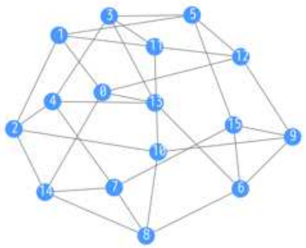
2018-11-27 GraphGolf Workshop

Graph enumeration and its applications featuring compressed data structures

Takeru Inoue (NTT Labs, Japan)

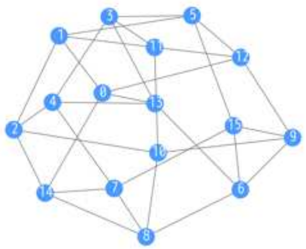
What's graph enumeration?

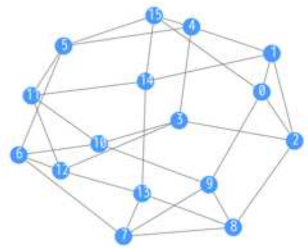
- GraphGolf: optimization problem
  - Minimize graph s.t. order/degree



A smallest graph (diameter = 3)
s.t. order n = 16, degree d = 4

- Enumeration version of GraphGolf
  - *Enumerate all* min graphs s.t. order/degree





…

Smallest graph*s* (diameter = 3)
s.t. order n = 16, degree d = 4

Why enumerate graphs?

- Fun! ☺
- Fundamental operation in discrete math
  - Basis for optimization, probability etc.
- Many applications
  - Communication networks, power distribution, GraphGolf etc.

- Is time-consuming?
  - NO, fast enough thanks to *compressed data structure*

Outline

- Graph enumeration problem
- Frontier-based Search: graph enumeration algorithm
- Real-world applications
- Graphillion

Graph enumeration problem

- Find all graphs
  - s.t. *constraints*
    - Order/degree
    - Connectivity
    - Class: path, cycle, tree, etc.

- Note: consider labeled graphs

Isomorphic but distinguished as labeled graphs

Complexity

- Often #P-complete
  - "how many" rather than "are there any"

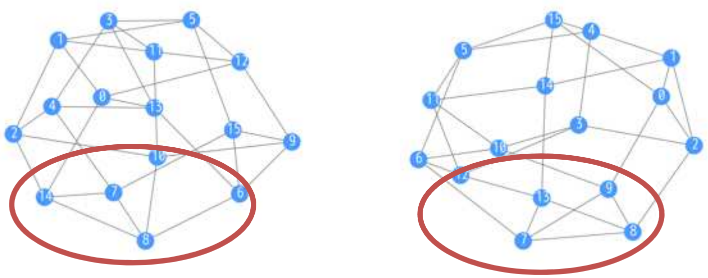
- Complexity depends on # of graphs to enumerate
  - There can be *exponential* # of graphs for n
  - e.g. ~10^{13} graphs for $n = 10$
    - $|E|$ is up to 45, which yields 2^45 graphs

- Disappointing ☹

Opportunity

- Need NOT to enumerate all graphs *explicitly*
  - OK if standard queries can be performed
    - min/max, counting etc.

- Opportunity: graphs often *share* common structures



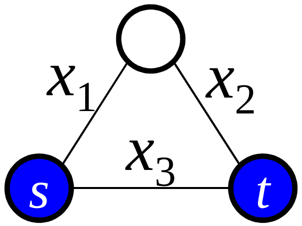
Can be merged for compression

- Redefines "enumeration" with compressed data structures

Example: Counting connected subgraphs

- Q. How many subgraphs <u>connecting *s*-*t*</u> ?

Constraint



Existence of edge i : $x_i \in \{0,1\}$

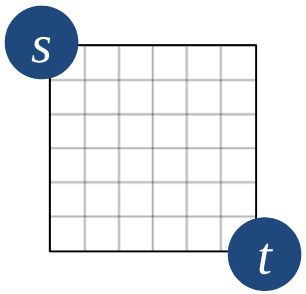
- A. 5

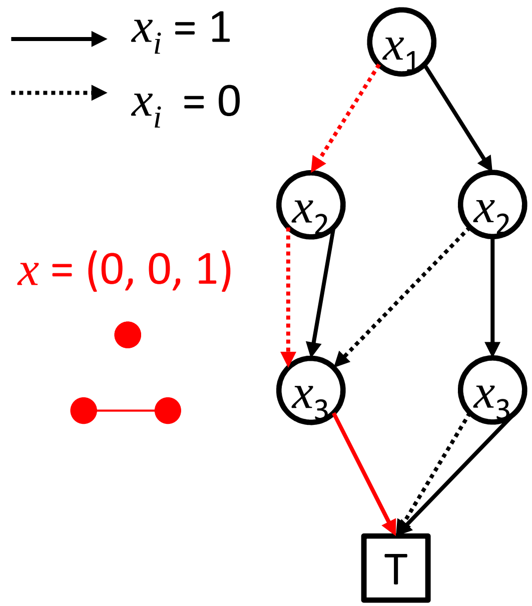
(x_1, x_2, x_3)
(1, 1, 1)
(1, 1, 0)
(1, 0, 1)
(0, 1, 1)
(0, 0, 1)

$O(2^{|E|})$

Binary decision diagrams (BDDs)

- Represents all connected subgraphs in *compressed* manner
  - Path corresponds to subgraph
  - Common substructures are shared among paths
- How much compressed?
  - e.g. 10^{25} subgraphs in BDD of 10^5 nodes





- Can we construct BDD *directly*?

Reviewing dynamic programming (DP)

- Find most valuable combo under weight constraint

Weight	2	1	1	2
Value	2	3	2	3

W't:	0	1	2	3
	0			
	0		2	
	0	3	2	5
	0	3	5	5
	0	3	5	6

→ Pack

⇢ Unpack

Counting with DP

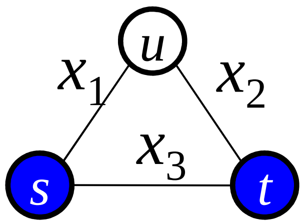
- Count combos with weight = 3

	0	1	2	3
	1			
	1		1	
	1	1	1	1
	1	2	2	2
	1	2	3	4

BDD is constructed by DP
used for min/max, counting etc.

How to apply DP to graphs?

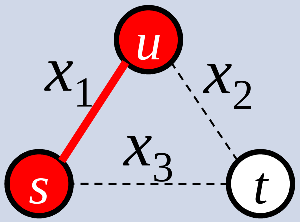
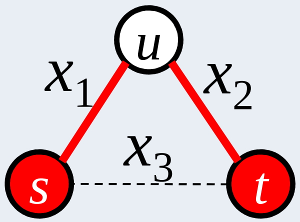
- In DP table, paths are merged if *equivalent* (e.g. equal weight)
  - How to define *equivalency* for graphs (e.g. connectivity)?



	?	?	?
x_1	1		
x_2			
x_3			
		Equivalent	

Definition: Frontier

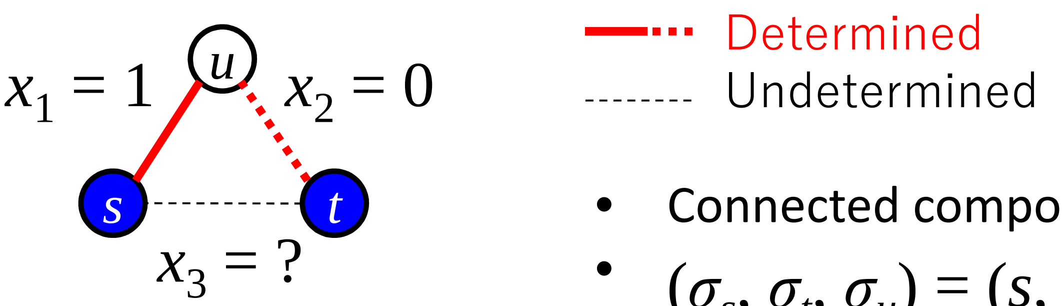
- Frontier: $F \subseteq V$
  - Set of vertices between determined/undetermined edges

Edge	F	Graph
x_1	$\{s, u\}$	
x_2	$\{s, t\}$	

Determined
Undetermined

Definition: Connected component

- To decide if s-t are connected, distinguish components
- Component ID of v: $\sigma_v \in V$
  - Min vertex in the connected component



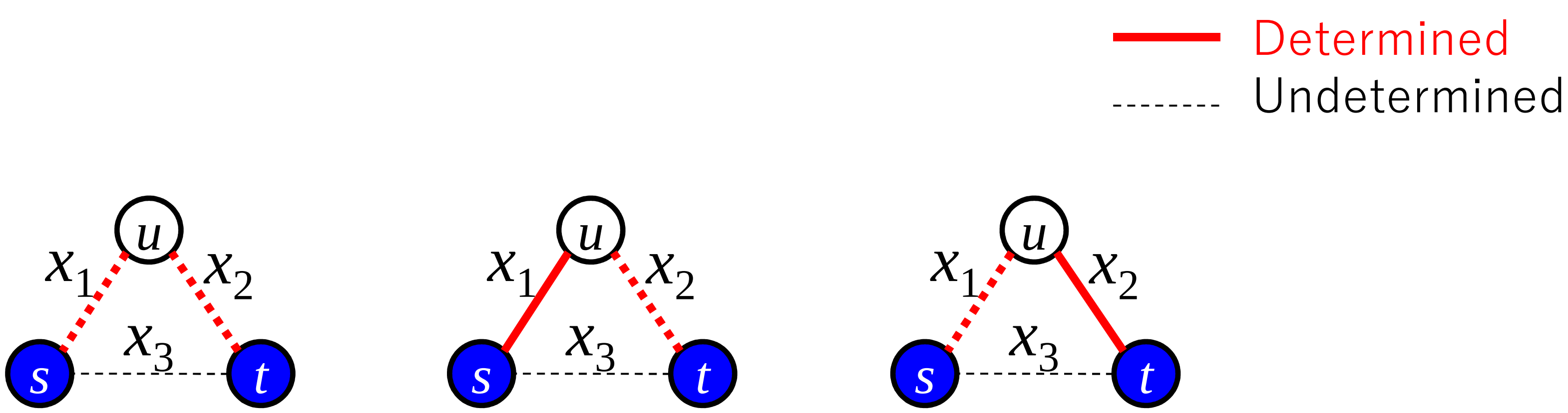
- Connected components: $\{s, u\}, \{t\}$
- $(\sigma_s, \sigma_t, \sigma_u) = (s, t, s)$

- $\sigma_t = s$ iff s and t are connected

Definition: Equivalency

- Defined with σ's of frontier vertices
  - Given subgraphs A and B,

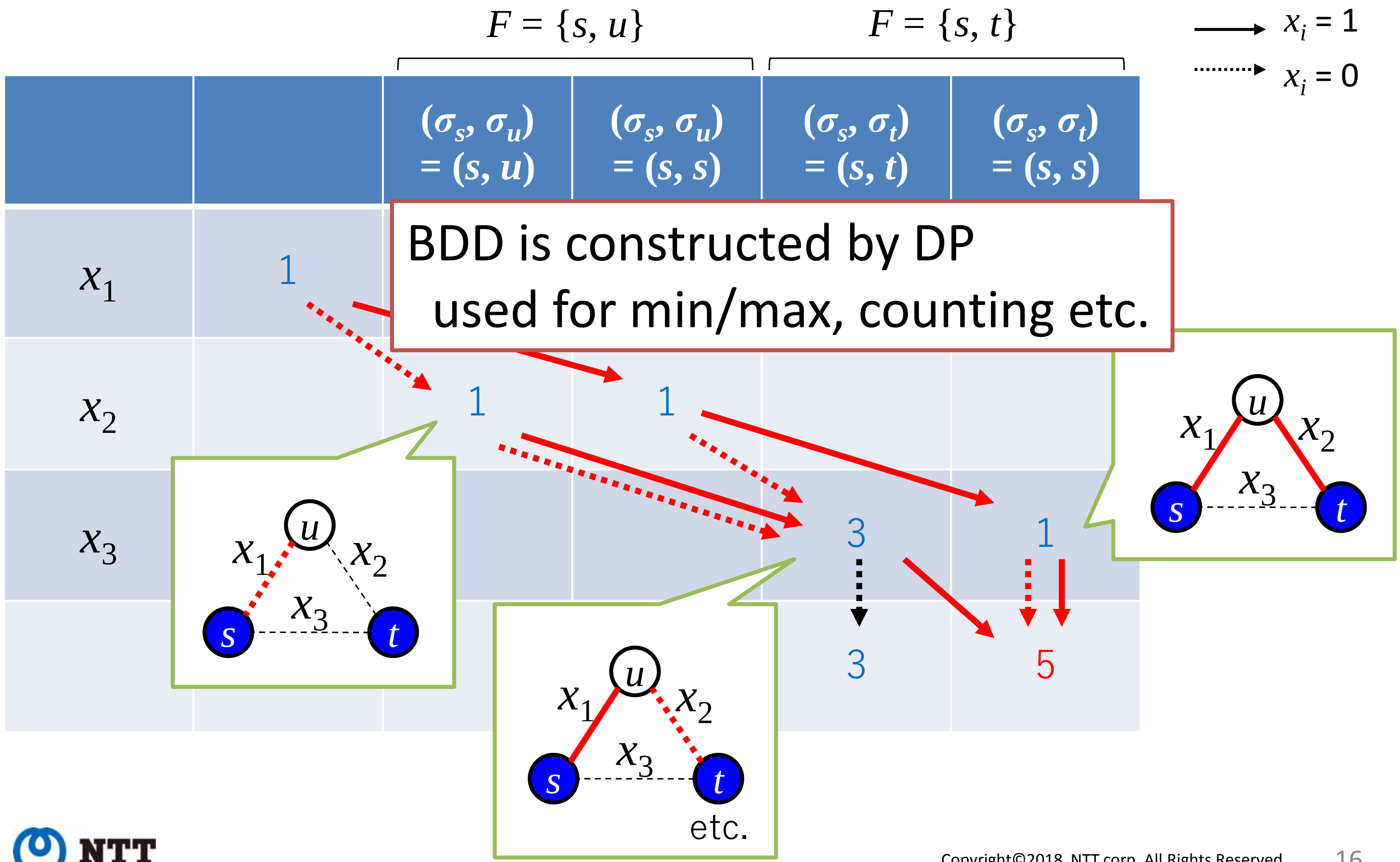
    they are equivalent iff $\forall u \in F_{\mathrm{A}}\ \forall v \in F_{\mathrm{B}}, \sigma_u = \sigma_v$



Equivalent subgraphs with $(\sigma_s, \sigma_t) = (s, t)$ at $F = \{s, t\}$

This is the enumeration algorithm

Frontier-based Search



Frontier-based Search

- Constructs BDD of graphs for given constraint
  - Constraint is specified by “equivalency”
    - connectivity, degree, graph class (path, cycle, tree etc.)

- Complexity depends on BDD size, not on # of graphs

- Standard queries (min/max, counting) are performed on BDD

- Historical notes
  - Originally developed for connectivity [1]
  - Generalized for various constraints [2]

[1] K. Sekine et al., “Computing the Tutte polynomial …,” in ISAC 1995.
[2] J. Kawahara, TI et al., “Frontier-based Search …,” IEICE EA 2017.

Application: Network reliability

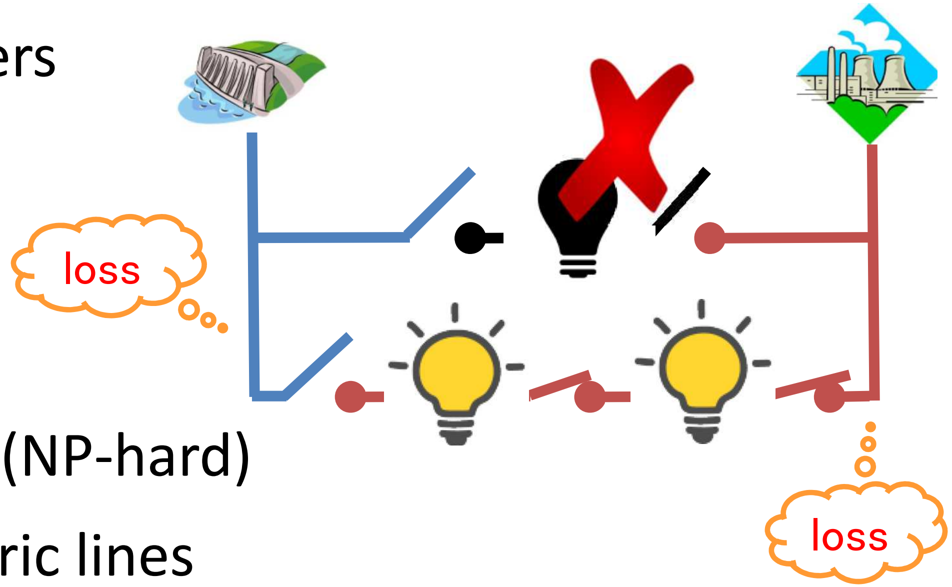
- Computes probability of connecting vertices when edges fail stochastically
  - Essentially same with counting connected subgraphs

- Extended for practical problems
  - Network design with *maximum* reliability [3]
  - Reliability analysis for *edge-disjoint paths* [4]
  - Scales with real Internet topologies ($|E| \sim 200$)

[3] M. Nishino, TI et al., “Optimizing Network Reliability …,” in INFOCOM 2018.
[4] TI “Reliability Analysis for Disjoint Paths,” IEEE Trans Rel (to appear).

Application: Power distribution

- Carries electricity to customers



- Tough *optimization* problem (NP-hard)
  - Min resistive loss on electric lines
  - Complex constraints on topology and voltage
- Solved largest benchmark ($|E|$ = 486, city of 300K people) [5]
  - Topological constraint is handled by Frontier-based Search

[5] TI et al., "Distribution Loss Minimization ...," IEEE Trans Smart Grid 2014.

国内初、スマートグリッド実現に向けた配電網の電力損失最小化の実証試験開始について

2016年4月20日
早稲田大学
国立研究開発法人科学技術振興機構
東京電力パワーグリッド株式会社

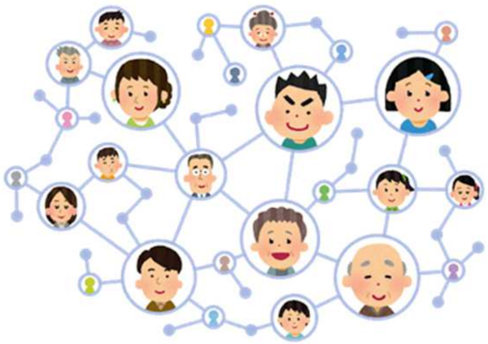
早稲田大学（本部：東京都新宿区　総長：鎌田薫　研究代表：理工学術院教授林泰弘）と、東京電力パワーグリッド株式会社（本社：東京都千代田区　社長：武部俊郎　以下「東京電力パワーグリッド」）は、国立研究開発法人科学技術振興機構（本部：埼玉県川口市

こうしたなか、早稲田大学と東京電力パワーグリッドは、一部地域の配電網を用いて、配電損失電力[※4]が最小となる配電網構成のための最適な運用方法の検討等を行うために本実証試験を行うこととしました。

http://www.tepco.co.jp/pg/company/press-information/press/2016/1277493_8622.html

More applications

- Railroad analysis [7]
- Logic puzzle solver/generator [8]
- Evacuation planning [9]
- Floor plans [10]
- Electoral partition [11]
- Social network analysis [12]



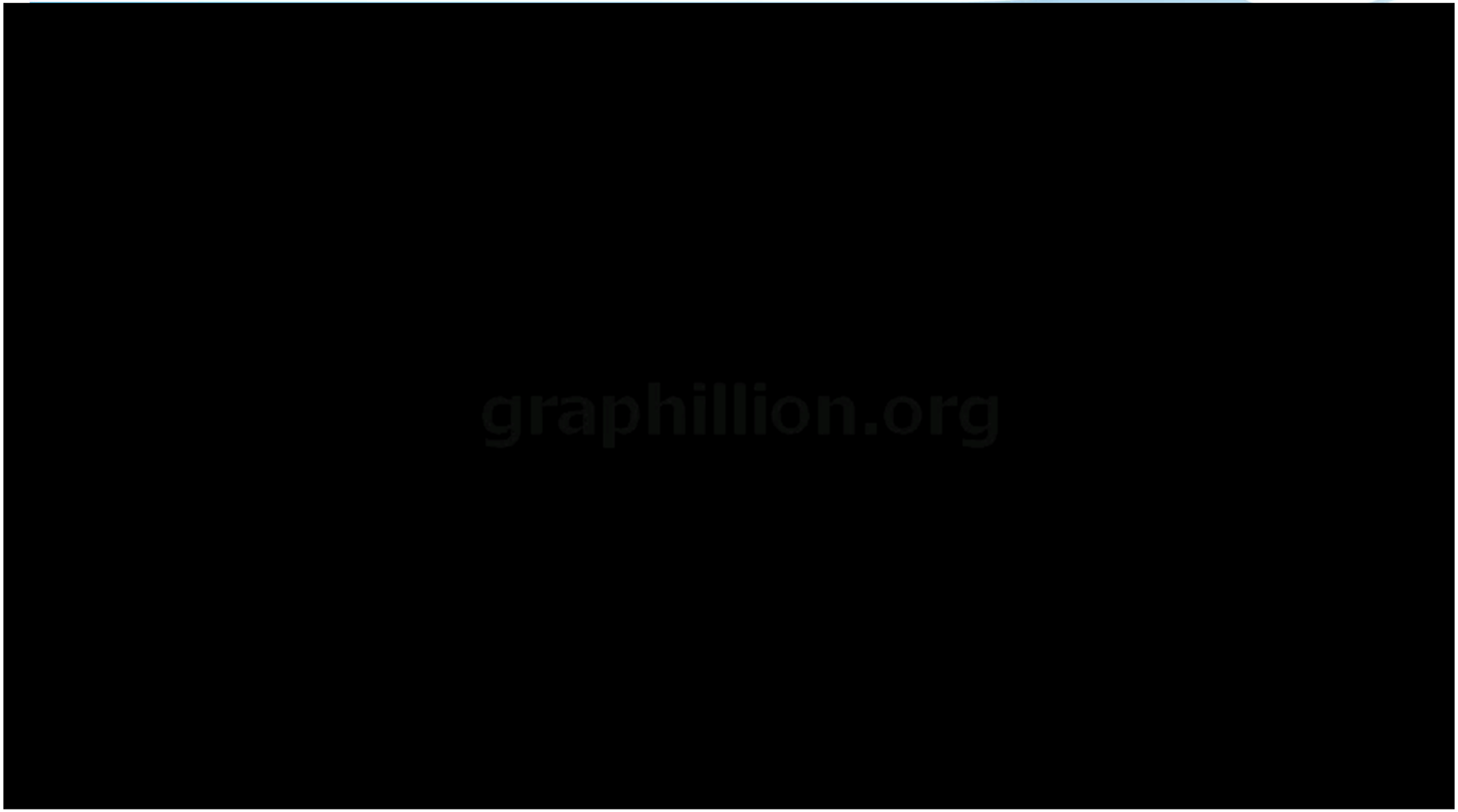
[7] http://www.nysol.jp/ekillion
[8] R. Yoshinaka et al., “Finding all solutions …,” Algorithms, 2012.
[9] A. Takizawa, TI et al., “Enumeration of Region Partitioning …,” in ISORA 2013.
[10] A. Takizawa et al., “Enumeration of Floor Plans …,” in CAADRIA 2014.
[11] J. Kawahara et al., “Generating All Patterns …,” in WALCOM 2017.
[12] T. Maehara et al., “Exact Computation …,” in WWW 2017.

Graphillion [13]

[13] TI et al., “Graphillion: ...,” Springer STTT 2016.

- Implements Frontier-based Search
  - Open-source! http://graphillion.org

Demo



http://youtu.be/R3Hp9k876Kk

Summary

- Frontier-based Search: graph enumeration algorithm
  - Constructs BDD with dynamic programming
  - Supports standard queries (min/max, counting)

- Real-world applications
  - Network reliability, power distribution etc.

- Graphillion
  - Implements Frontier-based Search, open-sourced