Diagnosability Planning for Controllable Discrete Event Systems

Hassan IBRAHIM[1] Philippe DAGUE[1] Alban GRASTIEN[2]
Lina YE[1] Laurent SIMON[3]

[1]LRI, Univ. Paris-Sud and CNRS, Univ. Paris-Saclay, Orsay, France
hassan.ibrahim@lri.fr, philippe.dague@lri.fr, lina.ye@lri.fr

[2] Data61 and Australian National University, Canberra, Australia
alban.grastien@data61.csiro.au

[3]LaBRI, Univ. Bordeaux and CNRS, Bordeaux, France
lsimon@labri.fr

27th International Workshop on Principles of Diagnosis
Denver, October 7, 2016

Outlines

About Diagnosability

Diagnosability of a fault f

It is the possibility to distinguish any possible f-faulty behavior from any other non-f-faulty one within a finite time after the occurrence of f.

Non-Diagnosability of a fault f

The fault f is non-diagnosable iff it exists a pair of infinite trajectories observation-equivalent, one with f and the other without f.

Diagnosability of a System

All faults are diagnosable. We consider one fault at a time.

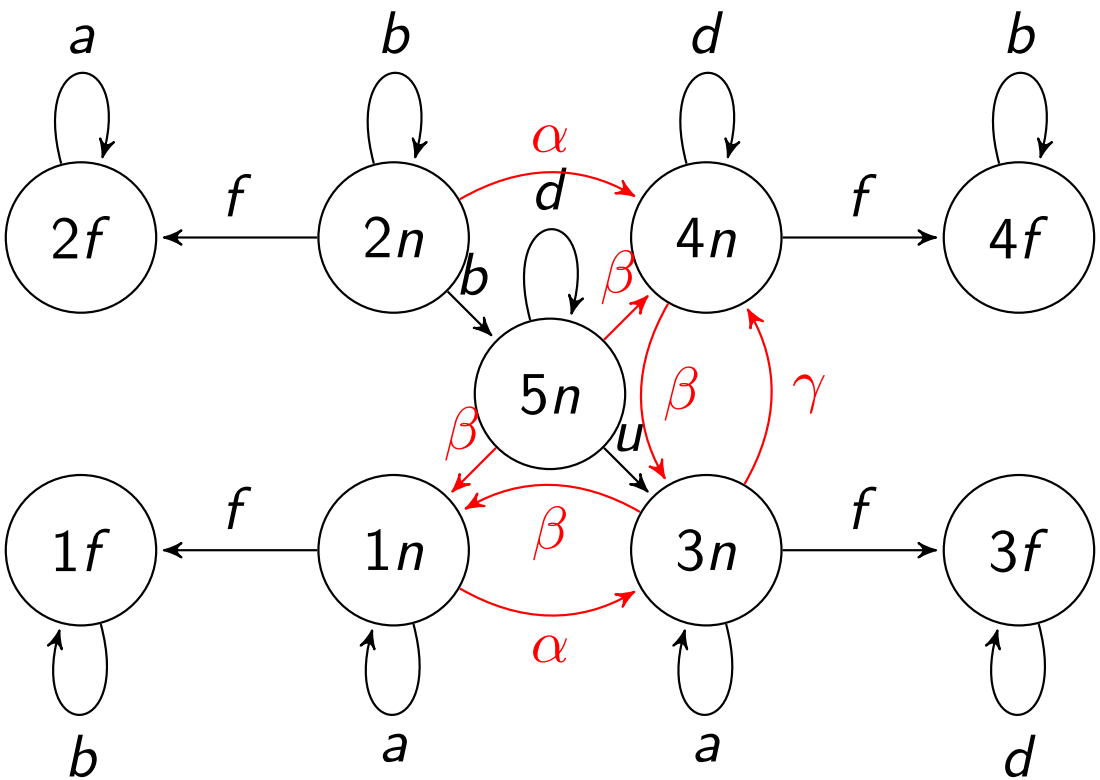
Motivation

- Diagnosability test is done generally only from one (or a small number of) given initial state.
- What happens when the system has to be stopped and restarted?
- For example following a fault occurrence and the execution of a repair plan?
- In general the new starting state will be different from the original initial state.
- With no guarantee for the system to be diagnosable from this new state, even if proved diagnosable from the initial state.
- In addition, following a disruption, the state of the system may be not precisely known.
- This uncertainty means that the starting state is actually a belief state, i.e., a set of possible states.

Problem statement

- Assume that, when not running freely, the system is partially controllable by a given set of (deterministic or not) predefined actions.
- Diagnosability planning problem: given one belief state (e.g., a possible output of a repair plan), determine if there exists a sequence of actions bringing the system from this belief state into a "safe" belief state, e.g., here a diagnosable one, from where it will start running freely again (up to the next disruption, e.g., the next fault occurrence followed by a repair action, where the process will be repeated).
- If it exists, compute such an (optimal) actions plan.
- Optimality being w.r.t. given criteria, e.g. minimizing the length of the plan or more generally its cost for given costs of elementary actions.

Example

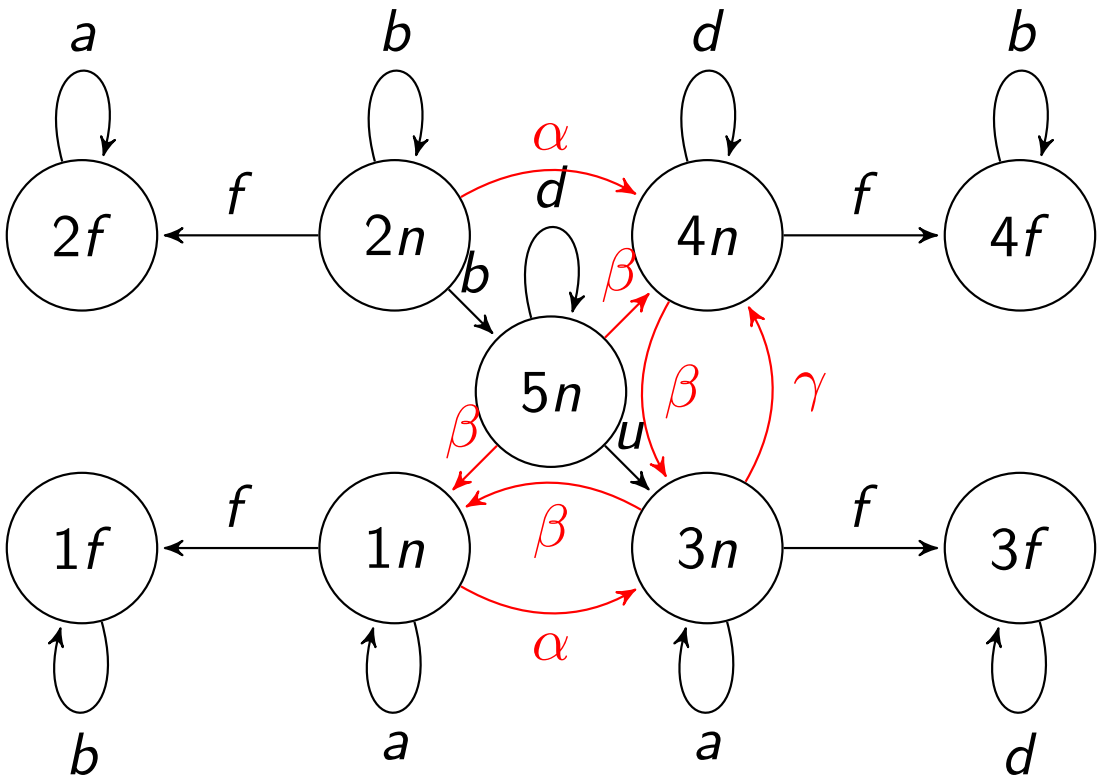


Autonomous system
f fault
u unobservable
a, b, d observable

Control
Exogenous actions
α, β, γ

The two modes
do not intertwine

Example



Assume initial belief state
$I = \{1n, 2n, 3n, 4n, 5n\}$
Non-diagnosable from I: a^+

Action α
Non-diagnosable from
$\alpha(I) = \{3n, 4n, 5n\}$: d^+

Action β
Non-diagnosable from
$\beta(\alpha(I)) = \{1n, 3n, 4n\}$: d^+

Action β
Diagnosable from
$\beta(\beta(\alpha(I))) = \{1n, 3n\}$
$\pi = \alpha\beta\beta$ diagnosability plan
(length-optimal)

Background I

Controllable Labeled Transition System (CLTS)

$G = \langle Q, \Sigma, \delta, I \rangle$, with $\Sigma = \mathcal{A} \cup \mathcal{E}$, where:

- Q finite set of states;
- $\mathcal{E} = \Sigma_o \cup \Sigma_u \cup \Sigma_f$ finite set of autonomous events (resp. observable, unobservable, faulty);
- $\mathcal{A}$ a finite set of (possibly nondeterministic) control actions (with every action applicable in every state);
- $\delta \subseteq Q \times \Sigma \times Q$ (active for labels in $\mathcal{E}$, reactive for labels in $\mathcal{A}$) transition relation;
- $I \subseteq Q$ initial belief state.

Background II

Notations

- (Active) path: $\rho = q_0 \xrightarrow{e_1} \cdots \xrightarrow{e_n} q_n$.
- Trajectory of ρ: $e_1 \dots e_n \in \mathcal{E}^*$, called I-trajectory if $q_0 \in I$.
- $L_I(G) \subseteq \mathcal{E}^*$: set of I-trajectories.
- s^f: I-trajectory ending by the fault f.
- $L_I(G)/s$: set of all extensions of s as I-trajectories.
- P: projection of $\mathcal{E}^*$ on Σ_o^*.

Assumptions

- $L_I(G)$ live.
- $L_I(G)$ convergent.

Background III

Definition of Diagnosability

$$\exists k \in \mathbb{N}, \forall s^f \in L_I(G), \forall t \in L_I(G)/s^f, |t| \geq k \Rightarrow$$
$$\forall \rho \in L_I(G), (P(\rho) = P(s^f.t) \Rightarrow f \in \rho)$$

Twin Plant

- Pre-diagnoser D: nondeterministic automaton obtained by keeping only observable events and attaching fault information to each remaining state.
- Twin plant: $TP = D \times D = D||_{\Sigma_o} D$.
- TP ambiguous state: f contained in one and only one of the two associated D states.
- TP ambiguous state cycle: cycle containing only ambiguous states.

Background IV

Twin Plant-based diagnosability checking

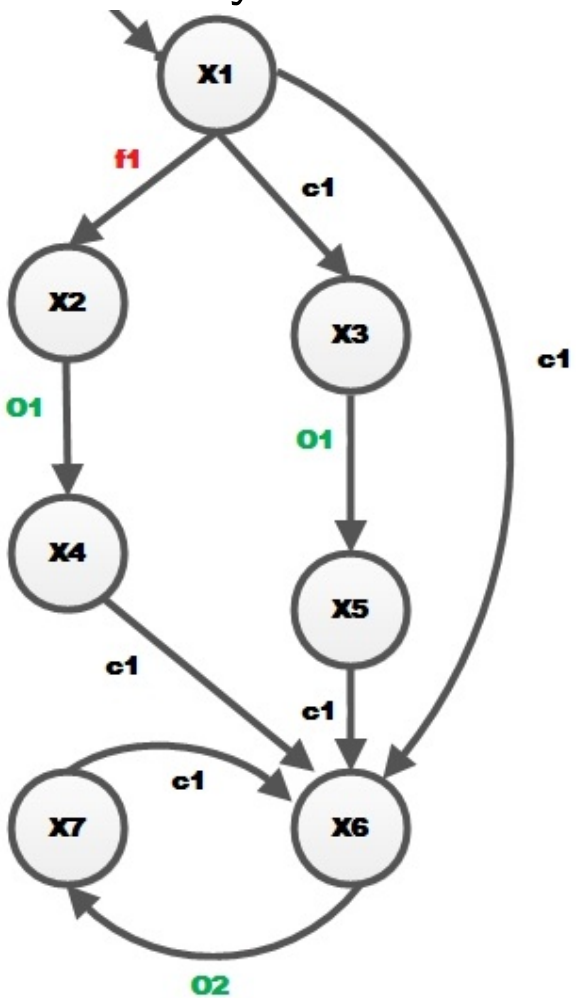
- G non-diagnosable $\Leftrightarrow \exists$ a pair of observation-equivalent infinite I-trajectories in G, one faulty and the other one correct $\Leftrightarrow \exists$ a critical path in TP, i.e. a $(I \times I)$-path made up of a prefix followed by an ambiguous cycle.
- TP-based diagnosability checking is polynomial in $|Q|$ (4th degree).

Example

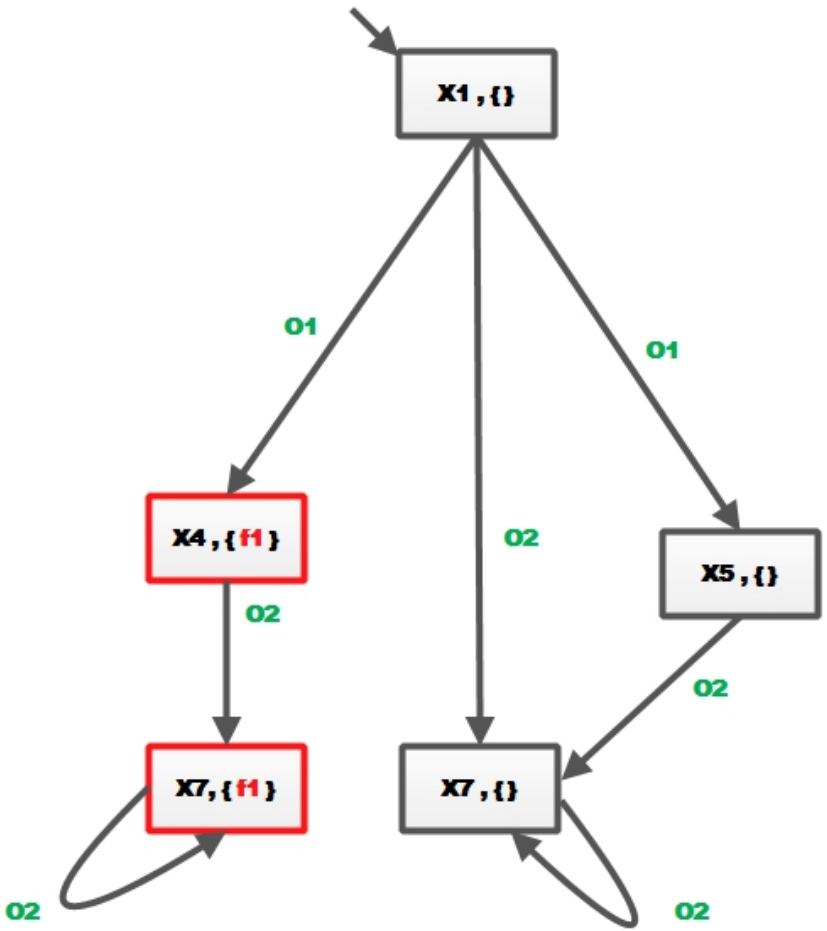
- $((1n, \emptyset), (2n, \emptyset)) \xrightarrow{a} ((1n, \emptyset), (2f, f)) \xrightarrow{a} ((1n, \emptyset), (2f, f))$ is a critical path.
- $\Rightarrow G$ is not diagnosable from I.

Background V

System

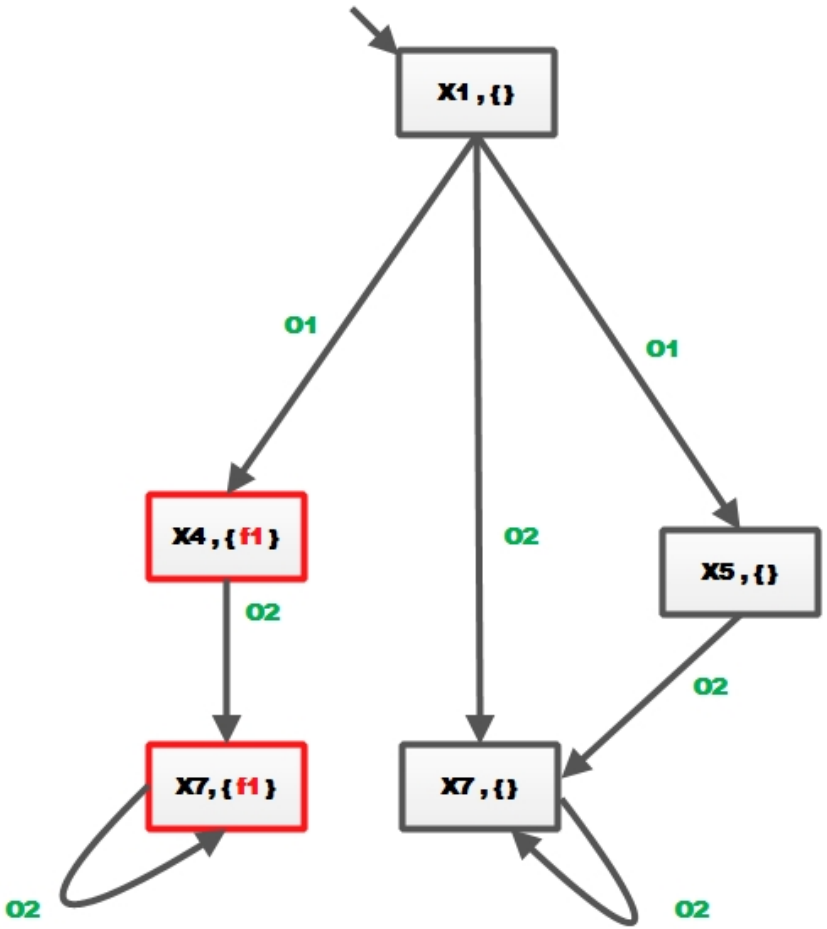


Pre-diagnoser

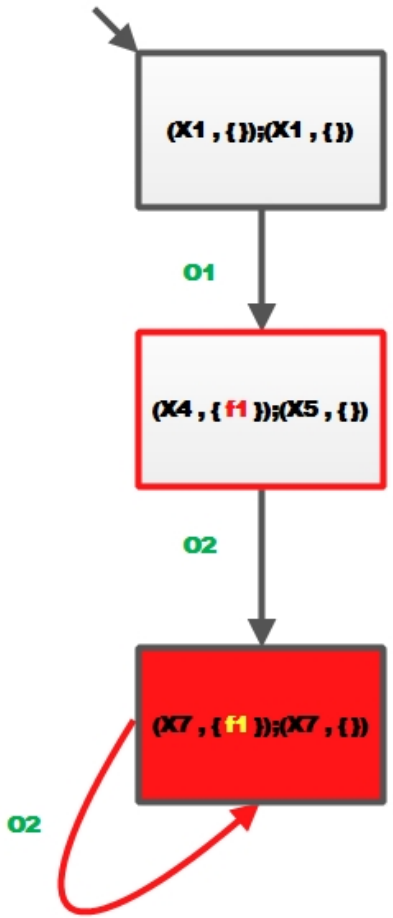


Background VI

Pre-diagnoser



Twin Plant



Background VII

Plan

- Plan for a CLTS G: finite sequence π of control actions in $\mathcal{A}$.

Planning problem

- $\langle G, O \rangle$ with $O \subseteq 2^Q$ collection of belief states.
- Solution: plan $\pi \mid \exists B \in O, \pi(I) \subseteq B$.

Diagnosability planning problem

- Case where O is the set of maximal belief states from where G is diagnosable.
- Solution: plan $\pi \mid G$ is diagnosable from $\pi(I)$.
- One can require π optimal (minimal size $|\pi|$, more generally minimal cost).

Diagnosability planning complexity I

Result

- Diagnosability planning is PSPACE-COMPLETE.
- Reminder: Active diagnosability checking is EXPTIME and PSPACE-COMPLETE $\subseteq$ EXPTIME.

Diagnosability planning complexity II

Sketch of the proof

- Diagnosability planning (DP) is a case of conformant planning, i.e. with undeterminism of initial state and/or actions.
- Membership: iterate over all possible belief states (PSPACE); verify this belief state is diagnosable (P); search for a conformant plan to this belief state (PSPACE).
- Hardness: reducing classical propositional planning P (known to be $\mathrm{PSPACE\text{-}COMPLETE}$) to conformant planning with explicit states CPXS and then to DP.

Diagnosability planning complexity III

Sketch of the proof

- P $\rightarrow$ CPXS. P is defined in a succinct way by propositional variables, each action being logically represented by a precondition and a set of effects. Variable v in P $\rightarrow$ states v_0 and v_1 in CPXS, s.t. v_0 (resp. v_1) belongs to the CPXS belief state iff v is true (resp. false) in the P state. Thus a belief state in CPXS represents a state in P. Action a in P $\rightarrow$ transitions between the v_0's and the v_1's, according to precondition and effects of a.
- CPXS $\rightarrow$ DP. One defines active transitions s.t. no fault can occur from the CPXS objective states, all other states having two observation-equivalent cycles, resp. faulty and correct. Thus a conformant plan reaching an objective state corresponds to a diagnosability plan.

Building a Diagnosability Plan

Normal method

- Generate a candidate plan π by browsing the search space, e.g. with Breadth First Search (BFS).
- Check the diagnosability of the reached belief state $\pi(I)$, with the Twin Plant method.
- Iterate until a diagnosable belief state $\pi(I)$ is reached (then π is a DP) or all the search space has been explored (nonexistence of a DP).
- BFS ensures minimality of $|\pi|$, if any. This size is bounded by $2^{|Q|}$ ($|Q|$ in the deterministic case).
- Inconvenient: builds from scratch at each step a new TP, without using possibly useful information from previously built TPs.

Recycling Twin Plant I

The idea of TP recycling

- TPs constructed from different belief states $\pi(I)$ will generally share some states with each other.
- Idea: recycling parts of previously built TPs (these TPs can be seen as parts of the implicit global TP of the system, that would correspond to $I = Q$, i.e. complete uncertainty).
- If a critical path is found during the test of a candidate π, we know that we will recover it each time we will meet again its starting state $((q_1, \emptyset), (q_2, \emptyset))$ with $q_1, q_2 \in \pi(I)$ in the construction of a next TP.
- Hence, label such pairs $\{q_1, q_2\}$ in order to avoid re-testing them later if they occur and use them to prune next TPs construction and to guide the DP search.

Recycling Twin Plant II

Bad pair

- $\{q_1, q_2\}$ bad pair $\Leftrightarrow ((q_1, \emptyset), (q_2, \emptyset))$ starting state of a critical path in TP of G.
- NB. Actually, when a critical path is discovered, not only its starting state but all its non-ambiguous states (i.e., before the fault occurrence) give rise to bad pairs, that can be learnt.
- Bad unit if $q_1 = q_2$ (i.e., $\langle Q, \Sigma, \delta, \{q_1\}\rangle$ not diagnosable).
- $\mathcal{B}$ = set of all (currently known) bad pairs.

Example

- $\{1n, 2n\}$ is a bad pair. Assume it is the one found when testing the plan $\pi = \emptyset$. So, at this step, $\mathcal{B} = \{\{1n, 2n\}\}$.

Recycling Twin Plant III

Exploiting bad pairs: the Lazy Learning method

- TP from a belief state I is built by processing I globally (adding one virtual initial state related by unobservable transitions to any state in I).
- Avoid testing π if $\mathcal{B} \cap (\pi(I) \times \pi(I)) \neq \emptyset$.
- Example: $1n, 2n \in \beta(I) = \{1n, 2n, 3n, 4n\}$ and $\in \gamma(I) = \{1n, 2n, 4n, 5n\}$. So, useless to test $\pi = \beta$ and $\pi = \gamma$.
- Stop the construction of the TP if a state $((q_1, \emptyset), (q_2, \emptyset))$ is reached with $\{q_1, q_2\} \in \mathcal{B}$.
- Example: if the bad unit $\{5n\}$ has been previously identified, testing $\{2n\}$ can be stopped just after the construction in TP of $((2n, \emptyset), (2n, \emptyset)) \xrightarrow{b} ((5n, \emptyset), (5n, \emptyset))$, concluding that $\{2n\}$ is a bad unit.

Recycling Twin Plant IV

Good pair

- $\{q_1, q_2\}$ good pair $\Leftrightarrow$ there is no critical path in TP issued from $((q_1, \emptyset), (q_2, \emptyset))$.
- NB. Actually, when no critical path exists, not only the starting state of TP but all its non-ambiguous states give rise to good pairs, that can be learnt.
- Good unit if $q_1 = q_2$ (i.e., $\langle Q, \Sigma, \delta, \{q_1\}\rangle$ diagnosable).
- $\mathcal{G}$ = set of all (currently known) good pairs.

Example

- $\{1n, 3n\}$ is a good pair, $\{1n\}$ and $\{3n\}$ are good units. So, $\langle Q, \Sigma, \delta, \{1n, 3n\}\rangle$ is diagnosable.

Recycling Twin Plant V

Exploiting bad and good pairs: the Eager Learning method

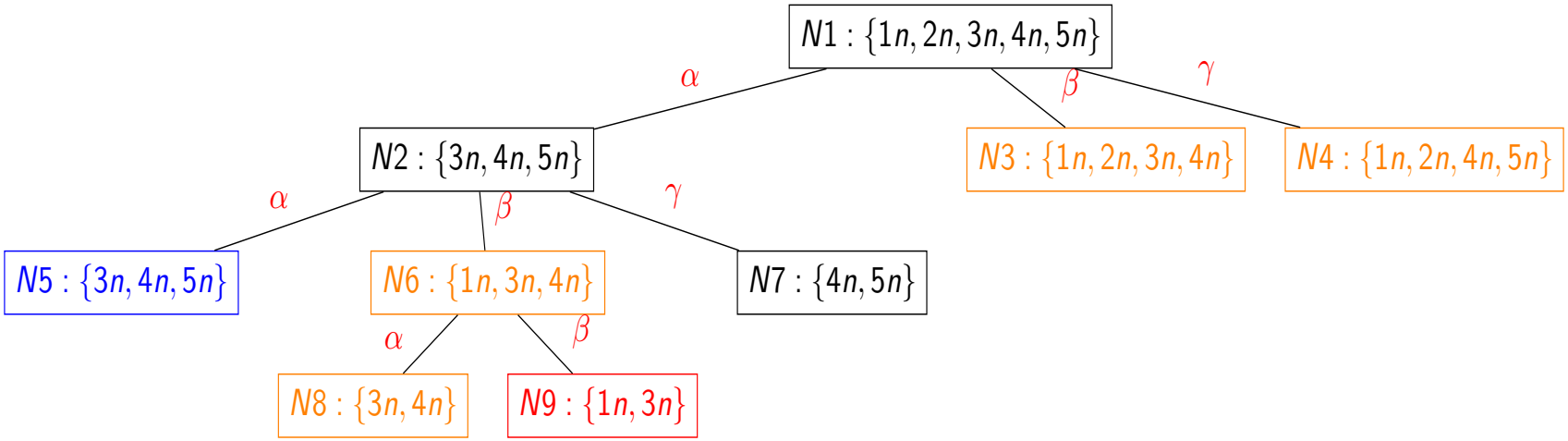
- TP from a belief state I is built by constructing successively the TPs for all pairs $\{q_1, q_2\}$ with $q_1, q_2 \in I$ (the process being stopped if a critical path is found in one of them).
- Thus, even if G is non-diagnosable from I, good pairs can be learned (which was not possible with the Lazy method).
- Avoid testing pairs $\{q_1, q_2\} \in \mathcal{G} \cap (\pi(I) \times \pi(I))$.
- Example: assume that, before finding the bad pair $\{1n, 2n\}$, the good pairs $\{1n, 1n\}$, $\{3n, 3n\}$ and $\{1n, 3n\}$ have been found from I. Then, useless to test $\pi = \alpha\beta\beta$ with $\pi(I) = \{1n, 3n\}$ to conclude diagnosability.
- Prune TP's building each time a state $((q_1, \emptyset), (q_2, \emptyset))$ with $\{q_1, q_2\} \in \mathcal{G}$ is reached (nothing to build from this state).

Recycling Twin Plant VI

Exploiting bad and good pairs to guide the search of a plan

- Guiding plan generation by a greedy algorithm that locally optimizes an objective function $f(\pi)$.
- For each candidate plan π, partition pairs of states in $\pi(I)$ according to current $\mathcal{B}$ and $\mathcal{G}$: $\langle \mathcal{B}_\pi, \mathcal{G}_\pi, \mathcal{U}_\pi \rangle$. Keep only those π with $\mathcal{B}_\pi = \emptyset$ and rank them according to $f(\pi)$ optimization.
- $f(\pi)$ built by combining minimization of the cost function $g(\pi)$ (e.g., $|\pi|$) and of the heuristic function $g(\pi)$ (e.g., ratio – or number – of unlabeled pairs).
- Take the best π and test all pairs in $\mathcal{U}_\pi$ (stop as soon as one is found bad). Update $\mathcal{B}$ and $\mathcal{G}$ and all $\langle \mathcal{B}_{\pi'}, \mathcal{G}_{\pi'}, \mathcal{U}_{\pi'} \rangle$. Repeat.

Recycling Twin Plant VII



Plan search space by Eager method with BFS strategy.

$N1$: Assume $\{1n, 1n\}, \{3n, 3n\}, \{1n, 3n\} \in \mathcal{G}$, then $\{1n, 2n\} \in \mathcal{B}$ are found.
$N2$: Assume $\{3n, 4n\} \in \mathcal{B}$ is found.
Exploiting $\mathcal{B}$, $N3, N4, N6, N8$ have not to be explored. $N5$ is closed.
$N7$: Assume $\{4n, 5n\} \in \mathcal{B}$ is found.
$N9$: diagnosable belief state by exploiting $\mathcal{G}$, without exploration.

Experimental Results I

Construction of the scalable benchmark

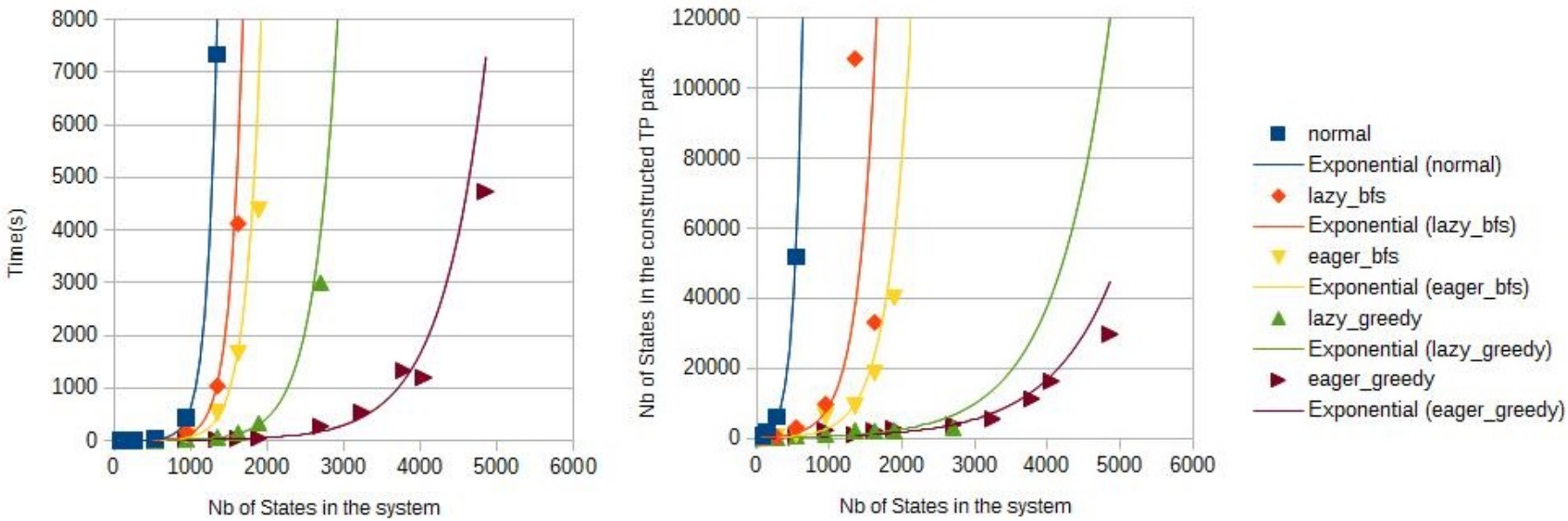
- Take the system example as a basic component and assemble copies of it in a rectangular grid.
- Connect them by active transitions: observable event c from $3n$ (resp. $1n$) of each component to $2n$ (resp. $4n$) of its bottom neighbor; unobservable one from $3f$ (resp. $4f$) of each component to $1f$ (resp. $2f$) of its right neighbor and from $1f$ (resp. $2f$) to $3f$ (resp. $4f$) of its left neighbor.

Experimental Results II

Construction of the scalable benchmark

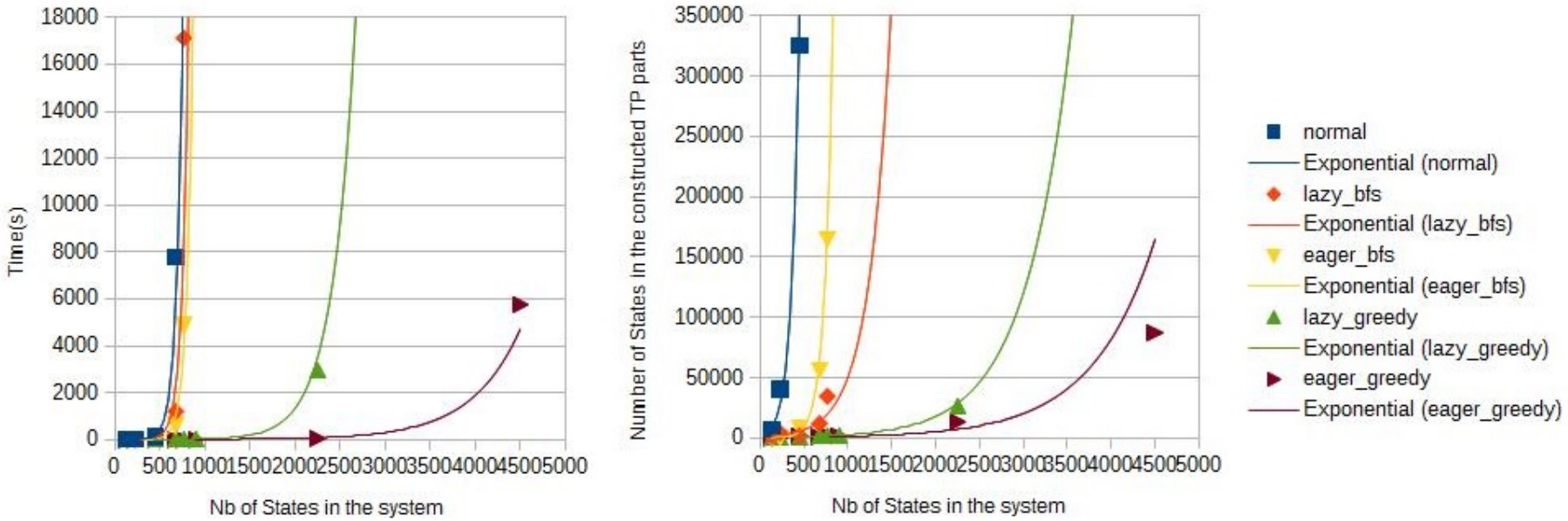
- Two different actions models for the (top and bottom) border and the internal components, with global "vertical" actions between one component and its top and bottom neighbors, such that, for any internal initial belief space, top and bottom border components have short local diagnosability plans but internal components require "vertical down" global plans up to the bottom border of the grid.
- We tested Normal, Lazy and Eager methods with BFS strategy (guaranteeing length optimality) and also, for the two last ones, with our greedy strategy (without guaranty of optimal length), where only the Eager method can exploit good pairs. We made vary the height of the grid and the initial belief space (local or scattered across several components).

Experimental Results III



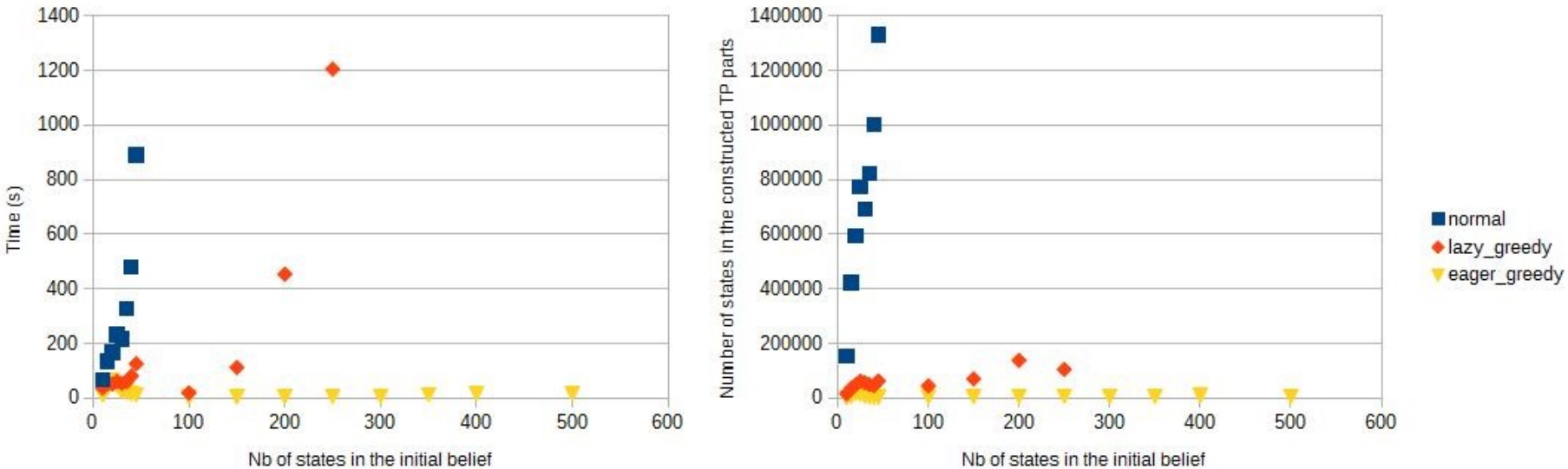
- Grids of increasing height (width 3).
- I made up of the 5 normal states of the central component.
- Efficiency of recycling: Eager method 15 times faster than Normal method for height 50 ($|\pi| = 74$).
- Efficiency of the greedy strategy: scales well up to height 180 ($|\pi| = 355$).

Experimental Results IV



- Grids of increasing height (width 5).
- I made up of the 10 normal states of two scattered internal components (at 1/3 and 2/3 of the height).
- Normal method explodes at height 17.
- With greedy strategy, scales up to height 100, with $|\pi|$ close to optimal for Lazy method.

Experimental Results V



- Fixed grid size 10×10.
- Varying initial belief state size $|I|$ firstly from 5 to 45, by adding incrementally to I the 5 normal states of "diagonal" components: Normal method explodes.
- Then compare Lazy and Eager methods by adding to I the 5 normal states of randomly chosen component up to 500 states (full grid): advantage of using good pairs in plan search.

Conclusion & Future Works I

Conclusion

- Definition of Diagnosability Planning problem.
- Demonstration of its PSPACE-COMPLETENESS.
- Design of an algorithm based on Twin Plant incremental construction and reusability via learned bad and good pairs, allowing pruning its construction and guiding the plan search.
- Experimental demonstration of its efficiency.

Conclusion & Future Works II

Future Works

- Defining more informative cost functions for better search strategies.
- Encoding this problem in SAT, improving incremental SAT method.
- Extending our approach to other Twin Plant-based properties, such as predictability.
- Merging repair and diagnosability plans construction for achieving an integrated framework with global optimality.
- Applying to real applications, e.g. power supply restoration.