Applications of HUGIN to Diagnosis and Control of Autonomous Vehicles

Anders L. Madsen[1] and Uffe B. Kjærulff[2]

[1] HUGIN Expert A/S, Gasværksvej 5, DK-9000, Aalborg, Denmark
[2] Aalborg University, Department of Computer Science, Fredrik Bajers Vej 7E, DK-9220, Aalborg, Denmark

Abstract. We present an application of HUGIN to solve problems related to diagnosis and control of autonomous vehicles. The application is based on a distributed architecture supporting diagnosis and control of autonomous units. The purpose of the architecture is to assist the operator or piloting system in managing fault detection, risk assessment, and recovery plans under uncertainty. To handle uncertainty, we focus on the use of probabilistic graphical models (PGMs) as implemented in the HUGIN tool.

We describe the application of PGMs to three problems of diagnosis and control of autonomous vehicles. Based on the HUGIN tool, limited memory influence diagrams (LIMIDs) are used to represent and solve complex problems of diagnosis and control of autonomous ground and underwater vehicles. In particular, we describe how battery monitoring and control problems related to an underwater and a ground vehicle are solved and how to solve the problem of assessing the quality of a sonar image related to an underwater vehicle.

1 Introduction

The HUGIN tool [1,5,13] supports construction and deployment of complex statistical models known as probabilistic graphical models (PGMs) for reasoning and decision making under uncertainty. We describe how a particular kind of PGMs, known as limited memory influence diagrams (LIMIDs), have been applied to solve complex reasoning and decision making problems related to autonomous ground vehicles and autonomous underwater vehicles.

The ADVOCATE project (acronym for *Advanced On-board Diagnosis and Control of Autonomous Systems*), which was formed in 1997 under the IST program of the European Commission, had as one of its objectives to increase the performance of unmanned underwater vehicles in terms of availability, efficiency, and reliability of the systems and in terms of safety for the systems themselves as well as for their environments. The aim of the follow-up project, ADVOCATE II, was partly to design and develop a general-purpose software architecture for Autonomous Underwater Vehicles (AUVs) and Autonomous Ground Vehicles (AGVs) and partly to develop software-based systems to increase the degree of automation, efficiency, and reliability of the vehicles. The interest of such a concept from the market point of view was demonstrated by a market study.

The latter objective was reached by adding artificial intelligence (AI) into existing and new control software to diagnose and recover from any dysfunction or failure situation of the system. To improve the management of uncertainty in AUVs and AGVs it was decided that the ADVOCATE II architecture should allow for easy incorporation and merging of different AI techniques in a highly modular fashion.

Three end-user partners were involved in the ADVOCATE II project: University of Alcalá designs piloting modules for AGVs for surveillance applications, Ifremer designs AUVs for scientific applications, and ATLAS Elektronik designs AUVs and semi-AUVs for industrial applications. Each end-user partner presented diagnosis and control problems related to a single vehicle. Several such problems were presented for each vehicle involving different kinds of dysfunctions and failures:

- Thruster or motor failure diagnosis and recovery in case of abnormal behaviour of the vehicle due to thrusters or motors.
- Sensor malfunction diagnosis and recovery on sensor state in order to account for failure situations in case of corrupt sensor signals due, for instance, to noise.
- Power consumption diagnosis and recovery monitoring the level of remaining energy in a vehicle battery in order to avoid mission abortion.
- Motion diagnosis and recovery monitoring and assessing the motion characteristics of a vehicle.

Three different AI techniques have been applied for solving these diagnosis and control problems, namely probabilistic graphical models (PGMs), neuro-symbolic systems (NSSs), and fuzzy logic (FL). In this chapter, we focus on the development and application of PGMs for a selected set of the problems. We discuss how LIMIDs have been used to represent and solve complex problems of diagnosis and control of AGVs and AUVs. In particular, we describe how battery monitoring and control problems related to an AUV and an AGV are solved and how a sonar image quality assessment problem related to an AUV is solved.

In Sect. 2, we briefly present the HUGIN tool used for constructing and executing the LIMIDs. Section 3 introduces the problem domain of semi-autonomous ground and underwater vehicles. The ADVOCATE II communication architecture is described in Sect. 4. Section 5 presents some preliminaries and notation on the LIMID representation used to model and solve the diagnosis and control problems. The knowledge extraction process and method developed as part of the ADVOCATE II project is described in Sect. 6. Section 7 describes the models developed to solve the diagnosis and control problems, while in Sect. 8 we discuss how the developed LIMIDs are solved. Section 9 describes model integration and validation, and presents the results of real world trials. Finally, Sect. 10 ends the chapter with a discussion of our work.

2 The HUGIN Tool

The HUGIN tool [1,5,13] is a general purpose tool for constructing and deploying probabilistic graphical models (PGMs) such as Bayesian networks and influence diagrams.

In the HUGIN tool, inference in PGMs is performed through message passing in a secondary computational structure known as a junction tree [7]. The junction tree is constructed from the PGM through processes known as moralization and triangulation [12]. The nodes of a junction tree are sometimes referred to as *cliques*. To each clique (containing a subset of the variables of the PGM) is associated tables representing joint probability and utility functions over variables of the clique. The messages passed between cliques represent joint probability and utility functions over variables common to both the sending and the receiving clique.

The HUGIN tool consists of a graphical user interface (HUGIN Graphical User Interface) and an inference engine (HUGIN Decision Engine). The HUGIN Decision Engine has Application Programming Interfaces (APIs) for four different programming languages: C, C++, Java, and Visual Basic for Applications.

The core functionality of the HUGIN Decision Engine is implemented in the C programming language according to the ANSI C standard. This makes the HUGIN Decision Engine highly efficient and portable. Interfaces for C++, Java and Visual Basic for Applications are constructed on top of the core implementation. The HUGIN Graphical User Interface is implemented in Java, which makes it highly portable.

The HUGIN Decision Engine has been deployed on a large number of different platforms ranging from PDAs to multiprocessor mainframes.

3 Problem Domain

The transition of autonomous vehicles from experimental research tools to real applications increases the need for reliable and safe performance of the vehicles. This includes detection, avoidance, and recovery from any dysfunction.

Both the AGVs and the AUVs considered in the ADVOCATE II project are supplied with energy from batteries. This poses the problem of monitoring the remaining energy level of the battery and providing diagnosis and recovery actions in order to manage the mission parameters related to the energy consumption and to avoid unnecessary mission aborts. One AUV is equipped with an advanced object detection and avoidance system. This system works well in situations where obstacles can be detected by sonar. Hence, it is important to assess the quality of the sonar image and to suggest recovery actions to improve the sonar image quality or to suggest reductions in speed in case of poor image quality.

The ADVOCATE II consortium consisted of eight partners from five different European countries. Each partner served different roles in the consortium. The roles were robot manufactures and end-users, technology providers, marketing and communication specialists, and project coordinator. HUGIN Expert A/S served the role as technology provider and developer of intelligent modules.

3.1 The DeepC AUV

ATLAS Elektronik is developing a new type of underwater vehicle operating with autonomous mission durations of up to 60 hours. This vehicle is referred to as DeepC (see Fig. 1).

The long mission durations impose the need for advanced AI techniques to detect, avoid, and recover from any dysfunction. All end-users and ATLAS Elektronik in particular were faced with problems, which could not easily be solved by existing systems. The current approach to handle mission faults is to abort the mission, which is, however, very expensive.

Fig. 1. The DeepC underwater vehicle

As a fully autonomous system, the DeepC vehicle has to rely on its sensors to survive operationally. The DeepC is equipped with an advanced object detection and avoidance system. The object detection system consists of a mechanically scanning, forward looking sonar and its control electronics. This system works well when the sonar image is of sufficient quality. The problem considered is to construct a model for assessing the sonar image quality and for suggesting actions to avoid object collisions.

3.2 The VORTEX AUV

Ifremer has developed the remotely operated underwater vehicle VORTEX (see Fig. 2), which for our purpose is functionally considered as an AUV, as it allows programming of autonomous complex missions.

The motivation for equipping the vehicle with AI technology is much the same as for the DeepC, including optimization of the mission plan, diagnosis

Fig. 2. The VORTEX underwater vehicle

of abnormalities, recovery planning in case of abnormalities, avoidance of mission abortion (which is very expensive), and avoidance of vehicle loss.

3.3 The BART AGV

To put the ADVOCATE II concept into practice also for autonomous ground vehicles, University of Alcalá was deploying a telesurveillance application using the BART AGV (see Fig. 3). Two independent actuators powered by an on-board battery drive the vehicle.

Fig. 3. The BART autonomous ground vehicle

In order to increase the probability of mission success in case of energy problems or in case the vehicle gets stalled, the overall mission of the vehicle as well as other navigational issues can be managed using ADVOCATE II technology. The ADVOCATE II system provides the AGV with intelligent diagnosis capabilities and ability to recommend optimal recovery actions resulting in more reliable and safer operations. The diagnosis and recovery capabilities are concerned with aspects of navigation, energy system, sensors, actuators, etc.

4 ADVOCATE II Architecture

The ADVOCATE II architecture is a distributed architecture based on a generic communication protocol. The architecture is modular and easy to evolve and adapt to future piloting systems.

The purpose of the architecture is to assist the operator or piloting system in managing fault detection, risk assessment, and recovery plans under uncertainty. The generic communication protocol is based on SOAP/XML technology implementing HTTP for communication between different types of modules (see Fig. 4).

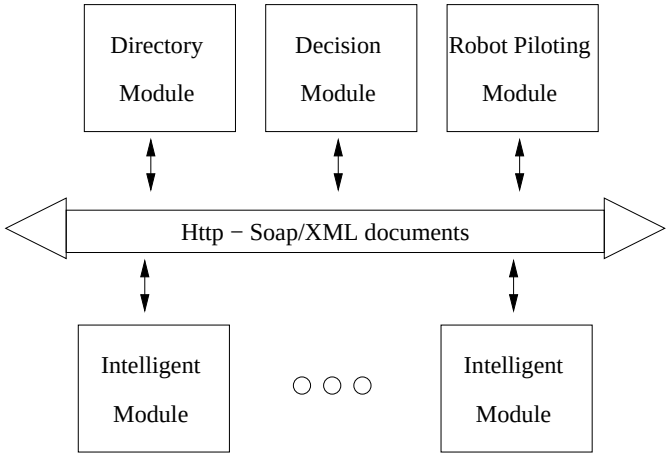

Fig. 4. The communication architecture

The architecture is generic, open, and modular consisting of a set of interacting modules including a decision module and a set of intelligent modules. The decision module communicates with the intelligent modules to request and obtain diagnosis and recovery action proposals based on data obtained from the robot piloting module.

The architecture is designed to allow easy integration of different AI techniques into preexisting systems. The decision to support the simultaneous use of multiple AI techniques was made to allow these techniques to collaborate on the task of reasoning and making decisions under uncertainty. This raises the question of how to most efficiently integrate different AI techniques into new and existing systems. We have found that this is most efficiently done through an open and generic architecture with a sophisticated communication interface.

The architecture consists of four different types of modules.

4.1 Robot Piloting Module

The robot piloting module manages mission plans and communicates directly with the sensors and actuators of the vehicle. This module also implements recovery plans received from the decision module into actions on the vehicle.

Each end-user partner of the ADVOCATE II project was responsible for the piloting module corresponding to the end-user vehicle.

4.2 Decision Module

The decision module manages the diagnosis and recovery action process. This includes integration of information provided by different intelligent modules, user validation of diagnosis and recovery actions when required by the system, and translation of recovery actions into recovery plans.

The decision module communicates with the intelligent modules receiving diagnoses and recovery actions, the robot piloting module, and the user.

4.3 Intelligent Module

The role of an intelligent module is to provide possible diagnoses, suggestions for recovery actions, or both. An intelligent module encapsulates a knowledge base to a specific problem domain and an inference engine.

A diagnosis on an operational vehicle corresponds to identification of system state while a recovery action corresponds to performing a sequence of actions on the vehicle (e.g., to avoid collision or to recover from any dysfunction).

The intelligent module communicates with the robot vehicle piloting module and the decision module. The robot vehicle piloting module supplies the intelligent module with data. These data are used in conjunction with the knowledge base to generate diagnoses and recovery actions. The diagnosis or recovery action is communicated to the decision module.

There may be multiple intelligent modules connected to the ADVOCATE II architecture. Multiple intelligent modules may consider the same or different problems related to the vehicle. Each intelligent module implements the communication protocol defined for the ADVOCATE II architecture.

In each application considered, the HUGIN Decision Engine is used for both reasoning and decision making under uncertainty, i.e., both to make a diagnosis and to generate recovery actions.

4.4 Directory Module

The architecture is organized around the directory module. The directory module is the central point of communication in the sense that it maintains a list of registered and on-line modules.

5 Limited Memory Influence Diagram

Limited memory influence diagrams are used to represent and solve the selected set of diagnosis and recovery action problems. An influence diagram [3] is a compact and intuitive probabilistic graphical model for reasoning about decision making under uncertainty. It is a graphical representation of a decision problem involving a sequence of interleaved decisions and observations. In essence, an influence diagram is a Bayesian network [2,6,14,15] augmented with decision variables and preference (or utility) functions.

An influence diagram $\mathcal{N} = (\mathcal{X}, \mathcal{G}, \mathcal{P}, \mathcal{U})$ over random variables $\mathcal{X}_C$ and decision variables $\mathcal{X}_D$ such that $\mathcal{X} = \mathcal{X}_C \cup \mathcal{X}_D$ consists of an acyclic, directed graph $\mathcal{G} = (V, E)$ over nodes V, connected by directed links $E \subseteq V \times V$, a set of conditional probability distributions $\mathcal{P}$, and a set of utility functions $\mathcal{U}$. The nodes V of $\mathcal{G}$ represent the random variables, decision variables, and utility functions of $\mathcal{N}$.

Each decision variable, D, represents a specific point in time under the model of the problem domain where the decision maker has to make a decision. The decision options or alternatives are the finite set of states $(d_1, \ldots, d_n)$ of D where n is the size of the state space of D. The usefulness of a decision option d_i is measured by local utility functions associated with D or one of its descendants in $\mathcal{G}$.

Mathematically, an influence diagram is a compact representation of a joint expected utility (EU) function:

$$\mathrm{EU}(V) = \prod_{X \in \mathcal{X}_C} P(X|\pi(X)) \sum_{u \in \mathcal{U}} u,$$

where $\pi(X)$ denotes the immediate predecessors (or parents) of variable X in $\mathcal{N}$.

To solve an influence diagram $\mathcal{N}$ is to determine an optimal strategy for the decision maker to follow. The strategy consists of one decision policy δ_D for each decision variable $D \in \mathcal{X}_D$. A policy δ_D is a mapping from the requisite past of D (i.e., past observations that may impact the choice of decision option for D) to the state space of $D \in \mathcal{X}_D$.

In the graphical representation of an influence diagram, random variables are represented as ovals, decision variables as rectangles, and utility functions as diamond-shaped nodes. A link into a node representing a random variable represents a probabilistic dependence relation, a link into a utility node identifies a domain variable of the corresponding utility function, and a link into a node representing a decision variable specifies that the parent is observed prior to the decision is made. Links into decision nodes are referred to as informational links.

An influence diagram supports the representation and solution of sequential decision problems under the no-forgetting assumption (i.e., assuming perfect recall of all observations and decisions made in the past that are influential in a given decision situation). A LIMID [11] is an influence diagram

relaxing the no-forgetting assumption to a limited memory assumption. This implies that all information available to the decision maker must be specified using informational links for each decision. This is contrary to an influence diagram where some informational links may be implicitly assumed present.

Figure 5 shows a LIMID representation of a simple decision problem involving two decisions D_1 and D_2.

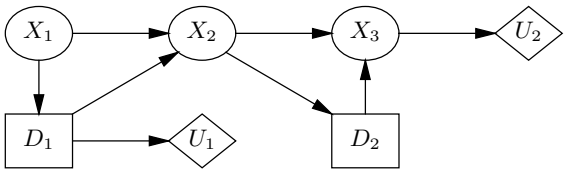

Fig. 5. A decision problem with two decisions

If the graph in Fig. 5 is interpreted as an influence diagram, then the domain of the decision policy for D_2 will consist of X_1, D_1, and X_2 (due to the no-forgetting assumption). If, on the other hand, the graph in Fig. 5 is interpreted as a LIMID, then the domain of the decision policy for D_2 will consist only of X_2. In a LIMID, all informational links are shown explicitly.

In the HUGIN tool, an influence diagram is solved by message passing in a so-called strong junction tree [4]. A LIMID, on the other hand, is solved by message passing in an ordinary junction tree. The difference in computational complexity between a strong junction tree and an ordinary junction tree can be tremendous [11].

An OO LIMID is an extension of a LIMID with support for object-oriented constructions [10], considering a LIMID as a class of which instances can exist in several other classes (LIMIDs). Thus, in addition to the elements of a LIMID, an OO LIMID contains instance nodes. An instance node represents an instantiation (or realization) of one LIMID class within another LIMID class following the object-oriented paradigm. In graphical representations of LIMIDs, instance nodes are represented using box-shaped nodes with rounded corners. The interface of a class is its input and output variables (indicated as nodes with a gray outer part, where input nodes have a dashed black border); see Fig. 9 for an example.

6 Knowledge Extraction Methodology

Unfortunately, the construction of a PGM can be a labour intensive task with respect to both knowledge acquisition and formulation. LIMIDs are not exceptional in this respect. The knowledge acquisition and formulation process associated with building the three LIMID models in the ADVOCATE II project involved knowledge engineers and domain experts located in four different countries. The knowledge engineers and domain experts had limited

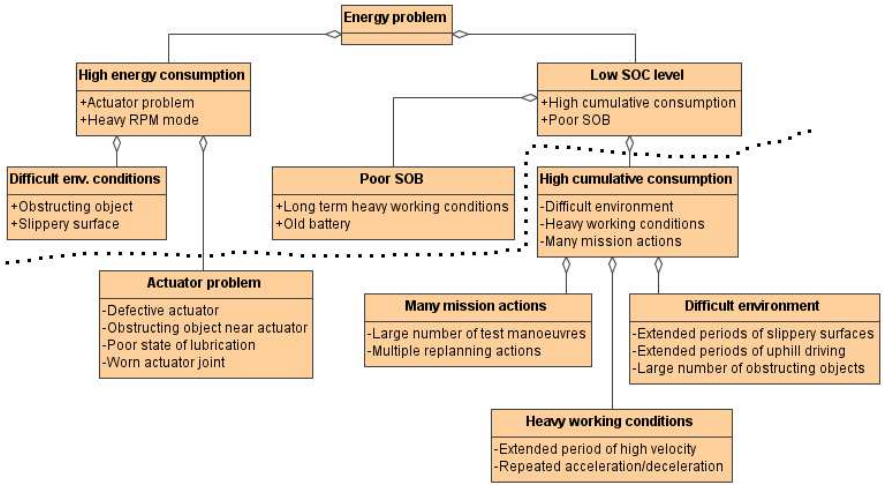

Fig. 6. Cause hierarchy for the BART AGV energy problem

possibilities for face-to-face meetings and the domain experts had limited knowledge of LIMIDs. Therefore, a knowledge acquisition scheme had to be developed that did not rely on familiarity with terminology of probabilistic graphical models and direct contact with the knowledge engineers.

The scheme is based on building a problem hierarchy for an overall problem. The problems (or causes) of the hierarchy relate to the states of the different parts of a vehicle and its environment.

Figure 6 shows such a cause hierarchy related to the energy problem of the BART AGV. The causes of the hierarchy are grouped into causes that qualify as satisfactory explanations of the overall problem and causes that do not. The first group of causes are referred to as *permissible diagnoses*. The subset of these that can actually be identified based on available information are referred to as *possible diagnoses*. Possible diagnoses are marked with a "+" in Fig. 6, and permissible diagnoses that are not possible are marked with a "−".

The cause hierarchy acts as a road-map for describing the relevant diagnostic information and the possible recovery actions. A cause of a sub-tree of the cause hierarchy that does not contain any possible diagnoses is unlikely to provide relevant diagnostic information or error recovery information. Thus, if there are no observable manifestations of the cause strong enough to identify a possible diagnosis for the cause, we need not worry about it when eliciting the diagnostic and error recovery information. In particular, none of the causes below the dotted line in Fig. 6 contain any possible diagnoses. The domain expert provides the relevant diagnostic information and the recovery actions in matrix form with one row for each cause "above the dotted line"

and one column for each kind of diagnostic information (i.e., background information and symptoms) and one column for possible recovery actions.

The qualitative knowledge elicited following such a scheme provides a sufficient basis for a knowledge engineer to construct the structure of a PGM, on the basis of which the quantitative knowledge can then be elicited.

For a detailed description of the knowledge acquisition scheme, we refer the reader to [9].

7 Models

Using the knowledge extraction method described in Sect. 6, one LIMID model for each of the vehicles has been developed in collaboration with the end-user. For reasons of space limitations, we include only a subset of the cause hierarchies and models developed.

7.1 The VORTEX AUV

The purpose of the PGM intelligent module of the VORTEX is to assess the status of the energy consumption of the actuators and the payload systems of the AUV. The payload systems consist of various sensors for scientific investigations. More concretely, the task of the module is to compute the probabilities of the various possible root causes of unexpected high energy consumption and the expected utilities of the various recovery actions given the information available.

There are two different aspects (or sub-causes) of "Energy consumption problem", namely "High energy consumption" indicating that the current level of energy consumption is significantly higher than recommended, and "Low state of charge (SOC)" indicating either an abnormally high level of cumulative energy consumption or a poor state of the battery (SOB). These two aspects relate to, respectively, the *present energy consumption* and the *cumulative energy consumption*. The present energy consumption is defined as the average consumption over the last 10 seconds.

To identify the cause of low SOC as a high cumulative energy consumption, the model should either be dynamic, capable of representing phenomena evolving over time, or rely on a measurement of the cumulative total consumption and an indication of the recommended cumulative total consumption at any given point of the mission. We decided to go with the latter approach, as a dynamic model would result in serious computational complexity problems. Also, given that periodic requests are issued from the decision module to the VORTEX intelligent module, determining that the cumulative energy consumption is high is a straightforward task that might as well be performed by the decision module itself.

Figure 7 shows the resulting LIMID. There are four groups of random variables in Fig. 7: Ten diagnosis variables, eleven background information

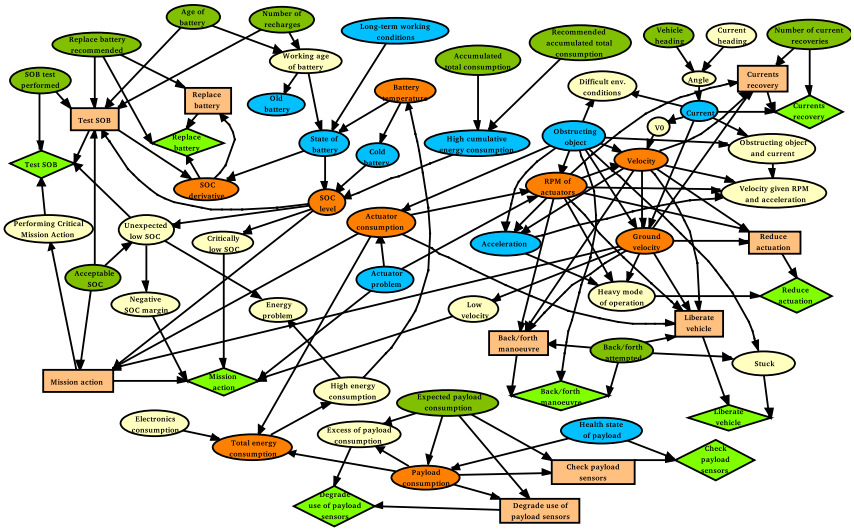

Fig. 7. The VORTEX LIMID

variables, nine symptom variables, and eighteen auxiliary variables. The ten diagnosis variables represent the following distinct root causes of an energy consumption problem of the VORTEX: *Old battery*, *Long-term heavy working conditions*, *Poor SOB*, *Cold battery*, *High cumulative energy consumption*, *Obstructing object*, *Strong currents*, *Fast acceleration*, *Actuator problem*, and *Unhealthy payload*. The posterior probability distributions for these diagnoses are computed on the basis of information provided through the twenty evidence variables (symptom measurements and background information).

The domain experts identified a group of nine different actions that can be performed in response to energy problems: *Mission action* (e.g., "Continue", "Reduce velocity", "Abort mission", etc.), *Test SOB*, *Replace battery*, *Back/forth manoeuvre* (i.e., to escape from an obstructing object), *Check payload sensors*, etc.

Except that *Replace battery* must be preceded by a *Test SOB* action there are no natural orderings among the actions. Also, observations will be provided for all twenty evidence variables (symptom measurements and background information) before any decisions are going to be made. These two facts imply that the model is not naturally represented as an influence diagram. Also, it would make exact inference absolutely intractable.

The LIMID framework therefore offers an ideal representation of this combined diagnosis and decision problem. In fact, the size of the junction tree for the network in Fig. 7 is only about 30K (measured as the sum of the sizes of the clique tables). We should note, however, that the "limited memory" aspect of the model contributes significantly to this fact, as there are observed

variables that belong to the "relevant past" [16] of some decision variables that do not appear as parents of these variables. For example, according to the model in Fig. 7 the observed variables *RPM of actuators* and *Velocity* appear to be relevant for the *Mission action* decision, but there are no information links from these variables to the decision variable, as the *Actuator consumption* and the *Ground velocity* variables are assumed to cater for their influences.

Despite the "limited memory" aspect of the model in Fig. 7, preliminary evaluations of the model provided satisfactory results.

7.2 The BART AGV

The purpose of the PGM intelligent module of the BART AGV is very similar to that of the VORTEX AUV. The fact that the BART carries no payload systems and the obvious difference that the BART is an AGV and the VORTEX an AUV, give rise to some differences in the two models, but for the most part, the BART model shown in Fig. 8 constitutes a subset of the VORTEX model. After some adjustments of the model, a preliminary evaluation

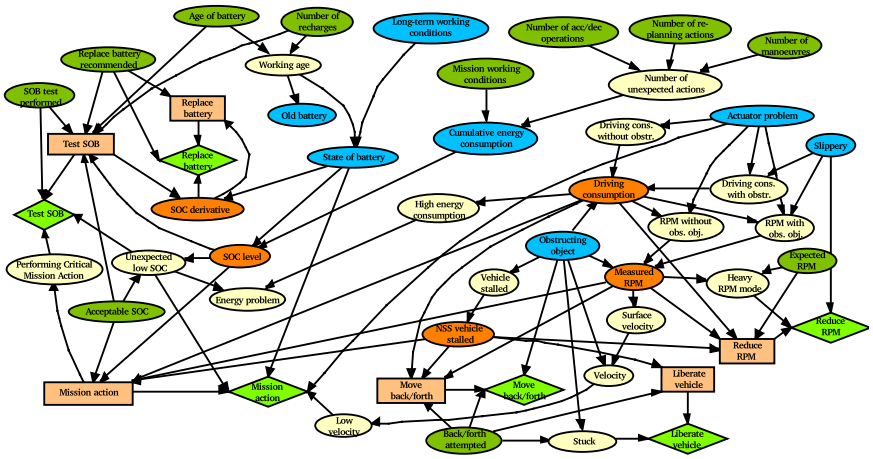

Fig. 8. The BART LIMID

of the model showed an almost complete agreement between expert diagnoses and recommendations and those provided by the model.

7.3 The DeepC AUV

The purpose of the PGM intelligent module of the DeepC is to assess the quality of the sonar image and in the case of bad sonar image quality to

suggest appropriate actions to avoid damage to the vehicle or even loss of vehicle.

The assessment of the sonar image quality is based on the computation of three sonar image quality indicators. The quality indicators are determined by the robot piloting module and fed into the model as evidence. The sonar image quality indicators are pixel entropy, pixel mean value, and pixel substance, see [8] for details. From the above description of the problem domain, it is clear that the amount of disturbance in the sonar image and the presence of objects is time dependent.

The main modeling challenges were to capture the dynamics of the process (how the quality indicators relate to the position and behaviour of the vehicle, the noise sources, and the quality of the image), to address the inherent infinite horizon problem, and to maintain a computationally efficient model (small cliques and policies).

We model the problem as a discrete time, finite horizon partially observed Markov decision process. The model is dynamic in the sense that it models the behaviour of the system over (discrete) time. The state of the system at any given point in time is partially observed as sensor readings are available, but not all entities of the problem domain are observed. The process is represented as an OO LIMID.

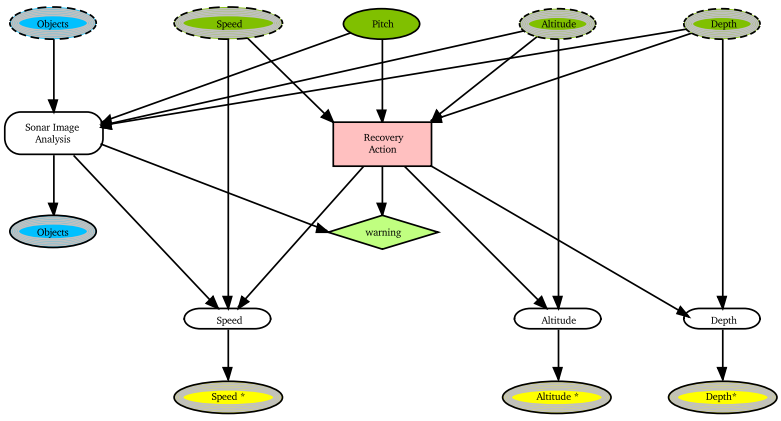

Fig. 9. The generic time-slice model for sonar image assessment

The top-level LIMID class $\mathcal{N}$ contains three instantiations, $\mathcal{M}_i$, $\mathcal{M}_{i+1}$, and $\mathcal{M}_{i+2}$, of the class $\mathcal{M}$ shown in Fig. 9; see Fig. 10 where $\mathcal{M}_i$ has label Ti. The input variables are located at the top of Fig. 9, while the output variables are located at the bottom. In $\mathcal{N}$, the output variables of $\mathcal{M}_i$ are connected to the input variables of the subsequent time-slice $\mathcal{M}_{i+1}$, and similarly for $\mathcal{M}_{i+1}$ and $\mathcal{M}_{i+2}$.

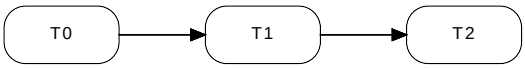

Fig. 10. The top-level LIMID class for sonar image assessment is a time-sliced model of three instances of the model (class) shown in Fig. 9

Each instantiation of $\mathcal{M}$ represents the system at a given point in time. The model $\mathcal{N}$ represents the system at three consecutive time steps with an 8 seconds interval, which is the time the image analysis component needs to analyze a single sonar image.

To avoid combinatorial explosion and thereby maintain computational efficiency, the model specifies that *Altitude*, *Depth*, *Pitch*, and *Speed* are observed prior to the decision *Recovery Action*, but not the image quality indicators. The values computed for the image quality indicators are inserted as evidence and subsequently policies are recomputed. Hence, the policy for the decision in the next time-slice will only depend on the most recent observations on *Altitude*, *Depth*, *Pitch*, and *Speed*. Since the image quality indicators are observed each time a sonar image is analyzed, we need to resolve the LIMID with the observations on the image quality indicators entered as evidence.

The decision has a potential impact on speed, altitude, and depth of the vehicle. Deciding on a recovery action changing any of these properties will impact the quality of the next sonar image. The probability of a collision is modeled in the class instance *Speed*.

The instance *Sonar Image Analysis*, which is an instance of the network class shown in Fig. 11, models the sonar image assessment process. The three image quality indicators are represented in this class by the variables *Entropy*, *Mean*, and *Substance*. The quality indicators are influenced by the presence of disturbance or objects in the sonar image. Disturbance may be caused by reverberation or noise.

The hierarchical construction of the LIMID enforced by the object-oriented paradigm has simplified the knowledge acquisition phase considerably as it is easy to focus on well-defined subparts of the LIMID in isolation. Using class instances, it is a simple task to create and maintain multiple instances of the same LIMID class. Furthermore, it is a simple task to change the class of an instance to another class. This is particularly useful in the knowledge acquisition phase where each LIMID class has been revised and updated multiple times.

For further details on the DeepC model, see [8].

8 Solving Models

A LIMID is solved using a message passing procedure in a junction tree as briefly described in Sect. 5. This procedure proceeds by iteratively passing

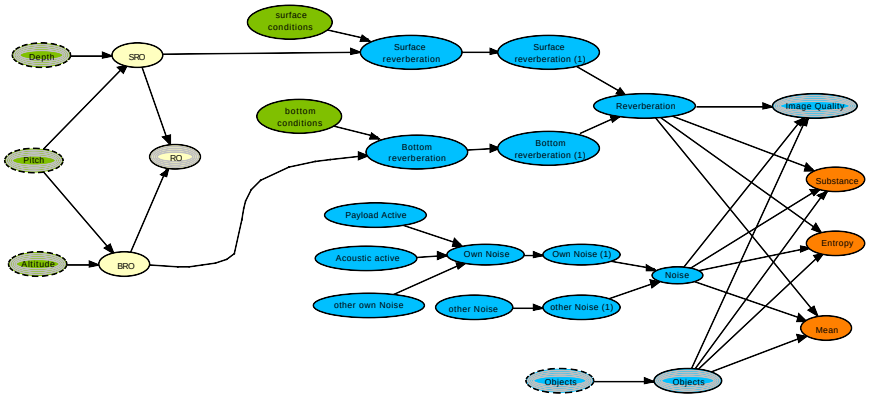

Fig. 11. The model (class) representing the sonar image assessment process

messages in a junction tree representation of the LIMID. The procedure for solving LIMIDs differs from algorithms solving influence diagrams [4,11]. The procedure for solving a LIMID is an iterative procedure working on a junction tree representation.

The complexity of solving a LIMID depends heavily on the structure of the junction tree, i.e., the sizes of cliques in the junction tree. The structure of the junction tree is determined by the connectivity of the graph of the LIMID, which in turn is determined by the number of nodes and links of the graph. The state space sizes of variables and the number of links determine the number of parameters of the model. Table 1 shows the number of chance and decision variables as well as the number of parameters to be specified in the three LIMID models constructed. For the DeepC model the number of variables in the instantiated network is shown.

Table 1. Number of variables and parameters of the three models

Model	Chance variables	Decision variables	Total	Parameters
DeepC	121	3	124	3,556
VORTEX	57	9	66	3,475
BART	49	6	51	901

From Table 1 it can be seen that the DeepC model contains the largest number of variables while the VORTEX and BART models have approximately the same number of variables. The VORTEX model has a higher number of parameters though.

Table 2 shows the total clique state space of the optimal junction trees used for solving each LIMID where the optimality criterion is clique state space size. The total clique state space size is defined as $\sum_{C \in \mathcal{C}} \prod_{X \in C} \|X\|$, where $\mathcal{C}$ denotes the set of cliques of the junction tree and $\|X\|$ denotes the number of states of variable X.

Table 2. Total clique state space size of the three models

Model	Influence Diagram	LIMID
DeepC	$> 2^{32} - 1$	108,841
VORTEX	$> 2^{32} - 1$	26,099
BART	14,017,742	39,322

Table 2 also shows the total clique state space size when each model is solved as an influence diagram. The table shows that the DeepC and VORTEX models when considered as influence diagrams cannot be solved on a standard 32 bits PC platform (each vehicle is equipped with a standard 32 bits PC platform running either Windows or Linux operating systems). Thus, applying probabilistic graphical models to diagnosis and control of autonomous vehicles would be infeasible without the use of LIMIDs.

9 Integration, Validation, and Real-World Trials

Each LIMID model is encapsulated in an intelligent module of the ADVOCATE II architecture (see Fig. 4). The intelligent module takes care of the communication with the decision module and robot piloting modules. Module integration was performed using a special-purpose integration tool, which has greatly simplified the integration process as it allows developers to integrate their modules into an architecture consisting of a mix of mock-up modules, man-machine interfaces, and real modules. This was very helpful as module developers were located in different countries with different working hours and with limited possibilities for face-to-face meetings.

The validation of each module is equivalent to validation of the knowledge bases (models). The validation of a model was performed by a careful investigation of the performance and behaviour of the system based on a selected set of test scenarios.

Extensive prototyping has helped to ensure appropriate performance and behaviour of each module. The test cases used to validate and measure the performance of a model covers all important, critical situations that possibly can occur in realistic operations. Finally, domain expert(s) and the knowledge engineers have evaluated the model against the test cases iteratively.

The usefulness of each LIMID was evaluated by a sequence of trials where each LIMID was deployed as an intelligent module as part of the ADVOCATE II architecture on the vehicles described above. In each trial the LIMID was deployed in an instantiation of the architecture consisting of a directory module, a decision module, a robot piloting module, and an intelligent module.

The DeepC model had an 88.4% accuracy on image quality assessment on the test set consisting of 1048 sonar images. The sonar images were recorded at the ATLAS test pond in Bremen where the real sea trials were performed.

The BART model had a 93.4% accuracy on 36 selected test cases. The test cases where hand generated to reflect operation of BART at the University of Alcalá campus.

No real trial data was available for the VORTEX model. Preliminary results on simulated data were promising, but the data is of insufficient quality for a proper validation.

10 Discussion

We have presented an application of HUGIN to solve reasoning and decision making problems under uncertainty related to diagnosis and control of autonomous vehicles. The application is based on the ADVOCATE II architecture. The ADVOCATE II architecture is a distributed architecture supporting diagnosis and control of autonomous vehicles.

The main objective of the ADVOCATE II project was to develop an architecture to allow the implementation of intelligent modules for AGVs and AUVs in order to increase their reliability and efficiency.

The performance of the ADVOCATE II architecture is constrained by (soft) real-time requirements. This implies that the performance of the communication protocol and the intelligent modules needs to be very high. This has implied a high focus on computational performance in the model construction.

Not only does the communication architecture enable efficient integration of different AI technologies into new and existing systems, but it also allows various AI techniques to interact (through the decision module and robot piloting module, though). This option of interactions has been used to dedicate PGMs to certain types of problems and to have variables in a PGM represent the output of an NSS intelligent module (e.g., variable "NSS vehicle stalled" of the BART LIMID; see Fig. 8). In addition, different AI techniques may be used to solve the same problem. This will often be the case for mission critical error handling. In our case, this raises the issue of a common scale of measurement of the usefulness of actions. We have chosen to use normalized expected utility as the measurement of usefulness of recovery actions.

Even though the framework of PGMs have been available for more than 15 years, it is our experience that the efficient use of these models in certain

domains still requires some research and development. It is often a problem for the PGM technology that only a few very convincing success stories are available to the public. Often a PGM captures all or almost all the knowledge a company has in a particular business area. This implies that the company is not interested in sharing this model or even sharing the knowledge that such a model exists. The results of the ADVOCATE II project, however, add to the increasing number of successful applications of PGMs available to the public.

One of our key experiences from research and development projects is that even though graphical models are intuitive, they are difficult to build for inexperienced domain experts. In the ADVOCATE II project, it was necessary to develop a new methodology to solve the knowledge acquisition task. We have developed a knowledge elicitation and formulation method, which is applicable in general to problems of reasoning and decision making on complex machinery such as AGVs and AUVs.

The LIMID representation [11] and the object-oriented knowledge representation paradigm [10] implemented in the HUGIN tool [1,5] have been two major cornerstones of the success of Bayesian modeling in the ADVOCATE II project. Still a lot of work remains to develop the object-oriented framework further.

Acknowledgment We acknowledge the partners of ADVOCATE II: University of Alcalá (Spain), Getronics (France), Technical University of Madrid (Spain), ATLAS Elektronik GmbH (Germany), Ifremer (France), HUGIN Expert A/S (Denmark), Innova S.p.A (Italy), and e-motive (France). The EU Commission supported the ADVOCATE II project under grant IST-2001-34508. Visit the project web-site for more information on ADVOCATE II:

http://www.advocate-2.com

Please visit the company web-site for more information on applications using HUGIN software:

http://www.hugin.com

References

1. S. K. Andersen, K. G. Olesen, F. V. Jensen, and F. Jensen. HUGIN — a Shell for Building Bayesian Belief Universes for Expert Systems. In *Proceedings of IJCAI'89*, pages 1080–1085, 1989.
2. R. G. Cowell, A. P. Dawid, S. L. Lauritzen, and D. J. Spiegelhalter. *Probabilistic Networks and Expert Systems.* Springer-Verlag, 1999.
3. R. A. Howard and J. E. Matheson. Influence Diagrams. In *The Principles and Applications of Decision Analysis*, volume 2, chapter 37, pages 721–762. 1981.
4. F. Jensen, F. V. Jensen, and S. L. Dittmer. From Influence Diagrams to Junction Trees. In *Proceedings of 10th Conference on UAI*, pages 367–373, 1994.

5. F. Jensen, U. B. Kjærulff, M. Lang, and A. L. Madsen. HUGIN - The Tool for Bayesian Networks and Influence Diagrams. In *Proceedings of PGM'02*, pages 212–221, 2002.
6. F. V. Jensen. *Bayesian Networks and Decision Graphs*. Springer-Verlag, 2001.
7. F. V. Jensen, S. L. Lauritzen, and K. G. Olesen. Bayesian updating in causal probabilistic networks by local computations. *Computational Statistics Quarterly*, 4:269–282, 1990.
8. J. Kalwa and A. L. Madsen. Sonar image quality assessment for an autonomous underwater vehicle. In *Proceedings of ISORA'04*, 2004.
9. U. B. Kjærulff and A. L. Madsen. A methodology for acquiring qualitative knowledge for probabilistic graphical models. In *Proceedings of IPMU'04*, pages 143–150, 2004.
10. D. Koller and A. Pfeffer. Object-oriented Bayesian networks. In *Proceedings of UAI'97*, pages 302–313, 1997.
11. S. L. Lauritzen and D. Nilsson. Representing and solving decision problems with limited information. *Management Science*, 47:1238–1251, 2001.
12. S. L. Lauritzen and D. J. Spiegelhalter. Local computations with probabilities on graphical structures and their application to expert systems. *Journal of the Royal Statistical Society, Series B*, 50(2):157–224, 1988.
13. A.L. Madsen, M. Lang, U. Kjærulff, and F. Jensen. The Hugin Tool for Learning Bayesian Networks. In *Proceedings of 7th European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty*, pages 594–605, 2003.
14. R. E. Neapolitan. *Learning Bayesian Networks*. Prentice Hall, 2003.
15. J. Pearl. *Probabilistic Reasoning in Intelligence Systems*. Series in Representation and Reasoning. Morgan Kaufmann Publishers, 1988.
16. R. Shachter. Efficient Value of Information Computation. In *Proceedings of UAI'99*, pages 594–601, 1999.