Bayesian networks in Mastermind

Jiří Vomlel
http://www.utia.cas.cz/vomlel/

Laboratory for Intelligent Systems
University of Economics
Ekonomická 957
148 01 Praha 4, Czech Republic

Inst. of Inf. Theory and Automation
Academy of Sciences
Pod vodárenskou věží 4
182 08 Praha 8, Czech Republic

Abstract

The game of Mastermind is a nice example of an adaptive test. We propose a modification of this game - a probabilistic Mastermind. In the probabilistic Mastermind the code-breaker is uncertain about the correctness of code-maker responses. This modification better corresponds to a real world setup for the adaptive testing. We will use the game to illustrate some of the challenges that one faces when Bayesian networks are used in adaptive testing.

1 Mastermind

Mastermind was invented in early 1970's by Mordecai Meirowitz. A small English company Invicta Plastics Ltd. bought up the property rights to the game, refined it, and released it in 1971-72. It was an immediate hit, and went on to win the first ever Game of the Year Award in 1973. It became the most successful new game of the 1970's [8].

Mastermind is a game played by two players, the code-maker and the code-breaker. The code-maker secretly selects a hidden code $H_1, \ldots, H_4$ consisting of an ordered sequence of four colors, each chosen from a set of six possible colors $\{1, 2, \ldots, 6\}$, with repetitions allowed. The code-breaker will then try to guess the code. After each guess $T = (T_1, \ldots, T_4)$ the code-maker responds with two numbers. He computes the number P of pegs with correctly guessed color with correct position, i.e.

$$P_j = \delta(T_j, H_j) \ , \quad \text{for } j = 1, \ldots, 4 \qquad (1)$$

$$P = \sum_{j=1}^{4} P_j \ , \qquad (2)$$

where $\delta(A, B)$ is the function that is equal to one if $A = B$ and zero otherwise. Second, the code-maker computes the number C of pegs with correctly guessed color that are in a wrong position. Exactly speaking, he computes

$$C_i = \sum_{j=1}^{4} \delta(H_j, i) \ , \quad \text{for } i = 1, \ldots, 6 \qquad (3)$$

$$G_i = \sum_{j=1}^{4} \delta(T_j, i) \ , \quad \text{for } i = 1, \ldots, 6 \qquad (4)$$

$$M_i = \min(C_i, G_i) \ , \quad \text{for } i = 1, \ldots, 6 \qquad (5)$$

$$C = \left(\sum_{i=1}^{6} M_i \right) - P \ . \qquad (6)$$

The numbers P, C are reported by number of black and white pegs, respectively.

Example 1 For $(1, 1, 2, 3)$ and $(3, 1, 1, 1)$ the response is $P = 1$ and $C = 2$. ◇

The code-breaker continues guessing until he guesses the code correctly or until he reaches a maximum allowable number of guesses without having correctly identified the secret code.

Probabilistic Mastermind

In the standard model of Mastermind all information provided by the code-maker is assumed to be deterministic, i.e. each response is defined by the hidden code and the current guess. But in many real world situations we cannot be sure that the information we get is correct. For example, a code-maker may not pay enough attention to the game and sometimes makes mistakes in counting the number of correctly guessed pegs. Thus the code-breaker is uncertain about the correctness of the responses of the code-maker.

In order to model such a situation in Mastermind we add two variables to the model: the reported number of pegs with a correct color in the correct position P' and reported number of pegs with a correct color in a wrong position C'. The dependency of P' on P is thus probabilistic, represented by a probability distribution $Q(P' \mid P)$ (with all probability values being non-zero). Similarly, $Q(C' \mid C)$ represents probabilistic dependency of C' on C.

2 Mastermind strategy

We can describe the Mastermind game using the probability framework. Let $Q(H_1, \ldots, H_4)$ denote the probability distribution over the possible codes. At the beginning of the game this distribution is uniform, i.e., for all possible states $h_1, \ldots, h_4$ of

$H_1, \ldots, H_4$ it holds that

$$Q(H_1 = h_1, \ldots, H_4 = h_4) \quad = \quad \frac{1}{6^4} \quad = \quad \frac{1}{1296}$$

During the game we update probability $Q(H_1, \ldots, H_4)$ using the obtained evidence $\mathbf{e}$ and compute the conditional probability $Q(H_1, \ldots, H_4 \mid \mathbf{e})$. Note that in the standard (deterministic) Mastermind it can be computed as

$$Q(H_1 = h_1, \ldots, H_4 = h_4 \mid \mathbf{e}) \quad = \quad \begin{cases} \frac{1}{n(\mathbf{e})} & \text{if } (h_1, \ldots, h_4) \text{ is a possible code} \\ 0 & \text{otherwise,} \end{cases}$$

where $n(\mathbf{e})$ is the total number of codes that are possible candidates for the hidden code.

A criteria suitable to measure the uncertainty about the hidden code is the Shannon entropy

$$H(Q(H_1, \ldots, H_4 \mid \mathbf{e})) \quad = \quad \sum_{h_1, \ldots, h_4} \begin{array}{c} Q(H_1 = h_1, \ldots, H_4 = h_4 \mid \mathbf{e}) \\ \cdot \log Q(H_1 = h_1, \ldots, H_4 = h_4 \mid \mathbf{e}) \end{array} , \tag{7}$$

where $0 \cdot \log 0$ is defined to be zero. Note that the Shannon entropy is zero if and only if the code is known. The Shannon entropy is maximal when nothing is known (i.e. when the probability distribution $Q(H_1, \ldots, H_4 \mid \mathbf{e})$ is uniform.

Mastermind strategy is defined as a tree with nodes corresponding to evidence collected by performing guesses $\mathbf{t} = (t_1, \ldots, t_4)$ and getting answers c, p (in case of standard Mastermind game) or c', p' (in the probabilistic Mastermind). The evidence corresponding to the root of the tree is $\emptyset$. For every node n in the tree with corresponding evidence $\mathbf{e}_n : H(Q(H_1, \ldots, H_4 \mid \mathbf{e}_n)) \neq 0$ it holds:

- it has specified a next guess $\mathbf{t}(\mathbf{e}_n)$ and
- it has one child for each possible evidence obtained after an answer c, p to the guess $\mathbf{t}(\mathbf{e}_n)$[1].

A node n with corresponding evidence $\mathbf{e}_n$ such that $H(Q(H_1, \ldots, H_4 \mid \mathbf{e}_n)) = 0$ is called a *terminal node* since it has no children (it is a leaf of the tree) and the strategy terminates there. *Depth* of a Mastermind strategy is the depth of the corresponding tree, i.e., the number of nodes of a longest path from the root to a leaf of the tree.

We say that a Mastermind strategy $\mathcal{T}$ of depth ℓ is an *optimal Mastermind strategy* if there is no other Mastermind strategy $\mathcal{T}'$ with depth $\ell' < \ell$.

The previous definition is appropriate when our main interest is the worst case behavior. When we are interested in an average behavior other criteria is needed. We can define expected length EL of a strategy as the weighted average of the length of the test:

$$EL \quad = \quad \sum_{n \in \mathcal{L}} Q(\mathbf{e}_n) \cdot \ell(n) \ ,$$

where $\mathcal{L}$ denotes the set of terminal nodes of the strategy, $Q(\mathbf{e}_n)$ is the probability of terminating strategy in node n, and $\ell(n)$ is the number of nodes in the path from the root to a leaf node n. We say that a Mastermind strategy $\mathcal{T}$ of depth ℓ is *optimal in average* if there is no other Mastermind strategy $\mathcal{T}'$ with expected length $EL' < EL$.

Remark Note that there are at maximum

$$\binom{3+4-1}{4} - 1 \quad = \quad 15 - 1 \quad = \quad 14$$

possible responses to a guess[2]. Therefore the lower bound on the minimal number of guesses is

$$\log_{14} 6^4 + 1 \quad = \quad \frac{4 \cdot \log 6}{\log 14} + 1 \quad \doteq \quad 3.716 \ .$$

When the number of guesses is restricted to be at maximum m then we may be interested in a *partial strategy*[3] that brings most information about the code within the limited number of guesses. If we use the Shannon entropy (formula 7) as the information criteria then we can define expected entropy EH of a strategy as

$$EH \quad = \quad \sum_{n \in \mathcal{L}} Q(\mathbf{e}_n) \cdot H(Q(H_1, \ldots, H_4 \mid \mathbf{e}_n)) \ ,$$

where $\mathcal{L}$ denotes the set of terminal nodes of the strategy and $Q(\mathbf{e}_n)$ is the probability of getting to node n. We say that a Mastermind strategy $\mathcal{T}$ is a *most informative* Mastermind strategy of depth ℓ if there is no other Mastermind strategy $\mathcal{T}'$ of depth ℓ with its $EH' < EH$.

In 1993, Kenji Koyama and Tony W. Lai [7] found a strategy (of deterministic Mastermind) optimal in average. It has $EL = 5625/1296 = 4.340$ moves. However, for larger problems it is hard to find an optimal strategy since we have to search a huge space of all possible strategies.

Myopic strategy

Already in 1976 D. E. Knuth [6] proposed a non-optimal strategy (of deterministic Mastermind) with

[1] Since there are at maximum 14 possible combinations of answers c, p node n has at most 14 children.

[2] It is the number of possible combinations (with repetition) of three elements *black peg*, *white peg*, and *no peg* on four positions, while the combination of three *black pegs* and one *white peg* is impossible.

[3] Partial strategy may have terminal nodes with corresponding evidence $\mathbf{e}_n$ such that $H(Q(H_1, \ldots, H_4 \mid \mathbf{e}_n)) \neq 0$.

the expected number of guesses equal to 4.478. His strategy is to choose a guess (by looking one step ahead) that minimizes the number of remaining possibilities for the worst possible response of the codemaker.

The approach suggested by Bestavros and Belal [2] uses information theory to solve the game: each guess is made in such a way that the answer maximizes information on the hidden code on the average. This corresponds to the myopic strategy selection based on minimal expected entropy in the next step.

Let $\mathbf{T}^k = (T_1^k, \ldots, T_4^k)$ denote the guess in the step k. Further let P'^k be the reported number of pegs with correctly guessed color and the position in the step k and C'^k be the reported number of pegs with correctly guessed color but in a wrong position in the step k. Let $\mathbf{e}^k$ denote the evidence collected in steps $1, \ldots, k$, i.e.

$$\mathbf{e}(\mathbf{t}^1, \ldots, \mathbf{t}^k) = \left(\begin{array}{l} \mathbf{T}^1 = \mathbf{t}^1, P^1 = p^1, C'^1 = c'^1, \\ \ldots, \mathbf{T}^k = \mathbf{t}^k, P'^k = p'^k, C'^k = c'^k \end{array} \right)$$

For each $\mathbf{e}(\mathbf{t}^1, \ldots, \mathbf{t}^{k-1})$ the next guess is a $\mathbf{t}^k$ that minimizes

$$EH(\mathbf{e}(\mathbf{t}^1, \ldots, \mathbf{t}^{k-1}, \mathbf{t}^k)) \ .$$

3 Bayesian network model of Mastermind

Different methods from the field of Artificial Intelligence were applied to the (deterministic version of the) Mastermind problem. In [10] Mastermind is solved as a constraint satisfaction problem. A genetic algorithm and a simulated annealing approach are described in [1]. These methods cannot be easilly generalized for the probabilistic modification of the Mastermind.

In this paper we sugest to use a Bayesian network model for the probabilistic version of Mastermind. In Figure 1 we define Bayesian network for the Mastermind game.

The graphical structure defines the joint probability distribution over all variables V as

$$\begin{aligned} Q(V) = \; & Q(C \mid M_1, \ldots, M_6, P) \cdot Q(P \mid P_1, \ldots, P_4) \\ & \cdot \left(\prod_{j=1}^{4} Q(P_j \mid H_j, T_j) \cdot Q(H_j) \cdot Q(T_j) \right) \\ & \cdot \left(\prod_{i=1}^{6} \begin{array}{l} Q(M_i \mid C_i, G_i) \cdot Q(C_i \mid H_1, \ldots, H_4) \\ \cdot Q(G_i \mid T_1, \ldots, T_4) \end{array} \right) \end{aligned}$$

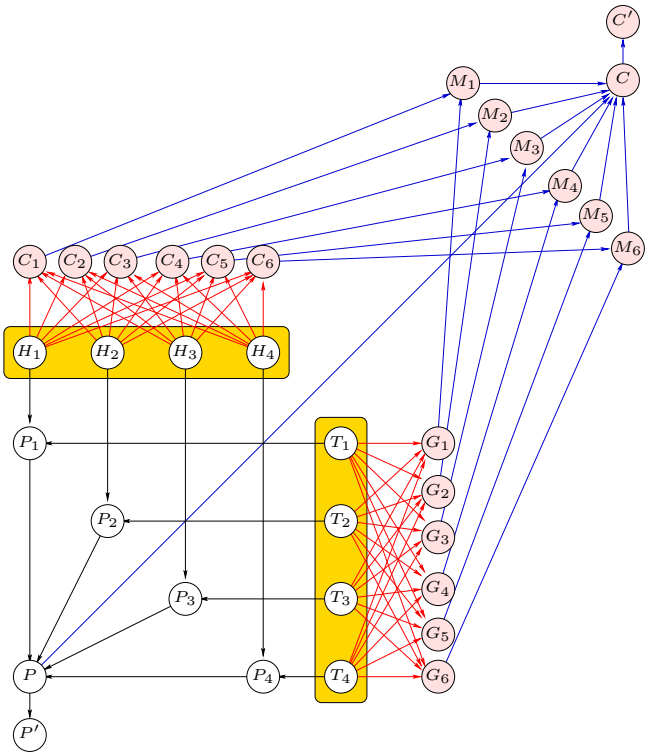

Figure 1: Bayesian network for the probabilistic Mastermind game

Conditional probability tables[4] $Q(X \mid pa(X)), X \in V$ represent the functional (deterministic) dependencies defined in (1)–(6). The prior probabilities $Q(H_i), i = 1, \ldots, 4$ are assumed to be uniform. The prior probabilities $Q(T_i), i = 1, \ldots, 4$ are defined to be uniform as well, but since variables $T_i, i = 1, \ldots, 4$ will be always present in the model with evidence the actual probability distribution does not have any influence.

4 Belief updating

The essential problem is how the conditional probability distribution $Q(H_1, \ldots, H_4 \mid \mathbf{t}, c, p)$ of variables $H_1, \ldots, H_4$ given evidence c, p and $\mathbf{t} = (t_1, \ldots, t_4)$ is computed. Inserting evidence corresponds to fixing states of variables with evidence to the observed states. It means that from each probability table we disregard all values that do not correspond to the observed states.

New evidence can also be used to simplify the model. In Figure 2 we show simplified Bayesian network model after the evidence $T_1 = t_1, \ldots, T_4 = t_4$ was inserted into the model from Figure 1. We eliminated all variables $T_1, \ldots, T_4$ from the model since their states were observed and incorporated the evi-

[4] $pa(X)$ denotes the set of variables that are parents of X in the graph, i.e. $pa(X)$ is the set of all nodes Y in the graph such that there is an edge $Y \rightarrow X$.

dence into probability tables $Q(M_i \mid C_i), i = 1, \ldots, 6$, $Q(C \mid C_1, \ldots, C_6)$, and $Q(P_j \mid H_j), j = 1, \ldots, 4$.

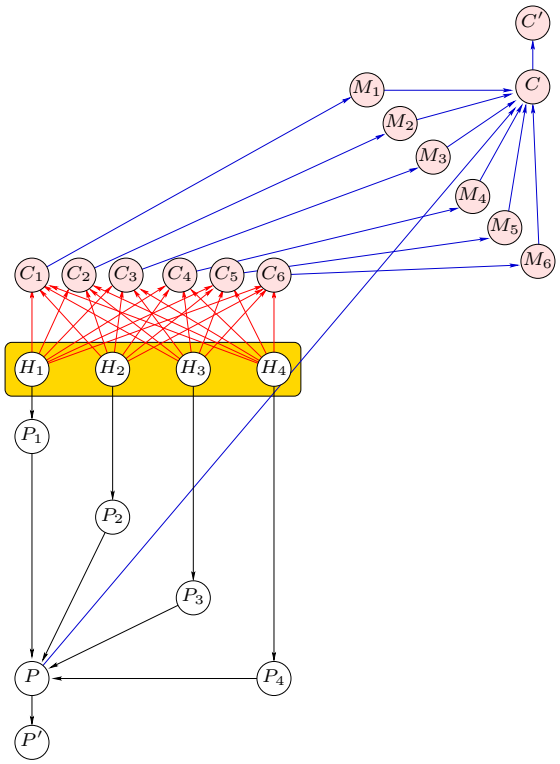

Figure 2: Bayesian network after the evidence $T_1 = t_1, \ldots, T_4 = t_4$ was inserted into the model.

Next, a naive approach would be to multiply all probability distributions and then marginalize out all variables except of variables of our interest. This would be computationally very inefficient. It is better to marginalize out variables as soon as possible and thus keep the intermediate tables smaller. It means the we need to find a sequence of multiplications of probability tables and marginalizations of certain variables - called *elimination sequence* - such that it minimizes the number of performed numerical operations. The elimination sequence must satisfy the condition that all tables containing variable, must be multiplied before this variable can be marginalized out.

A graphical representation of an elimination sequence of computations is *junction tree* [5]. It is the result of moralization and triangulation of the original graph of the Bayesian network (for details see, e.g., [4]). The total size of the optimal junction tree of the Bayesian network from Figure 2 is more than $20,526,445$. The Hugin [3] software, which we have used to find optimal junction trees, run out of memory in this case. However, Hugin was able to find an optimal junction tree (with the total size given above) for the Bayesian network from Figure 2 without the arc $P \to C$.

The total size of junction tree is proportional to the number of numerical operations performed. Thus we prefer the total size of a junction tree to be as small as possible.

We can further exploit the internal structure of the conditional probability table $Q(C \mid C_1, \ldots, C_6)$. We can use a multiplicative factorization of the table corresponding to variable C using an auxiliary variable B (having the same number of states as C, i.e. 5) described in [9]. The Bayesian network after this transformation is given in Figure 3.

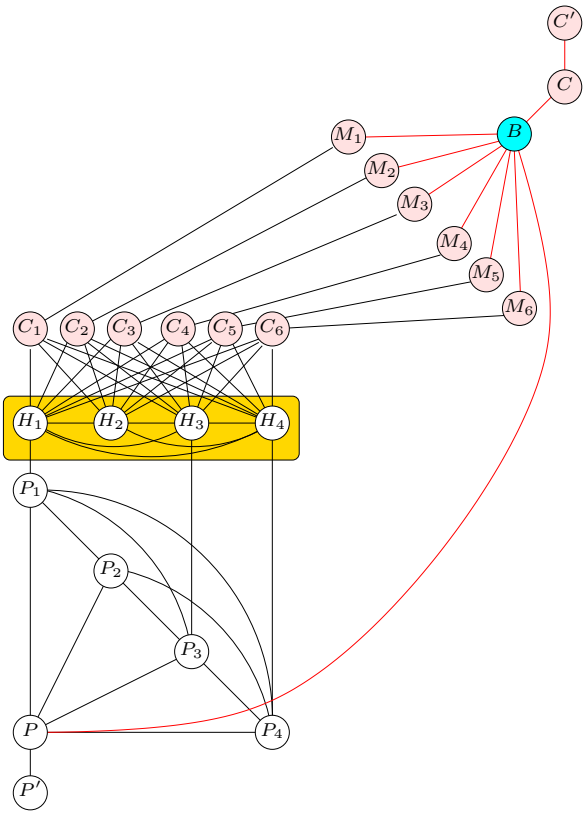

Figure 3: Bayesian network after the suggested transformation and moralization.

The total size of its junction tree (given in Figure 4) is $214,775$, i.e. it is more than 90 times smaller than the junction tree of Bayesian network before the transformation.

After each guess of a Mastermind game we first update the joint probability on $H_1, \ldots, H_4$. Then we retract all evidence and keep just the joint probability on $H_1, \ldots, H_4$. This allows to insert new evidence to the same junction tree. This process means that evidence from previous steps is combined with the new evidence by multiplication of distributions on $H_1, \ldots, H_4$ and consequent normalization, which corresponds to the standard updating using the Bayes rule.

Remark In the original deterministic version of Mastermind after each guess many combinations

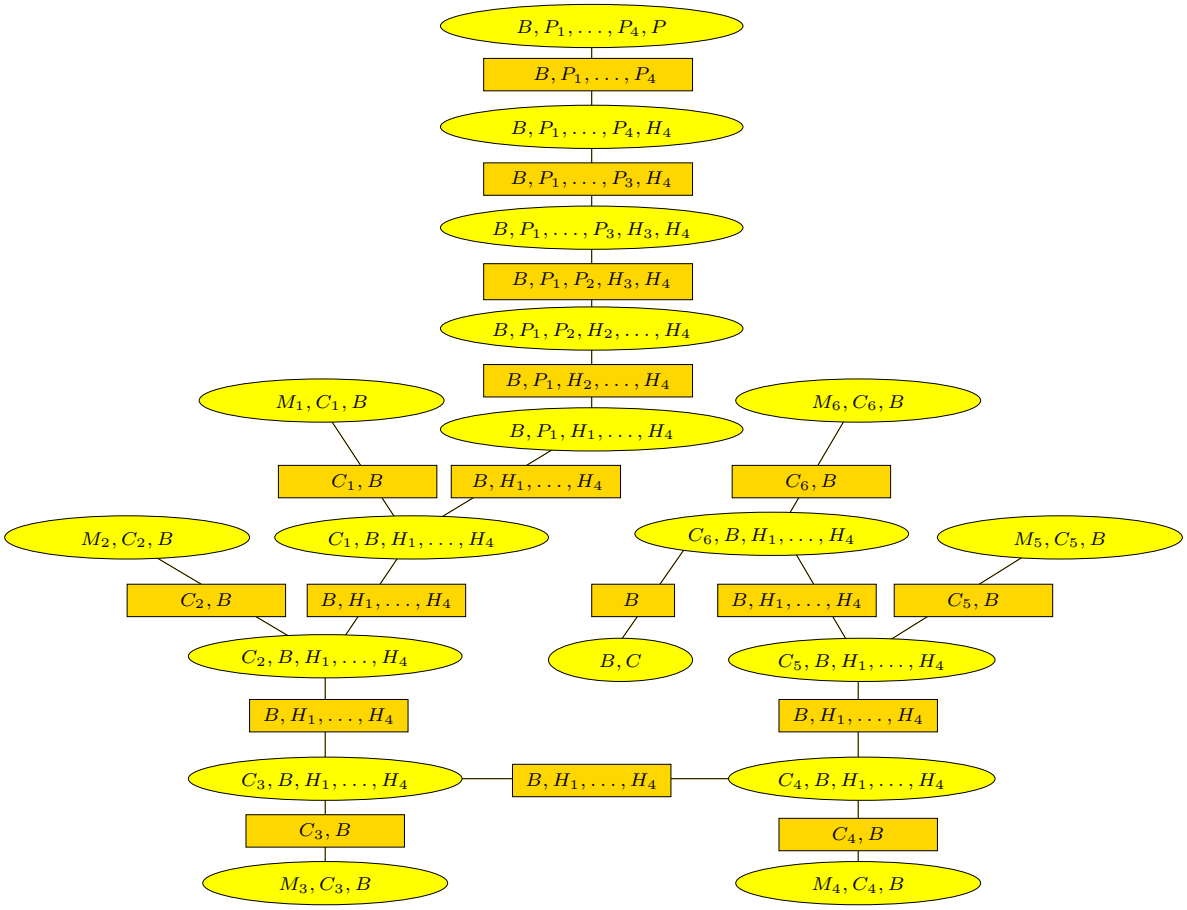

Figure 4: Junction tree of the transformed Bayesian network from Figure 3.

$h_1, \ldots, h_4$ of variables of the hidden code $H_1, \ldots, H_4$ get impossible, i.e. their probability is zero. We could eliminate all combination of hidden code with zero probability and just keep information about non-zero combinations. This makes the junction tree even smaller. Note that several nodes in the junction tree in Figure 4 contains all variables $H_1, \ldots, H_4$. The elimination of zero values means that instead of all 6^4 lists of probabilities for all combinations of values of other variables in the node we keep only those lists that corresponds to a non-zero combination of $H_1, \ldots, H_4$ values.

Before we select the next most informative guess the computation of $Q(H_1, \ldots, H_4 \mid \mathbf{t}, c', p')$ are done for all combinations of $t_1, \ldots, t_4, c', p'$. An open question is whether we can find a combination of $t_1, \ldots, t_4$ minimizing the expected entropy more efficiently than by evaluating all possible combinations of $t_1, \ldots, t_4$.

5 Conclusions

One advantage of the Bayesian network model is the natural visualization of the problem. The user creates the model of the problem and all computations are left to an automatic inference engine. However, we have observed that, in case of probabilistic Mastermind, the inference using the standard methods is not computationally feasible. It is necessary to exploit deterministic dependencies that are present in the model. We have shown that after the transformation using an auxiliary variable we get a tractable model.

Acknowledgements

The author was supported by the Grant Agency of the Czech Republic through grant number 201/02/1269.

References

[1] J. L. Bernier, C. Ilia Herraiz, J. J. Merelo, S. Olmeda, and A. Prieto. Solving mastermind using GA's and simulated annealing: a case of dynamic constraint optimization. In *Parallel Problem Solving from Nature IV*, number 1141 in Lecture Notes in Computer Science, pages 554–563. Springer Verlag, 1996.

[2] A. Bestavros and A. Belal. MasterMind a game of diagnosis strategies. *Bulletin of the Faculty of*

Engineering, Alexandria University, December 1986.

[3] Hugin Researcher 6.3. `http://www.hugin.com/`, 2003.

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

[5] F.V. Jensen, K.G. Olesen, and S.A.Andersen. An algebra of bayesian belief universes for knowledge-based systems. *Networks*, 20:637–659, 1990.

[6] D. E. Knuth. The computer as a Master Mind. *Journal of Recreational Mathematics*, 9:1–6, 1976–77.

[7] K. Koyama and T. W. Lai. An optimal Mastermind strategy. *Journal of Recreational Mathematics*, 25:251–256, 1993.

[8] T. Nelson. A brief history of the master mind board game. `http://www.tnelson.demon.co.uk/mastermind/history.html`.

[9] P. Savicky and J. Vomlel. Factorized representation of functional dependence. *(under review)*, 2004.

[10] P. Van Hentenryck. A constraint approach to mastermind in logic programming. *ACM SIGART Newsletter*, 103:31–35, January 1988.