Proceedings of the 38th
Conference on Decision & Control
Phoenix, Arizona USA • December 1999

Model-Based Diagnosis and Control Reconfiguration for Discrete Event Systems : An Integrated Approach †

Gregory Provan and Yi-Liang Chen‡
Rockwell Science Center
1049 Camino Dos Rios
Thousand Oaks, CA 91360
{gmprovan, ylchen}@rsc.rockwell.com

Abstract

We describe an approach to automate the dynamic computation of optimal control/reconfiguration actions that can achieve pre-specified control objectives. This approach, based on model-based diagnostic representations and algorithms, integrates diagnostics and control reconfiguration for discrete event systems using a single modeling mechanism and suite of algorithms. When the system functionality degrades (*i.e.*, failures occur in the systems), the diagnostic algorithm will isolate the most-likely failures, and then the control mechanism will generate the least-cost actions that attempt to recover from the failure and maintain the control objectives. Results about the quality of the control actions generated and the complexity of computing these actions are also presented. We illustrate our approach using a simple wireless sensor network.

Keywords: model-based diagnostics, discrete event systems, system control and reconfiguration.

1 Introduction

System Control, reconfiguration, and diagnosis are important aspects of managing real-world applications modeled as discrete event systems. By control we mean identifying control actions to achieve given control objective(s) given pre-defined system constraints and current system observations. By reconfiguration we mean (re)generating control actions that could rearrange the functions/roles of system components in order to maintain the control objectives given anomalous system observations. By diagnosis we mean isolating the cause of anomalous sensor readings (system observations). These aspects of discrete event systems are inter-related: the control actions generated by control and/or reconfiguration mechanisms will affect the next sensor readings; diagnostic isolation of the failed components leading to degraded system performance will affect the generation/modification to the control and possibly system reconfiguration.

Typically, control, reconfiguration and diagnosis tasks for discrete event systems are studied/performed independently, with distinct models designed for each task. Various studies in the automata-based supervisory control paradigm [9] have been focusing on identifying control actions within a given specification. Work in [10] addresses the system reconfiguration issues. For diagnosis, several model-based approaches based on artificial intelligence techniques have been widely studied (*e.g.*, GDE [4] or causal networks [3]). However, little attention has been given to leverage these existing results to provide an integrated framework for control, reconfiguration, and diagnosis. One recent work [11], an automata-based diagnosis approach, partially addresses this concern. An AI-based approach [12] has looked at this area, focusing primarily on the real-time aspects of this task.

A second key motivation for our work is that we provide a means for generating control for system failure states without having to pre-program all control conditions for anomalous states. In many existing approaches, control has to be pre-specified for all possible failure states, which can be an onerous process; moreover, it is typically not guaranteed to cover all possible failure states. In the event of system failure, our approach will automatically generate control actions that attempt to recover from the failure and maintain the control objectives; moreover, provide an automated method for computing control for failure states, and we give guarantees about the complete coverage of the control actions computed. Due to the highly related nature of system control and reconfiguration and our similar approach in addressing them, we do not specifically distinguish them hereon.

This paper describes how a single underlying representation, causal networks [3], can be used for a number of tasks, including simulation, control/reconfiguration, and diagnosis. Our main contributions are as follows: (1) We propose an integrated approach that uses a

†Research supported in part by The Office of Naval Research under contract number N00014-98-3-0012.

‡Corresponding author. Tel: (805) 373-4108.

single integrated model for both diagnosis and control of discrete event systems, regardless of system health. (2) We describe how to extend a traditional model-based diagnostic inference algorithm with focusing to efficiently compute least-cost control actions; as such, this framework allows us to provide guarantees for control similar to those possible for diagnosis. This framework enables recovery capabilities similar to [12], but using standard model-based diagnostic approaches. It also provides a powerful modeling/simulation platform that can incorporate other analysis mechanisms such as planning.

The the article is organized as follows. In Section 2 we introduce an illustrative simple wireless sensor network example. We summarize our modeling mechanism and diagnostic approach in Section 3. In Section 4 we show how we extend this diagnostic approach to system control. We present our integrated framework in Section 5. We then conclude with a discussion of the related work and comments on further extending our approach.

2 A simple wireless sensor network example

We now describe a simple wireless sensor network of four acoustic sensor nodes [6]. This much-simplified example is intended solely to illustrate our approach. We make various assumptions to simplify the model, to focus the readers on the concepts of our approach.

Figure 1 shows the relative positions of the four sensor nodes and their effective sensing ranges. Powered by a small battery, each node is equipped with an acoustic sensor and a wireless communication unit. The communication unit can perform node-to-node radio frequency communication and can also serve as a long-range hub that communicates back to the remote commanding center. We assume direct communication between a regular sensor node and a sensor node that serves as a long-range hub. The sensing and long-range communication functions of the sensor node operate independently. Hence, four different control instructions/actions can be sent to each of the sensor node: idle (sleep), sensor, hub, and sensor+hub.

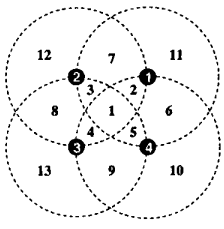

Figure 1: A wireless sensor network.

We quantify the remaining power of the battery to 7 levels, *i.e.*, from 0 (exhausted) to 6 (fully charged). We assume that a sensor node consumes negligible power when it is idle and consumes twice (three times) as much of the power when it serves as a long-range hub (both sensor and communication hub) as the power when it functions only as a sensor. Moreover, we assume sensor or long-range hub unit will not consume power when it fails.

We divide the effective sensing area of the network into 13 zones based on the coverage of the sensor nodes as shown in Figure 1. A functioning acoustic sensor will detect the sound generated in seven zones that it covers when it is activated. To simplified the problem, we do not address the issue of locating the source (*i.e.*, target) of the sound within the sensing area.

We assume the sensor and the communication unit in a node can fail. When a sensor fails, we assume that it will always generate negative output (*i.e.*, false-off). We assume that only the long-range hub function can fail and the node-to-node communication continues to function as long as the battery is not exhausted.

Since we do not address the target identification and locating issues, our diagnosis in this example is limited to the situation when the target can be positively identified by other means (*e.g.*, visual confirmation, or predelopyed testing sources). The typical control objective in this example is to provide consistent surveillance of certain zones over a period of time.

3 Causal network modeling and diagnosis

3.1 Causal Network Modeling

For discrete event applications, we model the underlying system (plant) behavior in terms of a causal network plant model Φ. The plant model describes the *physics* of the system: distances, speeds, and their relationships which can be viewed as an actuator-to-sensor map.[1] Note that the plant model Φ simulates the system behavior under both normal and abnormal *component modes* (*i.e.*, operating modes of the components).

A causal network plant model is a four-tuple $\Phi = (\mathbf{V}, \mathbf{A}, \mathcal{G}, \Delta)$, where $\mathbf{V}$ is a set of variables, $\mathbf{A}$ is a distinguished set of variables called *assumables*, $\mathcal{G}$ is a directed graph called a *causal structure* whose nodes are members in $\mathbf{V} \cup \mathbf{A}$, and Δ is a set of *sentences* constructed from members in $\mathbf{V} \cup \mathbf{A}$ based on the topological structure of $\mathcal{G}$.

Variables in $\mathbf{V}$ represent the components, modules, or abstract properties of the system, such as sensors, motors, or power level. Each variable in $\mathbf{V}$ has a finite set

[1] For some diagnosis applications, a similar "control" model can be built to describe the closed-loop controller behavior (*i.e.*, a sensor-to-actuator map) if the control plan is *given* rather than generated by the control approach described in this article.

of discrete values. We define $\mathbf{V}_{obs}$, the set of *observables*, as the subset of variables in $\mathbf{V}$ whose values can be either directly measured (sensor) or set (actuator). Similarly, $\mathbf{V}_{unobs}$ (the set of *unobservables*) consists of the remaining variables in $\mathbf{V}$ whose values cannot be directly detected. For control and discrete event applications, the observables are usually sensors and actuators while the unobservables represent internal components (*e.g.*, motors) or properties (*e.g.*, velocity and position).

Assumables in $\mathbf{A}$ are special variables describing the operating characteristics of system components. Each assumable A is associated with a *unique* variable in $\mathbf{V}$ and has a finite set of discrete modes. Each (failure) mode is assigned a reliability measure to specify the relative failure likelihoods.[2] These measures are used in a focusing algorithm to restrict search on only the most-likely diagnoses [3]. We will discuss the reliability measure and the focusing algorithm in more details in Sections 3.2 and 4. Note that $\mathbf{V} \cap \mathbf{A} = \emptyset$.

The causal structure $\mathcal{G}$ is a directed acyclic graph that specifies the causal relationships over the variables and assumables. Each node V_i of $\mathcal{G}$ represents a unique element in $\mathbf{V} \cup \mathbf{A}$ and each directed arc from V_1 to V_2 represents the causal influence of V_1 on V_2. Each assumable node has no predecessors and only one child.

To formally describe the syntax and semantics for the sentences in Δ that specify how the values of different variables interact, we first introduce some terminology [2]. Given a set of variables $\mathbf{V}$ and a set of assumables $\mathbf{A}$, a *sentence* is defined recursively as follows: (i) $[V = v]$ is a sentence, where $V \in \mathbf{V} \cup \mathbf{A}$, and v is a value of V; and (ii) $\neg\alpha, \alpha \vee \beta$, or $\alpha \wedge \beta$ is a sentence where α and β are sentences. We call $[V = v]$ a *V-literal.*

For each variable $V \in \mathbf{V}$, a set of equations Δ_V, called the *database* of V, is defined. These equations are of the form $\beta \supset \alpha$, where α and β are sentences that must satisfy modularity, locality, and consistency conditions similar to that described in [7]. The *database* Δ is then the union of Δ_V for all V in $\mathbf{V}$.

The plant model Φ we just described is for diagnosis purposes where we categorize the failure mode variables as assumables. We will show in Section 4 that by simply re-categorizing the variables, we can easily modify the plant model for our control approach.

3.2 Model-based diagnostics

A model-based diagnosis inference task using our causal network approach can be specified by the tuple $(\Phi, \Omega, \mathcal{I}, \mathcal{F})$, where Φ is the system (plant) model described in Section 3.1, Ω is the input observation, $\mathcal{I}$ denotes the inference algorithm, and $\mathcal{F}$ denotes the focusing mechanism. We now describe the remaining three elements of the tuple in turn.

The information about observables is the control and sensor data that allows us to determine the system state. The system observation Ω is a $\mathbf{V}_{obs}$-instantiation; *i.e.*, it describes the control and sensor values that are currently observed. This observation allows us to determine the current system status; the inference algorithm $\mathcal{I}$ then uses Ω in conjunction with the plant model to compute the possible diagnoses.

The underlying inference algorithm $\mathcal{I}$ for this approach [3] is based on computing a consequence, with which we can identify diagnoses, or as shown in Section 4, control actions. Our diagnosis inference task is: given a model $(\mathbf{V}, \mathbf{A}, \mathcal{G}, \Delta)$ and observation Ω, find the failure-mode specification $\mathbf{A}^*$ consistent with $\Delta \cup \Omega$.

The focusing algorithm is used to generate only the most-likely diagnoses, thereby reducing the inference complexity considerably over enumerating all minimal diagnoses. Computing the value of $\mathbf{A}^*$ is NP-hard [5], since it searches over all subsets of mode-instantiations. Hence, we apply a focusing algorithm to conduct a best-first search over the space of diagnoses [3]. Using the reliability measures (weights) assigned to the fault modes, this focusing algorithm applies an algebra to perform the best-first search over the diagnosis space, starting with the most-likely (*i.e.*, least-weight) diagnoses. For diagnosis purposes, this weight assignment is usually defined as a function $\mathcal{H} : \Sigma \rightarrow \mathcal{A}$, where Σ is a set of reliability measures and $\mathcal{A}$ is the union of all discrete failure mode values for all assumables in $\mathbf{A}$. Different weight assignments can also be implemented (refer to Section 4). Following [3], we can characterize the algebra using the triple $\langle \Sigma, \oplus, \geq_\oplus \rangle$, where $\oplus$ is a weight addition operation and $\geq_\oplus$ is a weight total ordering satisfying: (i) $\oplus$ is commutative, associative and has a zero element and (ii) $j \geq_\oplus i$ if and only if $i \oplus k = j$ for some k. Note that a diagnosis of weight j is said to be more likely than a diagnosis of weight i, if $j \geq_\oplus i$. An example of this is $\langle \mathbf{Z}, +, \geq \rangle$, where $\mathbf{Z}$ is the set of integers.

3.3 Causal network model for a sensor network

We now show our causal network model for the sensor network described in Section 2. The causal network model was constructed and simulated using our integrated tool of (a) Component-based Diagnostic Model Builder (CDMB) and (b) Causal NETwork diagnosis compilation and Simulation tool (CNETS). We describe an application of this tool to the condition-based monitoring of pump/motor systems in [8].

Figure 2 shows a high level causal network for the wireless sensor network depicted in Figure 1. Nodes (*i.e.*, variables and assumables) in this network are categorized into five groups. The control nodes represent the

[2]Different types of reliability measures, such as order-of-magnitude probabilistic, can be used.

control variables for the four sensor nodes. The "target in the zone" nodes indicate the presence of the targets in one of the 13 zones. The sensor coverage nodes represent whether the 13 different zones are monitored by some active sensors. The variable Comm.Coverage indicates the status of the long-range communication capability. The behaviors of a sensor node are represented by one of the four node subnets.

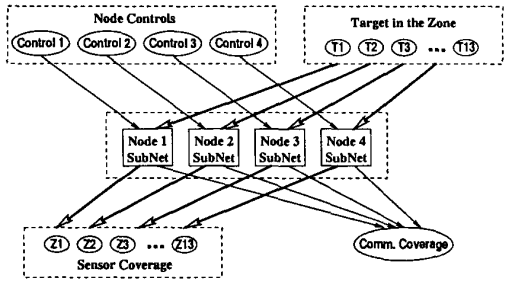

Figure 2: High level causal relationship.

Figure 3 shows the details of the subnet for Node 1 and its relationship to the rest of the causal network model. The unshaded oval nodes represent observables while the shaded oval nodes represent unobservables. The shaded rounded rectangle node NodeMode is the only assumable in this subnet. This assumable has four failure modes values: ok, s-bad (sensor failure), h-bad (long-range hub failure), and sh-bad (failure to both sensor and long-range hub unit). Node Control1 is the only actuator variable in this subnet. It has four possible values: idle (node idling), sensor (node serves as a sensor), hub (node serves as a long-range hub), and sensor+hub (node serves as both a sensor and a hub). Since the sensor node covers 7 zones, this subnet is connected to 7 corresponding target-presence nodes and 7 sensor-coverage nodes. Note that the current setting of this causal network model is for diagnostic purposes.

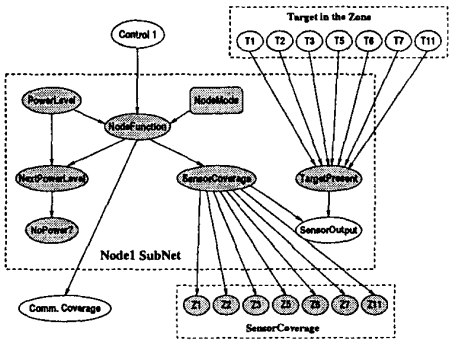

Figure 3: Diagnostic subnet for node 1.

We show one of the equations in the database for NodeFunction of Node 1 subnet in the following.

$$[\text{Control1} = \text{sensor}] \wedge [\text{NodeMode} = \text{ok}] \wedge \neg[\text{PowerLevel} = 0] \supset [\text{NodeFunction} = \text{sensor}].$$

Model	Variables		Assumables	Graph Structure
	Obs.	Unobs.		
Diagnosis	$\mathbf{C} \cup \mathbf{S}$	$\mathbf{U}$	$\mathbf{A}$	$\mathcal{G}$
Control	$\mathbf{A} \cup \mathbf{S}$	$\mathbf{U}$	$\mathbf{C}$	$\mathcal{G}$

Table 1: Relationship between primal and dual models.

4 Causal networks for control

4.1 A dual approach to system control

We now extend the causal network approach to diagnostic inference by creating a dual causal network for system control/reconfiguration. This involves recategorizing some nodes in the network, but preserving the structure of the network. To define a causal network for control purposes, we first partition the observables $\mathbf{V}_{obs}$ into a set $\mathbf{C}$ of control variables (*i.e.*, actuators) and a set $\mathbf{S}$ of sensors: $\mathbf{V}_{obs} = \mathbf{C} \cup \mathbf{S}$. We define the dual model for control purposes as $\Phi' := (\mathbf{S} \cup \mathbf{A} \cup \mathbf{U}, \mathbf{C}, \mathcal{G}, \Delta)$. Table 1 shows the relationship between the primal (diagnosis) model and the dual (control) model. The graph structure $\mathcal{G}$ and the unobservables $\mathbf{U}$ remain the same in both models. The key change is that we have swapped some observables (the control variables $\mathbf{C}$) and assumables $\mathbf{A}$.

The control objective in our approach is specified by a sentence, Θ, which consists of literals of some variables in $\mathbf{S}$ and $\mathbf{U}$. For example, a control objective for our wireless sensor network example can be

$$\Theta = [\text{Comm.Coverage} = \text{true}] \wedge [\text{Zone1Coverage} = \text{true}] \wedge [\text{Zone4Coverage} = \text{true}] \wedge [\text{Zone5Coverage} = \text{true}], \quad (1)$$

which specifies that the long range communication should be established and sensor surveillance should cover activities in zones 1, 4 and 5.

Similarly, the constraints to the system behavior are usually specified in terms of a sentence, Π, consisting of literals of some variables in $\mathbf{S}$ and $\mathbf{U}$. The sentence outlines the "proper" system behavior and excludes system states that should never be reached. An example of the constraints for the sensor network is

$$\Pi = \neg[\text{Node1Power} = 0] \wedge \neg[\text{Node2Power} = 0] \wedge \neg[\text{Node3Power} = 0] \wedge \neg[\text{Node4Power} = 0], \quad (2)$$

which indicates that none of the batteries in the four nodes should be exhausted.

As shown in Table 1, we treat the failure-mode variables $\mathbf{A}$ as observables in the control model. Hence, a diagnosis $\mathbf{A}^*$ that consists of propositional literals of variables in $\mathbf{A}$ is considered as a system observation in our control approach.

Our model-based inference for control purposes is as follows: given a control model Φ', a control objective Θ, constraints Π, and current diagnosis $\mathbf{A}^*$, find

the control action $\mathbf{C}^*$ such that $\mathbf{C}^*$ is consistent with $\Delta \cup \Theta \cup \Pi \cup \mathbf{A}^*$. Here $\mathbf{C}^*$ is an assignment of values to all system control variable (*i.e.*, next-step control actions). To efficiently compute the control action, we assign control costs Σ' to all the values of the control (actuator) variables and then adopt the same focusing algorithm [3] to guide our search. The control algebra for the focusing algorithm can be characterized using the triple $\langle \Sigma', \bar{\oplus}, \geq_{\bar{\oplus}} \rangle$, where $\bar{\oplus}$ is a cost addition operation and $\geq_{\bar{\oplus}}$ is a cost total ordering satisfying the properties defined in Section 3.2. To properly assign the control costs associated with the control actions (variables), we propose the three following approaches.

Our first approach, similar to how we assign the reliability measures to the failure modes in diagnosis algorithm, is to associate each value of a control variable with a constant control cost in Σ'. This approach assumes taking a control action would incur a constant control cost regardless of previous control setting and other factors. We assign a constant control cost proportional to, for instance, the energy consumption of the control action. For example, we can assign the control cost of a wireless sensor node to be 0 for idling, 1 for sensing, 2 for serving as a communication hub, and 3 for serving as both a sensor and a hub.

The second approach assigns a variable control cost to every value of a control variable. Hence, the control cost for a particular value of a control variable can be represented as a function of system states. For instance, we can assign the cost of a control action as an inverse function of the component life after the control action. Similarly, the cost for a certain control action of the sensor node (*e.g.*, idle, sensing, etc.) for a period of time can be defined as the square of the total accumulated power consumption after the period ended *i.e.*, $(\text{Charge}_{(\text{full})} - \text{Charge}_{(\text{remaining after the period})})^2$.

The above two approaches assume that transitions between any value of a control variable are possible. However, in some applications, the transitions between different control actions are sometimes restricted. In this case, the restrictions and the control cost of a control variable are best represented by a Moore machine. The states of the Moore machine are the possible values of the control variable and the outputs of transitions represent the switching cost from one control values to another. Figure 4 shows the Moore machine that we can use to define the control cost for a sensor node. Note that, with this approach, it is possible to restrict the available next control values to the ones that have a direct transition from the current control value in the Moore machine. This can be done by making the cost of any indirect transitions between control values infinitely large. Using this transition based approach, the allowable transitions add an extra constraint to the focusing algorithm for system control. That is, when searching for a least-cost control action, the search is constrained to the allowable set of transitions from any given place in a transition diagram.

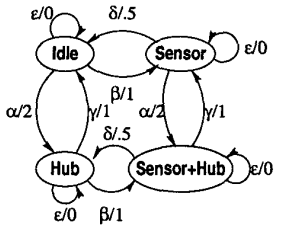

Figure 4: Control cost based on state transitions.

Applying the same inference algorithm $\mathcal{I}$ as that for diagnosis and a similar focusing mechanism $\mathcal{F}'$ based on the algebra $\langle \Sigma', \bar{\oplus}, \geq_{\bar{\oplus}} \rangle$, our model-based control approach can be specified by the tuple $(\Psi', \Theta \cup \Pi \cup \mathbf{A}^*, \mathcal{I}, \mathcal{F}')$. The results from our control algorithm are the least-cost control actions indicated in $\mathbf{C}^*$.

The diagnostic causal network model for a wireless sensor network described in Section 3.3 can be reconfigured for system control purposes as outlined in Table 1. The Node 1 subnet for control purposes is shown in Figure 5. Note that the shadings of the nodes in the figure will be explained later in Section 5.2 and do not mean whether the nodes are observable or unobservable. In this setting, Control1 becomes an assumable while NodeMode is treated as a variable.

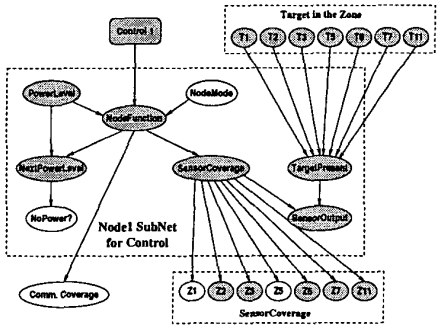

Figure 5: Control causal network for node 1.

4.2 Guarantees for control algorithm

We now summarize the guarantees made for the control algorithm. These results are derived using a modified diagnosis inference procedure as described by [3].[3]

Lemma 1 *(1) Computing the set of all control actions is NP-hard and (2) Computing the set of recovery actions using a causal network representation, in the worst case, is linear in the number of network nodes and exponential in the maximum graph width of the causal network.*

[3]Due to the page limit, proofs of all lemmas in this version of the article are omitted. Please contact the authors for proofs.

Lemma 2 *Given a control task characterized by* $(\Psi, \Theta \cup \Pi \cup \mathbf{A}^*, \mathcal{I}, \mathcal{F}')$, *and control algebra* $\langle \Sigma', \bar{\oplus}, \geq_{\bar{\oplus}} \rangle$, *the focusing algorithm* $\mathcal{F}'$ *is guaranteed to compute all sound, minimal-cost control actions, if any exist.*

Similar to the diagnostic compilation we have used to implement our diagnostic algorithm [3], in real-world implementations, we can also pre-compute a representation R for the set of all control actions, cache that function, and then compute control actions on a case-by-case basis using the focusing algorithm [3]. We call this technique control compilation. This means that we can perform the NP-hard task of generating R only once and then we can compute later on particular control actions efficiently, as shown below:

Lemma 3 *Given a representation* R *for the set of all control actions, computing a particular control action given a control objective, constraints, and diagnoses can be done in time* $O(|R||\bar{\oplus}|)$, *where* $|R|$ *denotes the number of elements in* R *and* $|\bar{\oplus}|$ *denotes the time for performing the* $\bar{\oplus}$ *operation followed by a minimization.*

Note that, for all the complexity analyses presented so far, we do not take into account the complexity of assigning costs to the values of control variables. If we assign constant costs to the values (the first approach described in Section 4), the computational cost is constant. If, however, we assign costs to the values dynamically, especially using the transition-based (third) approach in Section 4, the complexity of the cost assignment should be added to our total computational complexity. We note that the pre-compilation of R does not depend on the cost assignment [1]. Hence, the control compilation technique can still be applied with dynamic control costs.

5 An integrated diagnosis and control framework

5.1 Integration Architecture

With the diagnosis and control mechanisms based on causal network models described in the Sections 3.2 and 4, we now present how we integrate the two mechanisms. Figure 6 shows the flowchart of our integrated approach for the control and diagnosis of a discrete event system. Given the initial control objective (Θ), constraints (Π), observations (Ω), and Diagnoses ($\mathbf{A}^*$), we first apply our control algorithm to the control causal network model Φ' of the underlying system with the specification $(\Psi', \Theta \cup \Pi \cup \mathbf{A}^*, \mathcal{I}, \mathcal{F}')$. As outlined in Table 1, the observables and assumables of the control causal network consist of $\mathbf{A} \cup \mathbf{S}$ and $\mathbf{C}$, respectively.

If the given control objective is achievable, the control algorithm will generate all possible least-cost control actions $\mathbf{C}^*$. If there is more than one least-cost control action, a selection mechanism will be required to choose one control action for execution (actuation) based on domain knowledge. If the control objective cannot be achieved, then a planning mechanism will be required to revise the control objective or relax the imposed constraints, if possible, according to domain priorities. The revised objective/constraints will then trigger another execution of the control algorithm. Note that we do not address the control selection and planning mechanisms in this article, since these mechanisms are usually domain dependent.

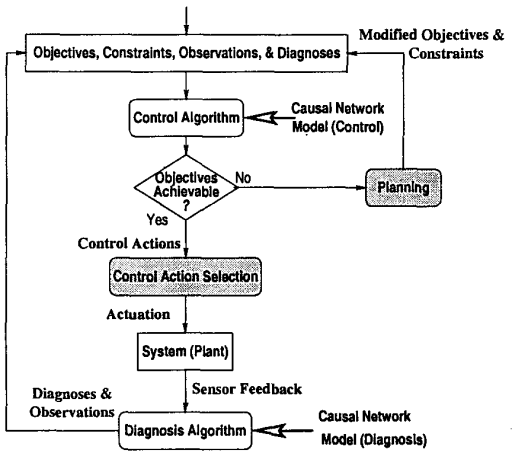

Figure 6: An integrated control and diagnosis framework.

The feedback from the underlying system after the execution of the selected control action is in terms of the values of different sensors (*i.e.*, system observations, Ω). Given these updated observations, we then apply our diagnosis algorithm to the diagnosis causal network model of the system with the specification $(\Psi, \Omega, \mathcal{I}, \mathcal{F})$. Note that the observables and assumables of the diagnosis causal network are $\mathbf{C} \cup \mathbf{S}$ and $\mathbf{A}$, respectively. The resulting diagnoses $\mathbf{A}^*$ along with the updated observations are then used for the next iteration of our integrated control-diagnosis approach.

5.2 Simulations and results

We now present the simulation results of applying our integrated approach for the control and diagnosis of the sensor network under a scenario. We assume all sensor nodes are functioning normally at the beginning. We set the objective, Θ, for our scenario as in Equation 1; *i.e.*, the long range communication should be established and sensor surveillance should cover activities in zones 1, 4, and 5. The initial constraints for our scenario are specified as in Equation 2; *i.e.*, none of the batteries in the four nodes should be exhausted. We assume the power consumption for each node under the four operational states are 0 (idle), 1 (sen.), 2 (hub), and 3 (sen.+hub) between each iteration of our

Iteration	1	2	3	4	5	6*
No. of actions generated	6	2	1	1	3	4
Node 1 Action (Power Level)	idle (6)	hub (6)	idle (4)	idle (4)	sen. (4)	hub (3)
Node 2 Action (Power Level)	idle (6)	idle (6)	hub (6)	idle (4)	sen. (4)	idle (3)
Node 3 Action (Power Level)	sen. (6)	idle (5)	idle (5)	hub (5)	hub (3)	idle (1)
Node 4 Action (Power Level)	hub (6)	sen. (4)	sen. (3)	sen. (2)	idle (1)	sen. (1)

Table 2: Control actions for our scenario.

integrated approach. We set a constant weight to each of the four failure mode values for a sensor node: 0 (ok), 1 (s-bad), 1 (h-bad), and 2 (sh-bad). We assume that an acoustic testing source is always present in zone 5.

To represent the control costs, we adopt our second approach described in Section 4.1. We define the control cost for each control value for the period between each iteration to be the square of the total accumulated power consumption after the iteration. This control cost function would favor more uniform usage among nodes in the network. Since we do not address the control action selection issue, we simply select the first control action generated by our simulation tool when there are multiple actions available. In the place of planning, we manually relax the constraints whenever the given objective is no longer achievable.

Table 2 shows our simulation results. In this scenario, we assume the sensor in node 3 fails during the first iteration. Hence, in the first iteration, the diagnosis algorithm detects this failure and sets the diagnosis as an observation for future applications of the control algorithm. As shown in the table, node 3 is never used as a sensor again. During the 5th iteration, the control action generated uses both nodes 1 and 2 as sensors. This is because our initial constraints force node 4 to no longer be used as a sensor; as a consequence, in achieving the objective, the remaining option is to use both nodes 1 and 2 as sensors. In the 6th iteration, our control algorithm first indicates that the control objective is no longer achievable without exhausting the battery. We then manually relax the constraints to allow node 4 to run out of power.

6 Conclusion

We have proposed a novel approach for integrating diagnostic and control inferences within a single formalism. We have shown how to extend the causal network diagnostic approach to compute control and reconfiguration actions. This extension allows us to develop analytical results for discrete event system control and reconfiguration similar to those already developed for diagnostic inference. We also illustrated our approach with a simple wireless sensor network example.

Our work is most closely related to the Livingstone system [12]. Our approach shares with Livingstone the computation of system state (or assumable binding) and recovery actions, called mode identification and mode reconfiguration, respectively, in [12]. Compared to our approach, Livingstone uses a different transition logic and algorithms, and it lacks the ability to compute arbitrary controls, relying on pre-specified control.

Although our approach currently generates only next-step control actions, in future work we plan to compute multiple-step control actions by modeling the underlying systems as *temporal* causal networks [2].

References

[1] A. Darwiche. Compiling devices: A structure-based approach. In *Proc. 8th Intl. Conf. on Principles of Knowledge Representation and Reasoning (KR'98)*, pages 156–166, 1998.

[2] A. Darwiche and G. Provan. Exploiting system structure in model-based diagnosis of discrete-event systems. In *Proc. 7th Intl. Workshop on Principles of Diagnosis*, pages 95–105, 1996.

[3] Adnan Darwiche. Model-based diagnosis using structured system descriptions. *Journal of Artificial Intelligence Research*, 8:165–222, 1998.

[4] J. de Kleer and B. Williams. Diagnosis with behavioral modes. In *Proc. 1995 Intl. Joint Conf. on Artificial Intelligence (IJCAI'89)*, pages 1324–1330. Morgan-Kaufmann Publishers, 1989.

[5] T. Eiter and G. Gottlob. The complexity of logic-based abduction. *J. of ACM*, 42:3–42, 1995.

[6] G. Pottie, W. Kaiser, L. Clare, and H. Marcy. Wireless integrated network sensors. Technical report, UCLA, September 1998.

[7] G. Provan and Y.-L. Chen. Modeling, diagnosis, and control of timed discrete event systems using temporal causal networks. In *Proc. 1998 Intl. Workshop on Discrete Event Systems (WODES'98)*, pages 152–154, Cagliari, Italy, August 1998.

[8] G. Provan and Y.-L. Chen. Component-based modeling and diagnosis of process-control systems. In *Proc. 1999 IEEE Intl. Symposium on Computer-Aided Control System Design*, pages 194–199, Kohala Coast, HI, August 1999.

[9] P. J. Ramadge and W. M. Wonham. Supervisory control of a class of discrete event processes. *SIAM J. Control and Optimization*, 25(1):206–230, January 1987.

[10] H. Rauch. Automonous control reconfiguration. *IEEE Control. Systems Mag.*, 15(6):37–48, 1995.

[11] M. Sampath, R. Sengupta, S. Lafortune, K. Sinnamohideen, and D. Teneketzis. Diagnosability of discrete event systems. *IEEE Trans. Automatic Control*, 40(9):1555–1575, September 1995.

[12] B. Williams and P. Nayak. A model-based approach to reactive self-configuring systems. In *Proc. 13th National Conf. on Artifical Intelligence (AAAI-96)*, pages 971–978, 1996.