Chapter 6
Diagnosis of Active Systems with Candidate Priority

Gianfranco Lamperti

Abstract Diagnosis of an *active system* (AS), an asynchronous and distributed discrete-event system, is typically *abduction-based*: given a temporal observation, the diagnoses, or *candidates*, are generated based on a complete model of the AS, where a candidate is a set of faults explaining the temporal observation. A critical problem, which is common to all approaches of model-based diagnosis, is a large number of candidates: this is a serious threat to diagnosticians, owing to the cognitive overload imposed by an overwhelming stream of information and, worse still, to the uncertainty raising from a large set of different diagnoses. This criticality is exacerbated by assuming that both the candidates and the relevant recovery actions, possibly performed by an artificial agent, are required in real time, like in a nuclear power plant or in a defense system. Since candidates with low cardinality are more probable than candidates with high cardinality, it seems appropriate to generate candidates in ascending order of cardinality, from most to least likely. This way, an agent is not required to wait for the complete generation of candidates to perform the recovery actions that are associated with the most probable diagnoses. A diagnosis technique for ASs with prioritization of candidates is presented. Evidence from experimental results shows that the diagnosis technique is not only sound and complete, inasmuch all and only correct candidates are generated, but also effective in providing the most likely candidates upfront.

6.1 Introduction

Diagnosis has always been a challenging task for Artificial Intelligence. Up to the mid-eighties, all approaches to diagnosis were *heuristics-based*: the knowledge of a human expert in a specific domain was embedded in a software system in form of rules mapping symptoms to possible diagnoses. No *deep* knowledge (i.e., knowledge on how the system works) was required, but only the experience of a diagnostician. Sub-

G. Lamperti (✉)
Department of Information Engineering, University of Brescia, Brescia, Italy
e-mail: gianfranco.lamperti@unibs.it

I. Czarnowski et al. (eds.), *Intelligent Decision Technologies*, Smart Innovation, Systems and Technologies 411, https://doi.org/10.1007/978-981-97-7419-7_6

sequently, the diagnosis became *model-based*: a model of the normal behavior of the system is given in input to a diagnostic engine, which, based on a set of observations, generates the relevant diagnoses, called *candidates*, a candidate being a set of faulty components. Model-based diagnosis is grounded on the seminal works of Reiter [13], where a general theory of diagnosis is formalized in first-order logic, and de Kleer and Williams [6], where a *general diagnostic engine* is presented and experimented in the domain of troubleshooting digital circuits. In contrast with heuristics-based diagnosis, model-based diagnosis does not require any domain-dependent human expertise, but only a model of the *normal* (correct) behavior of the system: the discrepancy between the predicted (correct) behavior and the observed (faulty) behavior suffices to enumerate the candidates (diagnoses). The process for candidate generation requires two steps: (1) identification of the *conflict sets*, and (2) computation of the *hitting sets*. A conflict set is a set of components (e.g., devices in a digital circuit) such that, assuming that all of them behave correctly is logically inconsistent, in other words, at least one of them must be faulty: this is why this approach is also called *consistency-based*. Hitting sets are then generated from conflict sets in such a way that each hitting set intersects all conflict sets, that is, it includes at least one component from each conflict set. For instance, given the conflict sets $\{a, b\}$ and $\{a, c\}$, possible hitting sets are $\{a\}$ and $\{b, c\}$. Since the number of possible hitting sets may be very large, the diagnosis technique generates *minimal* hitting sets only: there is no hitting set that contains another hitting set. In our example, the hitting set $\{a, b\}$ is *not* minimal, as it includes $\{a\}$. Since the generation of both the conflict sets and the hitting sets are NP-hard problems, several techniques were proposed to make these two steps more efficient, possibly at the expense of completeness, including [1, 5, 16]. Assuming that the diagnoses are required under stringent time constraints, as is, for example, in a nuclear power plant or in a defense system, waiting for all the candidates may be less than desirable, as candidates with low cardinality (including few components) are generally more probable (hence, more valuable) than candidates with high cardinality (including many components). This is why it is paramount to the viability of a diagnostic system that candidates are generated in ascending order of cardinality, from most to least likely, so that a (possibly artificial) agent is enabled to perform effective recovery actions in real time, based on most realistic diagnoses [15, 17]. Consistency-based diagnosis is not the only technique adhering to the model-based paradigm: when time-varying (dynamical) systems come into play, *abduction-based* diagnosis may be a better choice, especially for discrete-event systems (DESs), which are the subject of a vast literature in the Control Theory [3]. Unlike the consistency-based approach, abduction-based diagnosis requires a *complete* model of the system, including both normal and faulty behavior. When the DES is distributed, each component is modeled as a finite automaton, where each state transition may be qualified either as normal or faulty. A trajectory of the DES, namely, a sequence of component transitions from the initial state of the DES to a final state, generates a sequence of observations associated with some (observable) transitions, namely a *temporal observation*, which is regarded as a symptom of the DES: the diagnostic engine may generate the candidates (sets of faults) by finding out the trajectories of the DES that conform with a given temporal observation, each

trajectory associated with a (not necessarily distinct) candidate. Abduction-based diagnosis of DESs is grounded on the seminal work of Sampath et al. [14], where the notion of *diagnosability* of a DES is coined and a *diagnoser* is defined in order to support the online diagnostic task efficiently. Other approaches to the diagnosis of DESs include [2, 4, 8–10, 12]. For the same reasons expressed above for consistency-based diagnosis, *minimal* diagnosis of DESs is proposed in [18], where minimal diagnosability is studied and a minimal diagnoser is proposed. What makes the construction of a diagnoser impractical for a real DES, however, is the need to first generate the space of the DES, whose number of states is exponential at least in the number of components. This is why, in this paper, we assume the unavailability of both the space and the diagnoser of the AS. We also assume that the diagnosis is required as soon as possible in order to be handled by an artificial agent designed to perform relevant recovery actions in real time. We propose a diagnostic engine where candidates are generated in ascending order of cardinality, from most to least likely, so that the most valuable diagnoses are generated upfront.

6.2 System Modeling

An active system is a network of *components* that are modeled as communicating automata. Each component is equipped with some input and/or output *pins*, where each output pin is connected with an input pin of another component by a *link*. A component changes its state by a *transition* that is triggered by an *event* either occurring outside of the system or coming from another component, after which the event is consumed, while other events may be generated on output pins. The newly generated events are placed on input pins of other components via links, thereby possibly triggering a cascade of additional transitions of different components. The behavior of an active system is assumed to be asynchronous: only one component transition at a time can occur.

Example 1 Depicted on the top-left of Fig. 6.1 is an active system $\mathcal{P}$ (protection) that is designed to control a cooling system by means of two components: a transducer z (incorporating a temperature sensor) and a valve v, with a link from z to v. In normal conditions, when the temperature becomes high, the transducer commands the valve to open in order to let the cooling fluid flow; vice versa, when the temperature returns to normal, the transducer commands the valve to close. Outlined on the bottom-left of the figure are the communicating automata of z (top) and v (bottom). Specifically, the model of z involves two states and four transitions, while the model of v involves two states and eight transitions. Generally speaking, each component transition from a state s to a state s' that is triggered by an input event e and generates a set of output events E is denoted by a triple $\langle s, (e, E), s' \rangle$. Component transitions in $\mathcal{P}$ are detailed in Table 6.1.

Starting from an initial state x_0, an active system reacts to a triggering event by a sequence of component transitions that moves the active system from x_0 to another

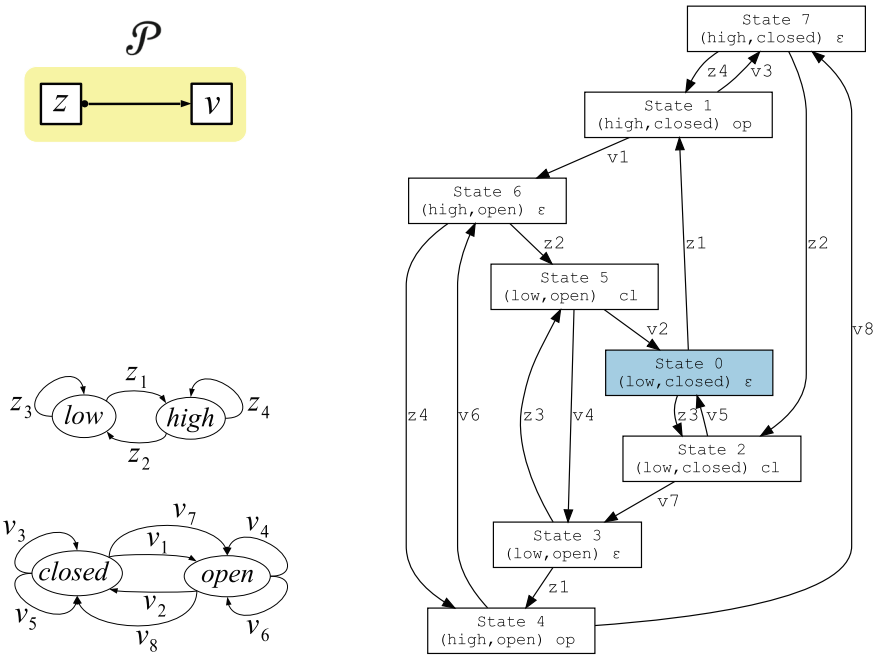

Fig. 6.1 Top-left: active system $\mathcal{P}$, involving transducer z and valve v; bottom-left: models of z (top) and v (bottom); right: *Space*($\mathcal{P}$), where the (filled) initial state is 0

Table 6.1 Details of transitions of z and v (cf. component models in Fig. 6.1)

Component transition	Details
$z_1 = \langle low, (ko, \{op\}), high\rangle$	The transducer detects high temperature and generates the open event
$z_2 = \langle high, (ok, \{cl\}), low\rangle$	The transducer detects low temperature and generates the close event
$z_3 = \langle low, (ko, \{cl\}, low\rangle$	The transducer detects high temperature, yet generates the close event
$z_4 = \langle high, (ok, \{op\}), high\rangle$	The transducer detects low temperature, yet generates the open event
$v_1 = \langle closed, (op, \emptyset), open\rangle$	The valve reacts to the open event by opening
$v_2 = \langle open, (cl, \emptyset), closed\rangle$	The valve reacts to the close event by closing
$v_3 = \langle closed, (op, \emptyset), closed\rangle$	The valve does not react to the open event and remains closed
$v_4 = \langle open, (cl, \emptyset), open\rangle$	The valve does not react to the close event and remains open
$v_5 = \langle closed, (cl, \emptyset), closed\rangle$	The valve reacts to the close event by remaining closed
$v_6 = \langle open, (op, \emptyset), open\rangle$	The valve reacts to the open event by remaining open
$v_7 = \langle closed, (cl, \emptyset), open\rangle$	The valve reacts to the close event by opening
$v_8 = \langle open, (op, \emptyset), closed\rangle$	The valve reacts to the open event by closing

state x, called a *trajectory*. The (possibly infinite) set of trajectories of the active system is the language of a *finite* automaton, called the *space* of the active system.

Definition 1 The *space* of an active system $\mathcal{X}$ is a finite automaton,

$$Space(\mathcal{X}) = (\Sigma, X, \tau, x_0) \tag{6.1}$$

where the alphabet Σ is the set of component transitions, X is the set of states, where a state is a pair (C, L), with C being the array of states of components and L being the array of the (possibly empty) events within links[1], τ is the transition function mapping a state and a component transition into a new state, $\tau : X \times \Sigma \mapsto X$, and x_0 is the initial state.

Example 2 The space of $\mathcal{P}$ (cf. Example 1), namely $Space(\mathcal{P})$, is shown on the right side of Fig. 6.1, where boxes and arcs indicate states and transitions, respectively, with initial state 0. Each state, which is identified by a number in $0\,..\,7$, embodies the pair of the component states of z and v, as well as the (possibly empty) event in the link. Due to cycles in the space, there is an infinite number of trajectories of $\mathcal{P}$, such as $T = [z_3, v_5, z_1, v_3, z_4, v_3, z_2, v_5]$, which, incidentally, terminates in the initial state.

6.3 Observability and Abnormality

The specification of an active system as given is insufficient for diagnosis purposes: it needs to be augmented with information about the *observability* and the *abnormality* of the system, both of them being specified in a *mapping table*.

Definition 2 Let $\mathbf{T}$ be the set of component transitions in an active system $\mathcal{X}$, let $\mathbf{O}$ be a finite set of *observations* for $\mathcal{X}$, let $\mathbf{F}$ be a finite set of *faults* for $\mathcal{X}$, and let ε denote the *empty* symbol. The *mapping table* of $\mathcal{X}$ is a function:

$$Map(\mathcal{X}) : \mathbf{T} \mapsto (\mathbf{O} \cup \{\varepsilon\}) \times (\mathbf{F} \cup \{\varepsilon\}). \tag{6.2}$$

In practice, $Map(\mathcal{X})$ can be represented as a set of triples (t, o, f), with $t \in \mathbf{T}$, $o \in \mathbf{O} \cup \{\varepsilon\}$, and $f \in \mathbf{F} \cup \{\varepsilon\}$, where each triple defines the observability and abnormality of t, specifically: if $o \neq \varepsilon$, then t is *observable*, else t is *unobservable*; also, if $f \neq \varepsilon$, then t is *faulty*, else t is *normal*.

Example 3 The mapping table of active system $\mathcal{P}$ is listed on the left side of Table 6.2, where $\mathbf{O} = \{oz, ov\}$ and $\mathbf{F} = \{fz_3, fz_4, fv_3, fv_4, fv_7, fv_8\}$, which are detailed on the right side of the figure. Only one observation label is provided for both the transducer and the valve, namely oz and ov, respectively, each being associated with

[1] Formally, an empty link contains an empty event, denoted ε.

Table 6.2 Mapping table $Map(\mathcal{P})$ (left), and details of observations and faults (right)

t	o	f
z_1	oz	ϵ
z_2	oz	ϵ
z_3	ϵ	fz_3
z_4	ϵ	fz_4
v_1	ov	ϵ
v_2	ov	ϵ
v_3	ϵ	fv_3
v_4	ϵ	fv_4
v_5	ov	ϵ
v_6	ov	ϵ
v_7	ov	fv_7
v_8	ov	fv_8

o	*Observation details*
oz	The transducer performs a normal action
ov	The valve reacts (possibly abnormally) to an event

f	*Fault details*
fz_3	The transducer generates the *cl* event instead of *op*
fz_4	The transducer generates the *op* event instead of *cl*
fv_3	The valve remains closed upon the open command
fv_4	The valve remains open upon the close command
fv_7	The valve opens upon the close command
fv_8	The valve closes upon the open command

several (still not all) transitions. For instance, transition z_1 is observable and normal, z_3 is unobservable and faulty, whereas v_7 is both observable and faulty. Owing to the possible association of the same observation with *several* transitions, uncertainty still remains in determining the actual component transition based solely on the observation occurred.

According to a mapping table, each trajectory of an active system can be associated with both a *temporal observation* and a *diagnosis*.

Definition 3 Let T be a trajectory of an active system $\mathcal{X}$. The *temporal observation* of T is the sequence of the observations associated with the component transitions in T,

$$Obs(T) = \left[\, o \mid t \in T, (t, o, f) \in Map(\mathcal{X}), o \neq \varepsilon \,\right]. \tag{6.3}$$

A temporal observation $\mathcal{O}$ *conforms* with a trajectory T iff $\mathcal{O} = Obs(T)$.[2]

Example 4 According to the mapping table $Map(\mathcal{P})$ in Table 6.2 and considering the trajectory $T = [z_3, v_5, z_1, v_3, z_4, v_3, z_2, v_5]$, we have $Obs(T) = [ov, oz, oz, ov]$.

Definition 4 Let T be a trajectory of an active system $\mathcal{X}$. The *diagnosis* of T is the set of faults associated with the component transitions in T,

$$Dgn(T) = \{\, f \mid t \in T, (t, o, f) \in Map(\mathcal{X}), f \neq \varepsilon \,\}. \tag{6.4}$$

A diagnosis δ *explains* a temporal observation $\mathcal{O}$ if $\delta = Dgn(T)$ and $\mathcal{O}$ conforms with T. The *cardinality* (number of faults) of δ is denoted $|\delta|$.

[2] The same temporal observation $\mathcal{O}$ may conform with several (possibly infinite) trajectories.

Example 5 Considering $Map(\mathcal{P})$ in Table 6.2 and $T = [z_3, v_5, z_1, v_3, z_4, v_3, z_2, v_5]$ in Example 2, we have $Dgn(T) = \{fz_3, fv_3, fz_4\}$.

Since a temporal observation $\mathcal{O}$ may conform with several trajectories of the active system, several (candidate) diagnoses may explain $\mathcal{O}$.

Definition 5 Let $\mathcal{O}$ be a temporal observation of an active system $\mathcal{X}$. The *candidate set* of $\mathcal{O}$ is the set of diagnoses of the trajectories of $\mathcal{X}$ conforming with $\mathcal{O}$,

$$\Delta(\mathcal{O}) = \{\delta \mid \delta = Dgn(T), T \in Space(\mathcal{X}), \mathcal{O} = Obs(T)\}. \tag{6.5}$$

Example 6 Let $\mathcal{O} = [ov, oz, oz, ov]$ be a temporal observation of active system $\mathcal{P}$. Based on $Space(\mathcal{P})$ in Fig. 6.1 and $Map(\mathcal{P})$ in Table 6.2, it is easy to find out that $\Delta(\mathcal{O})$ includes six diagnoses, or *candidates*, namely $\{fz_3, fv_3\}$, $\{fz_3, fz_4, fv_3\}$, $\{fz_3, fv_3, fv_7\}$, $\{fz_3, fz_4, fv_3, fv_7\}$, $\{fz_3, fv_3, fv_4, fv_7\}$, and $\{fz_3, fz_4, fv_3, fv_4, fv_7\}$.

6.4 Diagnosis Abduction

The candidate set of a temporal observation $\mathcal{O}$ of an active system $\mathcal{X}$ can be determined by generating a subspace of $Space(\mathcal{X})$ involving all and only the trajectories of $\mathcal{X}$ that conform with $\mathcal{O}$, called a *diagnosis abduction*.

Definition 6 Let $\mathcal{O} = [o_1, \ldots, o_n]$ be a temporal observation of an active system $\mathcal{X}$, with $Space(\mathcal{X}) = (\Sigma, X, \tau, x_0)$ and set of faults $\mathbf{F}$ involved in $Map(\mathcal{X})$. The *diagnosis abduction* of $\mathcal{X}$ based on $\mathcal{O}$ is a finite automaton,

$$Abd(\mathcal{X}, \mathcal{O}) = \left(\Sigma, A, \tau', a_0, A_\mathrm{f}\right) \tag{6.6}$$

where A is the set of states, with each state being a triple (x, δ, i), where $x \in X$, $\delta \in 2^{\mathbf{F}}$, and $i \in [0\,..\,n]$ is an *index* of $\mathcal{O}$; $a_0 = (x_0, \emptyset, 0)$ is the initial state; $A_\mathrm{f} \subseteq A$ is the set of final states, where the index i in each final state equals n (o_n being the last observation in $\mathcal{O}$); and $\tau' : A \times \Sigma \mapsto A$ is the transition function, where $\delta'((x, \delta, i), t) = (x', \delta', i')$ iff $\tau(x, t) = x'$, and, being (t, o, f) the triple in $Map(\mathcal{X})$: if $f = \varepsilon$ then $\delta' = \delta$ else $\delta' = \delta \cup \{f\}$, and, if $o = \varepsilon$ then $i' = i$, else, if $i < n$ and $o = o_{i+1}$, then $i' = i + 1$.

Example 7 Let $\mathcal{O} = [ov, oz, oz, ov]$ be a temporal observation of active system $\mathcal{P}$. Based on $Map(\mathcal{P})$ (Table 6.2), depicted in Fig. 6.2 is $Abd(\mathcal{P}, \mathcal{O})$, which includes 21 states (labeled $0, \ldots, 20$), where the initial state is 0, while the (twelve) final states are in bold. Each state is marked with a state in $Space(\mathcal{P})$ expressed as a pair of component states, a (possibly empty) event in the link, a vector of bits denoting a diagnosis, where each bit b_k, $k \in [1\,..\,6]$, indicates whether the k-th fault in the list $[fz_3, fz_4, fv_3, fv_4, fv_7, fv_8]$ is involved ($b_k = 1$) or not ($b_k = 0$) in the diagnosis, and an index $i \in [0\,..\,4]$ of $\mathcal{O}$. For instance, state 10 indicates that the states of z and v are

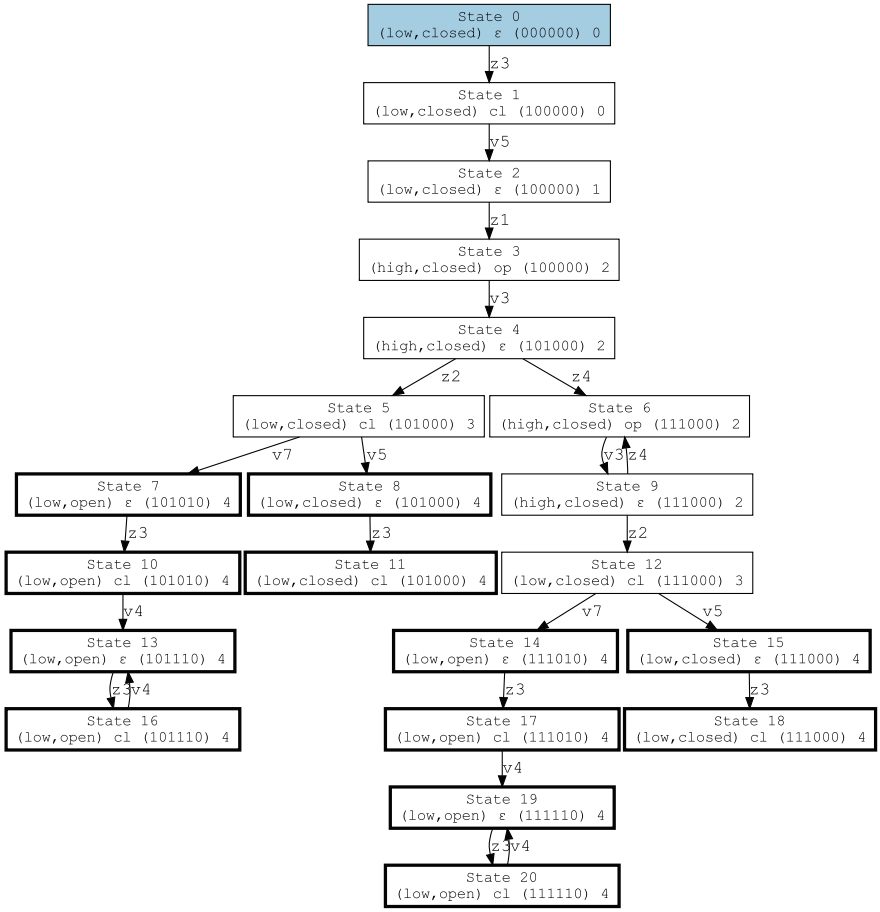

Fig. 6.2 $Abd(\mathcal{P}, \mathcal{O})$: the diagnosis abduction of $\mathcal{P}$ based on the temporal observation $\mathcal{O} = [ov, oz, oz, ov]$, with initial state 0 and final states being depicted in bold

low and *open*, respectively, the link contains event *cl*, the diagnosis is $\{fz_3, fv_3, fv_7\}$, and the index of $\mathcal{O}$ is $i = 4$ (hence, the trajectory ending in state 10 conforms with $\mathcal{O}$).

Proposition 1 *Let $\mathcal{O} = [o_1, \ldots, o_n]$ be a temporal observation of an active system $\mathcal{X}$, and let A_f be the set of final states in $Abd(\mathcal{X}, \mathcal{O})$. We have*

$$\Delta(\mathcal{O}) = \{\delta \mid (x, \delta, n) \in A_f\}\,. \tag{6.7}$$

Example 8 Based on $Abd(\mathcal{P}, \mathcal{O})$ in Fig. 6.2, the diagnoses associated with the final states are $\{fz_3, fv_3\}$ (states 8 and 11), $\{fz_3, fz_4, fv_3\}$ (states 15 and 18), $\{fz_3, fv_3, fv_7\}$ (states 7 and 10), $\{fz_3, fz_4, fv_3, fv_7\}$ (states 14 and 17), $\{fz_3, fv_3, fv_4, fv_7\}$ (states 13 and

16), and $\{fz_3, fz_4, fv_3, fv_4, fv_7\}$ (states 19 and 20), which, as claimed in Proposition 1, collectively constitute the candidate set $\Delta(\mathcal{O})$ determined in Example 6.

6.5 Prioritized Diagnosis Engine

Since not all candidates are equally probable, candidates are to be generated in ascending order based on their cardinality, on the grounds that the lower the cardinality, the more likely the candidate is the actual diagnosis (the diagnosis of the actual trajectory).

Definition 7 Let $\Delta(\mathcal{O}) = \{\delta_1, \ldots, \delta_m\}$ be a candidate set. An *ascending ordering* of $\Delta(\mathcal{O})$ is a sequence $\Delta^* = [\delta'_1, \ldots, \delta'_m]$ such that $\{\delta'_1, \ldots, \delta'_m\} = \{\delta_1, \ldots, \delta_m\}$ and $\forall k \in [1 \,..\, (m-1)] \left(|\delta'_k| \leq |\delta'_{k+1}|\right)$.

Example 9 Let $\Delta(\mathcal{O}) = \{\delta_1, \delta_2, \delta_3, \delta_4, \delta_5, \delta_6\}$, where $\delta_1 = \{fz_3, fz_4, fv_3\}$, $\delta_2 = \{fz_3, fz_4, fv_3, fv_7\}$, $\delta_3 = \{fz_3, fz_4, fv_3, fv_4, fv_7\}$, $\delta_4 = \{fz_3, fv_3\}$, $\delta_5 = \{fz_3, fv_3, fv_7\}$, and $\delta_6 = \{fz_3, fv_3, fv_4, fv_7\}$. An ascending ordering of $\Delta(\mathcal{O})$ is $[\delta_4, \delta_1, \delta_5, \delta_2, \delta_6, \delta_3]$, where $|\delta_4| = 2$, $|\delta_1| = |\delta_5| = 3$, $|\delta_2| = |\delta_6| = 4$, and $|\delta_3| = 5$.

The diagnosis engine is expected to output Δ^* incrementally so that more probable candidates are generated *before* less probable ones. This way, a (software) agent designed to monitor the active system can possibly perform relevant recovery actions in real time. An algorithm, called PRIORITIZED DIAGNOSIS ENGINE, for the incremental generation of Δ^*, is listed below (lines 1–18).[3] The idea is to generate the abduction $Abd(\mathcal{X}, \mathcal{O})$ based on an ascending cardinality of the candidates, namely $\mathcal{C}$.[4] Only when the index of $\mathcal{O}$ is complete ($i = n$), is a candidate δ generated, where $|\delta| = \mathcal{C}$, provided it is not in Δ^* already (lines 5 and 6), thereby avoiding duplication of the same candidate. The loop in lines 7–15 generates the transitions exiting the considered unmarked state y, by computing each target state $y' = (x', \delta', i')$ based on $Map(\mathcal{X})$. When there is no unmarked state with $|\delta| = \mathcal{C}$, the cardinality $\mathcal{C}$ is incremented (line 17), and the main loop (lines 3–18) is iterated based on the new cardinality. The algorithm stops when all states are marked, in other words, when $Abd(\mathcal{X}, \mathcal{O})$ is complete.

Example 10 Based on $Abd(\mathcal{P}, \mathcal{O})$ in Fig. 6.2, traced in Table 6.3 is the execution of PRIORITIZED DIAGNOSIS ENGINE for $\mathcal{P}$, where $\mathcal{O} = [ov, oz, oz, ov]$.[5] Within the

[3] The input parameter $Space(\mathcal{X})$ is assumed not to be materialized, since in real applications its generation is prohibitive owing to the exploding number of states. Hence, the loop statement in line 7, which considers each transition in $Space(\mathcal{X})$, is only a formal specification that is implemented in practice by considering each transition that is triggerable in state x, the latter being expressed as the vector of component states and the vector of events within links.

[4] As a side effect in constructing the abduction, the algorithm generates *spurious* states also, that is, states that are not connected to any final state, which are therefore irrelevant.

[5] Spurious states are missing in Table 6.3, as they do not contribute to candidates (cf. footnote 4).

Algorithm 1: PRIORITIZED DIAGNOSIS ENGINE

input : $\mathcal{X}$: active system with $Space(\mathcal{X}) = (\Sigma, X, \tau, x_0)$ and mapping table $Map(\mathcal{X})$,
$\mathcal{O} = [o_1, \ldots, o_n]$: a temporal observation of $\mathcal{X}$
output: $Abd(\mathcal{X}, \mathcal{O}) = (\Sigma, Y, \tau', y_0, F)$: the diagnosis abduction of $\mathcal{X}$ based on $\mathcal{O}$,
Δ^*: an ascending ordering of the candidate set $\Delta(\mathcal{O})$ generated incrementally

```
1  Δ* ← [ ], C ← 0        // C denotes the (increasing) cardinality of candidates
2  Create the unmarked initial state of Abd(X, O), namely y0 = (x0, ∅, 0)
3  repeat
4      while there is an unmarked state y = (x, δ, i) in Y where |δ| = C do
5          if i = n and δ ∉ Δ* then
6              Output δ and append it to Δ*
7          foreach transition ⟨x, t, x'⟩ in τ do
8              Let (t, o, f) be the triple in Map(X) relevant to component transition t
9              if o = ε or (i < n and o = o_{i+1}) then
10                 i' ← if o ≠ ε then i + 1 else i
11                 δ' ← if f ≠ ε then δ ∪ {f} else δ
12                 y' ← (x', δ', i')
13                 if y' ∉ Y then
14                     Insert y' into Y
15                 Insert ⟨y, t, y'⟩ into τ'
16         Mark y
17     C ← C + 1
18 until all states in Y are marked.
```

Table 6.3 Tracing of algorithm PRIORITIZED DIAGNOSIS ENGINE (cf. Fig. 6.2)

Cardinality ($\mathcal{C}$)	New states in $Abd(\mathcal{P}, \mathcal{O})$	Candidates generated
0	0, 1	
1	2, 3, 4	
2	5, 6, **7**, **8**, **11**	$\{fz_3, fv_3\}$
3	9, **10**, 12, **15**, **18**, **13**, **14**	$\{fz_3, fz_4, fv_3\}$, $\{fz_3, fv_3, fv_7\}$
4	**16**, **17**, **19**	$\{fz_3, fz_4, fv_3, fv_7\}$, $\{fz_3, fv_3, fv_4, fv_7\}$
5	**20**	$\{fz_3, fz_4, fv_3, fv_4, fv_7\}$

loop in lines 4–16, for each cardinality $\mathcal{C} \in [0\,..\,5]$, both the set of new states in $Abd(\mathcal{P}, \mathcal{O})$ and the set of new candidates are shown. Final states are in bold. Boxed (yellow) states are those generated for a given $\mathcal{C}$ but their cardinality is actually $\mathcal{C} + 1$. This comes from the loop in lines 7–15 generating the transition function of an unmarked state y, which amounts to computing *all* the transitions $\langle y, t, y' \rangle$ exiting

y, with $y' = (x', \delta', i')$, where $|\delta'|$ is possibly $\mathcal{C} + 1$ (when t is faulty) rather than $\mathcal{C}$. The candidates associated with these extra states are output at the next iteration, after incrementing $\mathcal{C}$ (lines 17).

6.6 Experimental Results

The diagnosis technique presented in this paper was implemented in the *C* programming language, under the *Linux* operating system (distribution *Kubuntu* 20.04), running on a laptop with 16GB of working memory. The software package allows for the specification of active systems by a specially-designed language. Once the description of an active system is compiled into internal data structures, the relevant space, such as $Space(\mathcal{P})$ in Fig. 6.1, can be generated as a graph specified in the *dot* language by exploiting the *Graphviz* package.[6] If a temporal observation is given, the relevant abduction can be generated also, like $Abd(\mathcal{P}, \mathcal{O})$ in Fig. 6.2. Two engines have been implemented for the generation of candidates: DIAGNOSIS ENGINE, which outputs the candidates without any prioritization, and PRIORITIZED DIAGNOSIS ENGINE, the algorithm presented in this paper. The two algorithms were compared based on several experiments. Shown in Fig. 6.3 are the results relevant to the abduction displayed in Fig. 6.2 for both DIAGNOSIS ENGINE (left) and PRIORITIZED DIAGNOSIS ENGINE (right). Specifically, in each bar chart, the x-axis indicates the order number of the candidate generated, while the y-axis on the left indicates the cardinality of the candidate. Hence, a bar in position $k \geq 1$ having height h corresponds to the output of the k-th candidate, whose cardinality is h. For instance, the first candidate generated by DIAGNOSIS ENGINE includes three faults, while the first candidate generated by PRIORITIZED DIAGNOSIS ENGINE includes two faults. The additional y-axis on the right indicates the processing time (in μs) spent so far, whose value is represented as a red bullet. For instance, the time spent by DIAGNOSIS ENGINE up to the generation of the fourth candidate, is 22 µs, while the generation of the same (most probable) candidate by PRIORITIZED DIAGNOSIS ENGINE is almost instantaneous. DIAGNOSIS ENGINE completes the generation of the candidate set in 26 µs, while PRIORITIZED DIAGNOSIS ENGINE takes 27 µs. Albeit these figures have little significance owing to the small dimension of the problem, they nevertheless reflect the need for a slight additional computation by PRIORITIZED DIAGNOSIS ENGINE for ranking and searching the candidates. On the other hand, unlike DIAGNOSIS ENGINE which generates an unordered set of candidates, PRIORITIZED DIAGNOSIS ENGINE generates the candidates in ascending order, thereby providing more probable candidates as soon as possible.

To compare more significant figures, another experiment with a different mapping table of $\mathcal{P}$ is presented, which involves ten faults and just one observation label. Given in input a temporal observation of length 60, the diagnosis results (including 205 candidates) are shown in Fig. 6.4, where the processing time is expressed in

[6] *Graphviz* is a shorthand for *Graph Visualization Software*, which is available at *graphviz.com*.

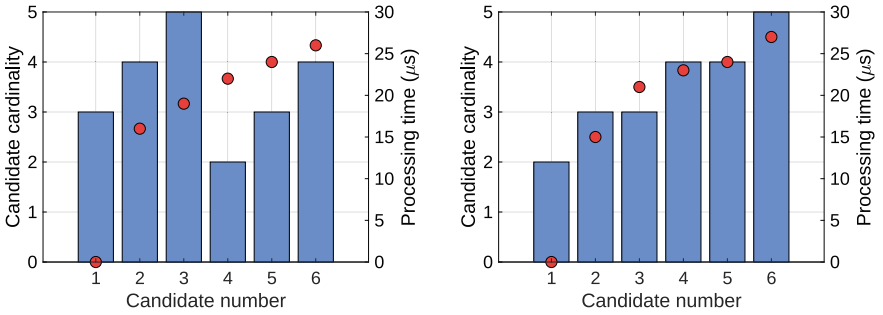

Fig. 6.3 Comparison for $\mathcal{P}$ (cf. $Abd(\mathcal{P}, \mathcal{O})$ in Fig. 6.2): DIAGNOSIS ENGINE (left) and PRIORITIZED DIAGNOSIS ENGINE (right), with processing time expressed in μs

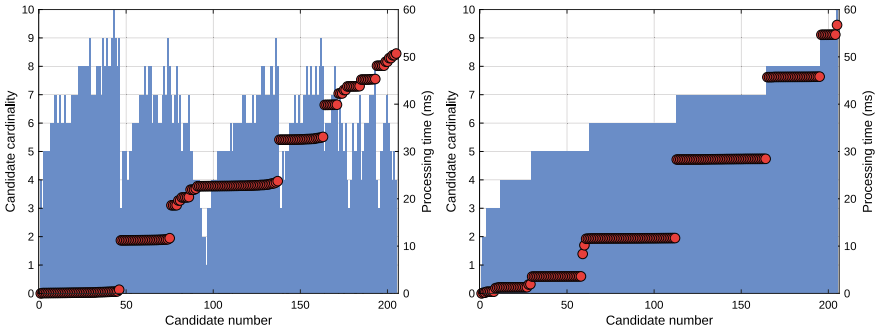

Fig. 6.4 Compared results (including 205 candidates) for a variation of the mapping table and the temporal observation of $\mathcal{P}$: DIAGNOSIS ENGINE (left) and PRIORITIZED DIAGNOSIS ENGINE (right), with processing time expressed in ms

ms. For the generation of the candidate set, DIAGNOSIS ENGINE and PRIORITIZED DIAGNOSIS ENGINE take 50ms and 56ms, respectively, with 55511 states generated in the abduction. The most probable candidate, however, is generated by DIAGNOSIS ENGINE after 22,67ms (in position 96), while it is generated by PRIORITIZED DIAGNOSIS ENGINE almost instantaneously (in first position). Even the two candidates with cardinality 2 (positions 2 and 3) are generated after 0.15ms by PRIORITIZED DIAGNOSIS ENGINE, while they are generated by DIAGNOSIS ENGINE after 22,66ms (positions 94 and 95). In comparison with DIAGNOSIS ENGINE, the generation of the three most probable candidates (one with cardinality 1 and two with cardinality 2) by PRIORITIZED DIAGNOSIS ENGINE allows for saving 99.33% of the time. If we add to the set of most probable candidates those with cardinality 3 and do the math, the time saved is 97.36%.

Results of a third experiment relevant to a different AS composed of five components are outlined in Fig. 6.5, where the model of each component involves two states and six transitions, four of which are spontaneous (that is, not triggered by events in links) and three of which are unobservable, thereby resulting in a diagnosis

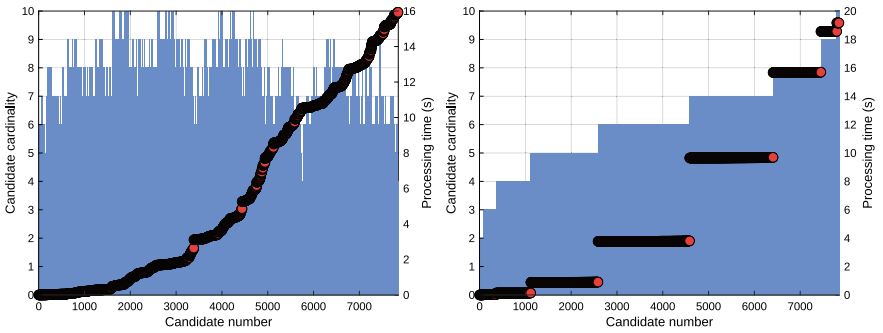

Fig. 6.5 Compared results (including 7848 candidates) for a DES with five components: DIAGNOSIS ENGINE (left) and PRIORITIZED DIAGNOSIS ENGINE (right), with processing time expressed in seconds

abduction with a large number of states (1967090). DIAGNOSIS ENGINE and PRIORITIZED DIAGNOSIS ENGINE took 15.93s and 19.20s, respectively, for generating all candidates (7848), with cardinality ranging from 1 to 10. Focusing on the most likely (low cardinality) candidates, however, figures are telling: the generation of singleton candidates is completed by DIAGNOSIS ENGINE and PRIORITIZED DIAGNOSIS ENGINE in 15.927152s and 0.000694s, respectively, while, considering candidates with one or two faults, the times are 15.930147s and 0.015191s, respectively: compared to DIAGNOSIS ENGINE, the time saved by PRIORITIZED DIAGNOSIS ENGINE for generating the most probable candidates are 99,9956% (one fault) and 99,9046% (one or two faults).

6.7 Conclusion

In the literature on (a posteriori) diagnosis of ASs, the (sound and complete) set of candidates has always been generated without any particular order. Since low-cardinality candidates are more likely than high-cardinality candidates, good heuristics is focusing on the most likely candidates (diagnoses with few faults). This is essential in diagnostic environments that are required to provide the most probable diagnoses as soon as possible so that relevant recovery actions are enabled (possibly automatically) in real time. The PRIORITIZED DIAGNOSIS ENGINE proposed in this paper serves this purpose: candidates are produced in ascending order by cardinality so that most probable diagnoses are generated upfront. Experimental results confirm the effectiveness of the approach. In the future, candidate prioritization may be extended to monitoring-based diagnosis of ASs [9], diagnosis of deep ASs [11], and sequence-oriented diagnosis of ASs [7].

Acknowledgements We acknowledge the support of the PNRR project FAIR - Future AI Research (PE00000013), Spoke 9 - Green-aware AI, under the NRRP MUR program funded by the NextGen-

erationEU, specifically by the project Argumentation for Informed Decisions with Applications to Energy Consumption in Computing – AIDECC (CUP D53C24000530001).

References

1. Abreu, R., van Gemund, A.: A low-cost approximate hitting set algorithm and its application to model-based diagnosis. In: Eighth Symposium on Abstraction, Reformulation, and Approximation (SARA'09), pp. 2–9. AAAI Press (2009)
2. Baroni, P., Lamperti, G., Pogliano, P., Zanella, M.: Diagnosis of large active systems. Artif. Intell **110**(1), 135–183 (1999). https://doi.org/10.1016/S0004-3702(99)00019-3
3. Cassandras, C., Lafortune, S.: Introduction to Discrete Event Systems, 2nd edn. Springer, New York (2008)
4. Grastien, A., Cordier, M., Largouët, C.: Incremental diagnosis of discrete-event systems. In: Sixteenth International Workshop on Principles of Diagnosis (DX 2005), pp. 119–124. Monterey, CA (2005)
5. de Kleer, J.: Hitting set algorithms for model-based diagnosis. In: 22nd International Workshop on Principles of Diagnosis (DX 2011), pp. 60–67. Murnau, Germany (2011)
6. de Kleer, J., Williams, B.: Diagnosing multiple faults. Artif. Intell. **32**(1), 97–130 (1987)
7. Lamperti, G., Trerotola, S., Zanella, M., Zhao, X.: Sequence-oriented diagnosis of discrete-event systems. J. Artif. Intell. Res. **78**, 69–141 (2023). https://doi.org/10.1613/jair.1.14630
8. Lamperti, G., Zanella, M.: Diagnosis of discrete-event systems from uncertain temporal observations. Artif. Intell. **137**(1–2), 91–163 (2002). https://doi.org/10.1016/S0004-3702(02)00123-6
9. Lamperti, G., Zanella, M.: Monitoring of active systems with stratified uncertain observations. IEEE Trans. Syst. Man Cybern. Part A Syst. Hum. **41**(2), 356–369 (2011). https://doi.org/10.1109/TSMCA.2010.2069096
10. Lamperti, G., Zanella, M., Zhao, X.: Introduction to Diagnosis of Active Systems. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-92733-6
11. Lamperti, G., Zanella, M., Zhao, X.: Diagnosis of deep discrete-event systems. J. Artif. Intell. Res. **69**, 1473–1532 (2020). https://doi.org/10.1613/jair.1.12171
12. Pencolé, Y., Cordier, M.: A formal framework for the decentralized diagnosis of large scale discrete event systems and its application to telecommunication networks. Artif. Intell. **164**(1–2), 121–170 (2005)
13. Reiter, R.: A theory of diagnosis from first principles. Artif. Intell. **32**(1), 57–95 (1987)
14. Sampath, M., Sengupta, R., Lafortune, S., Sinnamohideen, K., Teneketzis, D.: Failure diagnosis using discrete-event models. IEEE Trans. Control Syst. Technol. **4**(2), 105–124 (1996)
15. Siddiqi, S.: Computing minimum-cardinality diagnoses by model relaxation. In: Walsh, T. (ed.) Twenty-Second International Joint Conference on Artificial Intelligence (IJCAI 2011), vol. 2, pp. 1087–1092. AAAI Press, Barcelona, Spain (2011)
16. Stern, R., Kalech, M., Feldman, A., Provan, G.: Exploring the duality in conflict-directed model-based diagnosis. In: Twenty-Sixth AAAI Conference on Artificial Intelligence (AAAI-12), pp. 828–834 (2012)
17. Torasso, P., Torta, G.: Computing minimum-cardinality diagnoses using OBDDs. In: Günter, A., Kruse, R., Neumann, B. (eds.) KI-2003: Advances in Artificial Intelligence. Lecture Notes in Computer Science, vol. 2821, pp. 224–238. Springer, Berlin, Heidelberg (2003)
18. Zhao, X., Ouyang, D., Lamperti, G., Tong, X.: Minimal diagnosis and diagnosability of discrete-event systems modeled by automata. Complexity **2020**, 1–17 (2020). https://doi.org/10.1155/2020/4306261