An Aggregation and Disaggregation Procedure for the Maintenance of a Dynamic System under Partial Observations

Demet Özgür-Ünlüakın
Bahçeşehir University, Istanbul-Turkey
demet.unluakin@bahcesehir.edu.tr

Taner Bilgiç
Boğaziçi University, Istanbul-Turkey
taner@boun.edu.tr

Abstract

We study the maintenance of a dynamic system consisting of several components each of which age with a constant transition probability of failure. The state of the components are hidden. However, partial observations exist in each time period. The decision of whether to replace a component or to do nothing is to be made in each decision epoch. A hierarchical solution procedure is proposed to solve the problem. An aggregate model is developed by aggregating states and actions so that it can be solved with exact partially observable Markov decision process (POMDP) techniques. Then disaggregation is performed by simulating the process using a dynamic Bayesian network (DBN) and applying troubleshooting approaches in the decision epoch where replacement is planned in the aggregate policy.

1 Introduction

We consider the maintenance planning problem of a dynamic system whose status is not observable but is estimated through indirect signals. Such problems are common in space systems and hazardous material detection systems where the system is far away and cannot be directly observed. However, such systems can still be controlled remotely to perform diagnostic tests and repair actions.

The major difference of the problem we consider from other maintenance problems in the literature is the complex structure of the system due to several (possibly interacting) components which results in a huge state space.

Our approach to maintenance is akin to the decision-theoretic troubleshooting problems (DTTP) (Kalagnam and Henrion, 1988) in handling complex system structures. Complex system structure in DTTP is usually represented with Bayesian Networks (BNs) (Heckerman et al., 1995; Jensen et al., 2001) which encode the conditional probabilistic dependence relations of the components. We use the same kind of BN representation at each discrete decision epoch. In DTTP the task is to find a minimum-cost action plan. DTTP has always been studied as a static problem under the assumption that a system has an observable malfunction and then troubleshooting process starts. However, we study a dynamic problem, hence the objective is to minimize the total expected costs, comprised of replacement and observation costs for a finite planning horizon.

BNs have been used in reliability problems to represent the (complex) relations in the system (Torres-Toledano and Sucar, 1998; Bobbio et al., 2001; Langseth and Portinale, 2006). Recently there have also been studies using DBNs in reliability analysis (Welch and Thelen, 2000; Weber and Jouffe, 2003; Muller et al., 2004). A DBN is an extended BN which includes a temporal dimension. However all reliability studies with DBNs are *descriptive* (i.e.,the dynamic problem is represented with DBNs and the outcome of the analysis is how system reliability behaves in time). The impact of performing a repair at a specific time on this behavior is also reported in some of them. However optimization of maintenance activities (i.e., finding a minimum cost plan) is not considered which is one of the main motivations of this paper.

Therefore our approach is *prescriptive* as opposed to being descriptive.

The maintenance problem we consider can be modeled as a POMDP (Hauskrecht, 2000) in which the decision maker makes sequential decisions under partial information. POMDPs fall prey to the "curse of dimensionality" as their state space grows exponentially with the number of components in the system. The type of a POMDP model having several hidden variables is rarely studied due to its computational complexity. There are algorithms for computing optimal solutions to POMDPs. However these algorithms are applicable in practice only to relatively simple problems. Also some structural results have been presented in some specific machine maintenance applications (Ross, 1971; Rosenfield, 1976), but such results are not common for general POMDPs.

The rest of the paper is organized as follows: In Section 2, we define and represent the maintenance planning problem. In Section 3, we present the proposed solution. In Section 4, we give the experimental design and the results of computational study. In Section 5, we conclude and point to further research directions.

2 Problem Definition and Representation

2.1 Problem Definition

There is a system consisting of several components each of which age with a constant transition probability of failure. It is not possible to observe the system components, they are hidden. However the system gives signals at each decision epoch which may indicate some partial information about the state of the components. This is the only information one can get from the process. System components age with a constant rate constituting the dynamic behavior of the problem. It is possible to replace components in any period and when a component is replaced, a replacement cost is charged. On the other side, since observing a signal indicating faulty components in the next decision epoch is undesirable, it incurs another type of cost, which is called observation cost in this study. There is a trade off between replacing the components and observing undesirable signals. In each decision epoch, the decision maker can either prefer doing nothing or replacing only one of the components. The aim is to find a policy that minimizes expected total replacement and observation cost in a given horizon. The following assumptions are made:

(i) Every component has a constant transition probability of failure. (ii) All other conditional probability distributions are discrete. (iii) All components have two states ("w": working state, "nw": failure state). (iv) Components can only fail at the beginning of a time period. Once they fail they will be in state "nw" unless they are repaired. (v) Once a component is replaced in a period, its working state probability becomes 1 in the next period. (vi) Only one replacement can be done in a given epoch.

The first three assumptions are required for computational purposes as DBN tools usually work with discrete probabilities and states. The fourth and fifth assumptions are standard in reliability. The sixth assumption implies that there is a limit to replacements at each epoch (possibly due to a time or a budget constraint).

2.2 Problem Representation

The maintenance problem can be expressed as a POMDP with the following parameters:

I :	number of components in the process
i :	index for component
C_i:	set of states of component i, $i = 1, \cdots, I$
A :	set of actions
Θ :	set of observations
T_i':	set of transition probabilities of component i. T_i' :$C_i \times A \times C_i \to [0,1]$
O':	set of observation probabilities among component states and observations. O': $C_1 \times \cdots \times C_I \times \Theta \to [0,1]$
R :	reward function that assigns rewards to observations and actions. $R : A \times \Theta \to \mathbb{R}$

Part (only two epochs) of the influence diagram describing the problem is given in Figure 1 where c_{it}, a_t, o_t and r_t denote component state, action, observation and reward at time t.

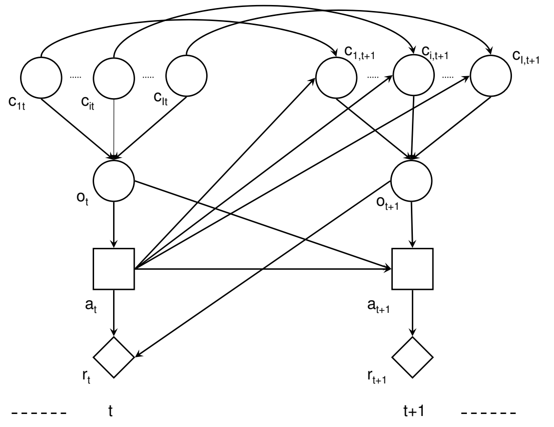

Figure 1: Part of the influence diagram describing the maintenance problem

The system gives two types of signals: g and r which stand for green and red respectively, $\Theta = \{g, r\}$. A green signal is preferred to a red signal since it implies a better condition of components. In each epoch, the decision maker can either prefer doing nothing or replacing one of the components. Hence $A = \{dn, rc_1, rc_2, \cdots, rc_I\}$, where dn denotes doing nothing and rc_i denotes replacing component i for $i = 1, 2, \cdots, I$. Table 1 shows the transition probabilities, T_i', for component i having two states, $C_i = \{w, nw\}$, given $a_t \in A\backslash\{rc_i\}$ and $a_t = rc_i$ respectively. Here, $a_t \in A\backslash\{rc_i\}$ means the decision maker takes an action other than replacing component i in decision epoch t. In this case, $P(c_{i,t+1} = w | c_{it} = w) = p_i$ where $0 < p_i < 1$ and $1 - p_i$ is the constant transition probability of failure for a working component i. When $a_t \in A\backslash\{rc_i\}$ and component i is non-working in period t, it will still be non-working in the subsequent period $t + 1$. So $P(c_{i,t+1} = nw | c_{it} = nw) = 1$. When $a_t = rc_i$, component i will be in its working state with probability one in the next period no matter whether it is working or not in the current period.

Table 1: Transition probability of component i

$a_t \in A_t\backslash\{rc_i\}$

	$c_{i,t+1}$	
c_{it}	w	nw
w	p_i	$1 - p_i$
nw	0	1

$a_t = rc_i$

	$c_{i,t+1}$	
c_{it}	w	nw
w	1	0
nw	1	0

3 Proposed Solution

The system under maintenance can be arbitrarily complex. There may be non-component nodes in the system which are not temporal and only transmits information from its parents to its children. After eliminating these nodes from the graph, we propose POMDPs as a solution to the maintenance problem with the reduced joint state space as the hidden process state space. This space grows exponentially with the number of component nodes in the graph. To overcome this difficulty we propose a hierarchical solution procedure where in the higher level we solve an aggregate model of the problem with an exact POMDP solver and in the lower level we disaggregate the aggregate solution using DBNs.

3.1 POMDP Formulation

The maintenance problem has a POMDP structure with the following exception: In a POMDP, there is usually a single variable defining the hidden process. However, in our problem the hidden process is more complex since it consists of several component nodes (Figure 1). One solution to this drawback is to merge all component nodes into a single mega node as in Figure 2. Here, the component nodes c_{1t},..c_{it},..,c_{It} are merged into the process node s_t. After the merge, the bold black arcs are added to the model, the grey arcs are deleted from the model and the rest of arcs are maintained as they are before the merge. Let S be the new process state space of s_t, T be the new set of transition probabilities of the process node, and O be the new set of observation probabilities. The following conversions should be done to formulate the problem as a POMDP. The new process state space S is the Cartesian product of the component state spaces and can be represented as $S = C_1 \times C_2 \times ... \times C_I$, where $|S| = \prod_{i=1}^{I} |C_i|$.

The new observation probabilities O can be constructed as follows:

$$P(o_t|s_t) \quad = \quad P(o_t|c_{1t}, c_{2t}, .., c_{It}), \qquad (1)$$

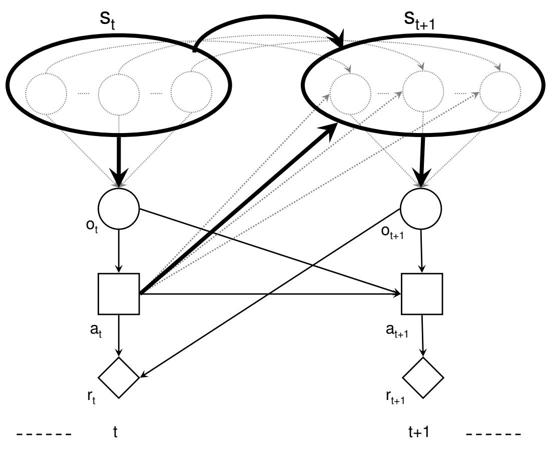

Figure 2: Part of the influence diagram after conversion to POMDP

which are in fact equal to the observation probabilities O' before merge. When components are independent as in Figure 2, the new transition probabilities T can be constructed from T'_i as in (2). For more general models where there are also dependencies among components, T can be constructed with (3), where $pa(c_{i,t+1})$ consists of all parents of $c_{i,t+1}$ including c_{it} and a_t.

$$P(s_{t+1}|s_t, a_t) = \prod_{i=1}^{I} P(c_{i,t+1}|c_{it}, a_t), \quad (2)$$

$$P(s_{t+1}|s_t, a_t) = \prod_{i=1}^{I} (c_{i,t+1}|pa(c_{i,t+1})). \quad (3)$$

3.2 Aggregate Model

We have formulated the maintenance problem as a POMDP with a hidden state space whose cardinality is $|S| = \prod_{i=1}^{I} |C_i|$. As i increases $|S|$ increases exponentially making the problem harder to solve. To overcome this difficulty, we can aggregate some states and form new aggregate states S_p. Let S^a be the new state space, built after aggregation, whose elements are S_p which are mutually exclusive and totally exhaustive sets (i.e., $\cup_{p=1}^{P} S_p = S$, $S_p \cap S_q = \emptyset$, $p \neq q\ p, q = 1..P$). Let P be the new cardinality of S^a, $|S^a| = P$. Let T^a and O^a be the new transition and observation probabilities of this aggregate model which are obtained from T and O respectively as follows:

$$P(s_{t+1} \in S_q|s_t \in S_p, a_t) =$$

$$\frac{\sum_{s_{t+1}\in S_q} \sum_{s_t \in S_p} P(s_{t+1}|s_t, a_t)}{|S_p|}, \quad (4)$$

$$P(o_t|s_t \in S_p) = \frac{\sum_{s_t \in S_p} P(o_t|s_t)}{|S_p|}, \quad (5)$$

where the cardinality $|S_p|$ is a normalizing factor. When process states are aggregated, it is more meaningful to aggregate the related actions. Let A^a denote the new action space, built after aggregation, whose elements are A_j which are mutually exclusive and totally exhaustive sets (i.e., $\cup_{j=1}^{J} A_j = A$, $A_j \cap A_l = \emptyset$, $j \neq l\ j, l = 1..J$). Let J be the new cardinality of A^a, $|A^a| = J$.

When actions are aggregated, it affects the transition probabilities and the reward function since they depend on actions performed in each decision epoch. Let T^{aa} and R^a be the new transition probabilities and the new reward function of this aggregate model where actions are also aggregated in addition to states. T^{aa} and R^a are obtained from the original data, T and R, respectively as follows:

$$P(s_{t+1} \in S_q|s_t \in S_p, a_t \in A_j) =$$

$$\frac{\sum_{a_t \in A_j} \sum_{s_{t+1}\in S_q} \sum_{s_t \in S_p} P(s_{t+1}|s_t, a_t)}{|A_j||S_p|}, \quad (6)$$

$$R(A_j, o) = \frac{\sum_{a \in A_j} R(a, o)}{|A_j|}, \quad (7)$$

where $|A_j||S_p|$ and $|A_j|$ are normalizing factors. In (6–7), actions are given equal probabilities to be performed during aggregation. However, when one prefers to replace some components more or less frequently than the others, giving weights to actions according to the desired frequency is more appropriate. Let W_a be the weight of action a. Let T_w^{aa} and R_w^a be the new transition probabilities and the new reward function of this weighted aggregate model where actions are also aggregated with respect to their weights in addition to the aggregation of states.

T_w^{aa} and R_w^a are obtained from the original data, T and R, respectively as follows:

$$P(s_{t+1} \in S_q | s_t \in S_p, a_t \in A_j) = \frac{\sum_{a_t \in A_j} \sum_{s_{t+1} \in S_q} \sum_{s_t \in S_p} W_a P(s_{t+1}|s_t, a_t)}{\sum_{a_t \in A_j} W_a |S_p|}, \quad (8)$$

$$R(A_j, o) = \frac{\sum_{a \in A_j} W_a R(a, o)}{\sum_{a \in A_j} W_a}. \quad (9)$$

Although some detailed information is lost after aggregation, the aggregate model has fewer process states and action states than the original model, which makes it easier to solve.

3.3 Disaggregation

The aggregate solution given by an exact POMDP solver is disaggregated to obtain an expected maintenance cost of the original problem. Disaggregation is performed by simulating the process with a DBN tool (Bayesian Network Toolbox) and applying troubleshooting approaches in the decision epoch, where replacement is planned in the aggregate policy, to obtain which component to replace.

3.3.1 DBN Formulation

The original problem is simulated with a DBN given in Figure 3, where action a_t is represented as a probabilistic node such that $a_t \in A$, $A = \{dn, rc_1, rc_2, ..., rc_I\}$. Solid arcs represent the causal relations between the components and the observation node. They constitute the conditional probabilities $P(o_t|c_{1t}..c_{It})$ for all t. The dashed arcs represent temporal relations of the components and actions between two consecutive time periods. They constitute the conditional probabilities $P(c_{i,t+1}|c_{it}, a_t)$. Temporal relations are the transition probabilities of components due to aging and replacement which are given in Table 1.

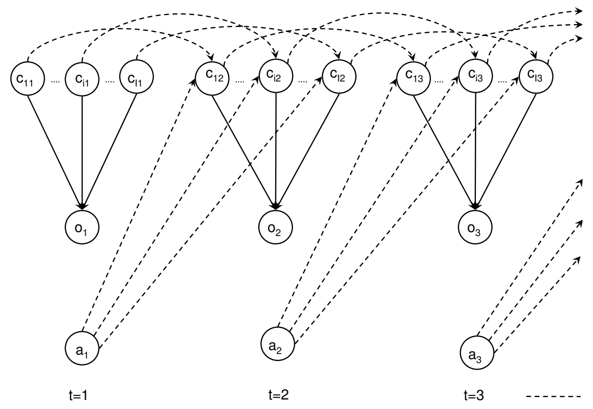
c_{11} .... c_{i1} .... c_{I1}
c_{12} .... c_{i2} .... c_{I2}
c_{13} .... c_{i3} .... c_{I3}
o_1
o_2
o_3
a_1
a_2
a_3
t=1
t=2
t=3 -------

Figure 3: DBN representation of the disaggregate problem

3.3.2 Disaggregation Procedure

The disaggregation procedure takes the optimal policy of the aggregate POMDP model as input. This policy is an aggregate policy in terms of aggregate actions. It does not specify which component to replace when it decides to do a replacement. This is done by the disaggregation procedure. It is performed by simulating the original problem (before merge) with a DBN represented in Figure 3. Let ε be the evidence set containing the total evidence gathered so far. It is accumulated by two means in every period: one is the selected disaggregated action of the current period and the other is the sampled observation of the next period. In each period, if a replacement is required by the aggregate policy, the component which has the highest efficiency is selected. This constitutes the current disaggregated action. The efficiency measure is defined as follows:

$$ef_{it} = \frac{P(c_{i,t+1} = nw| \, \varepsilon \cup \{o_{t+1} = r\})}{\pi_i}, \quad (10)$$

where the numerator is the probability of component i being nonworking given the up-to-date evidence and also the condition that a red signal is observed in the next period although this may not be the case. Appending the latter one to the evidence indicates the component which explains observing a red signal in the next period. The denominator, π_i, is the replacement cost of component i and it makes the efficiency measure to trade off component's nonworking probability to its replacement cost. The component with the highest efficiency is selected to be replaced.

4 Computational Study

The design structure given in Table 2 is used in the computational study. Four factors are determined in the experimental design. Two of them are related with costs and the other two are related with probabilities in the problem. The values of each factor are seen in the related column of the table. When observation probabilities are different, probability of observing green differs as number of faults differs for in between states (other than the all working and all nonworking states). As an alternative case, invariant observation probabilities mean probability of observing a green signal is same for all intermediate states and it is the average of the observation probabilities of the different case. Each combination of factor values leads to a total of 32 data sets. After elimination of four symmetric data, we have a total of 28 data sets.

Table 2: Experimental design

Transition Working Probability	Replacement Cost	Red Cost	Observation Probability
Increasing Constant-low Constant-high Decreasing	Increasing Constant	High Low	Different Indifferent

For a four component dynamic system, when components are merged into a single process node, we have a total of $2^4 = 16$ states and 5 actions. We try to solve the problem with an exact POMDP solver, but it cannot be solved exactly. Then states are aggregated into three states such that S_1 is the all working state, S_3 is the all nonworking state, and S_2 is the in between state (at least one working and one nonworking component exist). Actions are aggregated into two actions such that A_1 is doing nothing and A_2 is replacing one of the components. Initially no weights are given to the actions in A_2 while aggregating. Observation and transition probabilities, and reward function are aggregated accordingly as in Equations 5, 6 and 7 respectively.

After aggregating the problem into three compound states and two compound actions, it can be solved with an exact POMDP solver (pomdp-solve) implementing the Witness algorithm (Cassandra et al., 1994) for a finite horizon of 100 periods. All data sets are also solved with an approximate POMDP solver, ZMDP software package (Smith, 2007), for a finite horizon of 100 periods which finds upper bounds for the cost function and gives POMDP policies as an output. The ZMDP policy file is also simulated with the DBN in Figure 3 to achieve a complete solution as in the case of the disaggregate solution. Both disaggregation and ZMDP simulation are replicated 50 times and the average cost of these is reported with their standard deviations in Table 3.

We observe that, in data sets d01-d04, d08, d10, d11, d15-d18, d22, d24 and d25, only the first component is replaced in all replacement times and replications. So, a better aggregation can be possible by giving more weights to the replacement actions of components which are replaced more frequently and giving no weights to the replacement actions whose components are not even replaced. So we re-aggregate the data sets having unequal transition probabilities or unequal replacement costs by giving the average number of replacements of each component as its weight to the corresponding replacement action. Data sets d06, d07, d13, d14, d20, d21, d27 and d28 have equal transition probabilities and replacement costs, so giving unequal weights do not lead to different aggregations other than the ones already constituted. The computational results of the weighted aggregation are also tabulated in Table 3.

The results show that average cost of most of the re-aggregated data sets improve (d01-d04, d08-d11, d15, d16, d22-d25) or remain the same (d05, d12, d19 and d26) except data sets d17 and d18 which have performed already well with the equal weighted aggregation. One can make new aggregations of replacement actions with the new average number of replacements of components. This goes on until a cycle of aggregate policies is determined, which means no new solutions will be available to the decision maker. However, there is no guarantee that disaggregation results will improve every time a new aggregation is performed as in the case of data sets d17 and d18.

Table 3: Computational results

Set	Equal *Disagg*	*(std)*	Weighted *Disagg*	*(std)*	*ZMDP* *Sim.Avg*	*(std)*	t-test p-value
d01	1497.3	(131.9)	**1220.8**	(103.6)	854.7	(129.2)	.0000 *
d02	1500.8	(72.27)	**1308**	(104.3)	1295.2	(85.96	.5046
d03	1068.4	(163.8)	**717.5**	(127)	605.52	(112.16)	.0000 *
d04	1218.8	(84.19)	**904.2**	(104.4)	915.86	(110.26)	.5871
d05	**1308.1**	(123.4)	1308.1	(123.4)	908.6	(96.46)	.0000 *
d06	**1479.4**	(85.93)	-	-	1495.2	(96.77)	.3901
d07	**955.4**	(130.8)	-	-	562.30	(81.95)	.0000 *
d08	414.6	(41.79)	**375.1**	(30.73)	348.74	(42.23)	.0006 *
d09	496.2	(4.11)	**398.4**	(22.22)	396.6	(24.65)	.7021
d10	296.6	(36.54)	**269.7**	(44.09)	232.6	(40.46)	.0000 *
d11	324.9	(25.54)	**299.4**	(30.49)	292.38	(33.35)	.2747
d12	**400.8**	(43.16)	400.8	(43.16)	476.2	(.57)	.0000 +
d13	**554.4**	(23.12)	-	-	495.6	(4.70)	.0000 *
d14	**280.3**	(31.87)	-	-	227.7	(40.17)	.0000 *
d15	1336.2	(126.1)	**886.8**	(95.33)	851.3	(102.4)	.0761
d16	1259.7	(87.37)	**878.4**	(90.25)	898.8	(125.3)	.3525
d17	**853.26**	(105.8)	924.76	(125.9)	764.66	(115.60)	.0001 *
d18	**873.32**	(86.54)	1150.2	(105.87)	892.7	(97.9)	.2958
d19	**1059.8**	(123.2)	1059.8	(123.2)	1120.6	(102.2)	.0085 +
d20	**1298.6**	(103.4)	-	-	1306.4	(91.65)	.6906
d21	**853.9**	(119.4)	-	-	690.7	(105.09)	.0000 *
d22	381.38	(44.52)	**315.28**	(26.8)	377.1	(36.5)	.0000 +
d23	496.2	(4.58)	**295.7**	(21.75)	303.7	(21.3)	.0661
d24	238.32	(31.17)	**221.3**	(25.24)	242.12	(29.18)	.0002 +
d25	245.16	(25.35)	**236.8**	(23.82)	234.94	(20.85)	.6788
d26	**341**	(47.78)	341	(47.78)	274.4	(36.6)	.0000 *
d27	**501.1**	(23.28)	-	-	494.9	(5.1)	.0689
d28	**242.3**	(33.06)	-	-	236.7	(25.88)	.3479

It is of interest to analyze the performance of the two procedures in each data set in order to understand what type of data sets our procedure is successful at. So, for each data set, we perform hypothesis test for two samples and use two-sided t-test to test whether the means of simulated ZMDP cost and the best disaggregation cost (with bold number) are equal or not in each data set. The resulting p-value is given in the table for each data set. When significance level is taken as 0.05, data sets with p-value less than 0.05 are marked in the table with * or + indicating that the simulated ZMDP cost is significantly less than the disaggregation cost or vice versa, respectively. There are 15 data sets out of 28 where the two methods significantly differ in terms of cost values. 11 of these data sets belong to the case where simulated ZMDP cost is significantly lower than disaggregate cost whereas 4 of them belong to the case where disaggregate cost is significantly lower. If we look at the distribution of 11 data sets where ZMDP cost is significantly lower than disaggregate cost, we will see that 8 of them lie within the data sets with different observation probabilities for the in between states (d01-d14). Alternatively only 1 out of 4 data sets where disaggregate cost is significantly lower than simulated ZMDP cost, lies in d01-d14. These results give insight that our solution procedure is more successful at the data sets whose observation probabilities of the aggregated states are invariant.

One may be interested in obtaining the overall performance of the proposed procedure by using one-sided paired t-test. Experimental results already show that there are plenty of instances where our procedure is outperformed by the simulated ZMDP. However simulated ZMDP has two drawbacks: First, in order to run ZMDP in a finite horizon, time information is added explicitly into the state information. So, when the number of components increase, state space and hence size of the ZMDP input file grows exponentially. Second, when ZMDP is run, it gives an upper bound for the cost function. However to obtain full solutions as in the case of our procedure, we also simulate the ZMDP output file. This requires more computational time than simulating the aggregate model of our procedure.

5 Conclusion

We tackle the maintenance problem of a complex system where system components are partially observable via indirect signals, and present a hierarchical solution procedure to solve it. The complex POMDP problem is aggregated in terms of states and actions such that it can be solved exactly and the optimal policy is implemented on the system by simulating it with DBNs. Our solutions are compared with the solutions of the approximate POMDP solver, which uses full information state space, on 28 data sets. The results show that when observation probabilities are invariant among the states aggregated into the same compound state, our procedure performs better.

In the aggregation, we prefer to aggregate the states into three compound states which are all working, all nonworking and in between states; and the actions into two compound actions which are doing nothing and doing a replacement. A POMDP with three states and two actions can always be solved exactly. However, significant probabilistic information can be lost. The disaggregation result can be at most as good as the quality of the aggregate policy. The action aggregation can be improved by giving weights to replacement actions while the state aggregation is harder to improve. The average number of replacements of components obtained from the disaggregation solution can be used as weights, however determining the appropriate weights for action aggregation from the parameters of the problem will be a better way since this will reduce the number of aggregations and hence the effort. This can be a future study.

References

A. Bobbio, L. Portinale, M. Minichino, and E. Ciancamerla. 2001. Improving the analysis of dependable systems by mapping fault trees into Bayesian networks. *Reliability Engineering and System Safety*, 71:249–260.

Anthony R. Cassandra, Leslie P. Kaelband, and Michael L. Littman. 1994. Acting optimally in partially observable stochastic domains. Technical Report CS-94-20, Brown University, Providence, Rhode Island.

Milos Hauskrecht. 2000. Value function approximations for partially observable Markov decision processes. *Journal of Artificial Intelligence Research*, 13:33–94.

David Heckerman, John S. Breese, and Koos Rommelse. 1995. Decision-theoretic troubleshooting. *Communications of the ACM*, 38(3):49–57.

Finn V. Jensen, Uffe Kjærulff, Brian Kristiansen, Helge Langseth, Claus Skaanning, Jiri Vomlel, and Marta Vomlelováá. 2001. The SACSO methodology for troubleshooting complex systems. *Artificial Intelligence for Engineering, Design, Analysis and Manufacturing*, 15(4):321–333.

Jayant Kalagnam and Max Henrion. 1988. A comparison of decision analysis and expert rules for sequential analysis. In *Proceedings of 4th Conference on Uncertainty in Artificial Intelligence*, pages 271–281.

Helge Langseth and Luigi Portinale. 2006. Bayesian networks in reliability. *Reliability Engineering and System Safety*, 92:92–108.

Alexandre Muller, Philippe Weber, and A. Ben Salem. 2004. Process model-based dynamic Bayesian networks for prognostic. In *Proceedings of 4th International Conference on Intelligent Systems Design and Applications*, pages 849–854.

Donald Rosenfield. 1976. Markovian deterioration with uncertain information. *Operations Research*, 24(1):141–155.

Sheldon M. Ross. 1971. Quality control under Markovian deterioration. *Management Science*, 17(9):587–596.

Trey Smith. 2007. ZMDP software for POMDP and MDP planning. http://www.cs.cmu.edu/ trey/zmdp/README.

José Gerardo Torres-Toledano and Luis Enrique Sucar. 1998. Bayesian networks for reliability analysis of complex systems. In *Proceedings of 6th Ibero-American Conference on AI: Progress in Artificial Intelligence*, pages 195–206.

Philippe Weber and Lionel Jouffe. 2003. Reliability modeling with dynamic Bayesian networks. In *Proceedings of 5th IFAC Symposium SAFEPROCESS'03*, pages 57–62.

Robert L. Welch and Travis V. Thelen. 2000. Dynamic reliability analysis in an operational context: the Bayesian network perspective. In *Proceedings of Dynamic Reliability: Future Directions*, pages 277–307.