Some Thoughts on Dynamic Unit Disk Graphs

Neven VILLANI

ENS Paris-Saclay and LaBRI, France

joint work with Arnaud CASTEIGTS

Algorithmic Aspects of Temporal Graphs IV – July 2021

Outline

Static Unit Disk Graphs

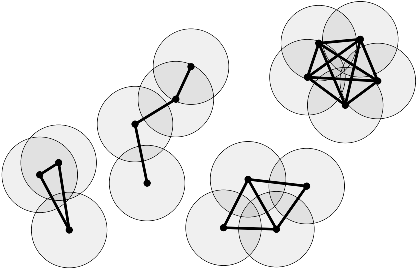
Definition (Unit Disk Graph)

$G = (V, E)$ an undirected graph is a Unit Disk Graph (UDG) in dimension n when there exists an embedding $\iota : V \to \mathbb{R}^n$ such that $\forall v, v' \in V,\ \{v, v'\} \in E \iff \|\iota(v) - \iota(v')\| \leqslant 1$

Static Unit Disk Graphs

Definition (Unit Disk Graph)

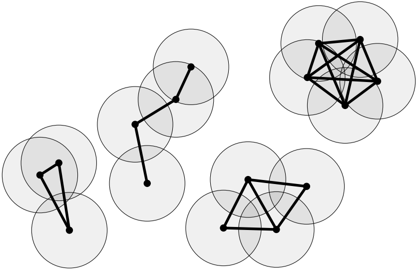
$G = (V, E)$ an undirected graph is a Unit Disk Graph (UDG) in dimension n when there exists an embedding $\iota : V \to \mathbb{R}^n$ such that $\forall v, v' \in V,\ \{v, v'\} \in E \iff \|\iota(v) - \iota(v')\| \leqslant 1$

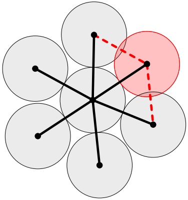


Static Unit Disk Graphs

Definition (Unit Disk Graph)

$G = (V, E)$ an undirected graph is a Unit Disk Graph (UDG) in dimension n when there exists an embedding $\iota : V \to \mathbb{R}^n$ such that $\forall v, v' \in V,\ \{v, v'\} \in E \iff \|\iota(v) - \iota(v')\| \leqslant 1$





Dynamic UDG

Definition

A dynamic UDG is $\mathcal{G} = (V, E_0, \cdots, E_\tau)$ such that all $G_i = (V, E_i)$ are UDG and successive embeddings change in limited ways.

G_i: "snapshots"
$(V, \bigcup_{0 \leqslant i \leqslant \tau} E_i)$: "footprint"

- To what extent can dynamic UDG be recognized ?
- How to define "limited ways" ?

Plausible Mobility

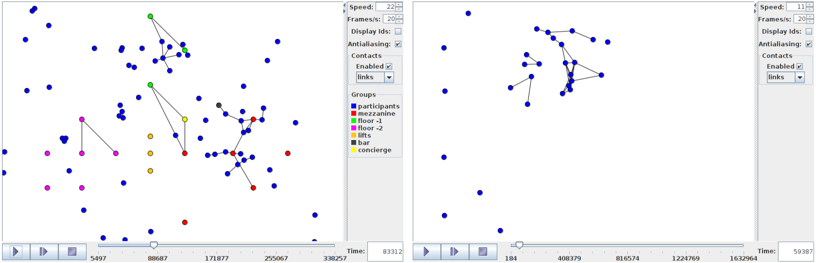


Figure: Inferring of positions from contact trace

Tolerates missing or extra links.
Reasonable assumption in the case of a low quality trace, but can we do better ?

Whitbeck, *Plausible Mobility*, `https://plausible.lip6.fr` (2011)

Results

setting	static	dynamic
unrestricted (2D)	NP-hard[(1)]	
tree (2D)	NP-hard[(2)]	
caterpillar (2D)	Linear[(2)]	
1D	Linear[(3)]	

[(1)] Breu & Kirkpatrick, *Unit disk graph recognition is NP-hard* (1998)
[(2)] Bhore & Nickel & Nöllenburg, *Recognition of Unit Disk Graphs for Caterpillars, Embedded Trees, and Outerplanar Graphs* (2021)
[(3)] Booth & Lueker, *Testing for the consecutive ones property, interval graphs, and graph planarity using PQ-tree algorithms* (1976) (And at least 3 other papers)

Results

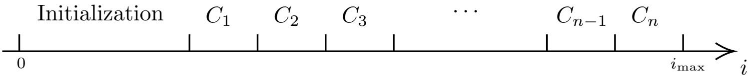
setting	static	dynamic
unrestricted (2D)	NP-hard[1]	NP-hard
tree (2D)	NP-hard[2]	NP-hard
caterpillar (2D)	Linear[2]	NP-hard[*]
1D	Linear[3]	Linear

- Seemingly no interesting tractable problem in two dimensions, simpler reduction than in the static problem
  [*] all snapshots are caterpillars
- An extension of a data structure for the 1-dimensional case can handle temporality.

[1] Breu & Kirkpatrick, *Unit disk graph recognition is NP-hard* (1998)
[2] Bhore & Nickel & Nöllenburg, *Recognition of Unit Disk Graphs for Caterpillars, Embedded Trees, and Outerplanar Graphs* (2021)
[3] Booth & Lueker, *Testing for the consecutive ones property, interval graphs, and graph planarity using PQ-tree algorithms* (1976) (And at least 3 other papers)

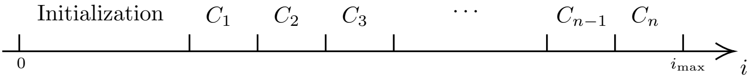
Overview and intuition

- reduction from 3-SAT
- one group of disks for each variable
- each variable can take two states, interpreted as true or false
- clauses are handled sequentially over a sequence of consecutive snapshots



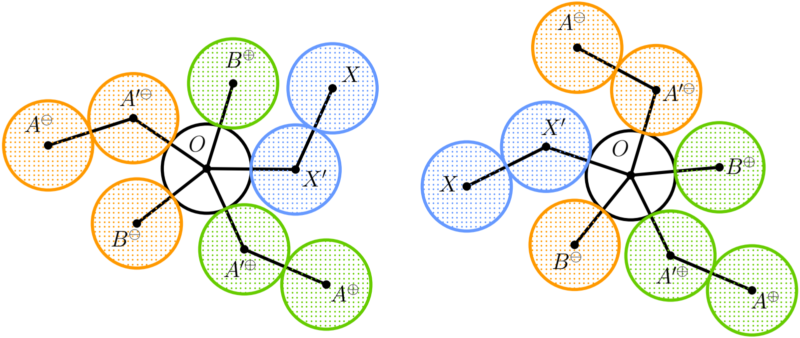
Overview and intuition

- reduction from 3-SAT
- one group of disks for each variable
- each variable can take two states, interpreted as true or false
- clauses are handled sequentially over a sequence of consecutive snapshots



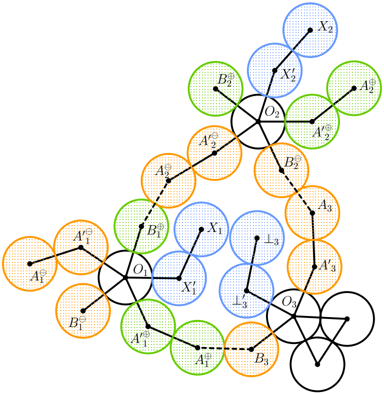
Hypothesis: “slow enough”. Speed is bounded by a constant fraction of the radius.
This makes variables unable to change state in the middle of the process.

Two configurations of variables



Left: **true**, Right: **false**

Clause assembling



The clause $C = \neg x_1 \vee x_2 \vee \perp$.
With $x_1 = x_2 =$ **true**.
Satisfied thanks to x_2.

The central 12-cycle can fit 4 disks but not 6.

Extension of the result

This shows NP-hardness in the general case.

Simpler proof than in the static case
+ linear number of disks instead of quadratic
+ fewer restrictions on initial 3-SAT instance

Extension of the result

This shows NP-hardness in the general case.

Simpler proof than in the static case
+ linear number of disks instead of quadratic
+ fewer restrictions on initial 3-SAT instance

Still NP-hard under the modified constraints (separately):

- integer coordinates
- footprint is a tree
- snapshots are caterpillars
- snapshots have CCs of size at most 2
- one event at a time

(caterpillar: tree with all vertices within distance 1 of a central path)

Extension of the result

This shows NP-hardness in the general case.

Simpler proof than in the static case
+ linear number of disks instead of quadratic
+ fewer restrictions on initial 3-SAT instance

Still NP-hard under the modified constraints (separately):

- integer coordinates (static: unknown)
- footprint is a tree (static: NP-hard)
- snapshots are caterpillars (static: linear)
- snapshots have CCs of size at most 2 (static: $O(1)$)
- one event at a time (static: irrelevant)

(caterpillar: tree with all vertices within distance 1 of a central path)

Takeaway and 1D restriction

Main source of problems: structures can be forced to "choose" one of several embeddings, which they are then unable to escape from.

In one dimension, an efficient representation of all possible configurations
$\longrightarrow$ extension of PQ-trees

Physical 1D model

- one event at a time LinkUp or LinkDown
  $\longrightarrow$ perfect trace
- continuous transition from one embedding to the next

Equivalent permutations

Theorem

For $\tau \in \mathfrak{S}(V)$, there exists an injective embedding ι of G with the same ordering of vertices iff all neighborhoods are contiguous subsequences of τ

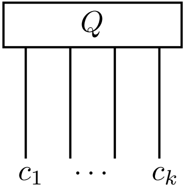
$\longrightarrow$ The set of all valid embeddings can be represented by a set of permutations.

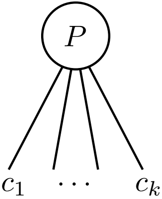
Theorem

There exists a continuous transition without event from ι to ι' iff ι and ι' differ only in the order of vertices that have the same neighborhood

$\longrightarrow$ From now on, only manipulations on sets of permutations

PQ-tree

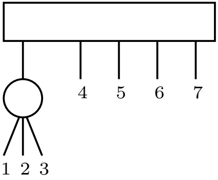




$$\{(c_1, \cdots, c_k), (c_k, \cdots, c_1)\}$$

$$\mathfrak{S}(c_1, \cdots, c_k)$$

Example:



A tree for the set

1234567, 1324567, 2134567,

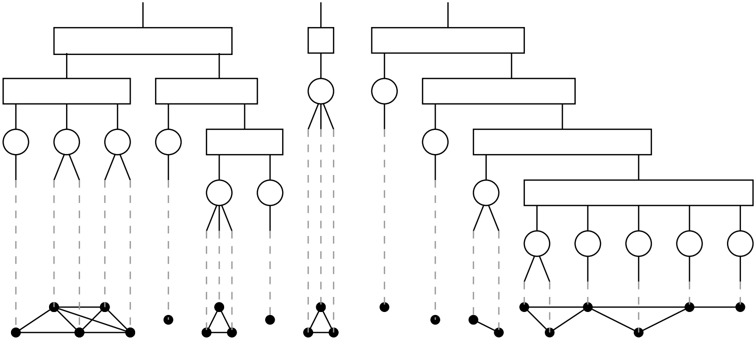
2314567, 3124567, 3214567,

7654321, 7654231, 7654312,

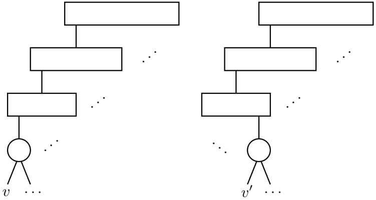
7654132, 7654213, 7654123,

PQ-forest

- set of PQ-trees
- P-nodes as leaves contain disks with the same neighborhood
- toplevel trees can be arbitrarily permuted

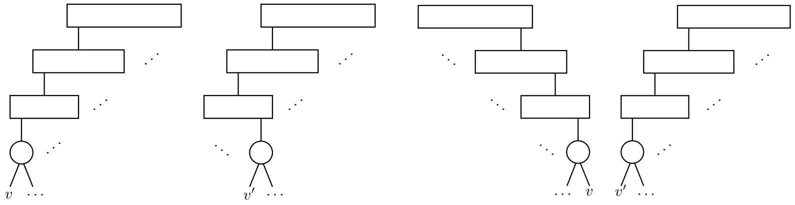


$\textsc{LinkUp}(v, v')$



Initial

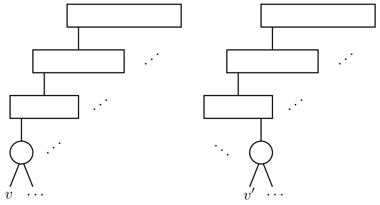
LinkUp(v, v')



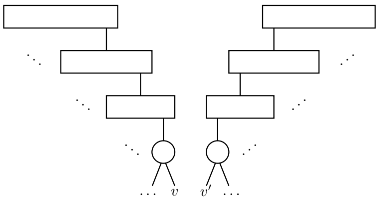
Initial

Rotate

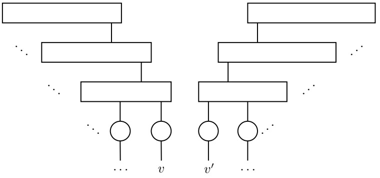
LinkUp(v, v')



Initial

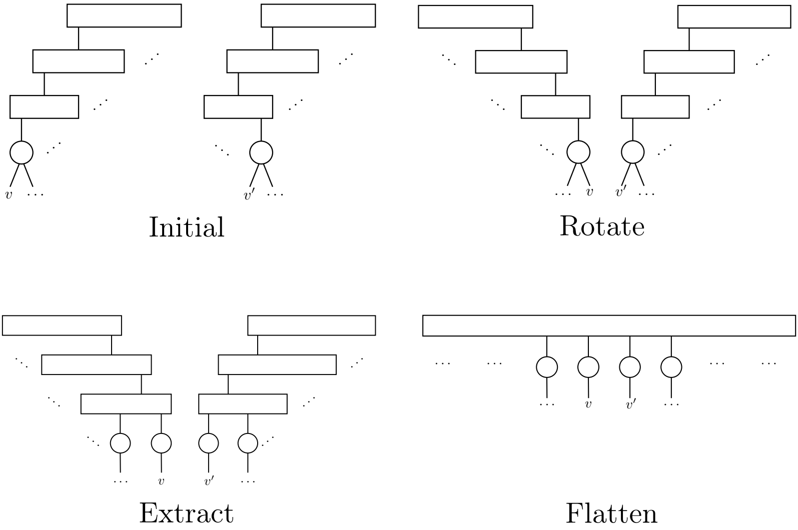


Rotate



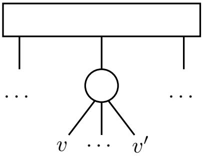
Extract

LINKUP(v, v')



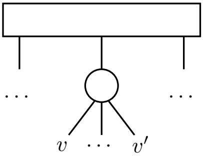
v
$\cdots$
v'
$\cdots$
Initial
$\cdots$
v
v'
$\cdots$
Rotate
$\cdots$
v
v'
$\cdots$
Extract
$\cdots$
$\cdots$
$\cdots$
v
v'
$\cdots$
$\cdots$
$\cdots$
Flatten

LINKDOWN(v, v')

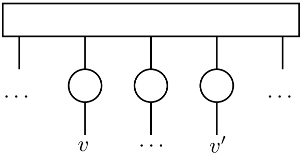


Initial

LinkDown(v, v')

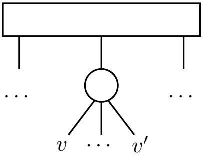


Initial

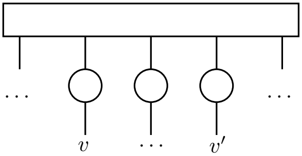


Extract

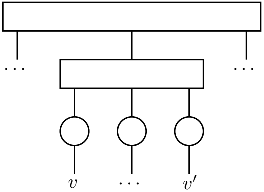
LinkDown(v, v')



Initial



Extract



Allow flip

Final result

- each new event requires amortized $O(\log n)$
  (n: number of vertices)
- linear overall: $O(\tau \cdot \log n)$
  (τ: number of events)
- online algorithm: updates the PQ-forest in real time

Open questions & future works

- characterization of forbidden 1D patterns
- exact algorithm for 2D (even if exponential) ?
- 2D when the *footprint* is a caterpillar
  (despite it being too restrictive for practical purposes)