Complexity in Infinite Games on Graphs and Temporal Constraint Networks

Carlo Comin

Advisor: Prof. Romeo Rizzi (University of Verona, Italy)
Co-Advisor: D.d.R. Stéphane Vialette (LIGM UPEM, CNRS, France)
Referee: Prof. Luke Hunsberger (Vassar College, NY, US)
Referee: Prof. Angelo Montanari (University of Udine, Italy)

University of Trento jointly University of Verona
in co-tutelle with
Université Paris-Est in Marne-la-Vallée
Chapter 1

*Introduction and Context*
Temporal constraint networks*

Rina Dechter**
Computer Science Department, Technion—Israel Institute of Technology, Haifa 32000, Israel

Itay Meiri and Judea Pearl
Cognitive Systems Laboratory, Computer Science Department, University of California, Los Angeles, CA 90024, USA

Received November 1989
Revised July 1990
This dissertation provides further evidence that game theoretic arguments help to study algorithmic problems in the area of *automated temporal planning* and *formal verification of finite state non-terminating systems*.

- *Automated temporal planning* is a branch of Artificial Intelligence (AI) that concerns the realization of temporal strategies or temporal action sequences, typically for execution by intelligent agents, autonomous robots and unmanned vehicles.

- *Non-terminating computing systems* abound in environments as varied as household appliances, medical equipment, industrial control systems, flight control systems in airplanes, etc. Failures caused by design faults may be very costly and they should be avoided as much as possible. Behaviour of such systems is typically very complex which makes their design and validation a challenge. *Formal methods* try to address this challenge by developing formal models of such systems, and methods to specify and reason about their properties.
Introduction and Context

- We began our research by studying algorithmic problems in automated temporal planning, particularly, *conditional temporal planning*.
- At some point, we have identified interesting connections between the algorithmics of these problems and that of a specific family of infinite 2-player pebble games played on finite graphs.
  - i.e. Energy (EGs) and Mean-Payoff Games (MPGs).
  - MPGs are intimately related to Parity Games and the semantics of the modal $\mu$-calculus, a well-known logic for formal verification.
In summary...

- Algorithmic Game Theory (especially, infinite two-player pebble games played on finite graphs) is the red thread that makes it possible to look at the various contributions of this dissertation in a sufficiently coherent way.

- A game theoretic formulation helps to abstract away from syntactic and semantic peculiarities of modelling formalisms and makes the conditional temporal constraints problems in question more easily amenable to algorithmic and complexity analysis.

- On the other side, this have led us to deepen the study of algorithmic and complexity issues in infinite games on finite graphs per se. Ultimately, we obtained faster algorithms and improved complexity bounds for some of these games, i.e., Update Games, Explicit McNaughton-Müller Games, and finally, for Mean Payoff Games.
Summary of Results (I)

**Temporal Constraint Networks** (Temporal Planning and Scheduling) [Dechter, 1991]

1. Hyper Temporal Networks (A Tractable Generalization of Simple Temporal Networks and its relation to Mean Payoff Games)

2. Singly-Exponential Time DC-Checking of Conditional Simple Temporal Networks via Mean Payoff Games
   - 22st International Symposium on Temporal Representation and Reasoning (TIME2015), Kassel, Germany, 2015
   - Information and Computation, Elsevier, Special Issue of TIME2015

3. Instantaneous Reaction-Time in DC-Checking of Conditional Simple Temporal Networks
   - 23st International Symposium on Temporal Representation and Reasoning (TIME2016), Copenhagen, Denmark, 2016

Complexity in Infinite Games on Graphs and Temporal Constraint Networks
Carlo Comin
Summary of Results (II)

Infinite Games on Finite Graphs (Games for Formal Verification) [Grädel, 2002]

Parity Games, Mean Payoff Games, Energy Games, Update Games, Müller-McNaughton, ...

1. Linear-Time Algorithm for Update Games via Strongly-Trap-Connected-Components
   - Submitted.

2. An Improved Upper Bound on the Time Complexity of the Value Problem and Optimal Strategy Synthesis in Mean Payoff Games

3. Faster $O(|V|^2|E|W)$-Time Energy Algorithms and Energy-Lattice Decomposition for Optimal Strategy Synthesis in Mean Payoff Games
   - Submitted.
Summary of Results (III)

Not part of the dissertation, but for completeness:

Algorithms and Complexity for Computational Biology

1. Sorting with Forbidden Intermediates
   - Submitted to IEEE/ACM Transactions on Computational Biology and Bioinformatics (TCBB), Special Issue of AlCoB 2016

2. An Improved Upper Bound on Maximal Clique Listing via Rectangular Fast Matrix Multiplication (i.e., from $O(n^{2.3728639})$ to $O(n^{2.093362})$ time-delay)
Chapter 2

Hyper Temporal Networks
Simple Temporal Problems

Simple Temporal Networks (STNs) [Dechter, Meiri, Pearl, 1991]

- Represent a general framework for analyzing systems (conjunctions) of difference constraints on ordered pairs of temporal variables.
- An STN can be encoded by a weighted directed graph:
  - a node represents a time-point variable (time-point);
  - an arc represents a temporal distance constraint:
    - $u \xrightarrow{a} v$ stands for $v - u \leq a$, $a \in \mathbb{R}$;
    - $(v \xleftarrow{a} u$ stands for $v - u \geq a$).

Represented Constraints:

- $4 \leq A - D \leq 5$
- $6 \leq B - A \leq 7$
- $1 \leq C - A \leq 4$
- $-1 \leq B - C \leq 1$
- $1 \leq C - D \leq 3$
Consistency of Simple Temporal Problems

STN Consistency [Dechter, Meiri, Pearl 1991]

- An STN $\langle G = (V, E), \ell \rangle$ is consistent if it admits a feasible scheduling function, i.e., we can assign a real value $s(v)$ to each time-point $v$, such that all constraints are satisfied:
  \[ \exists s : V \rightarrow \mathbb{R} \text{ such that:} \]
  \[ s(v) \leq s(u) + \ell_{(u,v)} \quad \forall (u, v) \in E. \]

- An STN is not consistent if it contains a negative cycle.

Represented Constraints:

- $4 \leq A - D \leq 5$
- $6 \leq B - A \leq 7$
- $1 \leq C - A \leq 4$
- $-1 \leq B - C \leq 1$
- $1 \leq C - D \leq 3$
Consistency of Simple Temporal Problems

STN Consistency [Dechter, Meiri, Pearl 1991]

- An STN $\langle G = (V, E), \ell \rangle$ is consistent if it admits a feasible scheduling function, i.e., we can assign a real value $s(v)$ to each time-point $v$, such that all constraints are satisfied:
  \[ \exists s : V \mapsto \mathbb{R} \text{ such that:} \]
  \[ s(v) \leq s(u) + \ell(u,v) \quad \forall (u, v) \in E. \]

- An STN is not consistent if it contains a negative cycle.

Represented Constraints:
- $4 \leq A - D \leq 5$
- $6 \leq B - A \leq 7$
- $1 \leq C - A \leq 4$
- $-1 \leq B - C \leq 1$
- $1 \leq C - D \leq 3$
Introduction - STNs

### STN Consistency

- An STN \((G = (V, E), \ell)\) is **consistent** if it admits a feasible scheduling function, i.e., we can assign a real value \(s(v)\) to each time-point \(v\), such that all constraints are satisfied:
  \[
  \exists s : V \rightarrow R \text{ such that: } s(v) \leq s(u) + \ell_{(u,v)} \quad \forall (u, v) \in E.
  \]
- An STN is **not consistent** if it contains a negative cycle.

![Graph](image)

Represented Constraints:
- \(4 \leq A - D \leq 5\)
- \(6 \leq B - A \leq 7\)
- \(1 \leq C - A \leq 4\)
- \(-1 \leq B - C \leq 1\)
- \(1 \leq C - D \leq 3\)
An STN \((G = (V, E), \ell)\) is **consistent** if it admits a feasible scheduling function, i.e., we can assign a real value \(s(v)\) to each time-point \(v\), such that all constraints are satisfied:

\[
\exists s : V \mapsto \mathbb{R} \text{ such that: } s(v) \leq s(u) + \ell_{(u,v)} \quad \forall (u, v) \in E.
\]

An STN is **not consistent** if it contains a negative cycle.

Represented Constraints:

- \(4 \leq A - D \leq 5\)
- \(6 \leq B - A \leq 7\)
- \(1 \leq C - A \leq 4\)
- \(-1 \leq B - C \leq 1\)
- \(1 \leq C - D \leq 3\)
An STN \((G = (V, E), \ell)\) is consistent if it admits a feasible scheduling function, i.e., we can assign a real value \(s(v)\) to each time-point \(v\), such that all constraints are satisfied:

\[
\exists s : V \rightarrow \mathbb{R} \text{ such that: } \\
s(v) \leq s(u) + \ell(u,v) \quad \forall (u, v) \in E.
\]

An STN is not consistent if it contains a negative cycle.
STN Consistency

- An STN \((G = (V, E), \ell)\) is consistent if it admits a feasible scheduling function, i.e., we can assign a real value \(s(v)\) to each time-point \(v\), such that all constraints are satisfied:

\[
\exists s : V \rightarrow \mathbb{R} \text{ such that: } s(v) \leq s(u) + \ell(u,v) \quad \forall (u, v) \in E.
\]

- An STN is *not consistent* if it contains a negative cycle.

Represented Constraints:

\[
\begin{align*}
4 & \leq A - D \leq 5 \\
6 & \leq B - A \leq 7 \\
1 & \leq C - A \leq 4 \\
-1 & \leq B - C \leq 1 \\
1 & \leq C - D \leq 3
\end{align*}
\]
Introduction - STNs

**STN Consistency**

- An STN \((G = (V, E), \ell)\) is **consistent** if it admits a feasible scheduling function, i.e., we can assign a real value \(s(v)\) to each time-point \(v\), such that all constraints are satisfied:

\[
\exists s : V \rightarrow R \text{ such that:}
\]

\[
s(v) \leq s(u) + \ell(u,v) \quad \forall (u, v) \in E.
\]

- An STN is **not consistent** if it contains a negative cycle.

![Graph](image)

Represented Constraints:

- \(4 \leq A - D \leq 5\)
- \(1 \leq C - A \leq 4\)
- \(-1 \leq B - C \leq 1\)
- \(1 \leq C - D \leq 3\)
An STN \((G = (V, E), \ell)\) is consistent if it admits a feasible scheduling function, i.e., we can assign a real value \(s(v)\) to each time-point \(v\), such that all constraints are satisfied:

\[
\exists s : V \rightarrow \mathbb{R} \text{ such that: }
\ s(v) \leq s(u) + \ell_{(u,v)} \quad \forall (u, v) \in E.
\]

An STN is not consistent if it contains a negative cycle.
Introduction - STNs

STN Consistency

- An STN \((G = (V, E), \ell)\) is consistent if it admits a feasible scheduling function, i.e., we can assign a real value \(s(v)\) to each time-point \(v\), such that all constraints are satisfied:
  \[
  \exists s : V \mapsto \mathbb{R} \text{ such that: } s(v) \leq s(u) + \ell(u,v) \quad \forall (u, v) \in E.
  \]
- An STN is not consistent if it contains a negative cycle.

Complexity in Infinite Games on Graphs and Temporal Constraint Networks
Carlo Comin
STN Consistency

- An STN \((G = (V, E), \ell)\) is consistent if it admits a feasible scheduling function, i.e., we can assign a real value \(s(v)\) to each time-point \(v\), such that all constraints are satisfied:

\[
\exists s : V \rightarrow \mathbb{R} \text{ such that:} \\
s(v) \leq s(u) + \ell(u,v) \quad \forall (u, v) \in E.
\]

- An STN is not consistent if it contains a negative cycle.
STN Consistency

- An STN $(G = (V, E), \ell)$ is consistent if it admits a feasible scheduling function, i.e., we can assign a real value $s(v)$ to each time-point $v$, such that all constraints are satisfied:

$$\exists s : V \rightarrow \mathbb{R} \text{ such that: } s(v) \leq s(u) + \ell(u, v) \quad \forall (u, v) \in E.$$ 

- An STN is not consistent if it contains a negative cycle.

Complexity in Infinite Games on Graphs and Temporal Constraint Networks

Carlo Comin
A hypergraph $\mathcal{H}$ is a pair $(V, \mathcal{A})$, where $V$ is the set of nodes, and $\mathcal{A}$ is the set of hyperarcs. Each hyperarc $A \in \mathcal{A}$ has a distinguished node $t_A$, called the tail of $A$, and a nonempty weighted set $(H_A, w_A)$, where $H_A \subseteq V \setminus \{t_A\}$ contains the heads of $A$, and each head $v \in H_A$ is associated with a weight $w_A(v) \in \mathbb{R}$. 
An HyTN $\mathcal{H} = (V, A)$ is consistent if it admits a feasible scheduling function, i.e., we can assign a real value $\phi(v)$ to each time-point $v$, such that each hyperarc constraint is satisfied:

$$\phi(t_A) \geq \min_{v \in H_A} \{ \phi(v) - w_A(v) \} \quad \forall A \in \mathcal{A}.$$  \hspace{1cm} (1)
We may consider multi-tail hyperarcs as well multi-head ones. This allows to introduce, in any HyTN model, an event bound to occur precisely when the last/first event of an associated set of events occurs. This allows to represent AND/OR join connectors from workflows. These could not be done before.
Our findings on the two versions of HyTNs in a nutshell

A complexity map, algorithms, and characterizations

- If we allow both multi-tails arcs and multi-heads arcs the Consistency-Checking problem becomes NP-complete.
- The multi-tails and the multi-heads versions of HyTNs are inter-reducible. (Direct reduction by reversing all arcs and inverting the time axis).
- If we allow either only multi-tails arcs or only multi-heads arcs the Consistency problem can be efficiently solved in pseudo-polynomial time.
- Both of these problems can be reduced to the Determinacy problem for Energy and Mean Payoff Games.
- In fact, both of these problems are equivalent to the Determinacy problem for Mean Payoff Games.
- HyTN’s Consistency-Checking is well characterized: we have introduced a notion of negative hyper-circuit as NO-certificate.
Our findings on the two versions of HyTNs in a nutshell

A complexity map, algorithms, and characterizations

- If we allow **both** multi-tails arcs and multi-heads arcs the Consistency-Checking problem becomes NP-complete.
- The multi-tails and the multi-heads versions of HyTNs are inter-reducible. (Direct reduction by reversing all arcs and inverting the time axis).
- If we allow **either** only multi-tails arcs or only multi-heads arcs the Consistency problem can be efficiently solved in pseudo-polynomial time.
- Both of these problems can be reduced to the Determinacy problem for Energy and Mean Payoff Games.
- In fact, both of these problems are equivalent to the Determinacy problem for Mean Payoff Games.
- HyTN’s Consistency-Checking is well characterized: we have introduced a notion of negative hyper-circuit as NO-certificate.
Our findings on the two versions of HyTNs in a nutshell

A complexity map, algorithms, and characterizations

- If we allow both multi-tails arcs and multi-heads arcs the Consistency-Checking problem becomes NP-complete.
- The multi-tails and the multi-heads versions of HyTNs are inter-reducible. (Direct reduction by reversing all arcs and inverting the time axis).
- If we allow either only multi-tails arcs or only multi-heads arcs the Consistency problem can be efficiently solved in pseudo-polynomial time.
- Both of these problems can be reduced to the Determinacy problem for Energy and Mean Payoff Games.
- In fact, both of these problems are equivalent to the Determinacy problem for Mean Payoff Games.
- HyTN’s Consistency-Checking is well characterized: we have introduced a notion of negative hyper-circuit as NO-certificate.
Our findings on the two versions of HyTNs in a nutshell

A complexity map, algorithms, and characterizations

- If we allow both multi-tails arcs and multi-heads arcs the Consistency-Checking problem becomes NP-complete.

- The multi-tails and the multi-heads versions of HyTNs are inter-reducible. (Direct reduction by reversing all arcs and inverting the time axis).

- If we allow either only multi-tails arcs or only multi-heads arcs the Consistency problem can be efficiently solved in pseudo-polynomial time.

- Both of these problems can be reduced to the Determinacy problem for Energy and Mean Payoff Games.

- In fact, both of these problems are equivalent to the Determinacy problem for Mean Payoff Games.

- HyTN's Consistency-Checking is well characterized: we have introduced a notion of negative hyper-circuit as NO-certificate.
Our findings on the two versions of HyTNs in a nutshell

A complexity map, algorithms, and characterizations

- If we allow both multi-tails arcs and multi-heads arcs the Consistency-Checking problem becomes NP-complete.
- The multi-tails and the multi-heads versions of HyTNs are inter-reducible. (Direct reduction by reversing all arcs and inverting the time axis).
- If we allow either only multi-tails arcs or only multi-heads arcs the Consistency problem can be efficiently solved in pseudo-polynomial time.
- Both of these problems can be reduced to the Determinacy problem for Energy and Mean Payoff Games.
- In fact, both of these problems are equivalent to the Determinacy problem for Mean Payoff Games.

- HyTN’s Consistency-Checking is well characterized: we have introduced a notion of negative hyper-circuit as NO-certificate.
Our findings on the two versions of HyTNs in a nutshell

A complexity map, algorithms, and characterizations

- If we allow both multi-tails arcs and multi-heads arcs the Consistency-Checking problem becomes NP-complete.

- The multi-tails and the multi-heads versions of HyTNs are inter-reducible. (Direct reduction by reversing all arcs and inverting the time axis).

- If we allow either only multi-tails arcs or only multi-heads arcs the Consistency problem can be efficiently solved in pseudo-polynomial time.

- Both of these problems can be reduced to the Determinacy problem for Energy and Mean Payoff Games.

- In fact, both of these problems are equivalent to the Determinacy problem for Mean Payoff Games.

- HyTN’s Consistency-Checking is well characterized: we have introduced a notion of negative hyper-circuit as NO-certificate.
Given a multi-head HyTN $\mathcal{H} = (V, A)$, a cycle is a pair $(S, C)$ with $S \subseteq V$ and $C \subseteq A$ such that:

1. $S = \bigcup_{A \in C} (H_A \cup \{t_A\})$ and $S \neq \emptyset$;
2. $\forall v \in S$ there exists an unique $A \in C$ such that $t_A = v$. 

Complexity in Infinite Games on Graphs and Temporal Constraint Networks

Carlo Comin
Extensions of HyTNs: NP-Completeness proofs

Let $\varphi(x_1, \ldots, x_n) = \bigwedge_{i=1}^{m} (\alpha_i \lor \beta_i \lor \gamma_i)$ be a 3-CNF formula,

\begin{itemize}
  \item[(a)] Gadget for a 3-SAT variable $x_i$.
  \item[(b)] Gadget for a 3-SAT clause $C_j = (\alpha_j \lor \beta_j \lor \gamma_j)$ where each $\alpha_j, \beta_j, \gamma_j$ is a positive or negative literal.
\end{itemize}
The local replacement reduction

How an hyperarc (multi-head) gets replaced:

If $H = (V, \mathcal{A})$ then $G_H = (V_0 \cup V_1, E)$, where:

- $V_1 = V$;
- $V_0 = \mathcal{A}$;
- $E = \{(t_A, A, 0) \mid A \in \mathcal{A}\} \cup \{(A, h, w_A(h)) \mid A \in \mathcal{A}, h \in H_A\}$. 
Mean Payoff Games: the model

A Mean-Payoff Game (MPG) is a two-player infinite game played on an arena $\Gamma = \langle V, E, w, (V_{\text{Max}}, V_{\text{Min}}) \rangle$.

$G^\Gamma = \langle V, E, w \rangle$ is a finite weighted directed graph whose nodes are partitioned in two classes, $V_{\text{Max}}$ and $V_{\text{Min}}$.

Every node has at least one outgoing arc.

Weights are integers, i.e., $w : E \to \mathbb{Z}$.

Nodes in $V_p$, where $p \in \{\text{Max}, \text{Min}\}$, are those under control of Player $p$. 
Each match starts with a pebble placed at some node $v \in V_{\text{Max}} \cup V_{\text{Min}}$.

Here $v = A \in V_{\text{Max}}$, the nodes controlled by Player Max.

Player Max chooses an arc $e \in E$ exiting $v$ and moves the pebble along $e$. 
The two players move the pebble ad infinitum along the arcs...

The infinite sequence of encountered nodes, i.e.,

$$\pi = v_0 v_1 \cdots v_n \cdots = ADC(DC)^*$$

is a play.
Mean Payoff Games: play example

In order to play well, Player Max wants to maximize the limit inferior of the long-run average weight, i.e., \( \lim \inf_{n \to \infty} \frac{1}{n} \sum_{i=0}^{n-1} w(v_i, v_{i+1}) \).

and Player Min wants to minimize \( \lim \sup_{n \to \infty} \frac{1}{n} \sum_{i=0}^{n-1} w(v_i, v_{i+1}) \).
Formally, the Player 0’s *payoff* of a play $v_0v_1 \ldots v_n \ldots$ in $\Gamma$ is:

$$\text{MP}_0(v_0v_1 \ldots v_n \ldots) := \lim_{n \to \infty} \inf \frac{1}{n} \sum_{i=0}^{n-1} w(v_i, v_{i+1}).$$

The value *secured* by a strategy $\sigma_0 \in \Sigma_0$ in a vertex $v$ is:

$$\text{val}^{\sigma_0}(v) := \inf_{\sigma_1 \in \Sigma_1} \text{MP}_0(\text{outcome}^{\Gamma}(v, \sigma_0, \sigma_1)).$$

The *(optimal) value* of a vertex $v \in V$ in the MPG $\Gamma$ turns out to be:

$$\text{val}^{\Gamma}(v) = \sup_{\sigma_0 \in \Sigma_0} \text{val}^{\sigma_0}(v) = \inf_{\sigma_1 \in \Sigma_1} \text{val}^{\sigma_1}(v).$$
Mean Payoff Games: main facts

Positional Strategy

In a **positional** strategy $\sigma_p$ for Player $p$, she decides once and for all the one arc to use for exiting node $v$, for every $v \in V_p$. She forgets about all her other options and lets the opponent play solitaire in the resulting arena $G^\Gamma_{\sigma_p}$.

![Diagram of a graph with nodes A, B, C, D, E, F, G and arrows with labels +3, 0, -5, -3, +3, and -5, representing the payoffs of the game.]

Determinacy Theorem (Ehrenfeucht, Mycielski 1979)

For every node $v \in V_{\text{Max}} \cup V_{\text{Min}}$, there exists a unique **value** $\text{val}^\Gamma(v) \in \mathbb{Q}$ such that Player **Max** has a positional strategy which secures game payoff at least $\text{val}^\Gamma(s)$ in any match starting from $s$, whereas Player **Min** has a positional strategy which secures game payoff at most $\text{val}^\Gamma(s)$ in any match starting from $s$. 
Mean Payoff Games: main facts

MPG Problems

- The DECISION problem asks, given a node $v \in V$, to decide whether or not $\text{val}^{\Gamma}(v) \geq 0$. The sets $W_{\text{Max}} = \{v \in V \mid \text{val}^{\Gamma}(v) \geq 0\}$ and $W_{\text{Min}} = V \setminus W_{\text{Max}}$ are called Winning Regions.
  - The DECISION problem lies in $\text{NP} \cap \text{coNP}$ (Zwick, Paterson 1996). It is recognizable with Unambiguous Polynomial-Time Non-Deterministic Turing Machines (Jurdziński 1998), i.e., it is in $\text{UP} \cap \text{coUP}$.

- The VALUE problem asks to compute the values of all nodes.

- The STRATEGY SYNTHESIS problem asks to compute, for each player, a positional strategy securing her payoffs corresponding to the values of the starting node.
Mean Payoff Games: main facts

Pseudopolynomial-time algorithms for MPGs:

1996, Zwick and Paterson gave a $O(|V|^3|E|W)$ time algorithm for the Value Problem and a $O(|V|^4|E|W \log(|E|/|V|))$ time algorithm for Optimal Strategy Synthesis.

2011, Brim, Raskin, et al. gave a $O(|V|^2|E|W \log(|V|W))$ time algorithm both for the Optimal Strategy Synthesis and for the Value Problem.
Energy Games: the model

- In EGs, given an initial credit $c \in \mathbb{N}$ and a node $v \in V$, Player $\text{Max}$ wins the game starting from $v$ if she can maintain the sum of the encountered weights always non-negative, i.e., if it holds on the play $\{v_i\}_{i=0}^{\infty}$ that:

$$c + \sum_{i=0}^{j} w(v_i, v_{i+1}) \geq 0, \text{ for all } j \geq 0;$$

otherwise, the winner is Player $\text{Min}$.

- The decision problem for EGs asks whether there exists an initial credit $c^*$ for which Player $\text{Max}$ wins from a given starting position node $v$. The corresponding winning regions are also denoted $\mathcal{W}_{\text{Max}}$ and $\mathcal{W}_{\text{Min}}$.

- EGs are log-space equivalent to MPGs [Patricia Bouyer (2008)].

- EGs can be solved in $O(|V||E|W)$ time with the Value-Iteration Algorithm [Brim, Raskin, et. al, 2011]
The Consistency-Checking algorithm

Feasible Scheduling Computation
- If every node in the corresponding game $G_H$ is winning for Player 0:
  - we compute a positional winning strategy $\pi_0$ for $G_H$;
  - the graph $G_{\pi_0}$ induced by $\pi_0$ is conservative, hence, we compute a potential function $p : G_{\pi_0} \rightarrow \mathbb{R}$;
  - $p$ is a feasible scheduling of $H$.

Negative Cycle Computation
- If there exist nodes in $G_H$ which are winning for Player 1:
  - let $G[W_1]$ be the subgraph of $G_H$ obtained by removing all nodes which are winning for Player 0;
  - on $G[W_1]$ we compute a positional winning strategy $\pi_1$ for Player 1;
  - let $\overline{W_1} = W_1 \cap V_1$ and consider the family of hyperarcs $C = \{\pi_1(v)\}_{v \in \overline{W_1}}$;
  - $(\overline{W_1}, C)$ is a negative cycle of $H$.
The Consistency-Checking algorithm

Feasible Scheduling Computation

- If every node in the corresponding game $G_H$ is winning for Player 0:
  - we compute a positional winning strategy $\pi_0$ for $G_H$;
  - the graph $G_{\pi_0}$ induced by $\pi_0$ is conservative, hence, we compute a potential function $p : G_{\pi_0} \rightarrow \mathbb{R}$;
  - $p$ is a feasible scheduling of $H$.

Negative Cycle Computation

- If there exist nodes in $G_H$ which are winning for Player 1:
  - let $G[W_1]$ be the subgraph of $G_H$ obtained by removing all nodes which are winning for Player 0;
  - on $G[W_1]$ we compute a positional winning strategy $\pi_1$ for Player 1;
  - let $W_1 = W_1 \cap V_1$ and consider the family of hyperarcs $C = \{\pi_1(v)\}_{v \in \overline{W_1}}$;
  - $(\overline{W_1}, C)$ is a negative cycle of $H$.
Theorem (Comin, Posenato, Rizzi, TIME2014 and Constraints, 2016)

The following propositions hold on HyTNs.

1. There exists an \( O((|V| + |A|)mAW) \) pseudo-polynomial time algorithm for checking HyTN-Consistency;

2. There exists an \( O((|V| + |A|)mAW) \) pseudo-polynomial time algorithm such that, given in input any consistent HyTN \( H = (V, A) \), it returns as output a feasible scheduling \( \phi : V \rightarrow \mathbb{R} \) of \( H \);

Here, \( W \triangleq \max_{A \in A, v \in H_A} |w_A(v)| \).

The approach was shown to be robust by experimental evaluations, e.g. (randomly generated) HyTNs of size up to \(|V| \sim 10^6\) and \( W \sim 10^3\) were solved.
A HyTN which requires $\Theta(W)$ computation time.
Chapter 3

*Conditional Hyper Temporal Networks and Dynamic Consistency Checking*
The CSTN formalism extends STNs in that:

1. some of the nodes are called **observation events** and to each of them is associated a boolean variable, to be disclosed only at execution time;
2. **labels** (i.e. conjunctions over the literals) are attached to all nodes and constraints, to indicate the situations in which each of them is required.
Scenario

A **scenario** \( s \) over a set \( P \) of boolean variables is a truth assignment:

\[
s : P \rightarrow \{\top, \bot\}.
\]

An example CSTN \( \Gamma = \langle V, A, L, O, OV, P \rangle \):

![Diagram](https://via.placeholder.com/150)
A **scenario** $s$ over a set $P$ of boolean variables is a truth assignment:

$$s : P \rightarrow \{\top, \bot\}.$$  

Let $s(p) = \top$ and $s(q) = \bot$...

---

Complexity in Infinite Games on Graphs and Temporal Constraint Networks

Carlo Comin
A scenario $s$ over a set $P$ of boolean variables is a truth assignment:

$$s : P \rightarrow \{\top, \bot\}.$$ 

Let $s(p) = \top$ and $s(q) = \bot$...

Complexity in Infinite Games on Graphs and Temporal Constraint Networks

Carlo Comin
A scenario \( s \) over a set \( P \) of boolean variables is a truth assignment:

\[
s : P \rightarrow \{\top, \bot\}.
\]

Let \( s(p) = \top \) and \( s(q) = \bot \), then \( \Gamma \) becomes:

![Graph diagram showing conditional simple temporal network with nodes A, B, C, O_p, O_q and edges labeled with integers representing delays or constraints.]
CSTNs: Consistency

CSTN’s Consistencies

Three notions of consistency arise for CSTNs: *weak*, *strong*, and...

dynamic consistency.

Complexity in Infinite Games on Graphs and Temporal Constraint Networks

Carlo Comin
CSTNs: Consistency

CSTN’s Dynamic Consistency

**Dynamic Consistency (DC)** requires the existence of conditional plans where decisions about precise timing of actions are postponed until exec. time, but it guarantees that all the relevant constraints will be ultimately satisfied.

Complexity in Infinite Games on Graphs and Temporal Constraint Networks
Carlo Comin
CSTNs: Dynamic Consistency

Difference Set $\Delta(s_1; s_2)$

$$\Delta(s_1; s_2) \triangleq \left\{ O_p \in V_{s_1}^+ \cap O V \mid s_1(p) \neq s_2(p) \right\}.$$ 

Dynamic Execution Strategy

Let $\sigma \in S_\Gamma$ be an execution strategy. Then, $\sigma$ is **dynamic** if and only if the following implication holds for every $s_1, s_2 \in \Sigma_P$, $u \in V_{s_1, s_2}^+$:

$$\bigg( \bigwedge_{v \in \Delta(s_1; s_2)} [\sigma(s_1)]_u \leq [\sigma(s_1)]_v \bigg) \Rightarrow [\sigma(s_1)]_u = [\sigma(s_2)]_u$$
An **Execution Strategy** for $\Gamma$ is a mapping $\sigma : \Sigma_P \rightarrow \Phi_V$ such that, for any scenario $s \in \Sigma_P$, the domain of the scheduling $\sigma(s) \in \Phi_V$ is $V_s^+$. 

\[
\begin{align*}
\phi(A) &= 0 \\
\phi(O_p) &= 1 \\
\phi(O_q) &= 2 \\
\phi(B) &= 8 \\
\phi(C) &= 10 \\
\end{align*}
\]

- $\phi(O_p) = 1$ 
- $\phi(O_q) = 2$ 
- $\phi(B) = 8$ 
- $\phi(C) = 10$ 

- $s(p) = T$ 
- $s(p) = \bot$ 
- $s(q) = T$ 
- $s(q) = \bot$ 
- $s(q) = T, \bot$
Main Results at TIME2015

Our Contribution:
- Lower-Bound: coNP-hard.
- Upper-Bound: NE \cap coNE \cap pseudo-E.

A (pseudo) Singly-Exponential Time DC-Checking for CSTNs.
- There exists an
  \[ O(|\Sigma_P|^3 |A|^2 |V| + |\Sigma_P|^4 |A||V|^2 |P| + |\Sigma_P|^5 |V|^3 |P|)W \]
  (pseudo) singly-exponential time algorithm for checking DC on any input CSTN \( \Gamma = \langle V, A, L, O, OV, P \rangle \).
  - Here, \( W \triangleq \max_{a \in A} w_a \) and \( |\Sigma_P| \leq 2^{|P|} \).
- In particular, given any dynamically-consistent CSTN \( \Gamma \), the algorithm returns a viable and dynamic execution strategy.

(here above, E is deterministic singly-exponential time, NE is nondeterministic)
Most importantly, we unveil a connection between the problem of checking DC in CSTNs and that of determining Mean Payoff Games (and Energy Games).
In order to analyze the algorithm, we introduce a novel and refined notion of dynamic-consistency, named $\varepsilon$-dynamic-consistency;

- Consider $\hat{\varepsilon}(\Gamma) = \sup\{\varepsilon \in \mathbb{R}_>0 \mid \Gamma \text{ is } \varepsilon \text{-dynamically-consistent}\}$.
  - $\hat{\varepsilon}(\Gamma)$ is the reaction time of $\Gamma$.

- We provide a sharp lower bounding analysis of the critical value of the reaction time $\hat{\varepsilon}(\Gamma)$ where the CSTN $\Gamma$ transits from being, to not being, dynamically-consistent.

- This clarifies the role of the reaction time $\hat{\varepsilon}$ in the DC-checking of CSTNs.
Sketch of the reduction from CSTN-Dynamic-Consistency to HyTN-Consistency

Sketch of the reduction

- Introduce $\epsilon$-dynamic-consistency.
- Prove that any execution strategy $\sigma$ is dynamic iff $\sigma$ is $\epsilon$-dynamic for some real number $\epsilon \in (0, +\infty)$.
- Consider $\hat{\epsilon}(\Gamma) = \sup\{\epsilon \in \mathbb{R}_{>0} \mid \Gamma$ is $\epsilon$-dynamically-consistent\}.
  - $\hat{\epsilon}(\Gamma)$ is the reaction time of $\Gamma$.
- Prove that for any dynamically-consistent CSTN $\Gamma$, where $V$ is the set of events and $\Sigma_P$ is the set of scenarios, it holds $\hat{\epsilon}(\Gamma) \geq |\Sigma_P|^{-1}|V|^{-1}$.
- Devise an algorithm for checking $\epsilon$-dynamic-consistency by reducing that problem to the consistency checking of HyTNs.
- Also, we proved that the bound $\hat{\epsilon}(\Gamma) \geq |\Sigma_P|^{-1}|V|^{-1}$ is optimal.
\(\epsilon\)-dynamic-consistency, for some small real \(\epsilon > 0\)

**\(\epsilon\)-dynamic-consistency**

Given any CSTN \(\langle V, A, L, \mathcal{O}, \mathcal{OV}, P \rangle\) and any real number \(\epsilon \in (0, +\infty)\), an execution strategy \(\sigma \in S_\Gamma\) is **\(\epsilon\)-dynamic** if it satisfies all the \(H_\epsilon\)-constraints, namely, for any two scenarios \(s_1, s_2 \in \Sigma_P\) and any event \(u \in V^+_{s_1, s_2}\), the execution strategy \(\sigma\) satisfies the following constraint, denoted \(H_\epsilon(s_1; s_2; u)\):

\[
[\sigma(s_1)]_u \geq \min \left( \{[\sigma(s_2)]_u\} \cup \{[\sigma(s_1)]_v + \epsilon \mid v \in \Delta(s_1; s_2)\} \right)
\]

A CSTN \(\Gamma\) is **\(\epsilon\)-DC** if it admits \(\sigma \in S_\Gamma\) which is both viable and \(\epsilon\)-dynamic.
Lemma 1

If $\Gamma$ is $\epsilon$-dynamically-consistent, for some $\epsilon > 0$, then $\Gamma$ is $\epsilon'$-dynamically-consistent for every $\epsilon' \in (0, \epsilon]$. 
Lemma 2

Let $\sigma$ be a dynamic execution strategy for the CSTN $\Gamma$. Then, there exists a sufficiently small real number $\epsilon \in (0, +\infty)$ such that $\sigma$ is $\epsilon$-dynamic.
Lemma 3

Let \( \sigma \) be an \( \epsilon \)-dynamic execution strategy for the CSTN \( \Gamma \), for some \( \epsilon \in (0, +\infty) \). Then, \( \sigma \) is dynamic.

\[ \Rightarrow \text{Dynamic-Consistency of CSTNs is expressible with Max-Plus (or Min-Plus) constraints.} \]
Solving $\epsilon$-dynamic-consistency: Expansion of a CSTN

![Diagram](image)

Complexity in Infinite Games on Graphs and Temporal Constraint Networks

Carlo Comin
Solving $\epsilon$-dynamic-consistency: Expansion of a CSTN

An excerpt of the expansion of the CSTN $\Gamma$ with two scenarios $s_1$ and $s_4$. 
Complexity in Infinite Games on Graphs and Temporal Constraint Networks

Carlo Comin
Complexity in Infinite Games on Graphs and Temporal Constraint Networks

Carlo Comin
Complexity in Infinite Games on Graphs and Temporal Constraint Networks

Carlo Comin
Complexity in Infinite Games on Graphs and Temporal Constraint Networks

Carlo Comin
Complexity in Infinite Games on Graphs and Temporal Constraint Networks

Carlo Comin
Complexity in Infinite Games on Graphs and Temporal Constraint Networks

Carlo Comin
Theorem (CSTNs and HyTNs)

Given any CSTN $\Gamma = \langle V, A, L, O, OV, P \rangle$, there exists a sufficiently small real number $\epsilon \in (0, +\infty)$ such that $\Gamma$ is dynamically-consistent if and only if $\mathcal{H}_\epsilon(\Gamma)$ is consistent.

Moreover, $\mathcal{H}_\epsilon(\Gamma)$ has at most $|V_{\mathcal{H}_\epsilon}| \leq |\Sigma_P| |V|$ nodes, $|A_{\mathcal{H}_\epsilon}| = O(|\Sigma_P| |A| + |\Sigma_P|^2 |V|)$ hyperarcs, and it has size at most $m_{A_{\mathcal{H}_\epsilon}} = O(|\Sigma_P| |A| + |\Sigma_P|^2 |V| |P|)$. 
The bound $\hat{\epsilon}(\Gamma) \geq |\Sigma_P|^{-1} |V|^{-1}$ is sharp.

- A natural question is whether the lower bound $\hat{\epsilon}(\Gamma) \geq |\Sigma_P|^{-1} |V|^{-1}$ can be improved up to $\hat{\epsilon}(\Gamma) = \Omega(|V|^{-1})$...
- ... this would improve the time complexity by a factor $|\Sigma_P|$.
- However, the following theorem shows that this is not the case by exhibiting a CSTN for which $\hat{\epsilon}(\Gamma) = 2^{-\Omega(|P|)}$.
- This proves that the bound $\hat{\epsilon}(\Gamma) \geq |\Sigma_P|^{-1} |V|^{-1}$ is (almost) sharp.

**Theorem**

*For each $n \in \mathbb{N}_0$ there exists a CSTN $\Gamma^n$ such that*

$$\hat{\epsilon}(\Gamma^n) < 2^{-n+1} = 2^{-|P^n|/3+1},$$

*where $P^n$ is the set of boolean variables of $\Gamma^n$.***
The bound $\hat{\epsilon}(\Gamma) \geq |\Sigma_P|^{-1} |V|^{-1}$ is sharp.

- A natural question is whether the lower bound $\hat{\epsilon}(\Gamma) \geq |\Sigma_P|^{-1} |V|^{-1}$ can be improved up to $\hat{\epsilon}(\Gamma) = \Omega(|V|^{-1})$...
- ... this would improve the time complexity by a factor $|\Sigma_P|$.
- However, the following theorem shows that this is not the case by exhibiting a CSTN for which $\hat{\epsilon}(\Gamma) = 2^{-\Omega(|P|)}$.
- This proves that the bound $\hat{\epsilon}(\Gamma) \geq |\Sigma_P|^{-1} |V|^{-1}$ is (almost) sharp.

**Theorem**

For each $n \in \mathbb{N}_0$ there exists a CSTN $\Gamma^n$ such that

$$\hat{\epsilon}(\Gamma^n) < 2^{-n+1} = 2^{-|P^n|/3+1},$$

where $P^n$ is the set of boolean variables of $\Gamma^n$. 
How $\Gamma^n$ looks like

![Graph Diagram]

Complexity in Infinite Games on Graphs and Temporal Constraint Networks

Carlo Comin
How $\Gamma^n$ looks like

Complexity in Infinite Games on Graphs and Temporal Constraint Networks
Carlo Comin
How $\Gamma^n$ looks like

Complexity in Infinite Games on Graphs and Temporal Constraint Networks

Carlo Comin
Chapter 4

Instantaneous-Reaction Time in DC-Checking of CSTNs
The $\varepsilon$-DC notion is interesting per se, and the $\varepsilon$-DC-Checking algorithm (TIME2015) rests on the assumption that $\varepsilon > 0$; i.e., leaving unsolved the question of what happens when $\varepsilon = 0$.

Then, we introduced and studied $\pi$-DC, a sound notion of DC with an instantaneous reaction-time. i.e., one in which the planner can react to any observation \textit{at the same instant of time} in which the observation is made.
Consider the following CSTN $\Gamma_{\square}$.

**Proposition**

The CSTN $\Gamma_{\square}$ is 0-DC.
Instantaneous-Reaction Time vs 0-DC (TIME2016)

Complexity in Infinite Games on Graphs and Temporal Constraint Networks

Carlo Comin
We need to take explicitly into account an *additional ordering* between the observation events scheduled *at the same execution time*.

**π-Execution-Strategy**

An *ordered-Execution-Strategy* (*π-ES*) for $\Gamma$ is a mapping:

$$\sigma : s \mapsto ([\sigma(s)]^t, [\sigma(s)]^\pi),$$

$s \in \Sigma_P$, $[\sigma(s)]^t \in \Phi_V$ and $[\sigma(s)]^\pi : OV^+_s \mapsto \{1, \ldots, |OV^+_s|\}$ is bijective.

**Remark**

We require positions to be *coherent* w.r.t. execution times, i.e.,

$$\forall (O_p, O_q \in OV^+_s) \text{ if } [\sigma(s)]^{t}_{O_p} < [\sigma(s)]^{t}_{O_q} \text{ then } [\sigma(s)]^{\pi}_{O_p} < [\sigma(s)]^{\pi}_{O_q}.$$
\section*{\textit{\pi}-DC}

\subsection*{\textit{\pi}-History}

Let $\sigma \in S_{\Gamma}$, $s \in \Sigma_P$, and let $\tau \in \mathbb{R}$ and $\psi \in \{1, \ldots, |V|\}$.

The \textit{ordered-history} $\pi\text{-Hst}(\tau, \psi, s, \sigma)$ of $\tau$ and $\psi$ in the scenario $s$, under the $\pi$-ES $\sigma$, is defined as: $\pi\text{-Hst}(\tau, \psi, s, \sigma) = \{ (p, s(p)) \in P \times \{0, 1\} \mid O_p \in OV_{s}^{+}, [\sigma(s)]_{O_p}^{t} \leq \tau, [\sigma(s)]_{O_p}^{\pi} < \psi \}$. 

\subsection*{\textit{\pi}-Dynamic-Consistency}

Any $\sigma \in S_{\Gamma}$ is called $\pi$-\textit{dynamic} when, for any two scenarios $s_1, s_2 \in \Sigma_P$ and any event $u \in V_{s_1, s_2}^{+}$, if $\tau \triangleq [\sigma(s_1)]_{u}^{t}$ and $\psi \triangleq [\sigma(s_1)]_{u}^{\pi}$, then:

$$\pi\text{-Hst}(\tau, \psi, s_1, \sigma) \wedge s_2 \Rightarrow [\sigma(s_2)]_{u}^{t} = \tau, [\sigma(s_2)]_{u}^{\pi} = \psi.$$ 

We say that $\Gamma$ is $\pi$-\textit{dynamically-consistent} ($\pi$-DC) if it admits $\sigma \in S_{\Gamma}$ which is both viable and $\pi$-dynamic. The problem of checking whether a given CSTN is $\pi$-DC is named $\pi$-DC-Checking.
Remark

Due to the strict inequality \( \left\lbrack \sigma(s) \right\rbrack_{\mathcal{O}_p}^{\pi} \prec \psi \) in the definition of \( \pi\text{-Hst}(\cdot) \), in a \( \pi \)-dynamic \( \pi \)-ES, there must be exactly one \( \mathcal{O}_{p'} \in \mathcal{O}V \), for some \( p' \in P \), which is executed at first (w.r.t. both execution time and position) under all possible scenarios \( s \in \Sigma_P \) (i.e., a root).

An example of a permutation-scenario-tree (ps-tree) over \( P = \{a, b, c, d\} \).
In summary, the following chain of implications hold on the various DCs:

\[
(\varepsilon\text{-DC, } \varepsilon > 0) \iff \text{DC} \not\iff \pi\text{-DC} \not\iff (\varepsilon\text{-DC, } \varepsilon = 0)
\]
Main Result

Main Algorithmic Result [CCR, TIME2016]

The time-complexity of \( \pi \)-DC-Checking problem remains \((pseudo)\) \textit{Singly-Exponential} in the number of propositional letters \( P \) of the input CSTN.
Chapter 5

Linear Time Algorithm for Update Games
via Strongly-Trap-Connected Components
Recall an arena is a finite directed graph whose vertices are divided into two classes, say $V = V_{\square} \cup V_{\circ}$. A play is thus an infinite path $\pi = v_0v_1v_2 \ldots \in V^\omega$ such that $(v_i, v_{i+1}) \in A$ for every $i \in \mathbb{N}$.

Given two strategies $\sigma_{\square} \in \Sigma^A_{\square}$ and $\sigma_{\circ} \in \Sigma^A_{\circ}$, and some $v_s \in V$, the outcome play,

$$\rho_A(v_s, \sigma_{\square}, \sigma_{\circ})$$

is the (unique) play that starts at $v_s$ and is consistent with both $\sigma_{\square}$, $\sigma_{\circ}$. 
Update Games: the model

Update Games [Dinneen, Khoussainov, 1999]

- Let $\text{Inf}(\pi)$ be the set of all and only those vertices $v \in V$ that appear infinitely often in $\pi$; namely,

  $$\text{Inf}(\pi) \triangleq \{ v \in V \mid \forall j \in \mathbb{N} \exists k \in \mathbb{N} \text{ such that } k > j \text{ and } v_k = v \}.$$ 

- Player $\Box$ wins the UG $\mathcal{A}$ iff $\exists (\sigma_\Box \in \Sigma_\Box^A) \forall (\sigma_\bigcirc \in \Sigma_\bigcirc^A)$:

  $$\forall v_s \in V \ \ \text{Inf}(\rho_A(v_s, \sigma_\Box, \sigma_\bigcirc)) = V.$$ 

- An $O(|V||A|)$ time algorithm was given in [Dinneen, Khoussainov, 1999].
An Arena $\mathcal{A}$. The rev-palm-tree generated by rev-DFS, with indices of vertices. (c) The order of arcs’ exploration.

Figure: An arena (a), and a rev-palm-tree (b), generated by rev-DFS (c).
Alphabet of a play

For any finite (or infinite) path \( p \in V^* \) (or \( p \in V^\omega \)), the alphabet of \( p \) is

\[
\Xi(p) \triangleq \{ v \in V \mid v \text{ appears in } p \}.
\]

Trap-Reachability

Given an arena \( \mathcal{A} \) on vertex set \( V \), let \( U \subseteq V \) and \( u, v \in U \). We say that \( v \) is \( U \)-trap-reachable from \( u \) when there exists \( \sigma \Box \in \Sigma_{\mathcal{A}} \Box \) (i.e., \( \sigma \Box = \sigma \Box(u, v) \)) such that for every \( \sigma \Diamond \in \Sigma_{\mathcal{A}} \Diamond \):

- [reachability] \( v \in \Xi(\rho_{\mathcal{A}}(u, \sigma \Box, \sigma \Diamond)) \); and,
- [entrapment] \( \Xi(\rho_{\mathcal{A}}(u, \sigma \Box, \sigma \Diamond, v)) \subseteq U \).
(a) An arena $\mathcal{A}$.  

(b) The tr-palm-tree generated by tr-DFS rooted at $A$, with indices of vertices and labelled arcs.

(c) The order of arcs’ exploration.

Figure: An arena (a), and a tr-palm-tree (b), generated by tr-DFS (c).
Figure: An arena (a), and a tr-palm-tree (b), generated by tr-DFS (c).
TR-Depth-First-Search

(a) An arena $\mathcal{A}$.

(b) The tr-palm-tree generated by tr-DFS rooted at $A$, with indices of vertices and labelled arcs.

(c) The order of arcs’ exploration.

Figure: An arena (a), and a tr-palm-tree (b), generated by tr-DFS (c).
Figure: An arena (a), and a tr-palm-tree (b), generated by tr-DFS (c).
TR-Depth-First-Search

(a) An arena $\mathcal{A}$.

(b) The tr-palm-tree generated by tr-DFS rooted at $A$, with indices of vertices and labelled arcs.

(c) The order of arcs’ exploration.

Figure: An arena (a), and a tr-palm-tree (b), generated by tr-DFS (c).
TR-Depth-First-Search

(a) An arena $\mathcal{A}$.

(b) The tr-palm-tree generated by tr-DFS rooted at $\mathcal{A}$, with indices of vertices and labelled arcs.

1. $(B, A)$
2. $(D, B)$
3. $(E, D)$
4. $(C, E)$
5. $(F, E)$
6. $(G, D)$
7. $(A, G)$
8. $(F, G)$
9. $(C, B)$
10. $(H, A)$
11. $(C, H)$
12. $(F, H)$

(c) The order of arcs’ exploration.

Figure: An arena (a), and a tr-palm-tree (b), generated by tr-DFS (c).
TR-Depth-First-Search

(a) An arena $\mathcal{A}$.

(b) The tr-palm-tree generated by tr-DFS rooted at $A$, with indices of vertices and labelled arcs.

1. $(B, A)$
2. $(D, B)$
3. $(E, D)$
4. $(C, E)$
5. $(F, E)$
6. $(G, D)$
7. $(A, G)$
8. $(F, G)$
9. $(C, B)$
10. $(H, A)$
11. $(C, H)$
12. $(F, H)$

(c) The order of arcs’ exploration.

Figure: An arena (a), and a tr-palm-tree (b), generated by tr-DFS (c).
TR-Depth-First-Search

(a) An arena $A$.

(b) The tr-palm-tree generated by tr-DFS rooted at $A$, with indices of vertices and labelled arcs.

(c) The order of arcs’ exploration.

Figure: An arena (a), and a tr-palm-tree (b), generated by tr-DFS (c).
TR-Depth-First-Search

(a) An arena $\mathcal{A}$. (b) The tr-palm-tree generated by tr-DFS rooted at $A$, with indices of vertices and labelled arcs.

(c) The order of arcs’ exploration.

Figure: An arena (a), and a tr-palm-tree (b), generated by tr-DFS (c).
Figure: An arena (a), and a tr-palm-tree (b), generated by tr-DFS (c).
TR-Depth-First-Search

(a) An arena $\mathcal{A}$.

(b) The tr-palm-tree generated by tr-DFS rooted at $A$, with indices of vertices and labelled arcs.

(c) The order of arcs’ exploration.

Figure: An arena (a), and a tr-palm-tree (b), generated by tr-DFS (c).
(a) An arena \( A \).

(b) The tr-palm-tree generated by tr-DFS rooted at \( A \), with indices of vertices and labelled arcs.

(c) The order of arcs’ exploration.

Figure: An arena (a), and a tr-palm-tree (b), generated by tr-DFS (c).
TR-Depth-First-Search

(a) An arena $\mathcal{A}$.

(b) The tr-palm-tree generated by tr-DFS rooted at $A$, with indices of vertices and labelled arcs.

(c) The order of arcs’ exploration.

Figure: An arena (a), and a tr-palm-tree (b), generated by tr-DFS (c).
### Complexity in Infinite Games on Graphs and Temporal Constraint Networks

Carlo Comin 79/89
Strongly-Trap-Connectedness

We say that $U \subseteq V$ is strongly-trap-connected when for every $(u, v) \in U \times U$ there exists some $\sigma \in \Sigma^A$ (i.e., $\sigma = \sigma(u, v)$) such that $\sigma : u \sim v$.

Strongly-Trap-Connected Components

The binary relation $\sim_{stc}$ on $V$ is defined as follows:

$$\sim_{stc} \triangleq \left\{ (u, v) \in V \times V \mid \exists U \subseteq V \text{ s.t. } U \text{ is STC and } \{u, v\} \subseteq U \right\}.$$

It holds that $\sim_{stc}$ is an equivalence relation on $V$. Equivalence classes of $\sim_{stc}$ are strongly-trap-connected components of $\mathcal{A}$. 
Main Results on UGs

**UGs and STC**

There is a *linear time* algorithm for decomposing any given arena into its strongly-trap-connected components.

**UGs and STC**

An UG $\mathcal{A}$ is winning for Player $\square$ if and only if $V$ is strongly-trap-connected.

**Deciding UGs in Linear Time**

Deciding whether or not a given UG $\mathcal{A}$ is winning for Player $\square$ takes $\Theta(|V| + |A|)$ time.
Application to Explicit McNaughton-Muller Games

Explicit McNaughton-Muller Games

The most straightforward way to represent a Müller winning condition $\mathcal{F} \subseteq 2^V$ is to provide an explicit list of subsets of vertices:

$$\mathcal{F} = \{ \mathcal{F}_i \subseteq V \mid 1 \leq i \leq \ell \}, \text{ for some } \ell \in \mathbb{N}.$$

A play $\rho \in V^\omega$ is winning for Player $\Box$ if and only if $\text{Inf}(\rho) \in \mathcal{F}$.

Polynomial Time Algorithm [Horn 2008]

Explicit MMGs can be solved in polynomial time, where an $O(|\mathcal{F}| \cdot (|\mathcal{A}| + |\mathcal{F}|)^2)$ time algorithm is given in [Horn 2008].

Quadratic Time Algorithm for Explicit MMGs

We can solve Explicit MMGs in $O(|\mathcal{F}| \cdot (|\mathcal{A}| + |\mathcal{F}|))$ quadratic time.
Chapter 6

An Improved Upper Bound on Value Problem and Optimal Strategy Synthesis in MPGs
## Table: Complexity of the main Algorithms for solving MPGs.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Value Problem</th>
<th>Optimal Strategy</th>
<th>Synthesis</th>
</tr>
</thead>
<tbody>
<tr>
<td>This work</td>
<td>$\Theta(</td>
<td>V</td>
<td>^2</td>
</tr>
<tr>
<td>Brim11</td>
<td>$O(</td>
<td>V</td>
<td>^2</td>
</tr>
<tr>
<td>ZP96</td>
<td>$\Theta(</td>
<td>V</td>
<td>^3</td>
</tr>
<tr>
<td>LP07</td>
<td>$O(2^{</td>
<td>V</td>
<td>}</td>
</tr>
<tr>
<td>An06</td>
<td>$O\left(</td>
<td>V</td>
<td>^2</td>
</tr>
</tbody>
</table>
An $O(|V|^2|E|W)$ Improved Upper Bound for OSS in MPGs

Our Contribution: Improved Upper Bound for MPGs [CR, Algorithmica, 2016]

There exists an algorithm for solving the STRATEGY SYNTHESIS and the VALUE problem in:

$$O(|V|^2|E|W)$$ time and $O(|V|)$ space.

Particularly, the time complexity is bounded as follows:

$$\Theta\left(|V|^2|E|W + \sum_{v\in V} \deg_{\Gamma}(v) \cdot \ell^0_{\Gamma}(v)\right),$$

where $\ell^0_{\Gamma}(v) \leq (|V| - 1)|V|W$ denotes the total number of times that an energy-lifting operator $\delta(\cdot, v)$ is applied to any $v \in V$.

So, the best previously known upper bound [Brim, et al. 2011] is improved by a factor $O(\log(|V|W))$. Here, $W = \max_{e \in E} |w_e|$. 
Surprisingly, in this problem, a Linear Search turns out to be faster than a Binary Search...

- The $O(|V|^2|E| \log(|V|W))$ algorithm of Brim, et al. (2011) is based on a **binary search** procedure that reduces the **VALUE** and the **STRATEGY SYNTHESIS** problems to multiple resolutions of the **DECISION** problem.

- Can the **binary search** be **avoided**?
  - The answer is **YES**.

- Our algorithm also reduces to multiple resolutions of the **DECISION** problem, but it succeeds in **amortizing the cost** of these.
  - Actually, each **DECISION** problem generated gets solved by the computation of a **Small Energy Progress Measure** (SEPM) into the corresponding **Energy Game** (EG).
Chapter 7

A Faster $O(|V|^2|E|W)$ Time Energy Algorithm for MPGs
A Faster $O(|V|^2 |E| W)$ Algorithm for MPGs which is not $\Omega(|V|^2 |E| W)$

A more technical but “truly” $O(|V|^2 |E| W)$ algorithm for MPGs

- The running time of our first solution turns out to be also $\Omega(|V|^2 |E| W)$, the actual time complexity being

$$\Theta(|V|^2 |E| W + \sum_{v \in V} \deg_{\Gamma}(v) \cdot \ell_0^{\Gamma}(v)).$$

- We introduce a novel algorithmic scheme, named Jumping, which tackles on some further regularities of the problem, thus reducing the estimate on the pseudo-polynomial time complexity of MPGs to:

$$O(|E| \log |V|) + \Theta \left( \sum_{v \in V} \deg_{\Gamma}(v) \cdot \ell_1^{\Gamma}(v) \right).$$

- $\ell_1^{\Gamma} \leq (|V| - 1)|V| W$ (worst-case; but experimentally, $\ell_1^{\Gamma} \ll \ell_0^{\Gamma}$)
- the working space is $\Theta(|V| + |E|)$. 
Thank you

Thank you for the attention