range of y^r is determined as follows:

$$\Omega(y^r) = \Omega(C^r) \setminus \Omega\left(C^r_{l3}\right) = \{r_\iota : \iota = 1, \ldots, 7\}$$

where $i, l = 1, 2$, and $A \setminus B$ denotes the complement set of B in A. A similar operation is used to determine $\Omega(y^w)$ and $\Omega(y^s)$. Since cards of different categories are independent, when a card is hidden or dealt to a player, it only influences cards of the same category that are dealt subsequently. Thus, the hidden cards and the cards dealt to p_1 and p_2 can be represented by the CLUE® BN in Fig. 8, for which inference is feasible. The BN CPTs are obtained by noting that cards of the same category never acquire the same value, and cards of different categories are independent. Let a binary function υ be defined over a set of discrete variables $X = \{x_1, \ldots, x_n\}$ that all have the same finite range $\Omega(x) = \{x^1, \ldots, x^n\}$, with n mutually exclusive values, such that

$$\upsilon(X) = \prod_l \prod_{k \neq l} (1 - \delta_{ij}), \qquad \text{for} \quad x_l = x^i, x_k = x^j, \quad \forall (x_l, x_k) \in X, x^i, x^j \in \Omega(x). \tag{23}$$

Then, the CPTs of the CLUE® BN shown in Fig. 8 are given by

$$P\left(C^\ell_{ji} | \pi(C^\ell_{ji})\right) = \frac{\upsilon\left(C^\ell_{ji} \cup \pi\left(C^\ell_{ji}\right)\right)}{\left|\Omega\left(C^\ell_{ji}\right)\right| - \left|\pi\left(C^\ell_{ji}\right)\right|} \tag{24}$$

$$P\left(y^\ell = \ell_j\right) = \frac{1}{|\Omega(y^\ell)|} \quad \forall \ell_j \in \Omega(y^\ell). \tag{25}$$

The symbol $|\cdot|$ denotes the cardinality of a set, $\cup$ denotes the union of sets, and $\ell = s, w, r$ denotes the card category.

During the game, new evidence for the CLUE® BN may become available during every player's turn in which a suggestion is proven or refuted. Game evidence is organized into three independent tables, namely, E^s, E^w, and E^r, which are used to store hard and soft evidence about each card category. *Hard evidence* refers to perfect knowledge of a variable's value, whereas *soft evidence* refers to an observed PMF [28]. Jeffrey's rule is used to update soft evidence in a BN with a hidden node, given soft evidence about its children, $\text{ch}(\cdot)$ [36]. Jeffrey's rule can also be used for updating hard evidence by means of a PMF in which the observed variable's value has probability one, and all other values in the variable's range have probability zero. Let E^ℓ denote a $|\text{ch}(y^\ell)| \times |\Omega(y^\ell)|$ matrix of known probabilities for all children of y^ℓ, such that every row in E^ℓ contains the observed PMF of C^ℓ_{ji}, for $\forall C^\ell_{ji} \in \text{ch}(y^\ell)$. At the onset of the game, all cards have uniform probability over their range. Hence, all evidence tables are initialized according to the following equation:

$$E^\ell = \left\{e^\ell_{kl}\right\} = \left\{\frac{1}{|\Omega(y^\ell)|}\right\}, \quad \text{at } t_0 \tag{26}$$

where e^ℓ_{kl} represents the element in the kth row and lth column of E^ℓ. After t_0, the evidence tables are updated at every player's turn, indexed by t_i. Evidence is obtained through the players' suggestions according to the following rules.

1) By the unity law, all probabilities in the same row must sum to one, i.e., $\sum_{l=1}^{|\Omega(y^\ell)|} e^\ell_{kl} = 1$.
2) Hard evidence negating a card's value is constant over time. Therefore, if $C^\ell_{ji} \neq \ell_\iota$ at t_ι, then $P(C^\ell_{ji} = \ell_\iota) = 0$ $\forall t_i > t_\iota$.
3) Hard evidence supporting a card's value is constant over time and also negates all other card values in the same category. Thus, if $C^\ell_{ni} = \ell_m \in \Omega(y^\ell)$ at t_ι, then $e^\ell_{nm} = 1$ and $e^\ell_{nl} = e^\ell_{km} = 0$ for $\forall k \neq n$, $\forall l \neq m$, and $\forall t_i > t_\iota$.

During the game, the evidence tables are updated at every player's turn using a rule-based system derived from the CLUE® game rules, and shown in Appendix V. The CLUE® BN (Fig. 8) is used to make suggestions and provides a convenient factorization of $P(y^r, M)$, which is used to compute the measurement-benefit function, as explained in Section IV.

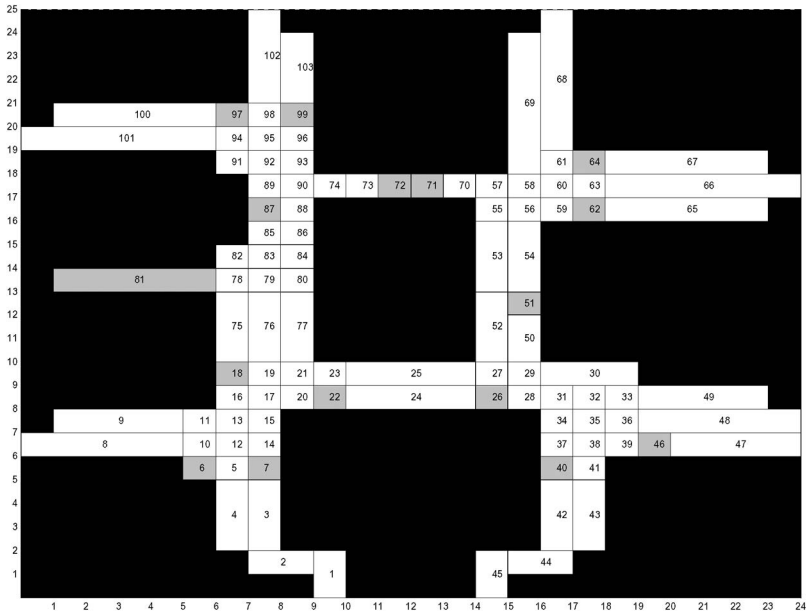

Fig. 9. Convex polygonal decomposition of the pawn's workspace, where void cells are shown in white, observation cells in gray, and obstacles in black.

Using the approach presented in Section VI, the ID representation of the game is obtained from $P(y^r, M)$ and from the connectivity graph $\mathcal{G}$, as shown in the next section.

B. ID Representation of CLUE®

A connectivity graph for the game of CLUE® is obtained via convex polygonal decomposition [37], using the approach presented in [1]. During the game, the pawn can translate in the horizontal or vertical directions inside a 2-D grid in the CLUE® mansion, which is viewed as the pawn's workspace and is shown in Fig. 1. Since the rooms can only be accessed through doors or secret passages, these locations represent the set $\bar{\mathcal{K}}_i$ of all observation cells that enable the measurement (or suggestion) m_i, associated with room r_i. The decomposition of this workspace is shown in Fig. 9, with void cells shown in white and observation cells shown in gray. The corresponding connectivity graph $\mathcal{G}$ is shown in Fig. 10. Using the pruning algorithm presented in Section VI-A, $\mathcal{G}$ is transformed into a pruned connectivity tree T_r using the pawn's initial position κ_0 and the desired final position κ_f, as shown by the example in Appendix VI. At the onset of the game, κ_0 is given by the starting position of the character impersonated by p_3. During the game, when the pawn is moved to a room by a player suggesting its character as the potential suspect, a new T_r is generated using its new position as κ_0. Also, a new T_r is required when κ_f changes as a consequence of the evidence obtained during the game.

Following the approach presented in Section VI, an ID representation of the game is obtained by folding T_r and by introducing test variables and decisions. The CLUE® ID has the structure shown in Fig. 7, where $x(t_k)$ represents the cells that can be visited at time t_k, and t_k is the time index for the turns of p_3. $z(t_k)$ represents the set of suggestions that can be performed at time t_k. The action decision $a(t_k)$ determines the cell to which the pawn moves at time t_{k+1}, and the test decision

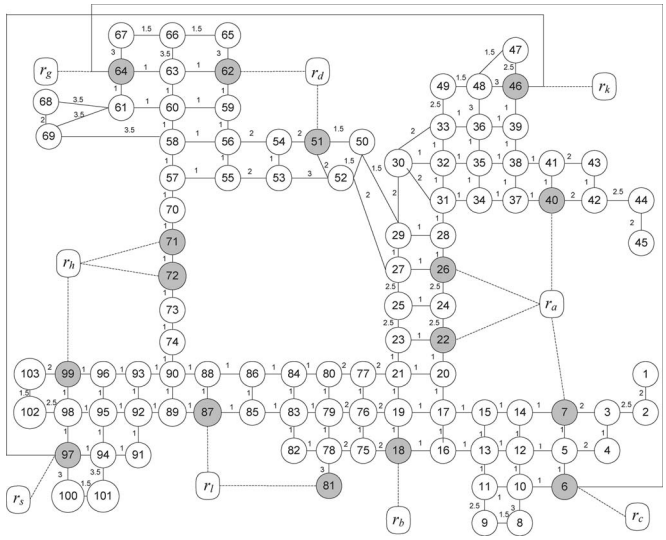

Fig. 10. Connectivity graph with observations corresponding to the decomposition in Fig. 9, with dashed lines indicating the room that can be entered through each observation cell, and with d_{ij} provided by the taxicab metric [38].

$u(t_k)$ determines whether one of the admissible suggestions will be performed. Since the turn ends every time a suggestion is made, the measurement cost $J(t_k)$ is the median distance that can be traveled in one turn, i.e., three bins. The desired tradeoff between the three objective functions, i.e., B, D, and J, is specified through the weights w_B, w_D, and w_J, respectively. A player that favors moving quickly about the board versus entering the rooms is obtained by choosing $w_B \ll w_D, w_J$ and vice versa. The set of CLUE® measurements consists of the suggestions that can be performed inside the nine rooms in the mansion and do not belong to p_3, i.e., $M = \{m_i : i = 1, \ldots, 7, r_i \in \Omega(y^r)\}$, where m_i can only be performed in r_i. Each suggestion in M has four the following mutually exclusive outcomes: $m_i^1 = \{r_i \text{ belongs to } p_1\}$, $m_i^2 = \{r_i \text{ belongs to } p_2\}$, $m_i^3 = \{r_i \text{ belongs to neither } p_1 \text{ nor } p_2\}$,

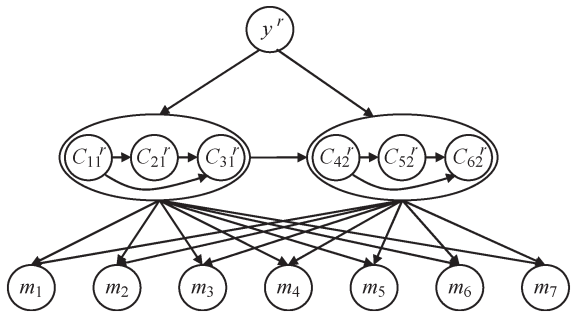

Fig. 11. BN factorization of the joint PMF $P(y^r, M)$ for CLUE®.

and $m_i^4 = \{r_i \text{ is unobserved}\}$. Each outcome corresponds to a possible reply by p_1 or p_2 in response to a suggestion by p_3 in room r_i. Thus, the CPT of a suggestion $m_i \in M$ in the BN model shown in Fig. 11 is

$$\begin{aligned} &P\left(m_i = m_i^1 | C_{l1}^r = r_i, C_{ji}^r\right) = 1, \\ &\quad \text{if } \upsilon\left(\pi(m_i)\right) = 1, \quad \text{for } l = 1,2,3 \\ &P\left(m_i = m_i^2 | C_{l2}^r = r_i, C_{ji}^r\right) = 1, \\ &\quad \text{if } \upsilon\left(\pi(m_i)\right) = 1, \quad \text{for } l = 4,5,6 \\ &P\left(m_i = m_i^3 | C_{ji}^r \neq r_i\right) = 1, \\ &\quad \text{if } \upsilon\left(\pi(m_i)\right) = 1, \quad \text{for } \forall j, i \\ &P\left(m_i = m_i^4 | C_{ji}^r\right) = 1, \\ &\quad \text{if } \upsilon\left(\pi(m_i)\right) = 0 \end{aligned} \tag{27}$$

and is defined $\forall C_{ji}^r \in \text{ch}(y^r)$, where $\upsilon(\cdot)$ is given by (23), and all other CPT entries are set equal to zero. Although the hidden weapon and suspect cards are also inferred using the CLUE® BN in Fig. 8, they need not be included in the ID because their suggestions can be made in any room and thus do not depend on the pawn's position $x(t_k)$.

The exact LIMID algorithm [33] is implemented using the MATLAB BN Toolbox [39] to obtain an optimal game strategy σ^* from the CLUE® ID. During the game, σ^* is updated whenever the pawn is moved to a room by one of the other players or when the evidence table E^r is updated. Some examples of optimal strategies are provided in Table III for various combinations of κ_0 and κ_f and weights w_B and w_D, where $w_J = 1$, and B_{tot} and D_{tot} denote the total measurement benefit and distance traveled over $(t_0, t_f]$, respectively. It can be seen from Table III that when $w_D \gg w_B$, the cell sequence specified by σ^* has fewer observation cells, unless entering a room shortens the distance traveled by means of a secret passage (test decisions not shown for simplicity). When $w_B = 0$, the optimal strategy includes $u^*(t_k) = m_{\text{un}}\ \forall k$, and the resulting cell sequence includes mostly void cells (Table III). On the other hand, when $w_D \ll w_B$, σ^* includes several observation cells and frequent decisions to enter rooms and make suggestions. Also, based on σ^*, when the expected measurement benefit is low, the pawn moves through a room without making a suggestion in order to use a secret passage to another room.

VIII. Game Results

A computer player implementing the optimal strategies obtained from the CLUE® ID was tested by making it compete against human and computer players implementing BNs [21], Q-learning [22], and CS [14] in order to demonstrate the ID representation and solution of the treasure hunt problem. Since humans are very effective at winning the game of CLUE® [21], [22], an interactive simulation of the game is developed through the MATLAB Graphical User Interface Toolbox [40] to allow the human players to confront the computer players. After a character and a pawn are assigned to each player, the simulation deals the cards. At any time during the game, the human players can see their cards on the screen at the command of a button. The die is replicated by a random number generator that produces an integer between one and six. The mansion is illustrated on an interactive board where the players can move the pawns, and enter rooms through doors or secret passages, as in the real game. Humans refute and make suggestions by means of pull-down menus, while the computer players exchange data in the MATLAB environment and display cards on the screen to the humans. The same simulation is also used to compare different computer players.

All of the computer players are fully autonomous. The ID player implements the action decisions in σ^* to move its pawn, and uses the test decisions, together with the CLUE® BN in Fig. 8, to make suggestions. The BN player uses a CLUE® BN to make suggestions and a heuristic rule described in [21] to navigate the board without implementing the ID presented in this paper. The CS player implements the heuristic rule in [21] to move the pawn but uses the approach in [14, Ch. 5] to make suggestions based on the following constraints: C1) $y^\ell \neq C_{ji}^\ell\ \forall \ell$, i, j and C2) $C_{ji}^\ell \neq C_{nm}^\ell$ for i, $m = 2,3$, j, $n = 1,2,3$, and $i \neq m$ or $j \neq n$. Suppose that the CS player makes a suggestion $G = \{s_G, w_G, r_G\}$ regarding the hidden suspect, weapon, and room cards. When the ith player refutes it by showing a card ℓ_G, $C_{ji}^\ell = \ell_G$ is obtained for some $\ell = s$, w, r and $j = 1,2,3$, and C1) and C2) are updated accordingly. Thus, in order to make a future suggestion, the CS player searches a new assignment for s_G, w_G, and r_G that satisfies the updated constraints [C1) and C2)].

The simulations in Table IV verify that when three players of the same type play against each other, they have an approximately equal chance of winning. As shown by the results of 100 games in Tables V and VI, both the ID and BN players outperform the CS player. It can be seen that, on average, the ID player determines the room card y^r in the smallest number of turns and wins more often against the CS players because it optimizes the total reward (2) representing a tradeoff between measurement benefit and distance. The effectiveness of the ID approach compared to the BN approach is confirmed by the results in Table VII that are obtained from 50 games in which the ID player confronts two BN players and wins approximately three times as many games as each BN player does. The ID player is found to always outperform the BN player and to display a higher winning rate not only against two CS players (Table VI) but also against two humans (Table VIII). It can be seen that the two humans perform differently against both the BN and ID players. As shown by the studies presented at the end of this section, this difference is caused by the different levels of skill and experience with the game.

Recently, a neural-network (NN) CLUE® player implementing Q-learning was presented in [22]. In this approach, a

TABLE III
EXAMPLES OF OPTIMAL CLUE® PATHS

κ_0, κ_f	w_B, w_D	Optimal cell sequence	B_{tot}	D_{tot}
1, 18	0, 0	[1 2 3 7 a 22 20 17 16 18]	0	10.5
	0, 12	[1 2 3 4 5 12 13 16 18]	0	11.5
	12, 0	[1 2 3 4 5 6 c 6 5 7 a 22 20 17 16 18]	9.55	14.5
1, 51	0, 0	[1 2 3 4 5 6 c g 64 63 62 d 51]	0	10.5
	0, 6	[1 2 3 7 14 15 17 19 21 23 25 27 52 51]	0	21.5
	10, 6	[1 2 3 4 5 6 c g 64 63 62 d 51]	7.96	10.5
100, 22	0, 0	[100 97 s k 46 39 38 37 40 a 22]	0	7
	0, 6	[100 97 94 91 92 89 87 85 83 79 76 19 17 20 22]	0	18
	12, 0	[100 97 98 99 h 99 98 97 s k 46 39 38 37 40 a 22]	13.84	11

TABLE IV
GAME RESULTS FOR THREE PLAYERS OF THE SAME TYPE

Players:	p_1	p_2	p_3
Games won / games played by BN players	15 / 50	18 / 50	17 / 50
Games won / games played by ID players	16 / 50	16 / 50	18 / 50

TABLE V
PERFORMANCE OF BN PLAYER COMPETING AGAINST TWO CS PLAYERS

Average performance:	BN player, p_1	CS player, p_2	CS player, p_3
Time to determine y^r	13.3 turns	12.5 turns	13.2 turns
Time to win the game	13.7 turns	12.5 turns	13.2 turns
Winning rate	64 %	22 %	14 %

TABLE VI
PERFORMANCE OF ID PLAYER COMPETING AGAINST TWO CS PLAYERS

Average performance:	ID player, p_1	CS player, p_2	CS player, p_3
Time to determine y^r	10.3 turns	14.5 turns	14.3 turns
Time to win the game	10.5 turns	14.5 turns	14.3 turns
Winning rate	70 %	18 %	12 %

TABLE VII
PERFORMANCE OF ID PLAYER COMPETING AGAINST TWO BN PLAYERS

Average performance:	ID player, p_1	BN player, p_2	BN player, p_3
Time to determine y^r	11.2 turns	11.5 turns	11.8 turns
Time to win the game	11.5 turns	11.7 turns	11.8 turns
Winning rate	68 %	18 %	14 %

sigmoidal NN is used to approximate the measurement-benefit function, or Q function, based on previous game histories. The PMF of the hidden cards is viewed as the state of a Markov decision process with an unknown transition probability function. Then, during the game, optimal stationary policies are computed via Q-learning and used by the NN player to move the pawn and make suggestions. The results obtained from 50 games in Table IX show that the ID player is the most effective at winning the game, when compared to the NN and BN players, and that the NN player is more effective than the BN player. The NN player determines the room card y^r in the smallest number of turns, indicating that it may be the most effective at approximating the measurement-benefit function. However, its winning rate is lower than that of the ID player because it is less effective at navigating the mansion based on the available suggestions, i.e., at solving the treasure hunt problem. Thus, the NN player may win, for example, when the hidden room is near its initial position or is entered by chance. The results also show that when more skilled players, such as humans, are involved, the games tend to be shorter because more useful evidence about the hidden cards is obtained from other players' suggestions (Appendix V). For example, when the ID player plays against experienced humans, most games are won after approximately eight turns, whereas against CS

TABLE VIII
WINNING RATE OF ID AND BN PLAYERS COMPETING AGAINST TWO HUMAN PLAYERS

BN player, p_1	Human, p_2	Human, p_3
20 %	50 %	30 %
ID player, p_1	**Human, p_2**	**Human, p_3**
42 %	20 %	38 %

TABLE IX
GAME RESULTS FOR ID, NN, AND BN PLAYERS

Average performance:	ID player, p_1	NN player, p_2	BN player, p_3
Time to determine y^r	9.89 turns	8.78 turns	11.2 turns
Time to win the game	10.0 turns	8.78 turns	11.2 turns
Winning rate	72 %	18 %	10 %

TABLE X
GAME RESULTS FOR INEXPERIENCED HUMAN PLAYERS

Players:	Games won / games played	Winning rate	Time to determine y^r	Time to win the game
ID player	14 / 25	56 %	8.21 turns	8.57 turns
Humans	10 / 25	40 %	n.a.	12.7 turns
CS player	1 / 25	4 %	12 turns	12 turns
BN player	20 / 37	54 %	10.2 turns	10.9 turns
Humans	16 / 37	43 %	n.a.	12.9 turns
CS player	1 / 37	2.7 %	4 turns	4 turns

TABLE XI
GAME RESULTS FOR EXPERIENCED HUMAN PLAYERS

Players:	Games won / games played	Winning rate	Time to determine y^r	Time to win the game
ID player	21 / 43	48.8 %	10.5 turns	10.7 turns
Humans	19 / 43	44 %	n.a.	8.89 turns
CS player	3 / 43	6.98 %	8.33 turns	8.33 turns
BN player	18 / 55	32.7 %	11.7 turns	12.0 turns
Humans	32 / 55	58 %	n.a.	11.4 turns
CS player	5 / 55	9 %	10.6 turns	10.6 turns

players, all games last for at least ten turns, and 30% of the games are won after approximately 14 turns.

A total of nine human players between the ages of 15 and 30 years old, and with different levels of experience, was recruited to confront the computer players. These players are referred to as experienced and inexperienced, based on whether they have previously played the board game or not, respectively. In this paper, the total number of games played varies based on the availability of the human players, and the results are averaged to obtain comparable performance metrics across trials. The final results summarized in Tables X and XI show that by utilizing the ID, the computer player wins more often than both inexperienced and experienced humans. Without an ID, the BN player wins more often than the CS player and the inexperienced humans, but less often than the experienced humans. The game records show that the player who first determines the hidden room card y^r wins the game and that the ID player typically does so before the other players. However, in games won by the experienced humans (Table X) and by the NN player (Table IX), on average, the winning players took less turns to determine y^r than the winning ID player. Moreover, by comparing Table X with Table XI, it can be seen that the use of an ID in the computer player is more important against the experienced players, whereas the ID and BN players perform similarly against the inexperienced human players (Table X). From these results, it can be concluded that the ID player's effectiveness is due to its strategy for deciding the pawn movements and suggestions that optimize the inference process for y^r while minimizing the distance traveled by the pawn in order to determine y^r and win the game as quickly as possible.

IX. SUMMARY AND CONCLUSIONS

The objective of the treasure hunt problem is to infer a hidden hypothesis variable or treasure from an available set of measurements or clues that are accessible only through observation cells in a given connectivity graph. Using the novel pruning algorithm and the measurement-benefit function presented in this paper, a reduced subset of feasible paths is obtained in the form of a pruned connectivity tree, which can be folded into a tractable ID. Also, under proper assumptions, the measurement-benefit function can be computed efficiently over time, without marginalizing over all possible measurements. The board game of CLUE® is found to be an excellent benchmark for illustrating the treasure hunt problem's formulation and solution. The game results show that a computer player implementing the optimal search strategies obtained from the ID representation of the game outperforms existing computer players implementing BNs, Q-learning, or CS, as well as human players. More importantly, the same algorithms can be used to manage robotic sensors in mine-hunting and pursuit-evasion applications, as shown in [1] and [3]. In these applications, the connectivity graph can be obtained by an approximate cell decomposition approach, which is presented in [1], and the BN factorization of the sensor model can be obtained from prior sensor data, as shown in [23] and [24]. Then, the measurement-benefit function and pruning algorithm presented in this paper are used to compute optimal sensing strategies that include the sensor's motion, mode, and measurement sequence [1].

APPENDIX I
PROOF OF THEOREM 4.1

At time t_0, before any measurements are taken, the uncertainty in the hypothesis variable y is the entropy $H(y)$, computed from the prior probability $P(y)$. At time t_1, when the first measurement z_1 is obtained, the uncertainty in y is given by $H(y|z_1)$ and using the result that conditioning reduces entropy [27]

$$H(y|z_1) \leq H(y) \tag{28}$$

where the equality holds if and only if $y \perp z_1$. At time t_2, when a second measurement z_2 is obtained, the chain rule for entropy [27] is applied such that

$$\begin{aligned} H(y, z_1, z_2) &= H(y, z_1|z_2) + H(z_2) \\ &= H(y|z_2, z_1) + H(z_1|z_2) + H(z_2) \qquad (29) \\ H(y, z_1) &= H(z_1) + H(y|z_1). \qquad (30) \end{aligned}$$

Then, the conditional entropy at t_2 can be written as

$$H(y|z_1, z_2) = H(y, z_1|z_2) - H(z_1|z_2) \leq H(y, z_1) - H(z_1|z_2) \qquad (31)$$

where the inequality applies because conditioning reduces entropy. Using (30), the aforesaid inequality is

$$H(y|z_1, z_2) \leq H(y|z_1) + H(z_1) - H(z_1|z_2). \qquad (32)$$

Furthermore, $H(z_1|z_2) \leq H(z_1)$, so $[H(z_1) - H(z_1|z_2)] \geq 0$, and it follows from (32) that conditioning the probability of y upon both z_1 and z_2 must reduce entropy with respect to conditioning y upon z_1 alone, regardless of the outcome of z_2

$$H(y|z_1, z_2) \leq H(y|z_1). \qquad (33)$$

Let z_i and z_j be any two measurements taken at times t_i and t_j, respectively, during the time interval $(t_0, t_f]$. Then, by induction

$$H(y|z_1, \ldots, z_f) \leq H(y|z_1, \ldots, z_j) \leq H(y|z_1, \ldots, z_i) \leq H(y) \qquad (34)$$

provided that $t_j > t_i$. Thus, the conditional entropy is a decreasing function of the measurements, and when $y \not\perp z_1, \ldots, z_f$, it is a monotonically decreasing function, since the aforementioned inequalities all become strict inequalities.

If we let $H(t_k)$ be the entropy of the hypothesis variable y conditioned upon all of the measurements obtained up to t_k, the entropy reduction during a time step in which z_k is obtained

$$\Delta H(t_k) \equiv H(t_{k-1}) - H(t_k) = I(y; z_k|z_{k-1}, \ldots, z_1) \geq 0 \qquad (35)$$

is the reduction in entropy brought about by z_k, and from (34), it is always a nonnegative quantity. From the definition in (4), it can be seen that the entropy reduction between two time steps is the mutual information between y and the latest measurement z_k, given all previous measurements up to z_{k-1}. The entropy reduction is an additive function over time because, for any two time instants t_i, $t_j \in (t_0, t_f]$, with $t_i < t_j$, the total reduction in entropy incurred over the time interval $[t_i, t_j]$ is equal to the sum of the entropy reductions during all time steps in $[t_i, t_j]$

$$\begin{aligned} \sum_{k=i}^{j} \Delta H(t_k) &= \Delta H(t_i) + \Delta H(t_{i+1}) + \cdots \\ &\quad + \Delta H(t_{j-1}) + \Delta H(t_j) \\ &= H(y|z_1, \ldots, z_{i-1}) - H(y|z_1, \ldots, z_i) \\ &\quad + H(y|z_1, \ldots, z_i) - H(y|z_1, \ldots, z_{i+1}) \\ &\quad + \cdots + H(y|z_1, \ldots, z_{j-2}) - H(y|z_1, \ldots, z_{j-1}) \\ &\quad + H(y|z_1, \ldots, z_{j-1}) - H(y|z_1, \ldots, z_j) \\ &= H(y|z_1, \ldots, z_{i-1}) - H(y|z_1, \ldots, z_j) \\ &= I(y; z_j, \ldots, z_i|z_{i-1}, \ldots, z_1). \qquad (36) \end{aligned}$$

The total entropy reduction simplifies to a mutual information because all of the intermediate terms cancel each other out

$$\begin{aligned} \sum_{k=1}^{f} \Delta H(t_k) &= H(y) - H(y|z_1, \ldots, z_f) \\ &= I(y; z_1, \ldots, z_f) = B_{\text{tot}} \qquad (37) \end{aligned}$$

and represents the reduction in the uncertainty of y due to all of the measurements obtained over $(t_0, t_f]$. □

Appendix II
Proof of Remark 4.2

Let $M_k = \{m_\ell, \ldots, m_l\} \subset M$ denote a subset of k measurements in $M = \{m_1, \ldots, m_r\}$, with $k < r$, which are performed during the time interval $(t_0, t_k] \in (t_0, t_f]$, leading to the measurement sequence $Z_{t_k} = \{z_1 = m_\ell, \ldots, z_k = m_l\}$. Then, the total measurement benefit up to time t_k is

$$B(t_k) = \sum_{i=1}^{k} \Delta H(t_i) = I(y; z_1, \ldots, z_k) = I(y; m_\ell, \ldots, m_l) \qquad (38)$$

according to Theorem IV-A. Where, M_k and M are unordered sets, while Z_{t_k} is a totally ordered set or sequence. Consider a different sequence of the same tests M_k, such that another totally ordered set is defined, for example, $Z'_{t_k} = \{z_1 = m_l, \ldots, z_k = m_j\}$. Then, the total measurement benefit up to time t_k is

$$B'(t_k) = \sum_{i=1}^{k} \Delta H(t_i) = I(y; z_1, \ldots, z_k) = I(y; m_l, \ldots, m_j). \qquad (39)$$

From the definition of mutual information [(4)]

$$\begin{aligned} &I(y; m_\ell, m_j, \ldots, m_l) \\ &\quad = \mathbb{E}_{y, m_\ell, m_j, \ldots, m_l} \left\{ \log_2 \frac{P(y|m_\ell, m_j, \ldots, m_l)}{P(y)} \right\} \\ &\quad = \mathbb{E}_{y, m_l, m_\ell, \ldots, m_j} \left\{ \log_2 \frac{P(y|m_l, m_\ell, \ldots, m_j)}{P(y)} \right\} \\ &\quad = I(y; m_l, m_\ell, \ldots, m_j) \qquad (40) \end{aligned}$$

where the expectation $\mathbb{E}_{y, m_\ell, m_j, \ldots, m_l}$ is computed by marginalizing its argument multiplied by the joint PMF $P(y, m_\ell, m_j, \ldots, m_l)$. Since both the marginalization operation and the union of sets (denoted by a comma in the PMFs) are commutative, the mutual information of y and Z_{t_k} can be written as in (40). Thus, $I(y; z_1, \ldots, z_k) = I(y; M_k)$, or $B(t_k) = B'(t_k)$, for any ordered sequence Z_{t_k} containing the set of measurements M_k. □

Appendix III
Label-Correcting Pruning Algorithm

The variable structures introduced in Section VI-A support the following operations.

1) ADJACENT$(n, n_{\text{root}}, n_s)$: Obtain all of the nodes adjacent to n in $\mathcal{G}$ that are within a distance d_M of n_s, and do not include n_{root}.

2) CUT(*branch*, t): Remove all nodes that follow the last observation cell in *branch* columnwise, and return the time index t of the last observation cell.
3) GROW(TREE, n, n_g): Add the arc $n \rightarrow n_g$ to the tree structure in TREE.
4) GETOBSERV(*branch*): Extracts all observation cells in *branch* and sorts them in ascending-order number (i.e., absolute value).
5) INSERT(LIST, n): Adds an item n at the end of the list in LIST.
6) PRUNE(TREE, n): For any branch in TREE with n as a leaf, cut the branch down to but not including its first joint, which is the last node going forward in time that generates other branches not ending in n (i.e., is repeated along the same column).

Then, the following algorithm produces the connectivity tree T_r associated with $\mathcal{G}$, κ_0, and κ_f, for a parameter d_M that is chosen by the user:

Pruning Algorithm {
procedure $T_r(\mathcal{G}, \kappa_0, \kappa_f, d_M)$
begin
initialize TREE $= \{\kappa_0\}$, and all other variables as empty
while $\neg$ END(TREE) do
$i_n \leftarrow$ index of first, shortest row in TREE that does not end in κ_f
$i_{\text{void}} \leftarrow$ index of first observation-free branch of shortest overall distance
$n =$ TREE(i_n, end), $n_{\text{root}} =$ TREE$(i_n, \text{end} - 1)$,
$n_s =$ TREE$(i_{\text{void}}, \text{end})$
adjacent $\leftarrow$ ADJACENT$(n, n_{\text{root}}, n_s)$
for every node $n_a \in$ adjacent do
$n_g =$ nil
branch$_{\text{new}} \leftarrow$ GROW(TREE$(i_n, \cdot), n, n_a)$
distance$_{\text{new}} =$ DIST$_{\text{short}}(n) + D(n, n_a)$
if $n_a \notin$ VISITED then
begin
$n_g = n_a$
VISITED $\leftarrow$ INSERT(VISITED, n_a)
DIST$_{\text{short}} \leftarrow$
INSERT(DIST$_{\text{short}}$, distance$_{\text{new}}$)
end;
else if $n_a > 0 [n_a \in \mathcal{K}_{\text{void}}]$ and distance$_{\text{new}} <$ DIST$_{\text{short}}(n_a)$ then
begin
TREE $\leftarrow$ PRUNE(TREE, n_a)
$n_g = n_a$
end;
else if $n_a < [n_a \in \mathcal{K}_z]$ then
begin
observ$_{\text{new}} =$ GETOBSERV(branch$_{\text{new}}$)
$j \leftarrow$ index value that gives
OBSERV$(j, \cdot) =$ observ$_{\text{new}}$
if $j =$ nil then $n_g = n_a$
else [observ$_{\text{new}} \in$ OBSERV]
begin
(branch$_{\text{old}}, t) \leftarrow$ CUT(TREE$(j, \cdot)$)
if distance$_{\text{new}} <$ DIST$_{\text{tot}}$ (j, t)
begin
TREE $\leftarrow$ PRUNE (TREE, TREE (j, t))
$n_g = n_a$
end;
end;
end;
TREE $\leftarrow$ GROW(TREE, n, n_g)
Update DIST, DIST$_{\text{tot}}$, DIST$_{\text{short}}$, and OBSERV based on n and n_g
$n_{\text{child}} \leftarrow$ INSERT(n_{child}, n_g)
end; [*for loop*]
if $n_{\text{child}} =$ nil then TREE $\leftarrow$ PRUNE (TREE, n)
$n_{child} =$ nil
end; [*while loop*]
end; [*procedure*]
}

APPENDIX IV
PROPERTIES OF PRUNED CONNECTIVITY TREE

The pruning algorithm applies the principle of optimality [34] to paths connecting two cells κ_0 and κ_f in $\mathcal{G}$. First, we demonstrate that T_r contains the path of shortest overall distance $d_{0f} = d_{f0}$. Since κ_f is fixed, we seek the shortest path from κ_f to κ_0 by means of dynamic programming, working *backward* from κ_0 to κ_f. At the second-to-the-last stage, $(t_0, t_1]$, the admissible cells, denoted by the set $\mathcal{N}(t_1)$, are those adjacent to κ_0, and they are all kept by the algorithm, along with their distance, and marked "visited." At t_1, all paths are already optimal because there is only one path to each admissible cell. At t_2, however, the set of admissible cells $\mathcal{N}(t_2)$ contains all cells adjacent to every cell in $\mathcal{N}(t_1)$. Suppose that a void cell κ_a is revisited, then only the path with the shortest overall distance d^*_{0a} is kept by the pruning algorithm. Thus, at any time index t_k, with $t_0 \leq t_k \leq t_f$, the set $\mathcal{N}^*(t_k) > 0$ of admissible and void cells that are kept in T_r lies on the path of minimum distance between t_0 and t_k. Suppose that $\mathcal{N}^*(t_k) = \{\kappa_a, \kappa_b, \kappa_c\}$, then the paths kept are those with the optimal distances d^*_{0a}, d^*_{0b}, and d^*_{0c}, respectively. By the principle of optimality, it follows that for each of these paths

$$d^*_{fa0} = d^*_{0a} + d_{af}$$

$$d^*_{fb0} = d^*_{0b} + d_{bf}$$

$$d^*_{fc0} = d^*_{0c} + d_{cf}$$

where d^*_{fa0} corresponds to the path of minimum distance from κ_f to κ_0, through κ_a. Thus, if any of the cells in $\mathcal{N}^*(t_k)$, for example, κ_b, lie on the path of minimum overall distance, i.e., $d^*_{f0} = d^*_{fb0}$, then the optimal path kept between κ_0 and κ_b also lies on the optimal path between κ_f and κ_0 (which is found when the algorithm terminates). Since the shortest path connecting κ_f to κ_0 is equivalent to the shortest path connecting κ_0 to κ_f, and $d^*_{f0} = d^*_{0f}$, then the pruning algorithm keeps the optimal path, provided that all cells are void.

Now, suppose that some of the cells in $\mathcal{G}$ are observation cells and are thus not always eliminated based solely on distance. If an observation cell $\bar{\kappa}_e$ is visited for the first time, or is revisited through a noninformation-equivalent branch ($j = \text{nil}$), then it is always kept regardless of distance. Since this step does not eliminate but only adds paths, it cannot eliminate the overall optimal path, with distance d^*_{0f}. Instead, suppose that $\bar{\kappa}_e$ is visited twice by the algorithm: once at time t_i by a path through a cell κ_a and once at time $t_j \geq t_i$ by a path through κ_b that is characterized by a shorter distance, i.e., $d^*_{0e} = d^*_{0be} < d^*_{0ae}$. Then, if the two paths are information equivalent, the observation cell is treated like a void cell, and the path $\kappa_0 \ldots \rightarrow \kappa_b \ldots \rightarrow \bar{\kappa}_e$ is kept such that

$$d^*_{fe0} = d^*_{0e} + d_{ef}.$$

Thus, if $\bar{\kappa}_e$ lies on the shortest path from κ_f to κ_0, then so does the path through κ_b, with distance d^*_{0be}, which is kept by the pruning algorithm. The same argument applies to any subsequent time when $\bar{\kappa}_e$ is revisited through an information-equivalent branch that is shorter than the one stored in TREE. Hence, the branch that remains when the tree is complete is the shortest of all information-equivalent branches connecting κ_0 and κ_f through $\bar{\kappa}_e$.

The final case is that of an observation cell, for example, $\bar{\kappa}_g$, which is revisited through an information-equivalent branch through a cell κ_a. Suppose that the existing information-equivalent branch in TREE, through κ_b, contains $\bar{\kappa}_g$ (by definition of information-equivalent branches) but terminates in another cell κ_c. In other words, the two branches are distinct and information equivalent, but the paths connecting κ_0 and $\bar{\kappa}_g$ along them are not information equivalent. In this case, the shortest branch can still be eliminated. Consider the existing branch up to the last observation cell in it, for instance, $\bar{\kappa}_e$, such that the new branch $\kappa_0 \ldots \rightarrow \kappa_a \ldots \rightarrow \bar{\kappa}_g$ can be compared to the shortest information-equivalent path in the existing branch, namely, $\kappa_0 \ldots \rightarrow \kappa_b \ldots \rightarrow \bar{\kappa}_g \ldots \rightarrow \bar{\kappa}_e$ (which has been pruned of the path connecting $\bar{\kappa}_e$ to κ_c but may still contain κ_b as well as any other cells anywhere along the path). If $d_{0ge} > d_{0ag}$, then $d_{0ag} < d_{0bg}$ because d_{ge} cannot be negative. By the principle of optimality $d^*_{0g} = d_{0ag}$, and if $\bar{\kappa}_g$ lies on the overall optimal path from κ_f to κ_0, so does the path $\kappa_0 \ldots \rightarrow \kappa_a \ldots \rightarrow \bar{\kappa}_g$. Since the existing path through κ_b to $\bar{\kappa}_g$ is distinct and suboptimal with respect to the former, then it cannot lie on the shortest path from κ_f to κ_0. Thus, this existing branch can be eliminated because it is information equivalent and longer in distance than the new branch. Moreover, eliminating this branch does not eliminate another possibly optimal path because if such a path went through the existing branch, then it would go through $\bar{\kappa}_g$, and if $\bar{\kappa}_g$ lies on the optimal path, then, by the principle of optimality, the optimal path must include $\kappa_0 \ldots \rightarrow \kappa_a \ldots \rightarrow \bar{\kappa}_g$, or $d^*_{f0} = d^*_{0ag} + d_{gf}$.

Finally, through the operation ADJACENT, the pruning algorithm disallows the parent of a cell to be its direct child because this branch always leads to a suboptimal path. Consider a branch $\kappa_0 \ldots \rightarrow \kappa_p \rightarrow \kappa_c$ in TREE that can be grown through the cells adjacent to its last cell κ_c. By definition, the parent κ_p is always adjacent to the child cell in TREE (in this case, κ_c). Also, if κ_p is in TREE, it implies that its branch contains the shortest information-equivalent path from κ_0 to κ_p with the shortest distance d^*_{0p}, and $d^*_{f0} = d^*_{0p} + d_{pf}$. Thus, if κ_c is added to the branch along with κ_p as its direct child, producing $\kappa_0 \ldots \rightarrow \kappa_p \rightarrow \kappa_c \rightarrow \kappa_p \ldots \rightarrow \kappa_f$, the resulting total distance is $d^*_{0p} + 2 \cdot d_{pc} + d_{pf} > d^*_{0p} + d_{pc} + d_{pf}$ and is thus suboptimal, even when κ_p lies on the optimal path from κ_f to κ_0. Moreover, since the order of the measurements is irrelevant (Remark 4-2), if κ_p is an observation cell, then revisiting it at a later time never adds information benefit to the path, and the branch remains information equivalent to itself.

APPENDIX V
EVIDENCE-TABLE UPDATE PROCEDURE

Let the set of values $G = \{s_G, w_G, r_G\}$ denote the suggestion of a player whose pawn is in room r_G, where $s_G \in \Omega(C^s)$, $w_G \in \Omega(C^w)$, and $r_G \in \Omega(C^r)$. The following are the two cases that allow p_3 to obtain soft evidence from the suggestion of another player: 1) One or more players are not able to refute the suggestion of p_i ($\forall i$), revealing that they do not possess any of the three cards, or 2) a player refutes the suggestion made by a player other than p_3. In these cases, p_3 obtains soft evidence about G that must be placed in the context of prior hard evidence. Hard evidence is obtained only when a player refutes a suggestion made by p_3. The evidence tables are updated by steps 1) and 2) hereinafter, which update the probabilities and guarantee that rules (1)–(3) in Section VII-A are satisfied, respectively.

1) The evidence tables are updated to reflect the evidence gathered from the latest turn t_ι.
   a) *Soft evidence*: Let G denote a suggestion by p_i that cannot be refuted by a player. Then, for any $p_k (k \neq i, k \neq 3)$ that cannot refute G, set

$$e^\ell_{jm} = 0, \quad \text{for } \forall \ell_G \in G, \text{ and } C^\ell_{jk} \in ch(y^\ell) \forall j \tag{41}$$

   where $\ell_G = \ell_m \in \Omega(y^\ell)$.
   b) *Soft evidence*: Let G denote a suggestion by p_i that is refuted by $p_k (i, k \neq 3, i \neq k)$, using a card C_G that is unknown to p_3, but such that $\Omega(C_G) = G$. Let $\phi^\ell = \Omega(y^\ell) \setminus \{\ell_m : e^\ell_{nm} = 1, \forall C^\ell_{ni} \in ch(y^\ell)\}$ denote the set of free values for cards of the ℓth category, based on E^ℓ_{old}. The set $\phi_G \equiv G \cap (\phi^s \cup \phi^w \cup \phi^r)$ contains the free values in G, and the evidence table is updated by

$$\begin{aligned} e^\ell_{ln} &= P_{\text{new}}\left(C^\ell_{lk} = \ell_G\right) = P\left(C^\ell_{lk} = \ell_G | C_G = \ell_G\right) P(C_G = \ell_G) \\ &\quad + P\left(C^\ell_{lk} = \ell_G | C_G \neq \ell_G\right) [1 - P(C_G = \ell_G)] \\ &\approx P(C_G = \ell_G) + P_{\text{old}}\left(C^\ell_{lk} = \ell_G\right) [1 - P(C_G = \ell_G)] \end{aligned} \tag{42}$$

   where $\ell_G = \ell_n \in \Omega(y^\ell)$, $P_{\text{old}}(\cdot)$ is available from E^ℓ_{old}

$$l = \max_j P\left(C^\ell_{jk} = \ell_G\right)$$

$$P(C_G = \ell_G) = \begin{cases} 1/|\phi_G|, & \text{if } \ell_G \in \phi_G \\ 0, & \text{if } \ell_G \notin \phi_G. \end{cases}$$

c) *Hard evidence*: Let G denote a suggestion by p_3 that is refuted by p_k ($k = 1$ or 2), using a card $\ell_G = \ell_m \in \Omega(y^\ell)$. Let $h = \max_j H(C^\ell_{jk})$, where H is the entropy. Then, $e^\ell_{hm} = 1$ and $e^\ell_{hl} = 0$ for $\forall l \neq m$ in table E^ℓ.

2) Normalize the row and column containing the element e^ℓ_{ij} updated in Step I:

a) Let $|\Omega(y^\ell)|$ be the number of columns in E^ℓ. Then, if $e^\ell_{il} = 1$, set $e^\ell_{im} = 0$ for $\forall m \neq l$. Otherwise, let

$$\Delta e^\ell_i = 1 - \sum_{m=1}^{|\Omega(y^\ell)|} e^\ell_{im}$$

$$e^\ell_{il} = e^\ell_{il} + \frac{\Delta e^\ell_i}{|\Omega(y^\ell)|}, \qquad \text{for} \quad \forall l \neq j \quad \text{and} \quad \forall e^\ell_{il} \neq 0.$$

Set all negative elements e^ℓ_{il} equal to zero, and repeat the aforementioned procedure until they all are nonnegative.

b) Consider the column in E^ℓ that contains e^ℓ_{ij}. If $e^\ell_{kj} = 1$, set $e^\ell_{nj} = 0$ for $\forall n \neq k$ and $n \neq i$. Otherwise, leave the column unaltered.

APPENDIX VI
EXAMPLE OF CONNECTIVITY TREE IN THE GAME OF CLUE®

NODES OF T_r OBTAINED FROM $\mathcal{G}$ IN FIG. 10 FOR $d_M = 5$, $\kappa_0 = \kappa_1$, AND $\kappa_f = \kappa_{51}$ (ARCS AND COSTS ARE SHOWN IN FIG. 10)

t_0	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9	t_{10}	t_{11}	t_{12}	t_{13}
1	2	3	4	a	6	a	6	c	g	a	g	7	d
t_{14}	t_{15}	t_{16}	7	5	7	c	7	5	b	c	b	14	14
15	b	b	t_{17}	14	15	5	15	14	a	7	a	15	15
17	a	18	b	t_{18}	22	14	16	17	6	14	7	17	16
18	17	19	a	18	26	17	19	18	7	17	14	18	17
19	18	20	18	19	t_{19}	23	20	21	15	18	15	21	19
20	19	21	19	21	19	24	22	22	19	21	19	22	20
21	20	22	21	22	21	27	25	23	20	22	20	23	21
22	21	23	22	23	23	t_{20}	26	24	23	25	22	24	23
23	22	24	23	24	24	21	52	27	24	26	23	25	24
25	23	25	24	25	25	23	t_{21}	51	25	27	24	26	25
26	24	26	25	26	26	25	23	t_{22}	26	51	26	27	26
27	25	27	26	27	27	26	25	25	51	64	27	51	27
51	27	51	27	51	51	27	27	27	52		51	52	51
52	51	52	51	52	52	51	51	51	t_{23}	t_{24}	52	60	52
56	52	56	52			52	52	52	27	51	61	62	58
60	54		54						51	52	63	64	59
62	58								52		t_{25}	69	61
69	59										51		63

ACKNOWLEDGMENT

The authors would like to thank Prof. R. Parr at Duke University and Prof. S. Russell at UC Berkeley for their helpful guidance and suggestions.

REFERENCES

[1] C. Cai and S. Ferrari, "Information-driven sensor path planning by approximate cell decomposition," *IEEE Trans. Syst., Man Cybern. B, Cybern.*, vol. 39, no. 3, Jun. 2009.

[2] S.-M. Lucas and G. Kendall, "Evolutionary computation and games," *IEEE Comput. Intell. Mag.*, vol. 1, no. 1, pp. 10–18, Feb. 2006.

[3] S. Ferrari, C. Cai, R. Fierro, and B. Perteet, "A multi-objective optimization approach to detecting and tracking dynamic targets in pursuit-evasion games," in *Proc. Amer. Control Conf.*, New York, 2007, pp. 5316–5321.

[4] R. Siegel, "Land mine detection," *IEEE Instrum. Meas. Mag.*, vol. 5, no. 4, pp. 22–28, Dec. 2002.

[5] J. Colegrave and A. Branch, "A case study of autonomous household vacuum cleaner," in *Proc. AIAA/NASA CIRFFSS*, Houston, TX, 1994.

[6] D. Culler, D. Estrin, and M. Srivastava, "Overview of sensor networks," *Computer*, vol. 37, no. 8, pp. 41–49, Aug. 2004.

[7] P. Juang, H. Oki, Y. Wang, M. Martonosi, L. Peh, and D. Rubenstein, "Energy efficient computing for wildlife tracking: Design tradeoffs and early experiences with ZebraNet," in *Proc. 10th Int. Conf. ASPLOS*, San Jose, CA, 2002, pp. 96–107.

[8] W. Schmaedeke, "Information based sensor management," in *Proc. SPIE Signal Process., Sensor Fusion, Target Recognit. II*, Orlando, FL, 1993, vol. 1955, pp. 156–164.

[9] K. Kastella, "Discrimination gain to optimize detection and classification," *IEEE Trans. Syst., Man Cybern. A, Syst., Humans*, vol. 27, no. 1, pp. 112–116, Jan. 1997.

[10] F. Zhao, J. Shin, and J. Reich, "Information-driven dynamic sensor collaboration," *IEEE Signal Process. Mag.*, vol. 19, no. 2, pp. 61–72, Mar. 2002.

[11] X. Liao and L. Carin, "Application of the theory of optimal experiments to adaptive electromagnetic-induction sensing of buried targets," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 26, no. 8, pp. 961–972, Aug. 2004.

[12] C. Kreucher, K. Kastella, and A. Hero, "Multi-platform information-based sensor management," in *Proc. SPIE Defense Transformation Netw.-Centric Syst. Symp.*, Orlando, FL, 2005, vol. 5820, pp. 141–151.

[13] C. Kreucher, K. Kastella, and A. Hero, "Sensor management using an active sensing approach," *Signal Process.*, vol. 85, no. 3, pp. 607–624, Mar. 2005.

[14] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*. Upper Saddle River, NJ: Prentice-Hall, 2003.

[15] S. Ji, R. Parr, and L. Carin, "Nonmyopic multiaspect sensing with partially observable Markov decision processes," *IEEE Trans. Signal Process.*, vol. 55, no. 1, pp. 2720–2730, Jun. 2007.

[16] M. R. Garey, "Optimal task sequencing with precedence constraints," *Discrete Math.*, vol. 4, pp. 37–56, 1973.

[17] H. A. Simon and J. B. Kadane, "Optimal problem-solving search: All-or-none solutions," *Artif. Intell.*, vol. 6, no. 3, pp. 235–247, 1975.

[18] D. Castañon, "Optimal search strategies in dynamic hypothesis testing," *IEEE Trans. Syst., Man Cybern.*, vol. 25, no. 7, pp. 1130–1138, Jul. 1995.

[19] D. P. Bertsekas, "A simple and fast label correcting algorithm for shortest paths," *Networks*, vol. 23, no. 7, pp. 703–709, Jul. 1993.

[20] A. J. Viterbi, "Error bounds for convolutional codes and an asymptotically optimum decoding algorithm," *IEEE Trans. Inf. Theory*, vol. IT-32, no. 2, pp. 260–269, Apr. 1967.

[21] C. Cai and S. Ferrari, "On the development of an intelligent computer player for CLUE: A case study on preposterior decision analysis," in *Proc. Amer. Control Conf.*, Minneapolis, MN, 2006, pp. 4350–4355.

[22] C. Cai and S. Ferrari, "A Q-learning approach to developing an automated neural computer player for the board game of CLUE," in *Proc. Int. Joint Conf. Neural Netw.*, Hong Kong, 2008, pp. 2347–2353.

[23] S. Ferrari and A. Vaghi, "Demining sensor modeling and feature-level fusion by Bayesian networks," *IEEE Sensors J.*, vol. 6, no. 2, pp. 471–483, Apr. 2006.

[24] C. Cai, S. Ferrari, and Q. Ming, "Bayesian network modeling of acoustic sensor measurements," in *Proc. IEEE Sensors*, Atlanta, GA, 2007, pp. 345–348.

[25] H. Choset, "Coverage for robotics—A survey of recent results," *Ann. Math. Artif. Intell.*, vol. 31, no. 1–4, pp. 113–126, Oct. 2001.

[26] E. Lawer, J. Lenstra, A. Kan, and D. Shmoys, *The Traveling Salesman Problem*. New York: Wiley, 1985.
[27] T. M. Cover and J. A. Thomas, *Elements of Information Theory*. New York: Wiley, 1991.
[28] F. V. Jensen, *Bayesian Networks and Decision Graphs*. Berlin, Germany: Springer-Verlag, 2001.
[29] M. I. Jordan, *Learning in Graphical Models*. Cambridge, MA: MIT Press, 1998.
[30] H. Raiffa and R. Schlaifer, *Applied Statistical Decision Theory*. Cambridge, MA: MIT Press, 1961.
[31] D. Poole, "The independent choice logic for modelling multiple agents under uncertainty," *Artif. Intell.*, vol. 94, no. 1/2, pp. 7–56, Jul. 1997.
[32] M. Diehl and Y.-Y. Haimes, "Influence diagrams with multiple objectives and tradeoff analysis," *IEEE Trans. Syst., Man Cybern. A, Syst., Humans*, vol. 34, no. 3, pp. 293–304, May 2004.
[33] S. L. Lauritzen and D. Nilsson, "Representing and solving decision problems with limited information," *Manage. Sci.*, vol. 47, no. 9, pp. 1235–1251, Sep. 2001.
[34] R. E. Bellman and S. E. Dreyfus, *Applied Dynamic Programming*. Princeton, NJ: Princeton Univ. Press, 1962.
[35] L. C. Polymenakos, D. P. Bertsekas, and J. N. Tsitsiklis, "Implementation of efficient algorithms for globally optimal trajectories," *IEEE Trans. Autom. Control*, vol. 43, no. 2, pp. 278–283, Feb. 1998.
[36] J. Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Los Altos, CA: Morgan Kaufmann, 1988.
[37] J. C. Latombe, *Robot Motion Planning*. Norwell, MA: Kluwer, 1991.
[38] S. Thurn, "Learning metric-topological maps for indoor mobile robot navigation," *Artif. Intell.*, vol. 99, no. 1, pp. 21–71, Feb. 1998.
[39] K. Murphy. (2004). *How to Use Bayes Net Toolbox*. [Online]. Available: http://www.ai.mit.edu/ murphyk/Software/BNT/bnt.html
[40] Mathworks, Matlab, 2004. [Online]. Available: http://www.mathworks.com

Silvia Ferrari (S'01–M'02–SM'08) received the B.S. degree from Embry–Riddle Aeronautical University, Daytona Beach, FL, and the M.A. and Ph.D. degrees from Princeton University, Princeton, NJ.

She is an Assistant Professor of mechanical engineering and materials science with the Department of Mechanical Engineering and Materials Science, Duke University, Durham, NC, where she directs the Laboratory for Intelligent Systems and Controls. Her principal research interests include robust adaptive control of aircraft, learning and approximate dynamic programming, and optimal control of mobile sensor networks.

Dr. Ferrari is a member of ASME, SPIE, and AIAA. She is the recipient of the ONR Young Investigator Award (2004), the NSF CAREER Award (2005), and the Presidential Early Career Award for Scientists and Engineers Award (2006).

Chenghui Cai (S'02–M'08) received the B.S. and M.S. degrees from Tsinghua University, Beijing, China, in 2000 and 2003, respectively, and the Ph.D. degree from Duke University, Durham, NC, in 2008.

He is currently a Postdoctoral Research Associate of electrical and computer engineering with Duke University. His research interests include robotic sensor planning and management, machine learning and data mining, multiagent systems, Bayesian statistics, decision making under uncertainty, and computational intelligence in games.

Dr. Cai is a member of ASME, Sigma Xi, and SIAM.