Franck Cassez, Stavros Tripakis and Karine Altisen.
Synthesis Of Optimal-Cost Dynamic Observers for Fault Diagnosis of Discrete-Event Systems (Extended version).
In 1st IEEE & IFIP International International Symposium on Theoretical Aspects of Software Engineering (TASE'07), Shangai, ROC, June 2007. IEEE Computer Society.

Synthesis Of Optimal-Cost Dynamic Observers for Fault Diagnosis of Discrete-Event Systems*

Franck Cassez† Stavros Tripakis‡ Karine Altisen§

Abstract

Fault diagnosis consists in synthesizing a diagnoser that observes a given plant through a set of observable events, and identifies faults which are not observable as soon as possible after their occurrence. Existing literature on this problem has considered the case of static *observers, where the set of observable events does not change during execution of the system. In this paper, we consider* dynamic *observers, where the observer can switch sensors on or off, thus dynamically changing the set of events it wishes to observe. We define a notion of cost for such dynamic observers and show that (i) the cost of a given dynamic observer can be computed and (ii) an optimal dynamic observer can be synthesized.*

1. Introduction

Fault Diagnosis. Discrete-event systems (DES) can be formalized by using finite automata over a set of *observable* events Σ, plus a set of *unobservable* events [8, 10].

Fault diagnosis consists in observing a DES and detecting whether a fault has occurred or not. We follow the DES setting of [9] where the behavior of the plant is known and a model of it is available as a finite-state automaton over $\Sigma \cup \{\varepsilon, f\}$ where Σ is the set of observable events, ε represents the unobservable events, and f is a special unobservable event that corresponds to the faults. Checking *diagnosability* (whether a fault can be detected) for a given plant and a *fixed* set of observable events can be done in polynomial time [9, 11, 5]. (Notice that synthesizing a diagnoser involves determinization in general, thus cannot be done in polynomial time.)

The usual assumption in this setting is that the set of observable events is fixed (and this in turn determines the set of unobservable events as well). Observing an event usually requires some detection mechanism, i.e. , a *sensor* of some sort. Which sensors to use, how many of them, and where to place them, are some of the design questions that are often difficult to answer, especially without knowing what these sensors are to be used for.

Dynamic Observers. *Dynamic sensors' selection* consist in selecting the sensors to switch on after each new observation, thus dynamically changing the set of events to observe. A device that chooses the set of events to observe dynamically is a *dynamic observer*. We are interested in synthesizing a dynamic observer in the hope that not all the observable events are always needed to diagnose a DES.

This problem is interesting since observing an event can be costly in terms of time or energy: computation time must be spent to read and process the information provided by the sensor, and power is required to operate the sensor (as well as to perform the computations). It is then essential that the sensors used really provide useful information. It is also important for the computer to discard any information given by a sensor that is not really needed. In the case of a fixed set of observable events, it is not the case that all sensors always provide useful information and sometimes energy (sensor operation and computer treatment) is spent for nothing. For example, to diagnose a fault in the system described by the automaton $\mathcal{B}$, Figure 1, a diagnoser only has to watch event a, and *when* a *has occurred*, to watch event b: if the sequence $a.b$ occurs, for sure a fault has occurred and the diagnoser can raise an alarm. It is then not useful to switch on sensor b before an a has occurred.

Optimal-Cost Dynamic Sensors' Selection. Given an observer we can define a notion of *cost* (i.e. how expensive it is) to diagnose a DES using this observer. The first problem we address in this paper is to compute the cost of diagnosing the DES with a given observer. We then focus on a more

*Work supported by the project CORTOS, a program of the french government. `http://www.lsv.ens-cachan.fr/aci-cortos`

†CNRS/IRCCyN, 1 rue de la Noë, BP 92101, 44321 Nantes Cedex 3, France. Email: franck.cassez@cnrs.irccyn.fr

‡Cadence Berkeley Labs, 1995 University Avenue, Berkeley, CA, 94704, USA, and CNRS, Verimag Laboratory, Centre Equation, 2, avenue de Vignate, 38610 Gières, France. Email: tripakis@cadence.com

§INPG and Verimag Laboratory, Centre Equation, 2, avenue de Vignate, 38610 Gières, France.

challenging problem which is to synthesize an optimal observer, in the sense that the cost of diagnosing a DES with such an observer is minimal.

Related work. In the case of *static* observers where the set of observable events is fixed a priori some papers have already considered optimization problems. NP-hardness of finding minimum-cardinality sets of observable events so that diagnosability holds under the standard, projection-based setting has been previously reported in [11].

The complexity of finding "optimal" observation masks, i.e. a set that cannot be reduced, has been considered in [6] where it was shown that the problem is NP-hard for general properties. [6] also shows that finding optimal observation masks is polynomial for "mask-monotonic" properties where increasing the set of observable (or distinguishable) events preserves the property in question. Diagnosability is a mask-monotonic property. Computing an optimal observation masks is not the same as finding a minimum-cardinality mask. We have recently considered this latter problem in [1] and proved it is NP-complete.

In [4], the authors investigate the problem of computing a minimal-cost strategy that allows to find a subset of the set of observable events s.t. the system is diagnosable. It is assumed that each such subset has a known associated cost, as well as a known a-priori probability for achieving diagnosability.

To our knowledge, dynamic observers have not been considered up to now. Consequently, the problem of synthesizing optimal-cost dynamic observers for diagnosability purposes, have not been addressed previously in the literature. In a recent paper [2], we have adressed the problem of synthesizing dynamic observers. No optimization criterion is used in this work. The present paper is a follow-up of [2] and extends it by considering optimization problems. A full version with proofs is available as a research report [1].

Organisation of the paper. In Section 2 we fix notations and introduce finite automata with faults to model DES. In Section 3 we introduce dynamic observers and define the cost of a dynamic observer. We also show how to compute it. In Section 4, we address the problem of computing an optimal observer.

2. Preliminaries

2.1. Words and Languages

Let Σ be a finite alphabet and $\Sigma^\varepsilon = \Sigma \cup \{\varepsilon\}$. Σ^* is the set of finite words over Σ and contains ε which is also the empty word[1]. A *language* L is any subset of Σ^*. $\Sigma^+ = \Sigma^* \setminus \{\varepsilon\}$. Given two words ρ, ρ' we denote $\rho.\rho'$ the concatenation of ρ and ρ' (which is defined in the usual way). $|\rho|$ stands for the length of the word ρ and $|\rho|_\lambda$ with $\lambda \in \Sigma$ stands for the number of occurrences of λ in ρ. Given $\Sigma_1 \subseteq \Sigma$, we define the *projection* $\boldsymbol{\pi}_{/\Sigma_1} : \Sigma^* \to \Sigma_1^*$ by: $\boldsymbol{\pi}_{/\Sigma_1}(\varepsilon) = \varepsilon$ and for $a \in \Sigma, \rho \in \Sigma^*$, $\boldsymbol{\pi}_{/\Sigma_1}(a.\rho) = a.\boldsymbol{\pi}_{/\Sigma_1}(\rho)$ if $a \in \Sigma_1$ and $\boldsymbol{\pi}_{/\Sigma_1}(\rho)$ otherwise.

2.2. Finite Automata

Let $f \notin \Sigma^\varepsilon$ be a fresh letter that corresponds to the fault action. An *automaton* A is a tuple[2] $(Q, q_0, \Sigma^{\varepsilon,f}, \to)$ with Q a set of states, $q_0 \in Q$ is the initial state, $\to \subseteq Q \times \Sigma^{\varepsilon,f} \times Q$ is the transition relation. If Q is finite, A is a *finite automaton*. We write $q \xrightarrow{\lambda} q'$ if $(q, \lambda, q') \in \to$. For $q \in Q$, $\mathit{en}(q)$ is the set of actions enabled at q. A *run* ρ from state s in A is a sequence of transitions $s_0 \xrightarrow{\lambda_1} s_1 \xrightarrow{\lambda_2} s_2 \cdots s_{n-1} \xrightarrow{\lambda_n} s_n$ s.t. $\lambda_i \in \Sigma^{\varepsilon,f}$ and $s_0 = s$. We let $\mathit{tgt}(\rho) = s_n$. The set of runs from s in A is denoted $\mathit{Runs}(s, A)$ and we define $\mathit{Runs}(A) = \mathit{Runs}(q_0, A)$. The *trace* of the run ρ, denoted $\mathit{tr}(\rho)$, is the word obtained by concatenating the symbols λ_i appearing in ρ, for those λ_i different from ε. Given a set $R \subseteq \mathit{Runs}(A)$, $\mathit{Tr}(R) = \{\mathit{tr}(\rho) \text{ for } \rho \in R\}$ is the set of traces of the runs in R. A run ρ is *k-faulty* if there is some $1 \leq i \leq n$ s.t. $\lambda_i = f$ and $n - i \geq k$. $\mathit{Faulty}_{\geq k}(A)$ is the set of k-faulty runs of A. A run is *faulty* if it is k-faulty for some $k \in \mathbb{N}$ and $\mathit{Faulty}(A)$ denotes the set of faulty runs. It follows that $\mathit{Faulty}_{\geq k+1}(A) \subseteq \mathit{Faulty}_{\geq k}(A) \subseteq \cdots \subseteq \mathit{Faulty}_{\geq 0}(A) = \mathit{Faulty}(A)$. Finally $\mathit{NonFaulty}(A) = \mathit{Runs}(A) \setminus \mathit{Faulty}(A)$ is the set on *non-faulty* runs of A. We let $\mathit{Faulty}^{tr}_{\geq k}(A) = \mathit{Tr}(\mathit{Faulty}_{\geq k}(A))$ and $\mathit{NonFaulty}^{tr}(A) = \mathit{Tr}(\mathit{NonFaulty}(A))$ be the sets of traces of faulty and non-faulty runs.

A word w is *accepted* by A if $w = \mathit{tr}(\rho)$ for some $\rho \in \mathit{Runs}(A)$. The *language* $\mathcal{L}(A)$ of A is the set of words accepted by A.

We assume that each run of A of length n can be extended into a run of length $n + 1$. This is required for technical reasons and can be achieved by adding ε loop transitions to each deadlock state of A. Notice that this transformation does not change the observations produced by the plant, thus, any observer synthesized for the transformed plant also applies to the original one.

2.3. Product of Automata

Let $A_1 = (Q_1, q_0^1, \Sigma_1, \to_1)$ and $A_2 = (Q_2, q_0^2, \Sigma_2, \to_2)$. The *product* of A_1 and A_2 is the automaton $A_1 \times A_2 = (Q, q_0, \Sigma, \to)$ where:

- $Q = Q_1 \times Q_2$,

[1] We use ε both for the unobservable event and the empty word as the type is always clear from the context.

[2] In this paper we only use finite automata that generate prefix-closed languages, hence we do not need to use a set of final or accepting states.

- $q_0 = (q_0^1, q_0^2)$,
- $\Sigma = \Sigma_1 \cup \Sigma_2$,
- $\rightarrow \subseteq Q \times \Sigma \times Q$ is defined by $(q_1, q_2) \xrightarrow{\sigma} (q_1', q_2')$ if:
  - either $\sigma \in \Sigma_i \setminus \Sigma_{3-i}$ and $q_i \xrightarrow{\sigma}_i q_i'$ and $q_{3-i}' = q_{3-i}$ or
  - $\sigma \in \Sigma_1 \cap \Sigma_2$ and $q_k \xrightarrow{\sigma}_k q_k'$ for $k = 1, 2$.

3. Sensor Minimization & Dynamic Observers

In this section we introduce *dynamic observers*. To illustrate why dynamic observers can be useful consider the following example.

Example 1 (Dynamic Observation) *Assume we want to detect faults in automaton $\mathcal{B}$ of Fig. 1. A static diagnoser that observes $\Sigma = \{a, b\}$ works, however, no proper subset of Σ can be used to detect faults in $\mathcal{B}$. Thus the minimum cardinality of the set of observable events for diagnosing $\mathcal{B}$ is 2 i.e. a static observer will have to monitor two events during the execution of the DES. If we want to use a mask, the minimum-cardinality for a mask is 2 as well. This means that an observer will have to be receptive to at least two inputs at each point in time to detect a fault in $\mathcal{B}$. One can think of being receptive as switching on a device to sense an event. This consumes energy. We can be more efficient using a dynamic observer, that only turns on sensors when needed, thus saving energy. In the case of $\mathcal{B}$, this can be done as follows: in the beginning we only switch on the a-sensor; once an a occurs the a-sensor is switched off and the b-sensor is switched on. Compared to the previous diagnosers we use twice as less energy.*

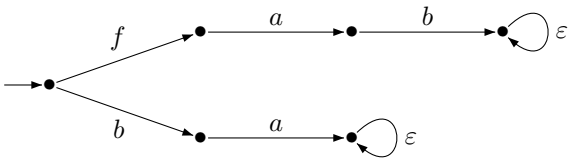

Figure 1. The automaton $\mathcal{B}$

3.1. Dynamic Observers

We now formalize the above notion of dynamic observation. The choice of the events to observe can depend on the choices the observer has made before and on the observations it has made. Moreover an observer may have *unbounded* memory.

Definition 1 (Observer) *An* observer Obs *over Σ is a deterministic labeled automaton* $\text{Obs} = (S, s_0, \Sigma, \delta, L)$, *where S is a (possibly infinite) set of states, $s_0 \in S$ is the initial state, Σ is the set of observable events, $\delta : S \times \Sigma \rightarrow S$ is the transition function (a total function), and $L : S \rightarrow 2^{\Sigma}$ is a labeling function that specifies the set of events that the observer wishes to observe when it is at state s. We require for any state s and any $a \in \Sigma$, if $a \notin L(s)$ then $\delta(s, a) = s$: this means the observer does not change its state when an event it chose not to observe occurs. We use the notation $\delta(s_0, w)$ to denote the state s reached by reading the word w and $L(\delta(s_0, w))$ for the set of events obs observes after w.* ■

An observer implicitly defines a *transducer* that consumes an input event $a \in \Sigma$ and, depending on the current state s, either outputs a (when $a \in L(s)$) and moves to a new state $\delta(s, a)$, or outputs nothing or ε, (when $a \notin L(s)$) and remains in the same state waiting for a new event. Thus, an observer defines a mapping Obs from Σ^* to Σ^* (we use the same name "Obs" for the automaton and the mapping). Given a run ρ, $\text{Obs}(\boldsymbol{\pi}_{/\Sigma}(tr(\rho)))$ is the output of the transducer on ρ. It is called the *observation* of ρ by Obs. We next provide an example of a particular case of observer which can be represented by a finite-state machine.

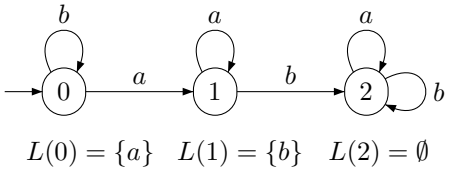

Figure 2. A Finite-State Observer Obs

Example 2 *Let Obs be the observer of Fig. 2. Obs maps the following inputs as follows: Obs$(baab) = ab$, Obs$(bababbaab) = ab$, Obs$(bbbbba) = a$ and Obs$(bbaaa) = a$. If Obs operates on the DES $\mathcal{B}$ of Fig 1 and $\mathcal{B}$ generates $f.a.b$, Obs will have as input $\boldsymbol{\pi}_{/\Sigma}(f.a.b) = a.b$ with $\Sigma = \{a, b\}$. Consequently the observation of Obs is Obs$(\boldsymbol{\pi}_{/\Sigma}(f.a.b)) = a.b$.*

3.2. Fault Diagnosis with Dynamic Diagnosers

Definition 2 ((Obs, k)-diagnoser) *Let* Obs *be an observer over Σ. $D : \Sigma^* \rightarrow \{0, 1\}$ is an* (Obs, k)-diagnoser *for A if (i) $\forall \rho \in$ NonFaulty$(\mathcal{A})$, $D(\text{Obs}(\boldsymbol{\pi}_{/\Sigma}(tr(\rho)))) = 0$ and (ii) $\forall \rho \in$ Faulty$_{\geq k}(\mathcal{A})$, $D(\text{Obs}(\boldsymbol{\pi}_{/\Sigma}(tr(\rho)))) = 1$.* ■

A is (Obs, k)-diagnosable if there is an (Obs, k)-diagnoser for A. A is Obs-diagnosable if there is some k such that A is (Obs, k)-diagnosable.

If a diagnoser always selects Σ as the set of observable events, it is a static observer and (Obs, k)-diagnosability

amounts to the standard (Σ, k)-diagnosis problem [9]. In this case A is (Σ, k)-diagnosable iff $\boldsymbol{\pi}_{/\Sigma}(\mathit{Faulty}^{tr}_{\geq k}(\mathcal{A})) \cap \boldsymbol{\pi}_{/\Sigma}(\mathit{NonFaulty}^{tr}(\mathcal{A})) = \emptyset$.

As for Σ-diagnosability, we have the following equivalence for dynamic observers: A is (Obs, k)-diagnosable iff $\mathrm{Obs}(\boldsymbol{\pi}_{/\Sigma}(\mathit{Faulty}^{tr}_{\geq k}(\mathcal{A}))) \cap \mathrm{Obs}(\boldsymbol{\pi}_{/\Sigma}(\mathit{NonFaulty}^{tr}(\mathcal{A}))) = \emptyset$. This follows directly from definition 2.

If an observer is given as a finite state automaton we can state the following problem:

Problem 1 (Finite-State Obs-Diagnosability)
INPUT: A, Obs *a finite-state observer.*
PROBLEM:

(A) Is A Obs*-diagnosable ?*

(B) If the answer to (A) is "yes", compute the minimum k *such that* A *is* (Obs, k)*-diagnosable.*

As proved in [1] Problem 1 can be solved in polynomial time. To solve it we build a *product* automaton[3] $A \otimes \mathrm{Obs}$ such that: A is (Obs, k)-diagnosable $\iff$ $A \otimes \mathrm{Obs}$ is (Σ, k)-diagnosable. As (Σ, k)-diagnosability can be checked in polynomial time and $A \otimes \mathrm{Obs}$ has polynomial size in the size of A and Obs the result follows.

The automaton $A \otimes \mathrm{Obs} = (Q \times S, (q_0, s_0), \Sigma^{\varepsilon,f}, \rightarrow)$ is defined as follows:

1. $(q, s) \xrightarrow{\beta} (q', s')$ iff $\exists \lambda \in \Sigma$ s.t. $q \xrightarrow{\lambda} q'$, $s' = \delta(s, \lambda)$ and $\beta = \lambda$ if $\lambda \in L(s)$, $\beta = \varepsilon$ otherwise;

2. $(q, s) \xrightarrow{\lambda} (q', s)$ iff $\exists \lambda \in \{\varepsilon, f\}$ s.t. $q \xrightarrow{\lambda} q'$.

Example 3 *Let* $\mathcal{A}$ *be the DES given in Fig. 3 and* Obs *the observer of Fig. 2. The product* $\mathcal{A} \otimes$ Obs *is given in Fig. 4. Using an algorithm for checking* Σ*-diagnosability of* $\mathcal{A} \otimes$ Obs *we obtain that it is* $(\Sigma, 2)$*-diagnosable (and* 2 *is the minimum value). Hence* A *is* $(\mathrm{Obs}, 2)$*-diagnosable with* 2 *the minimum value.*

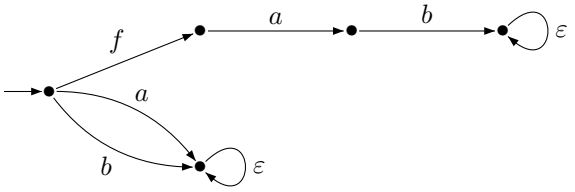

Figure 3. The automaton $\mathcal{A}$

We are going to define a notion of cost for observers. This notion is inspired by *weighted automata*.

[3] We use $\otimes$ to clearly distinguish this product from the synchronous product $\times$.

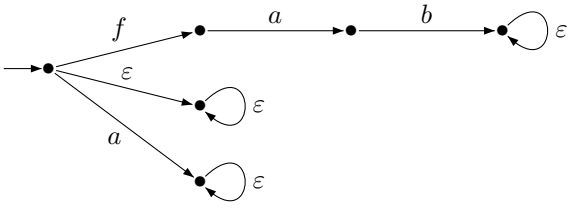

Figure 4. The automaton $\mathcal{A} \otimes \mathrm{Obs}$

3.3. Weighted Automata

The notion of cost for automata has already been defined and algorithms to compute some optimal values related to this model are described in many papers. We recall here the results of [7] which will be used later.

Definition 3 (Weighted Automaton) *A* weighted automaton *is a pair* (A, w) *s.t.* $A = (Q, q_0, \Sigma, \delta)$ *is a finite automaton and* $w : Q \rightarrow \mathbb{N}$ *associates a weight with each state.* ■

Definition 4 (Mean Cost) *Let* $\rho = q_0 \xrightarrow{a_2} q_1 \xrightarrow{a_1} \cdots \xrightarrow{a_n} q_n$ *be a run of* A*. The* mean cost *of* ρ *is*

$$\mu(\rho) = \frac{1}{n+1} \cdot \Sigma_{i=0}^{n} w(q_i)\,.$$

■

We remind that the length of $\rho = q_0 \xrightarrow{a_1} q_1 \xrightarrow{a_2} \cdots \xrightarrow{a_n} q_n$ is $|\rho| = n$. We assume that A is complete w.r.t. Σ (and $\Sigma \neq \emptyset$) and thus contains at least one run for any arbitrary length n. Let $\mathit{Runs}^n(A)$ be the set of runs of length n in $\mathit{Runs}(A)$. The *maximum mean-weight* of runs of length n for A is $\nu(A, n) = \max\{\mu(\rho) \text{ for } \rho \in \mathit{Runs}^n(A)\}$. The *maximum mean weight* of A is $\nu(A) = \limsup_{n\rightarrow\infty} \nu(A, n)$. Actually the value $\nu(A)$ can be computed using Karp's maximum mean-weight cycle algorithm [7] on weighted graphs. If $c = s_0 \xrightarrow{a_1} s_1 \xrightarrow{a_2} \cdots \xrightarrow{a_n} s_n$ is a cycle of A i.e. $s_0 = s_n$, the *mean weight* of c is $\mu(c) = \frac{1}{n+1} \cdot \Sigma_{i=0}^{n} w(s_i)$. The *maximum mean-weight cycle* of A is the value $\nu^*(A) = \max\{\mu(c) \text{ for } c \text{ a cycle of } A\}$. K As stated in [12], for weighted automata, $\nu(A) = \limsup_{n\rightarrow\infty} \nu(A, n) = \lim_{n\rightarrow\infty} \nu(A, n) = \nu^*(A)$. Karp's maximum mean-weight cycle algorithm [7] on weighted graphs is explained in Appendix A.

3.4. Cost of a Dynamic Observer

Let $\mathrm{Obs} = (S, s_0, \Sigma, \delta, L)$ be an observer and $A = (Q, q_0, \Sigma^{\varepsilon,f}, \rightarrow)$. We would like to define a notion of *cost* for observers in order to select an optimal one among all of those which are valid, i.e. s.t. A is (Obs, k)-diagnosable. Intuitively this notion of cost should capture the fact that the more events we observe at each time, the more expensive it is.

Definition of Cost. There is not one way of defining a notion of cost for observers and we first discuss two different notions:

- the first one is to define the cost of a word w generated by the DES w.r.t. to $\text{Obs}(w)$:

$$\textit{Cost}_1(w) = \frac{\Sigma_{i=0}^{i=n} |L(\delta(s_0, \text{Obs}(w)(i)))|}{n+1}$$

  with $n = |\text{Obs}(w)|$. Using the observer of Fig. 5, we obtain that $\textit{Cost}_1(b^n.a) = \frac{1+0}{2} = \frac{1}{2}$. And this regardless of the value of n.

- the second one is to define the cost of w w.r.t. to w itself:

$$\textit{Cost}_2(w) = \frac{\Sigma_{i=0}^{i=n} |L(\delta(s_0, w(i)))|}{n+1}$$

  with $n = |w|$. Using the observer of Fig. 5, we obtain $\textit{Cost}_2(b^n.a) = \frac{n+1+0}{n+2} = \frac{n+1}{n+2}$. And by simple arithmetic, it is true that $\textit{Cost}_2(b^n.a) < \textit{Cost}_2(b^{n+1}.a)$.

The example of Fig. 5 shows that the two notions are different. In the sequel we will use the second one $\textit{Cost}_2$ because $\textit{Cost}_2$ also captures the notion of the *time* we have been observing a set of events. Indeed, if the word b^{n+1} occurs, we have been observing the set $L(0)$ $n+1$ times in a logical time. It is natural that this is more expensive than observing $L(0)$ n times. Thus $\textit{Cost}_2$ is more satisfying than abstracting away the length of the input word as in $\textit{Cost}_1$.

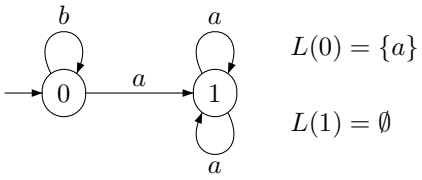

Figure 5. The Finite-State Observer Obs

Cost of an Observer. We now show how to define and compute the cost of an observer Obs operating on a DES A.

Given a run $\rho \in \textit{Runs}(A)$, the observer only processes $\boldsymbol{\pi}_{/\Sigma}(\textit{tr}(\rho))$ (ε and f-transitions are not processed). To have a consistent notion of costs that takes into account the logical time elapsed from the beginning, we need to take into account one way or another the number of *steps* of ρ even if some of them are non observable. A simple way to do this is to consider that ε and f are now observable events, let's say u, but that the observer never chooses to observe them. Indeed we assume we have already checked that A is (Obs, k)-diagnosable, and the problem is now to compute the cost of the observer we have used.

Definition 5 (Cost of a Run) *Given a run* $\rho = q_0 \xrightarrow{a_1} q_1 \cdots q_{n-1} \xrightarrow{a_n} q_n \in \textit{Runs}(A)$, *let* $w_i = \text{Obs}(\boldsymbol{\pi}_{/\Sigma}(\textit{tr}(\rho(i)))), 0 \leq i \leq n$. *The* cost *of* $\rho \in \textit{Runs}(A)$ *is defined by:*

$$\textit{Cost}_2(\rho, A, \text{Obs}) = \frac{1}{n+1} \cdot \sum_{i=0}^{n} |L(\delta(s_0, w_i)|.$$

■

We recall that $\textit{Runs}^n(A)$ is the set of runs of length n in $\textit{Runs}(A)$. The cost of the runs of length n of A is $\textit{Cost}_2(n, A, \text{Obs}) = \max\{\textit{Cost}_2(\rho, A, \text{Obs})$ *for* $\rho \in \textit{Runs}^n(A)\}$. The cost of the pair (Obs, A) is $\textit{Cost}_2(A, \text{Obs}) = \limsup_{n\to\infty} \textit{Cost}_2(n, A, \rho)$. Notice that $\textit{Cost}_2(n, A, \text{Obs})$ is defined for each n because we have assumed A generates runs of arbitrary large length.

To compute $\textit{Cost}_2(n, A, \text{Obs})$ we consider that ε and f are now observable events, let's say u, but that the observer never chooses to observe them.

Let $\text{Obs}^+ = (S, s_0, \Sigma^u, \delta', L)$ where δ' is δ augmented with u-transitions that loop on each state $s \in S$. Let A^+ be A where ε and f transitions are renamed u. Let $A^+ \times \text{Obs}^+$ be the synchronized product of A^+ and Obs^+. $A^+ \times \text{Obs}^+ = (Z, z_0, \Sigma^u, \Delta)$ is complete w.r.t. Σ^u and we let $w(q, s) = |L(s)|$ so that $(A^+ \times \text{Obs}^+, w)$ is a weighted automaton.

Theorem 1 $\textit{Cost}_2(A, \text{Obs}) = \nu^*(A^+ \times \text{Obs}^+)$.

Proof: The proof follows easily from the definitions. Let ρ be a run of A. There exists a run $\tilde{\rho}$ in $A^+ \times \text{Obs}^+$ s.t. $\textit{Cost}_2(\rho, A, \text{Obs}) = \mu(\tilde{\rho})$. $\tilde{\rho}$ is obtained from ρ by replacing ε and f transitions by some u transitions. Conversely for any run $\tilde{\rho}$ in $A^+ \times \text{Obs}^+$ there is a run ρ in A s.t. $\mu(\tilde{\rho}) = \textit{Cost}_2(\rho, A, \text{Obs})$. ■

We can compute the cost of a given pair (A, Obs): this can be done using Karp's maximum mean weight cycle algorithm [7] on weighted graphs. This algorithm is polynomial in the size of the weighted graph and thus we have:

Theorem 2 *Computing the cost of* (A, Obs) *is in P.*

Proof: The size of $A^+ \times \text{Obs}^+$ is polynomial in the size of A and Obs. ■

Notice that instead of the values $|L(s)|$ we could use any mapping from states of Obs to $\mathbb{Z}$ and consider that the cost of observing $\{a, b\}$ is less than observing a.

Example 4 *We give the results for the computation of the cost of two observers for the DES* $\mathcal{A}$ *given in Fig. 3. Let* O_1 *be the most powerful observer that observes* $\{a, b\}$ *at each step, and* O_2 *be the observer given in Fig. 2.*

The automata $\mathcal{A}^+ \times O_1^+$ *and* $\mathcal{A}^+ \times O_2^+$ *are given in Fig. 6 and Fig. 7. The weight function is pictured above each state.*

Notice that to compute $\nu^*(A^+ \times O_i^+)$ we do not need the labels of the transitions as we are dealing with weighted graphs: if two transitions (s, a, s') and (s, b, s') are in $A^+ \times O_i^+$ we only need one of them. For instance in Fig. 3 one of the transitions $(0, a, 4)$ and $(0, b, 4)$ is redundant. We apply the algorithm of Appendix A. The values $D_k(v)$ and $\min(v)$ for each state v of $A^+ \times O_i^+$ are given in Table 1 and Table 2. The maximum mean-weight value ν^* is the maximum value $\max_v \min(v)$ for v ranging over the set of states of $A^+ \times O_i^+$. We obtain Cost$_2(A, O_1) = 2$ and Cost$_2(A, O_2) = 1$.

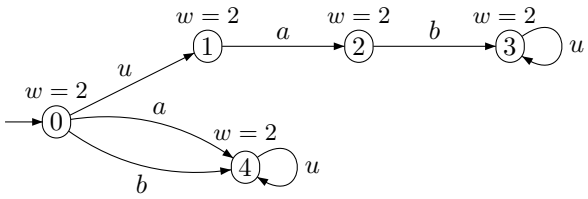

Figure 6. $\mathcal{A}^+ \times O_1^+$

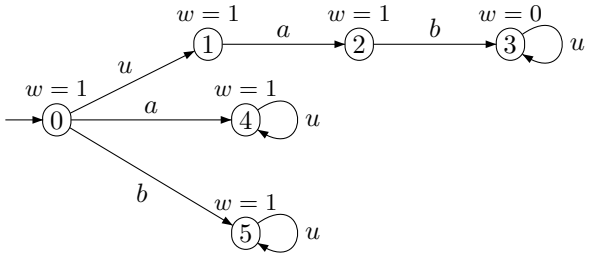

Figure 7. $\mathcal{A}^+ \times O_2^+$

	0	1	2	3	4
D_0	1	$-\infty$	$-\infty$	$-\infty$	$-\infty$
D_1	$-\infty$	4	$-\infty$	$-\infty$	4
D_2	$-\infty$	$-\infty$	6	$-\infty$	6
D_3	$-\infty$	$-\infty$	$-\infty$	8	8
D_4	$-\infty$	$-\infty$	$-\infty$	10	10
min	$-\infty$	$-\infty$	$-\infty$	**2**	**2**

Table 1. Iterations for $\mathcal{A}^+ \otimes O_1^+$

4. Optimal Dynamic Diagnosers

In this section, we focus on the problem of computing a best observer in the sense that diagnosing the DES with it has minimal cost. We address the following problem:

Problem 2 (Bounded Cost Observer)
INPUT: *A, $k \in \mathbb{N}$ and $c \in \mathbb{N}$.*
PROBLEM:

	0	1	2	3	4	5
D_0	1	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$
D_1	$-\infty$	2	$-\infty$	$-\infty$	2	2
D_2	$-\infty$	$-\infty$	3	$-\infty$	3	3
D_3	$-\infty$	$-\infty$	$-\infty$	4	4	4
D_4	$-\infty$	$-\infty$	$-\infty$	4	5	5
D_5	$-\infty$	$-\infty$	$-\infty$	4	6	6
min	$-\infty$	$-\infty$	$-\infty$	0	**1**	**1**

Table 2. Iterations for $\mathcal{A}^+ \otimes O_2^+$

(A). Is there an observer Obs *s.t. A is (*Obs*,k)-diagnosable and Cost$_2$(Obs) $\leq c$?*

(B). If the answer to (A) is "yes", compute a witness optimal observer Obs *with Cost$_2$(Obs) $\leq c$.*

Before dealing with Problem 2 we recall some results from [1].

4.1. Most Permissive Observer

For an observer $O = (S, s_0, \Sigma, \delta, L)$ and $w \in \Sigma^*$ we let $L(w)$ be the set $L(\delta(s_0, w))$: this is the set of events O chooses to observe on input w. Given a word $\rho \in \pi_{/\Sigma}(\mathcal{L}(A))$, we recall that $O(\rho)$ is the observation of ρ by O. Assume $O(\rho) = a_0 \cdots a_k$. Let $\overline{\rho} = L(\varepsilon).\varepsilon.L(a_0).a_0. \cdots L(O(\rho)(k)).a_k$ i.e. $\overline{\rho}$ contains the history of what O has chosen to observe at each step and the events that occurred after each choice.

Let $\mathcal{O} : (2^\Sigma \times \Sigma^\varepsilon)^+ \to 2^{2^\Sigma}$. $\mathcal{O}$ is the most permissive observer for (A, k) if the following holds:

$$\begin{array}{c} O = (S, s_0, \Sigma, \delta, L) \\ \textit{is an observer and} \\ A \textit{ is } (O, k)\textit{-diagnosable} \end{array} \iff \begin{array}{c} \forall w \in \Sigma^* \\ L(\delta(s_0, w)) \in \mathcal{O}(\overline{w}) \end{array}$$

The definition of the most permissive observer states that:

- any good observer Obs (one such that A is (Obs, k)-diagnosable) must choose a set of observable events in $\mathcal{O}(\overline{w})$ on input w;
- if an observer chooses its set of observable events in $\mathcal{O}(\overline{w})$ on input w, then it is a good observer.

Theorem 6 of [1] establishes that there is a most permissive observer $\mathcal{F}_A$ in case A is (Σ, k)-diagnosable and it can be computed in exponential time in the size of A and k, doubly exponential time in $|\Sigma|$, and has size exponential in A and k, and doubly exponential in $|\Sigma|$. Moreover the most permissive observer $\mathcal{F}_A$ can be represented by a finite state machine $S_{\mathcal{F}_A} = (\{0, 2 \cdots, l\} \cup (\{1, 3, \cdots, 2l'+1\} \times 2^\Sigma), 0, \Sigma \cup 2^\Sigma, \delta)$ which has the following properties:

- even states are states where the observer chooses a set of events to observe;
- odd states $(2i+1, X)$ are states where the observer waits for an observable event in X to occur;
- if $\delta(2i, X) = (2i'+1, X)$ with $X \in 2^{\Sigma}$, it means that from an even state $2i$, the automaton $S_{\mathcal{F}_A}$ can select a set X of events to observe. The successor state is an odd state together with the set X of events that are being observed;
- if $\delta((2i+1, X), a) = 2i'$ with $a \in X$, it means that from $(2i+1, X)$, $S_{\mathcal{F}_A}$ is waiting for an observable event to occur. When some occurs it switches to an even state.

By definition of $\mathcal{F}_A$, any observer O s.t. A is (O, k)-diagnosable must select a set of observable events in $\mathcal{F}_A(tr(\overline{w}))$ after having observed $w \in \boldsymbol{\pi}_{/\Sigma}(\mathcal{L}(A))$.

Example 5 *For the automaton $\mathcal{A}$ of Fig. 3, we obtain the most permissive observer $\mathcal{F}_{\mathcal{A}}$ of Fig. 8. For odd states we have not mentionned the component X that has last been picked up by the observer. X is the label of the unique incoming transition. In the even states, the observer chooses what to observe and in the odd states it moves according to what it observes. When the system starts, it can choose either $\{a, b\}$ or $\{a\}$. Once an a has been observed it can choose any subset containing b. When a b has been observed the observer can choose to observe the empty set.*

We point out that from an odd state $(2i+1, X)$, outgoing transitions are labeled by elements of X. This does not mean that the DES under observation cannot do other actions from state $2i+1$: it might be able to do so but there are unobservable for the observer.

4.2. Optimal Dynamic Observers

To compute an optimal observer, we use a result by Zwick and Paterson [12] on *weighted graph games* (see Appendix B for Zwick and Paterson algorithm). These are graphs (V, E) with the set of nodes partitioned into two sets: V_1 for Player 1 and V_2 for Player 2. In a V_i state it is Player i's turn to play. There is a *weight* function that associates with each edge e the weight $w(e)$. The players build paths $e_1 . \cdots . e_n$ by choosing an edge when it is their turn to play. The goal of the game is for Player 1 to maximize the value $\liminf_{n\to\infty} \frac{1}{n}\sum_{i=1}^{n} w(e_i)$ and for Player 2 to minimize $\limsup_{n\to\infty} \frac{1}{n}\sum_{i=1}^{n} w(e_i)$. One of the results by Zwick and Paterson [12] is that:

- there is a value $\nu \in \mathbb{Q}$, called the *value of the game* s.t. Player 1 has a strategy to ensure that $\liminf_{n\to\infty} \frac{1}{n}\sum_{i=1}^{n} w(e_i) \geq \nu$ and Player 2 has a strategy to ensure that $\limsup_{n\to\infty} \frac{1}{n}\sum_{i=1}^{n} w(e_i) \leq \nu$; this value can be computed in $O(|V|^3 \times |E| \times W)$ where W is the range of the weight function (assuming the weights are in the interval $[-W..W]$). Note that deciding whether this value satisfies $\nu \bowtie c$ for $\bowtie \in \{=, <, >\}$ for $c \in \mathbb{Q}$ can be done in $O(|V|^2 \times |E| \times W)$.
- there are optimal memoryless strategies for both players that can be computed in $O(|V|^4 \times |E| \times \log(|E|/|V|) \times W)$.

To solve the Problem 2, we use the most permissive observer we computed in section 4.1. Given A and $\mathcal{F}_A$, we build a weighted graph game $G(A, \mathcal{F}_A)$ s.t. the value of the game is the optimal cost for the set of all observers. Moreover an optimal observer can be obtained by taking an optimal memoryless strategy in $G(A, \mathcal{F}_A)$.

To build $G(A, \mathcal{F}_A)$ we use the same idea as in section 3.4: we replace ε and f transitions in A by u obtaining A^+. We also modify $\mathcal{F}_A$ to obtain a weighted graph game $(\mathcal{F}_A^+, w)$ by adding transitions so that each state $2k+1$ is complete w.r.t. Σ^u. This is done as follows:

- from each $(2i+1, X)$ state, create a new even state i.e. pick some $2i'$ that has not already been used. Add transitions $((2i+1, X), \sigma, 2i')$ for each $\sigma \in \Sigma^u \setminus en(2i+1, X)$. Add also a transition $(2i', X, (2i+1, X))$. This step means that if a A produces an event and it is not observable, $\mathcal{F}_A^+$ just reads the event and makes the same choice again.
- the weight of a transition $(2i, X, (2i'+1, X))$ is $|X|$.

The automaton $\mathcal{F}_A^+$ obtained from $\mathcal{F}_A$ is depicted on Fig. 9. The game $G(A, \mathcal{F}_A)$ is then $A^+ \times \mathcal{F}_A^+$. This way we can obtain a weighted graph game $WG(A, \mathcal{F}_A)$ by abstracting away the labels of the transitions. Notice that it still enables us to convert any strategy in $WG(A, \mathcal{F}_A)$ to a strategy in $\mathcal{F}_A$. A strategy in $WG(A, \mathcal{F}_A)$ will define an edge $(2i, (2i'+1, X))$ to take. As the target vertex contains the set of events we chose to observe we can define a corresponding strategy in $\mathcal{F}_A$.

By construction of $G(A, \mathcal{F}_A)$ and the definition of the value of a weighted graph game, the value of the game is the optimal cost for the set of all observers O s.t. A is (O, k)-diagnosable.

Assume A has n states and m transitions. From [1] we know that $\mathcal{F}_A$ has at most $O(2^{n^2} \times 2^k \times 2^{2^{|\Sigma|}})$ states and $O(2^{n^2} \times 2^k \times 2^{2^{|\Sigma|}} \times n^2 \times k \times m)$ transitions. Hence $G(A, \mathcal{F}_A)$ has at most $O(n \times 2^{n^2} \times 2^k \times 2^{2^{|\Sigma|}})$ vertices and $O(m \times 2^{n^2} \times 2^k \times 2^{2^{|\Sigma|}}$ edges. To make the game complete we may add at most half the number of states and hence $WG(A, \mathcal{F}_A)$ has the same size. We thus obtain the following results:

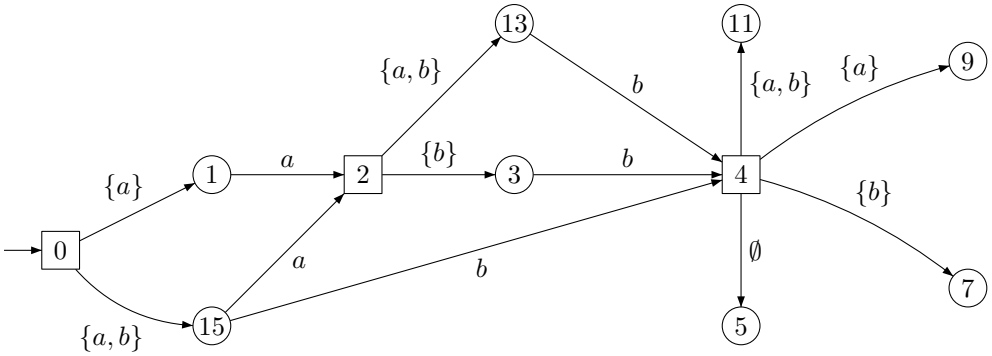

Figure 8. Most Permissive Observer for the Automaton $\mathcal{A}$ of Fig. 3

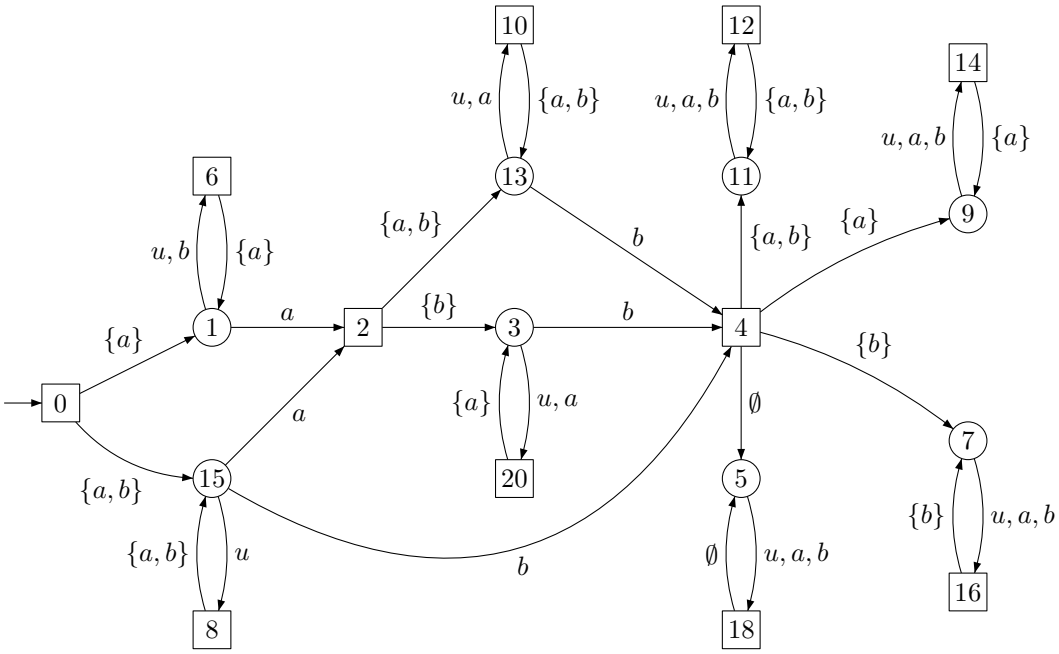

Figure 9. The Automaton $\mathcal{F}_A^+$

Theorem 3 *Problem 2 can be solved in time* $O(|\Sigma| \times m \times 2^{n^2} \times 2^k \times 2^{2^{|\Sigma|}})$.

We can even solve the optimal cost computation problem:

Problem 3 (Optimal Cost Observer)
INPUT: A, $k \in \mathbb{N}$.
PROBLEM: *Compute the least value* m *s.t. there exists an observer* Obs *s.t.* A *is (*Obs*,k)-diagnosable and Cost*$_2$(Obs) $\leq m$.

Theorem 4 *Problem 3 can be solved in time* $O(|\Sigma| \times m \times 2^{n^2} \times 2^k \times 2^{2^{|\Sigma|}})$.

A consequence of Theorem 3 is that the cost of the optimal observer is a rational number (see Appendix B).

Example 6 *For the example* $\mathcal{A}$ *of Fig. 4 and* $\mathcal{F}_A^+$ *of Fig. 9, using Zwick and Paterson's algorithm (Appendix B) we obtain that the optimal cost is* 1 *and the optimal strategy is to use the observer* Obs *of Fig. 2.*

5. Conclusion & Future Work

In this paper we have addressed sensor optimization problems in the context of fault diagnosis, using dynamic observers. We have defined a suitable notion of cost for such observers. Then we have proved that, given such observer, we could compute the cost of diagnosing a DES. This is done by reducing this problem to the computation of a maximum mean-weight cycle in a weighted graph. Hence we can solve it in polynomial time. We have also solved the optimal observer synthesis problem i.e. compute an observer of optimal cost by reducing it to an optimization on weighted graph game.

Further work will include:

- finding the exact complexity class of Problems 2 and 3;
- dealing with more realistic examples. This requires an implementation of our algorithms and of the algorithms described in Appendix A and B.

References

[1] Franck Cassez, Stavros Tripakis, and Karine Altisen. Sensor minimization problems with static or dynamic observers for fault diagnosis. Technical Report RI-2007-1, IRCCyN/CNRS, 1 rue de la Noë, BP 92101, 44321 Nantes Cedes, France, January 2007. Available at `http://www.irccyn.fr/franck`.

[2] Franck Cassez, Stavros Tripakis, and Karine Altisen. Sensor minimization problems with static or dynamic observers for fault diagnosis. In *7th International Conference on Application of Concurrency to System Design (ACSD'07)*, Bratislava, Slovak Republic, July 2007. IEEE Computer Society.

[3] Ali Dasdan, Sandy S. Irani, and Rajesh K. Gupta. Efficient algorithms for optimum cycle mean and optimum cost to time ratio problems. In *Annual ACM IEEE Design Automation Conference*, pages 37–42, New Orleans, Louisiana, United States, 1999. ACM Press New York, NY, USA. ISBN:1-58133-109-7.

[4] Rami Debouk, Stéphane Lafortune, and Demosthenis Teneketzis. On an optimization problem in sensor selection. *Discrete Event Dynamic Systems*, 4(12), November 2004.

[5] Shengbing Jiang, Zhongdong Huang, Vigyan Chandra, and Ratnesh Kumar. A polynomial algorithm for testing diagnosability of discrete event systems. *IEEE Transactions on Automatic Control*, 46(8), August 2001.

[6] Shengbing Jiang, Ratnesh Kumar, and Humberto E. Garcia. Optimal sensor selection for discrete event systems with partial observation. *IEEE Transactions on Automatic Control*, 48(3):369–381, March 2003.

[7] Richard M. Karp. A characterization of the minimum mean cycle in a digraph. *Discrete Mathematics*, 23:309–311, 1978.

[8] Peter Ramadge and W. Murray Wonham. Supervisory control of a class of discrete event processes. *SIAM J. Control Optim.*, 25(1), January 1987.

[9] Meera Sampath, Raja Sengupta, Stephane Lafortune, Kasim Sinnamohideen, and Demosthenis C. Teneketzis. Diagnosability of discrete event systems. *IEEE Transactions on Automatic Control*, 40(9), September 1995.

[10] John N. Tsitsiklis. On the control of discrete event dynamical systems. *Mathematics of Control, Signals and Systems*, 2(2), 1989.

[11] Tae-Sic Yoo and Stéphane Lafortune. On the computational complexity of some problems arising in partially-observed discrete event systems. In *American Control Conference (ACC'01)*, 2001. Arlington, VA.

[12] Uri Zwick and Michael S. Paterson. The complexity of mean payoff games on graphs. *Theoretical Computer Science*, 158(1–2):343–359, 1996.

A. Karp's Maximum Mean Cycle Algorithm

In this section, we recall Karp's maximum mean cycle algorithm [7]. The original algorithm works for weighted graph where the weights are on the edges. We give a version where weights are on vertices.

Definition 6 (Weighted Graph) *A* weighted directed graph *is a pair (G, w) s.t. $G = (V, E)$ is a directed graph and $w : V \to \mathbb{R}$. We assume that each vertex $v \in V$ is reachable from a unique* source *vertex s_0.* ■

Definition 7 (Mean Weight of a Cycle) *Let $c = (v_1, v_2, \cdots, v_k)$ be a sequence of vertices s.t. each $(v_i, v_{i+1}) \in E, 1 \leq i \leq k-1$ which is a cycle i.e. $v_1 = v_k$. The* mean weight *of c is $\mu(c) = \frac{1}{k} \cdot \Sigma_{i=1}^{k} w(v_i)$.* ■

Let $\nu^* = \max_c \mu(c)$ where c ranges over all directed cycles in G. A cycle c with $\mu(c) = \nu^*$ is a *maximum mean-weight cycle*.

Let $D(v)$ be the weight of a most expensive path from s_0 to v and $D_k(v)$ be the weight of a most expensive path which has exactly k edges (if there is no such path $D_k(v) = -\infty$).

Assume $|V| = n$. Karp's algorithm is based on the fact that

$$\nu^* = \max_{v \in V} \min_{0 \leq k \leq n-1} \frac{D_n(v) - D_k(v)}{n - k}$$

The values $D_k(v)$ can be computed iteratively:

$$D_0(s_0) = w(s_0) \quad (1)$$

$$D_0(v) = -\infty \quad \textit{for} \quad v \neq s_0 \quad (2)$$

$$D_{k+1}(v) = \max_{(u,v) \in E} \{D_k(u) + w(v)\} \quad (3)$$

Thus for each vertex we can compute $\min(v) = \min_{0 \leq k \leq n-1} \frac{D_n(v) - D_k(v)}{n-k}$ and then compute the value $\max_{v \in V} \min(v)$ to obtain ν^*. This algorithm runs in $O(n.m)$ where $|V| = n$ and $|E| = m$. Improvements [3] can be made to this algorithm still the worst case run-time is $O(n.m)$.

B. Zwick and Paterson's Algorithm

In this section we give an overview of some results of [12]. Assume $G = (V, E)$ is a weighted graph as in definition 6 except that the weight function is defined on edges: $w : E \to \{-W, \cdots, 0, \cdots, W\}$ assigns an integral weight to each edge of G with $W \in \mathbb{N}$. We assume each vertex has at least one outgoing transition.

Definition 8 (Weighted Graph Game) *A* weighted graph game *$G = (V, E)$ is a bipartite weighted graph with $V = V_1 \cup V_2$ and $E = E_1 \cup E_2$, $E_1 \subseteq V_1 \times V_2$ and $E_2 \subseteq E_2 \times E_1$. We assume the initial vertex v_0 of G belongs to V_1.* ■

Vertices V_i are Player i's vertex. A weighted graph game is a turn based game in which the turn alternates between Player 1 and Player 2. The game starts at a vertex $v_0 \in V_1$. Player 1 chooses an edge $e_1 = (v_0, v_1)$ and then Player 2 chooses an edge $e_2 = (v_1, v_2)$ and so on and they build an infinite sequence of edges. Player 1 wants to maximise $\liminf_{n \to \infty} \frac{1}{n} \cdot \sum_{i=1}^{n} w(e_i)$ and Player 2 wants to minimize $\limsup_{n \to \infty} \frac{1}{n} \cdot \sum_{i=1}^{n} w(e_i)$.

One of the result of [12] is that there is a rational value $\nu \in \mathbb{Q}$ s.t. Player 1 has a strategy to ensure $\liminf_{n \to \infty} \frac{1}{n} \cdot \sum_{i=1}^{n} w(e_i) \geq \nu$ and Player 2 has a strategy to ensure that $\limsup_{n \to \infty} \frac{1}{n} \cdot \sum_{i=1}^{n} w(e_i) \leq \nu$. ν is called the value of the game.

Let $n = |V|$. To compute ν, proceed as follows ([12]):

1. Let $\nu_0(v) = 0$ for $v \in V$. For $v \in V$ and $k \geq 1$, $\nu_k(v)$ is defined by:

$$\nu_k(v) = \begin{cases} \max_{(v,w) \in E} \{w(v, w) + \nu_{k-1}(w)\} \textit{ if } v \in V_1 \\ \min_{(v,w) \in E} \{w(v, w) + \nu_{k-1}(w)\} \textit{ if } v \in V_2 \end{cases}$$

   This is the equivalent of the $D_k(v)$ values for Karp's algorithm using a min max strategy depending on which player is playing;

2. for each $v \in V$, compute $\nu'(v) = \nu_k(v)/k$ for $k = 4 \cdot n^3 \cdot W$.

3. for each vertex, the value of the game from v is the only rational number with a denominator at most n that lies in the interval $]\nu'(v) - \alpha, \nu'(v) + \alpha[$ with $\alpha = \frac{1}{2n(n-1)}$.

The value of the game is $\nu = \nu(v_0)$ where v_0 is the initial vertex.

To compute an optimal strategy for Player 1, proceed as follows:

1. compute the values $\nu(v)$ for each $v \in V$;

2. if all the vertices of V_1 have outgoing degree 1, there is a unique strategy and it is positional and optimal;

3. otherwise, take a vertex $v \in V_1$ with outgoing degree $d \geq 2$. Remove $\lceil \frac{d}{2} \rceil$ edges from v leaving at least one. Recompute the value m_v for each v. If $m_v = \nu(v)$, there is an optimal positional strategy which uses the remaining edges from v. Otherwise there is a positional strategy that uses one of the removed edges.

We can iterate the previous scheme to find an optimal strategy for Player 1.