
Fully Homomorphic Encryption

Mohammad Ahmadian

Ashkan Paya

{ahmadian,ashkan_paya}@knights.ucf.edu

Computing on Encrypted Data

- ❑ Storing my files on the cloud
 - Encrypt them to protect my information
 - Search through them for emails with “homomorphic” in the subject line
 - Cloud should return only these (encrypted) messages, w/o knowing the key
- ❑ Private Internet search
 - Encrypt my query, send to Google
 - I still want to get the same results
 - Results would be encrypted too

Public-key Encryption

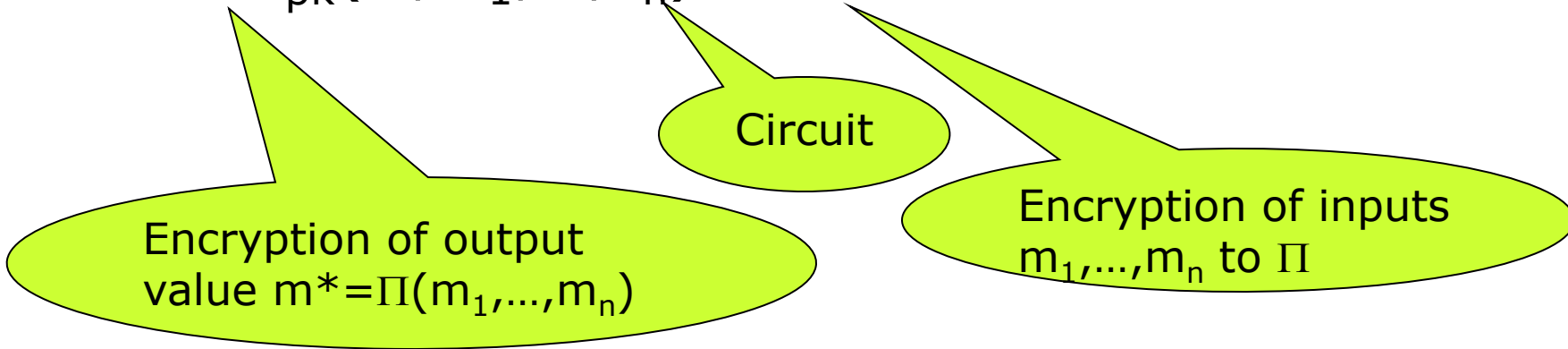
- Three procedures: **KeyGen**, **Enc**, **Dec**
 - $(sk, pk) \leftarrow \text{KeyGen}(\$)$
 - Generate random public/secret key-pair
 - $c \leftarrow \text{Enc}_{pk}(m)$
 - Encrypt a message with the public key
 - $m \leftarrow \text{Dec}_{sk}(c)$
 - Decrypt a ciphertext with the secret key

- E.g., RSA: $c \leftarrow m^e \bmod N$, $m \leftarrow c^d \bmod N$
 - (N, e) public key, d secret key

Homomorphic Public-key Encryption

□ Also another procedure: **Eval**

■ $c^* \leftarrow \text{Eval}_{pk}(\Pi, c_1, \dots, c_n)$



■ Π a Boolean circuit with ADD, MULT mod 2

An Analogy: Alice's Jewelry Store

❑ Alice's workers need to assemble raw materials into jewelry

❑ But Alice is worried about theft

How can the workers process the raw materials without having access to them?



An Analogy: Alice's Jewelry Store

- ❑ Alice puts materials in locked glove box
 - For which only she has the key
- ❑ Workers assemble jewelry in the box
- ❑ Alice unlocks box to get "results"



The Analogy

- **Enc**: putting things inside the box
 - Anyone can do this (imagine a mail-drop)
 - $c_i \leftarrow \text{Enc}_{pk}(m_i)$
- **Dec**: Taking things out of the box
 - Only Alice can do it, requires the key
 - $m^* \leftarrow \text{Dec}_{sk}(c^*)$
- **Eval**: Assembling the jewelry
 - Anyone can do it, computing on ciphertext
 - $c^* \leftarrow \text{Eval}_{pk}(\Pi, c_1, \dots, c_n)$
- $m^* = \Pi(m_1, \dots, m_n)$ is “the ring”, made from “raw materials” m_1, \dots, m_n

A homomorphic symmetric encryption

- ❑ Shared secret key: odd number p
- ❑ To encrypt a bit m :
 - Choose at random large q , small r
 - Output $c = pq + 2r + m$
 - Ciphertext is close to a multiple of p
 - $m = \text{LSB of distance to nearest multiple of } p$
- ❑ To decrypt c :
 - Output $m = (c \bmod p) \bmod 2$

$2r+m$ much smaller than p

Why this scheme is Homomorphic?

$$\square C_1 = q_1 p + 2r_1 + m_1, C_2 = q_2 p + 2r_2 + m_2$$

Distance to nearest multiple of p

$$\square C_1 + C_2 = (q_1 + q_2)p + 2(r_1 + r_2) + (m_1 + m_2)$$

➤ $2(r_1 + r_2) + (m_1 + m_2)$ still much smaller than p

$$\blacksquare C_1 + C_2 \bmod p = 2(r_1 + r_2) + (m_1 + m_2)$$

$$\square C_1 \times C_2 = (C_1 q_2 + q_1 C_2 - q_1 q_2 p)p$$

$$+ 2(2r_1 r_2 + r_1 m_2 + m_1 r_2) + m_1 m_2$$

$$\blacksquare 2(2r_1 r_2 + r_1 m_2 + m_1 r_2) + m_1 m_2$$

➤ $C_1 \times C_2 \bmod p = 2(2r_1 r_2 + r_1 m_2 + m_1 r_2) + m_1 m_2$ still smaller than p

Security

- The approximate-GCD problem:
 - Input: integers x_1, x_2, x_3, \dots
 - Chosen as $x_i = q_i p + r_i$ for a secret odd p
 - $p \in_{\$} [0, P], q_i \in_{\$} [0, Q], r_i \in_{\$} [0, R]$ (with $R \ll P \ll Q$)
 - Task: find p
- Thm: If we can distinguish $\text{Enc}(0)/\text{Enc}(1)$ for some p , then we can find that p
 - Roughly: the LSB of r_i is a “hard core bit”
 - ➔ Scheme is secure if approx-GCD is NP-hard
- Is approx-GCD really a NP-hard problem?



Hard-core-bit theorem

A. The approximate-GCD problem:

- Input: $w_0 = q_0 p, \{w_i = q_i p + r_i\}$
- $p \in [0, P], q_i \in [0, Q], r_i \in [0, R]$ (with $R \ll P \ll Q$)

Task: find p

B. The cryptosystem

- Input: $N = q_0 p, \{m_j, c_j = q_j p + 2r_j + m_j\}, c = qp + 2r + m$
- $p \in [0, P], q_i \in [0, Q], r_i \in [0, R']$ (with $R' \ll P \ll Q$)

Task: distinguish $m=0$ from $m=1$

□ Thm: Solving B \rightarrow solving A

Questions

- ❑ What will happen if we continue to running too many function on encrypted data?
- ❑ Why Approximate GCD is NP-Hard?
- ❑ How if could covert Partially HE to Fully HE?

Turing Test as a Defining Feature of AI-Completeness

Kutalmis Akpinar Kai Li Jun Ye

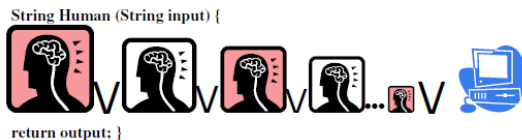
Data Systems Group, UCF



April 25, 2014

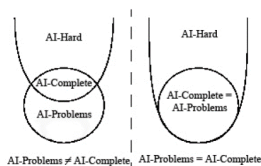
- What can we solve that machines can't?
- What needs human mind?
- An analogy to NP-Completeness?

- Human Oracle (HO):
 - $Human_{Best}$: Capable of computing any function computable by the union of all minds



- AI Problem:
 - Can be *solved* by a human oracle.

- AI-Complete:
 - It is in the set of AI-problems (Human Oracle solvable).
 - Any AI problem can be converted into it by some polynomial time algorithm.
- AI-Hard:
 - A problem is AI-Hard iff there is an AI-Complete problem that is poly-time Turing reducible to it.
- AI-Easy:
 - Can be solvable in poly-time using an oracle for an AI-problem.

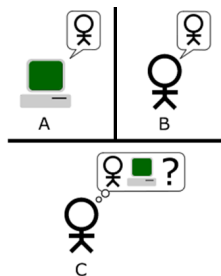


Turing Test

Test of a machine's ability to exhibit intelligent behavior equivalent to, or indistinguishable from, that of a human

(Alan Turing, *Computing Machinery and Intelligence*, 1950)

- A human judge (C) engages in natural language conversations with a human (B) and a machine (A).
- If the judge cannot reliably tell the machine from the human, the machine is said to have passed the test.
- A, B and C are in separate rooms.
- Conversation is limited to a text-only channel by keyboards and screens.



(Turing test. Figure from Wikipedia.)

TT as the first AI-complete problem

(SAT as the first NP-complete problem)

- How to prove?
 - TT is in the set of AI problems (HO solvable)
 - Any other AI problems reduce to TT in polynomial time under Turing reduction.
- For any AI problem h , we have a String input which encodes the problem and a String output which encodes the solution.
- By taking the input as a question to be used in the TT and output as the answer to be expected, h reduces to TT.
- TT is AI-complete.

An AI-Hard Problem

The Programming problem: Take a natural language description of a program and produce a source code.

e.g. "Write a program to play Tic-Tac-Toe".

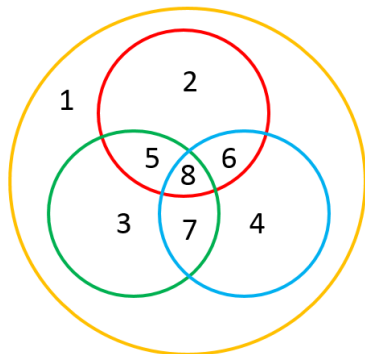
Proof: For any statement S of TT , transform the statement into a programming assignment of the form: Write a program which would respond to S with a statement indistinguishable from a statement provided by an average human.

This says $TT \leq_{TM}^P Programming$.

However, *Programming* itself is not in AI as there are many instances of *Programming* which are not solvable by Human Oracles. (e.g. Write a program to pass Turing Test is not an AI problem). This shows that $TT \notin AI$.

Beyond AI-completeness

Venn diagram of Problem spaces produced by different types of intelligent computational devices.



- 1. Universal Intelligence
- 2. Human Intelligence
- 3. Artificial Intelligence
- 4. Animal Intelligence

Consciousness-Complete Problems (C-Complete): the problem of reproducing internal states of human mind in artificial way

- The current Turing/Von Neumann architecture cannot deal with problems related to internal states of human mind.
- Unlike Turing Machine Oracle, C-Oracles do not produce any symbolic output
- Consciousness is suspected to be the first C-Complete problem. Yet no proof is provided yet.

Question

There are many other AI problems believed to be AI-complete. E.g. Natural Language Understanding, vision/image/video Understanding. Try to prove the following two sub-problems in Natural language Understanding are AI-complete .

1. Question Answering (QA)
2. Speech Understanding (SU)

Hint: You can first prove they are AI problems and then find a way to reduce Turing Test (TT) to them. ($TT \leq_{TM}^P QA$, $TT \leq_{TM}^P SU$)

Roman V. Yampolskiy,
“Turing Test as a Defining Feature of AI-Completeness” .
*Artificial Intelligence, Evolutionary Computing and Metaheuristics Studies
in Computational Intelligence*. Volume 427, 2013, pp 3-17

On NP-Completeness of Modularity Based Algorithms for Community Detection in Graphs

Presented by: Alireza Hajibagheri, Hamidreza Alvari

April 2014

Outline

- ▶ Introduction
- ▶ Modularity
- ▶ NP-Completeness
- ▶ Optimization Approach
- ▶ Conclusion and Question

Introduction

- ▶ Social Networks
 - ▶ Graphs with millions of nodes and relations
 - ▶ Interesting Features
- ▶ Social Networks Analysis
 - ▶ Improve performance in Internet-based applications (e.g. networking websites, recommendation networks, etc.)

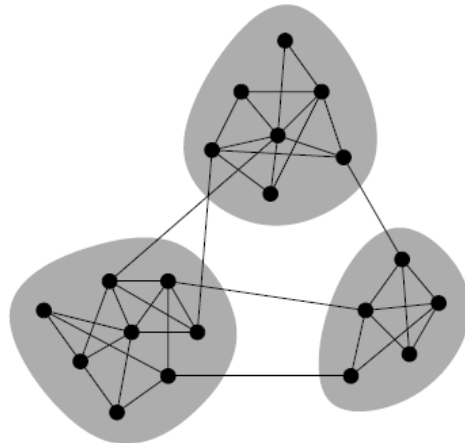
Introduction (contd.)

- ▶ Interesting property
 - ▶ Presence of social communities and their structures
 - ▶ Graph partitioning problem
- ▶ Idea goes back to Newman (2007) [1]
 - ▶ Graphs (networks) are found to divide naturally into communities or modules

Modularity

- ▶ Detection method which is optimization of the quality function known as “modularity”

$$Q = \frac{1}{4m} \sum_{ij} \left(A_{ij} - \frac{k_i k_j}{2m} \right) s_i s_j$$

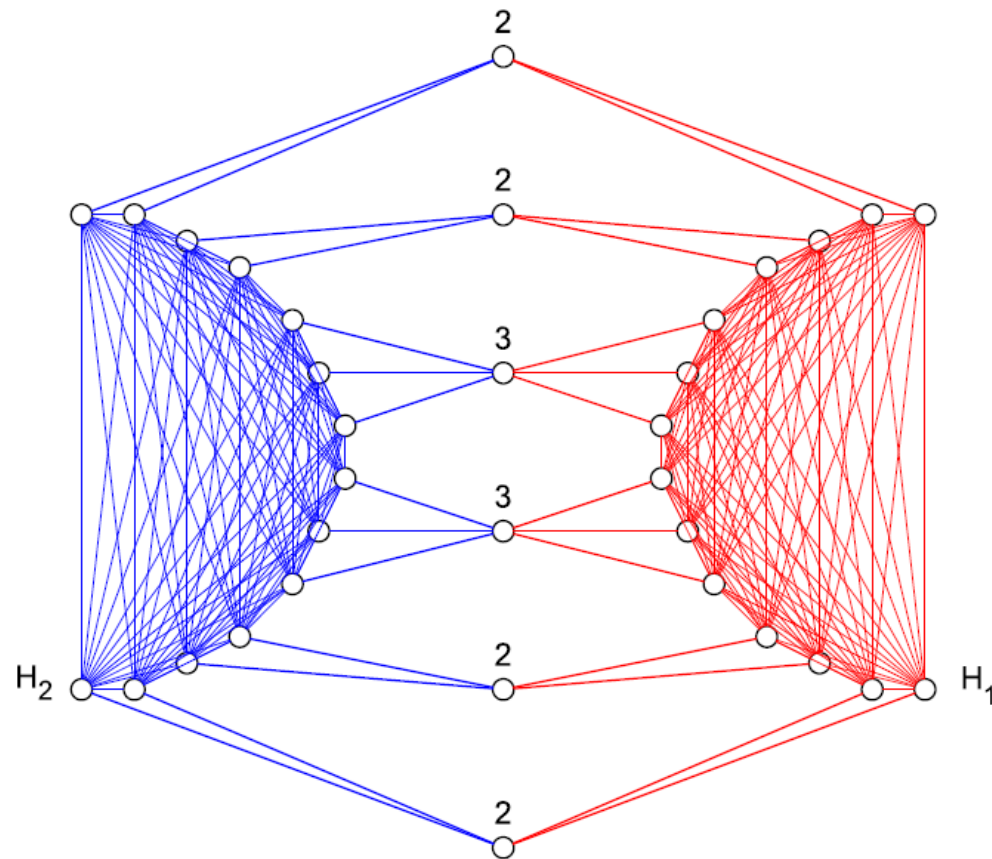


NP-Completeness

- ▶ We use reduction from 3-Partition to Modularity
- ▶ Modularity: Given a graph G and a number K , is there a clustering C of G , for which $q(C) \geq K$?
- ▶ 3-Partition: Given $3k$ positive integer numbers sum up to kb and $b/4 < a_i < b/2$, is there a partition of these numbers into k sets each of which sum up to b ?

NP-Completeness (contd.)

- ▶ An instance $A = \{a_1, \dots, a_{3k}\}$ and instance $(G(A), K(A))$
- ▶ We construct $G(A)$ with k cliques H_1, \dots, H_k
- ▶ Example:



Greedy Algorithm

- ▶ Divisive hierarchical clustering algorithm [2]
- ▶ Starts with the singleton clustering and iteratively merges those two clusters that yield a clustering with the best modularity

Algorithm 1: GREEDY ALGORITHM FOR MAXIMIZING MODULARITY

Input: graph $G = (V, E)$
Output: clustering \mathcal{C} of G
 $\mathcal{C} \leftarrow$ singletons
initialize matrix Δ
while $|\mathcal{C}| > 1$ **do**
 find $\{i, j\}$ with $\Delta_{i,j}$ is the maximum entry in the matrix Δ
 merge clusters i and j
 update Δ
return clustering with highest modularity

Conclusion

- ▶ Identifying communities is of great interest!
- ▶ We presented a greedy algorithm based on maximizing the quality function modularity.
- ▶ We used reduction from 3-Partition to Modularity to show that maximizing modularity is NP-Complete.

References

1. Modularity and community structure in networks, M. E. J. Newman, Proc. Natl. Acad. Sci. USA 103, 8577-8582 (2006)
2. Newman, M.E.J., Girvan, M.: Finding and evaluating community structure in networks. Physical Review E69 (2004)



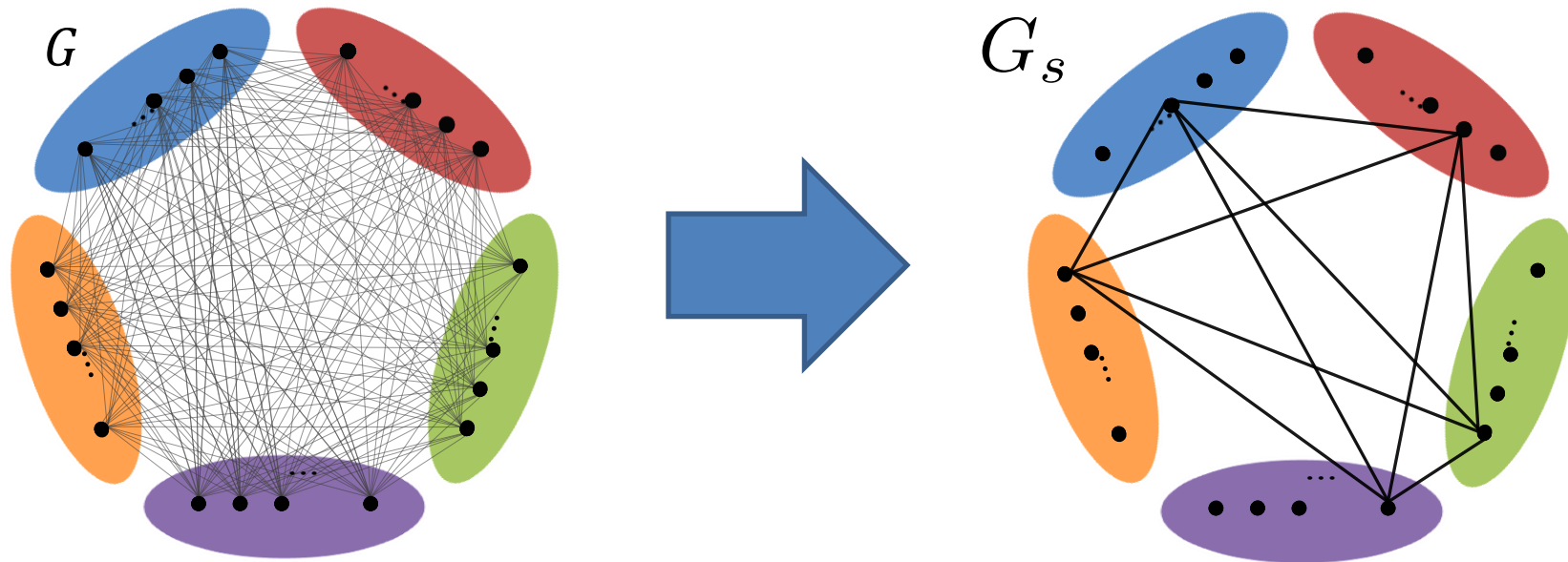
Thank You 😊

Generalized Minimum Clique Problem (GMCP)

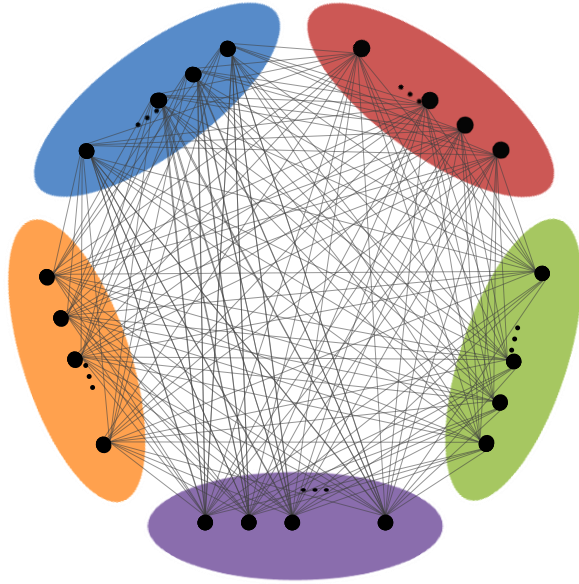
Shervin Ardeshir, Mahdi Kalayeh, Afshin Dehghan

Generalized Minimum Clique Problem (GMCP)

- **Traveling Salesman** : find the minimal cycle which visits all the nodes exactly once
- **Generalized Traveling Salesman** : Find the minimal cycle which connects all the clusters while exactly one node from each is visited
- **GMCP**: Find a subset of the nodes that includes exactly one node from each cluster while the cost of the complete graph that the subset forms is minimized (Can be reduced to Traveling Salesman, therefore its NP-Hard)

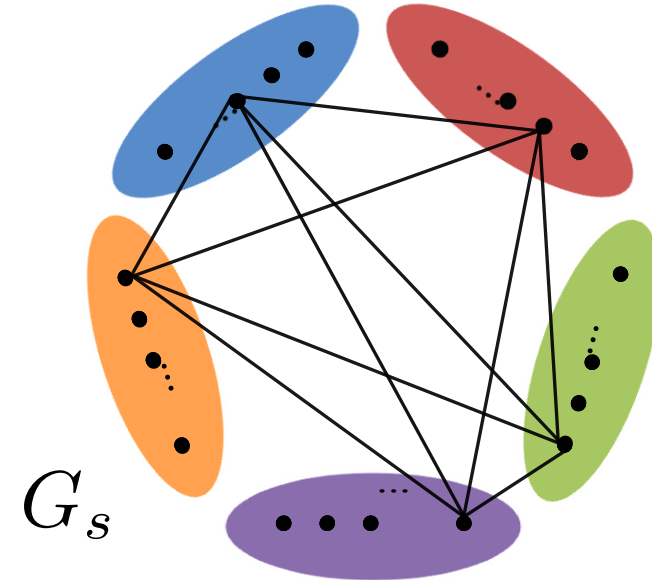
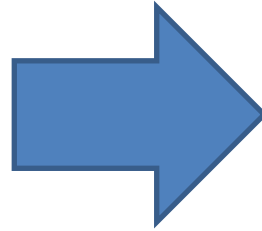


Generalized Minimum Clique Problem (GMCP)



$$G = (V, E, w)$$

V Nodes E Edges w Edge Weights



$$G_s = (V_s, E_s, w_s)$$

$$\hat{V}_s = \operatorname{argmin}_{V_s} C(V_s)$$

Approximate Solution (Heuristic)


An Approximate Solution for GMCP

- Best solution is initialized by picking one random node from each cluster
 - Complexity $O(L)$ for verification where L : number of clusters
- Fix neighborhood size to 1 and identify the δ best solutions
 - Complexity $O((K-1)L)$, where K : number of nodes
- At each iteration neighbor solutions of the size $|V|-\delta$ to $|V|-1$ are evaluated and compared with the Best solution
 - Complexity $O(K^\delta L)$
- Iterations continue until either convergence criteria(Minimum found) or termination criteria(Maximum time/Maximum iterations) are met
- Complexity of the whole process :
 - $O(L) \times O((K-1)L) + O(K^\delta L) = O((K-1)L^2) + O(K^\delta L) = O(KL^2 + K^\delta L) \rightarrow$
Polynomial Time

Exact Solution (BIP)

Solving GMCP using Binary Integer Programming

- Finds the optimal solution


$$\begin{cases} W^T X \\ AX = B \\ MX \in N \end{cases}$$

- W Is the object weights
- X Binary variable for the nodes and edges

- Constraints ensure the solution is a valid GMCP instance
- $$\begin{matrix} AX = B \\ MX \in N \end{matrix}$$

Constraints

Ensures that one and only one node is selected from each cluster

$$\{\forall j | 1 \leq j \leq h\} : \sum_{i=1}^k \nu_i^j = 1.$$

states if one node is selected to be in Gs, then exactly $(h - 1)$ of its edges should be included in Gs

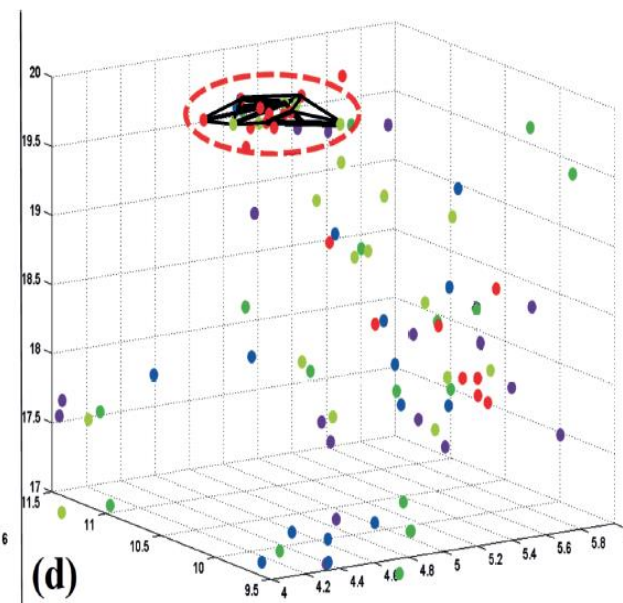
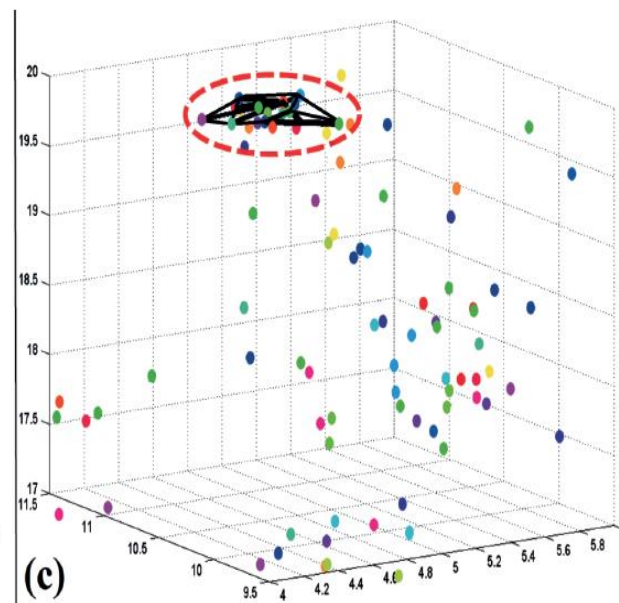
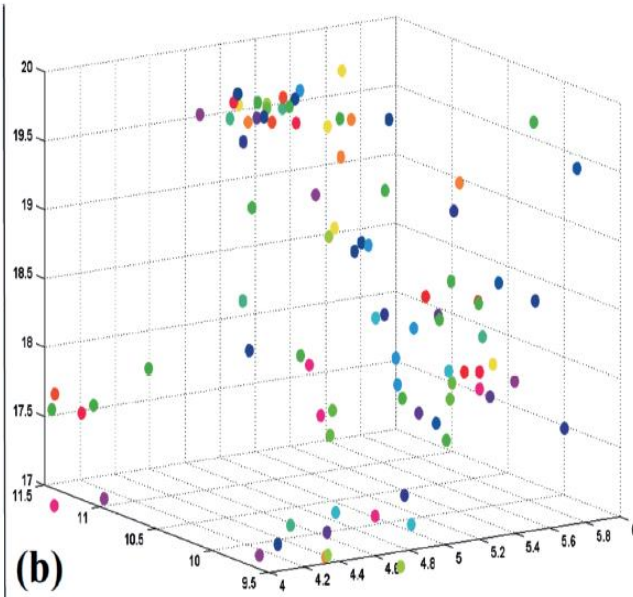
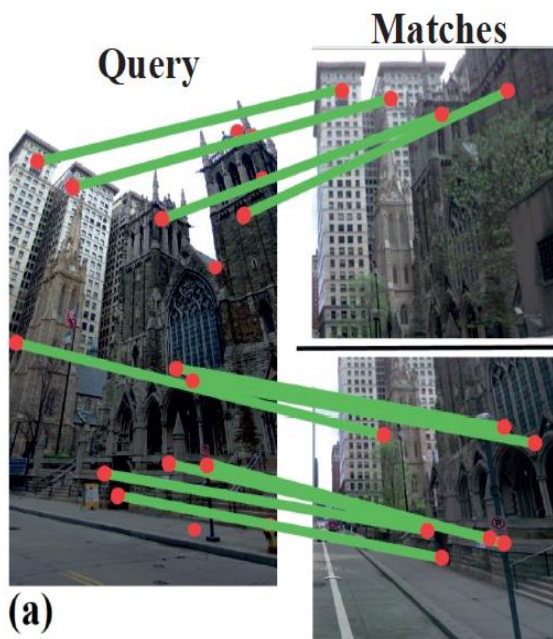
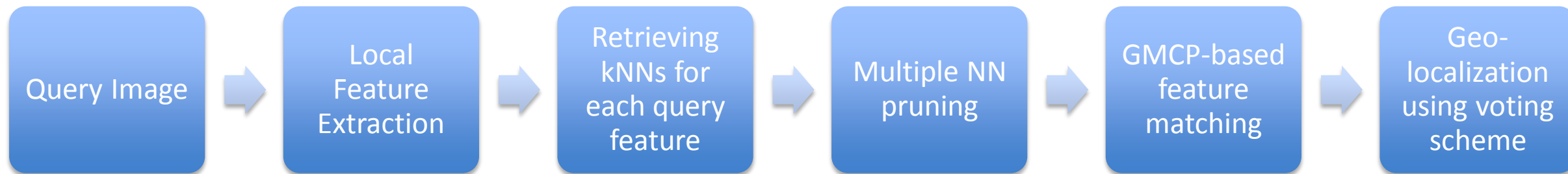
$$\{\forall m, n | 1 \leq m \leq h, 1 \leq n \leq k\} : \sum_{i=1}^h \sum_{j=1}^k \varepsilon_{mn}^{ij} = \nu_n^m \cdot (h - 1).$$

ensures if one edge is included in Gs, then the variables of its parent nodes have to be 1 and vice versa

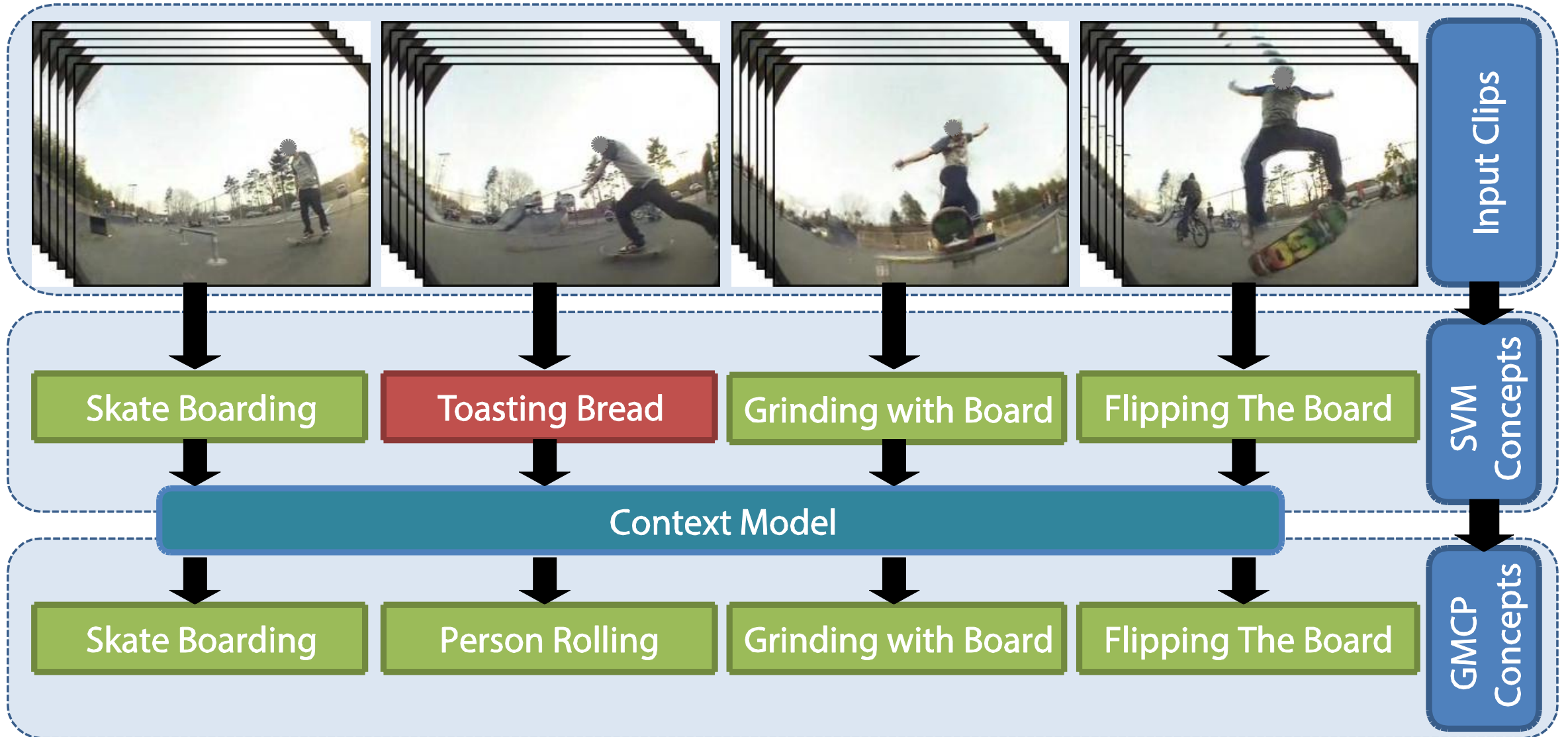
$$\{\forall m, n, i, j | 1 \leq m, j \leq h, 1 \leq n, i \leq k\} : \nu_n^m \wedge \nu_i^j = \varepsilon_{mn}^{ji}.$$

Applications

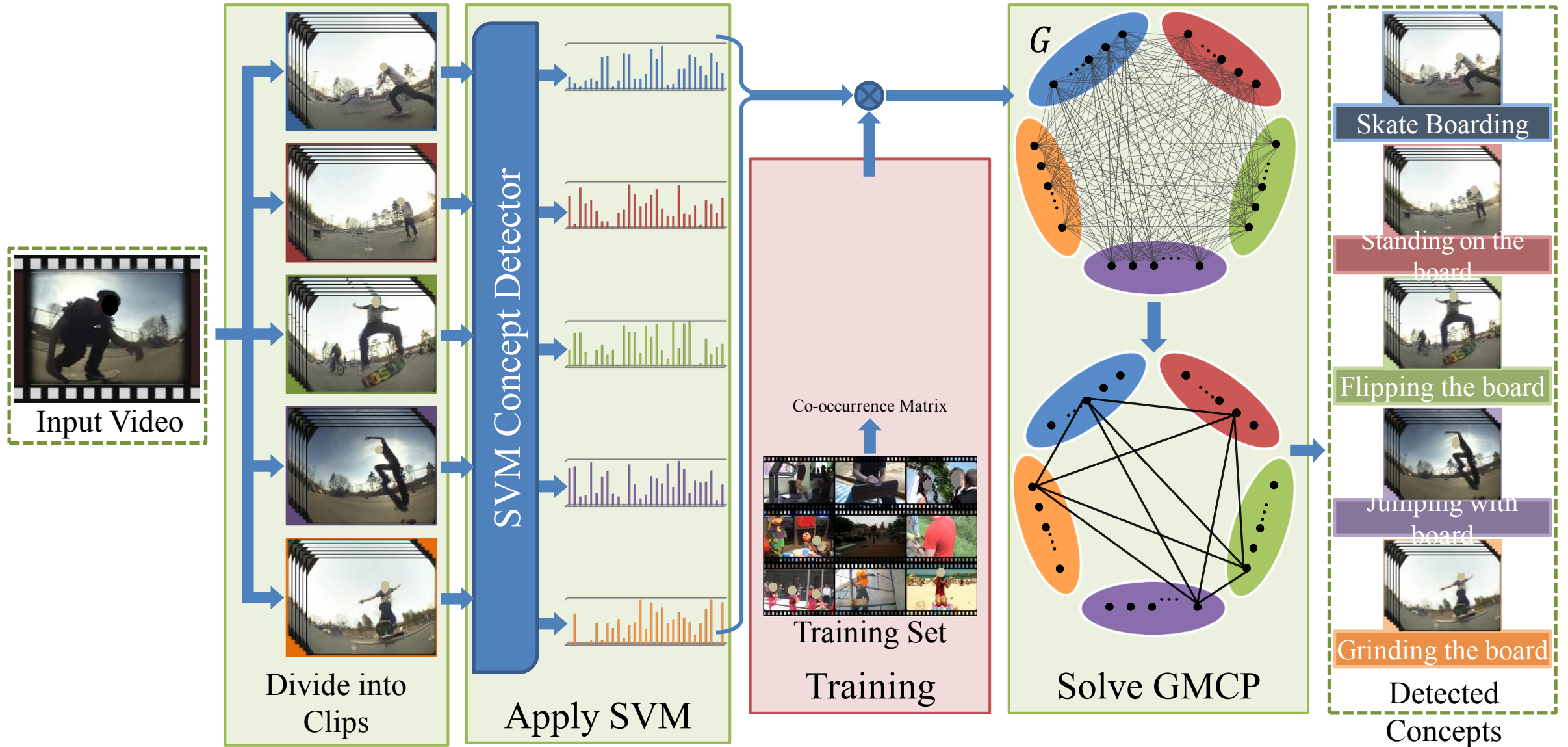
Problem : Image Geo-Localization



Problem : Concept Detection in Complex Videos



Problem : Concept Detection in Complex Videos



Thanks

Questions??

- In BIP solution for GMCP there is a Boolean variable associated to each node and each edges. What happens if we allow the solver to pick any number between $[0, 1]$ instead of a binary variable 0 and 1?
- Show that GMCP is NP-HARD (Show how does it reduce to TSP)?



On the undecidability of probabilistic planning and related stochastic optimization problems

Team members:

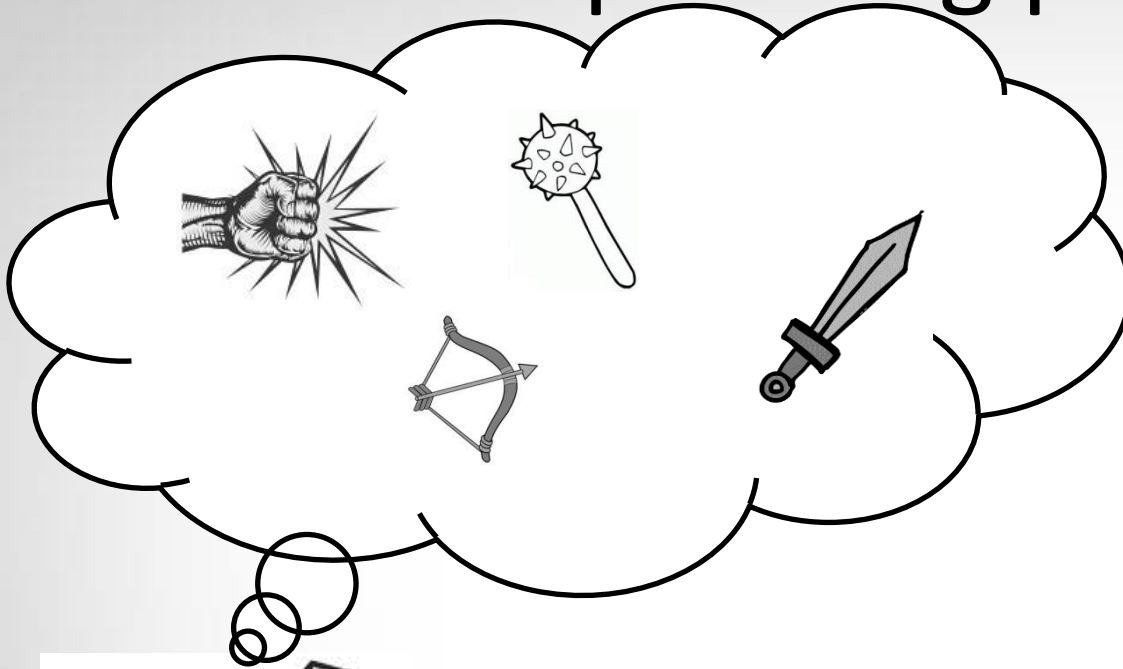
Dustin Morley and Edward Aymerich



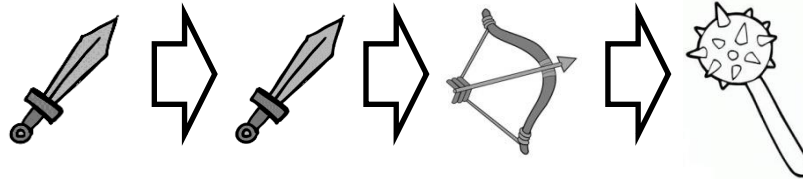
The planning problem

- The planning problem consists of:
 - A finite set of states.
 - A finite set of actions.
 - A start state.
 - A set of goal states.
 - A threshold τ on the probability of success.
- Find any sequence of actions that would move the system from the start state to any goal state with probability higher than τ .

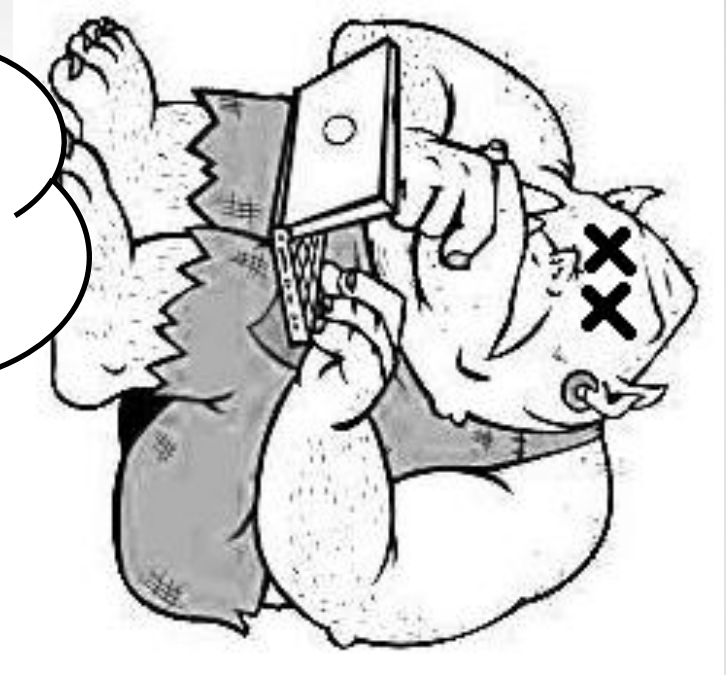
The planning problem



The planning problem



95% chance of
WIN!!!!



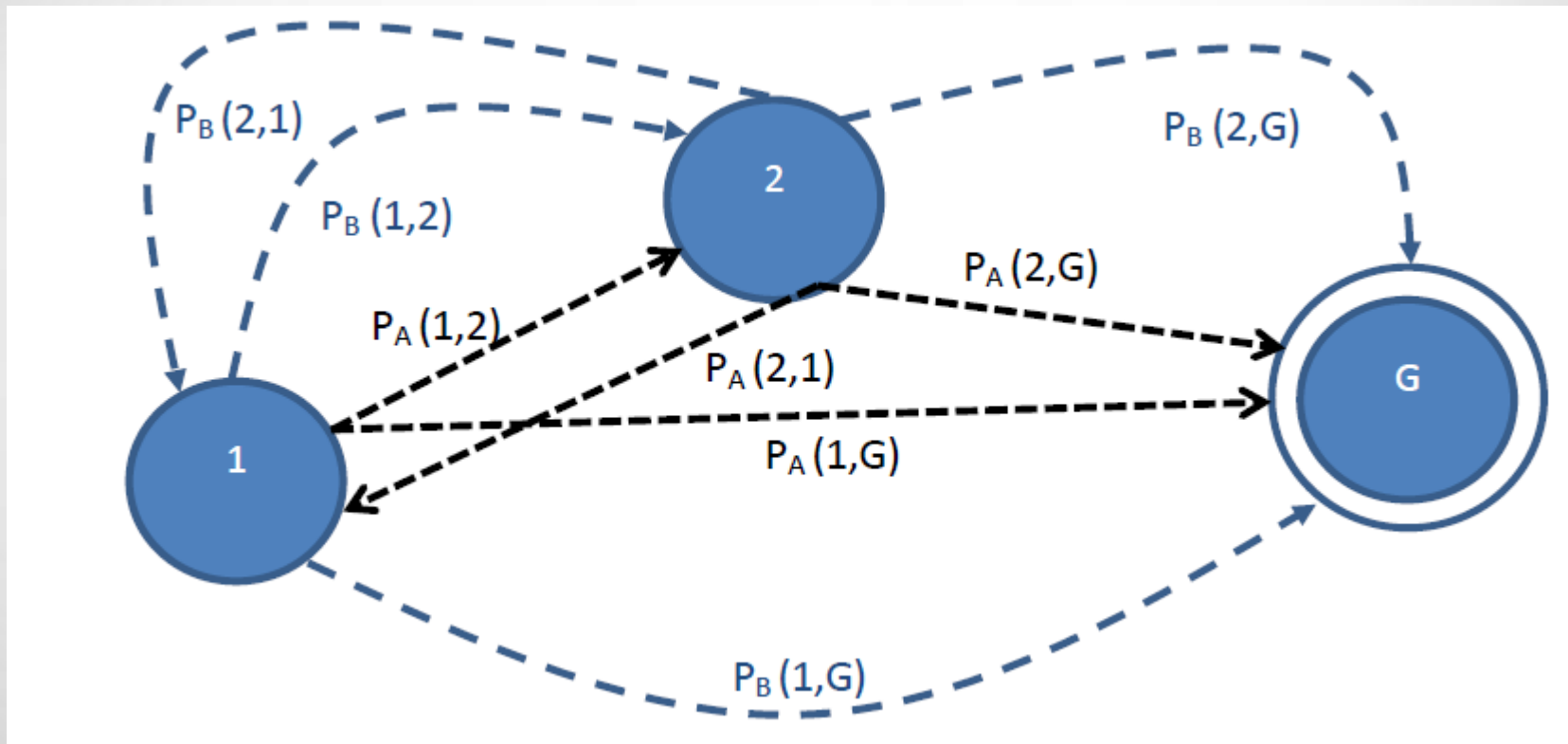
The planning problem

Undecidable



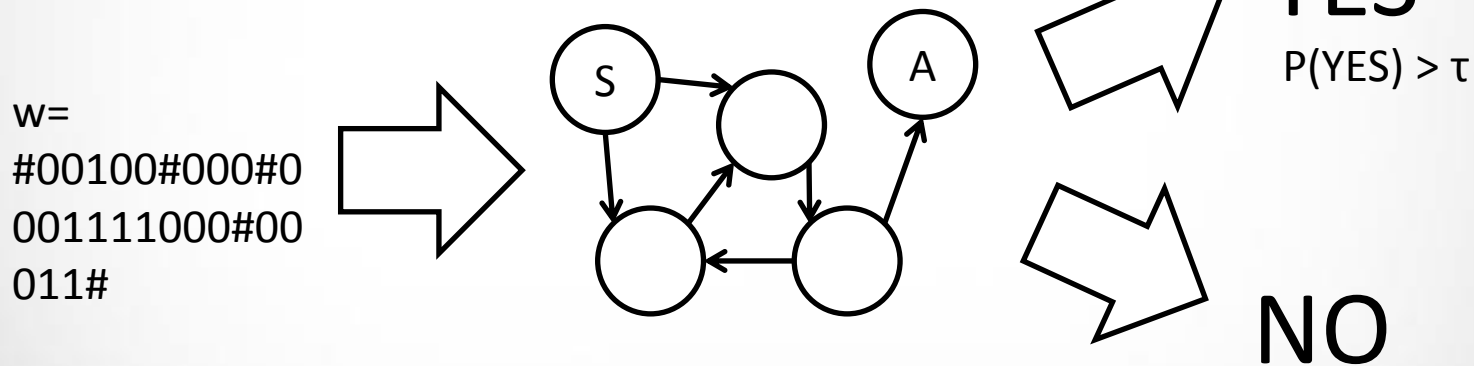
Probabilistic Finite Automata

- Reducible to probabilistic planning



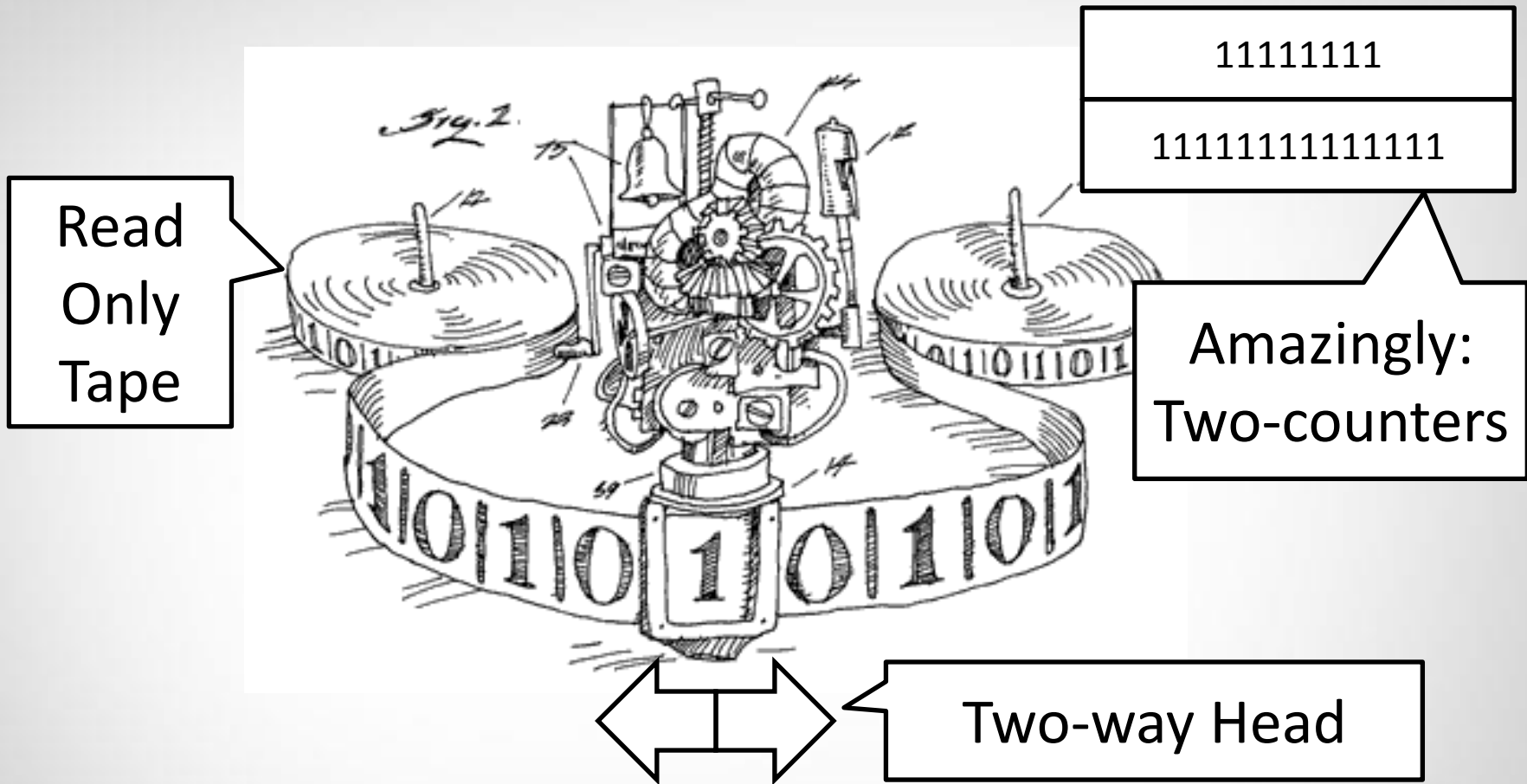
The emptiness problem

- Given a PFA and a threshold τ , decide whether or not there is some input string $w \in \Sigma^*$ that the PFA accepts with probability exceeding threshold τ .



- Undecidable by reduction from two-counter TM.

Two-Counter TM



$$w = \#C_1\#C_2\#C_3\#\dots\#C_{n-2}\#C_{n-1}\#C_n\#$$

TM reduction to PFA

- The constructed PFA must check if a sequence of computations is valid.

$$w = \#C_1\#C_2\#C_3\#\dots\#C_{n-2}\#C_{n-1}\#C_n\#$$

Start State

Valid Transitions

Accepting State

(PFA can't check valid counter contents exactly)

Check counters' validity reduces to:

$a^i b^j, i=j?$ → undecidable for DFA.



Weak Equality Test

- Goal – radically different probabilities of success between $i=j$ and $i \neq j$, for string $a^i b^j$
- Four series of coin tosses
 - Series 1: two tosses per 'a'
 - Series 2: two tosses per 'b'
 - Series 3 and 4: one toss per 'a' and 'b'
- Notice – if $i=j$, all-heads outcome equally likely
- Otherwise, more likely for series 1 or 2
 - One of them has fewer tosses



Weak Equality Test

- Easy part: Suspect if $i \neq j \pmod k$
- $H(x)$: “all-heads outcome on series x ”
- Indecisive if $\sim H(x) \forall x$ or $[H(1)|H(2)] \&\& [H(3)|H(4)]$
- Suspect if decisive and $[H(1)|H(2)]$
- Correct if decisive and $[H(3)|H(4)]$
- When decisive and $i=j$, $\Pr(\text{Suspect}) = \Pr(\text{Correct})$
- When decisive and $i \neq j$, $\Pr(\text{Suspect}) > \Pr(\text{Correct})$
 - Can show: $\Pr(\text{Suspect}) > 2^{k-1} \Pr(\text{Correct})$

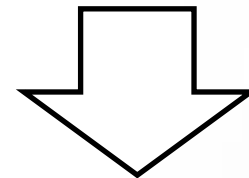
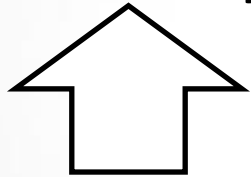


Completing the Reduction

- PFA overview
 - TM trace w as input
 - Enter Indecisive state, set $z=w$
 - While(Indecisive), run W.E.T., $z = zw$
- Loop forces a decision to be made
 - If valid trace, PFA accepts w with 50% probability
 - This can approach 100% if force more decisions
 - Otherwise, probability $< 1 - \frac{2^{k-1}}{1+2^{k-1}}$

Undecidability for Probabilistic Planning

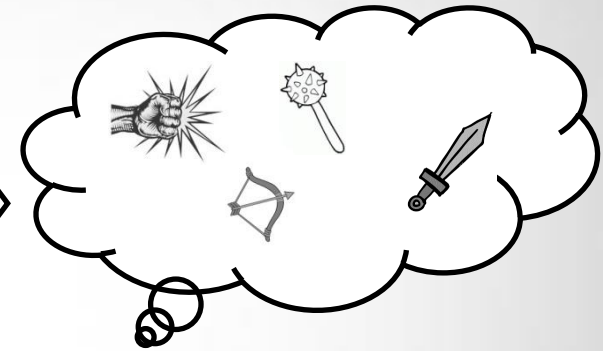
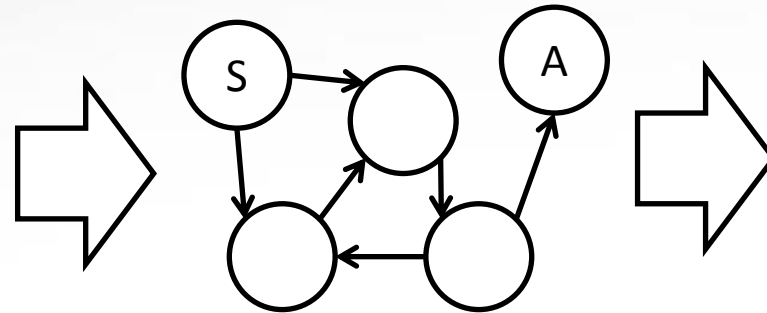
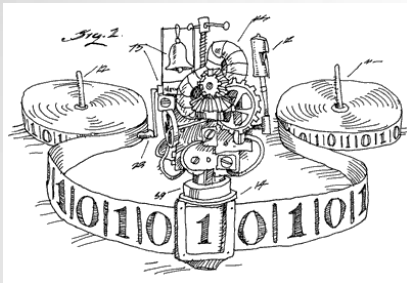
- An algorithm that solves the planning problem would solve the emptiness problem.
- emptiness problem \leq planning problem.



Undecidable

Undecidable

Summary of the Proof



Turing
Machine:
Empty string
acceptance

PFA:
Emptiness
problem

Probabilistic
planning:
The planning
problem

Extensions

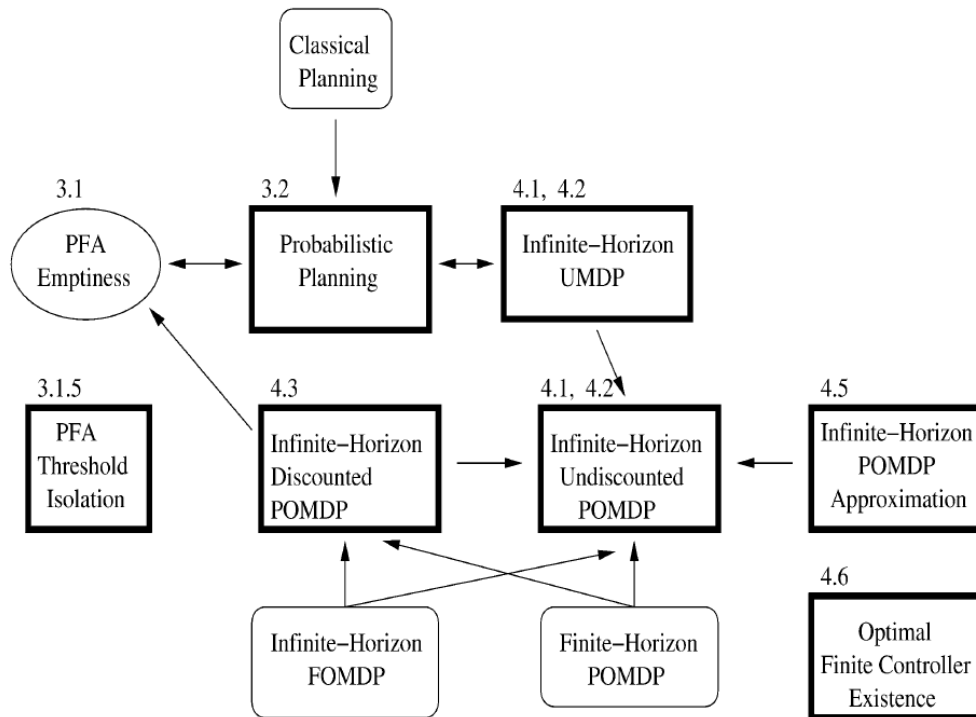


Fig. 1. Summary of Undecidability Results. Problems in bold rectangles are those established as undecidable in this paper, with the proofs starting from the result in the oval. In the rounded rectangles are related problems with previously known complexity results. Arrows point from “easier” to “harder” problems. Above each problem is the section number where the problem is addressed.



Open Questions

- What about special classes of PFAs that are not covered by this reduction?
 - Ex: PFA with just two actions and two states?
 - Others?
- Are there any other problems in computer science that the weak equality test be applied to?

A Fast Algorithm for Equitable Coloring

SteVen Batten

John Boswell

Mike Galletti

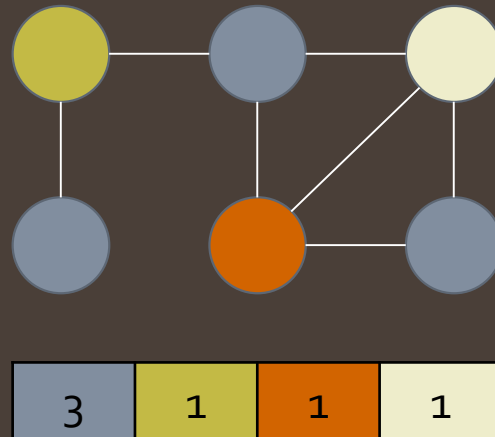
Outline

- Problem Description
- Terms and Concepts
- Data Structure
- Algorithm
- Procedure P

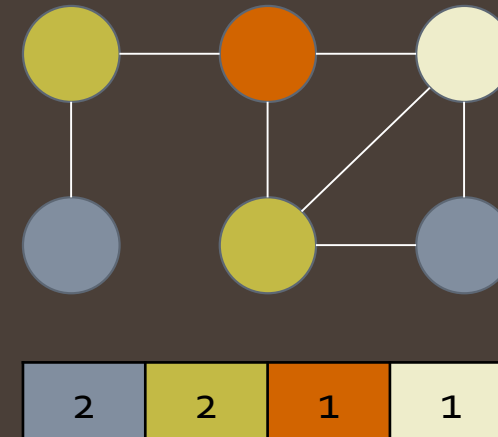
Problem Statement

- Given a graph G with maximal degree r :
 - Color the graph using $r + 1$ colors
 - Sizes of color classes differ by at most 1
- Input: Graph G , nonnegative integer r , s.t. $\Delta(G) \leq r$
 - $|G| = n = s(r + 1)$
- Output: F where F is an equitable coloring of G utilizing the $r + 1$ color classes

Nearly Equitable



Equitable



Terms and Concepts

- Color classes
 - Set of nodes in G colored a certain color W
 - Denoted via capital letters
 - Members of color classes denoted via lower case letters
- Digraph H
 - A directed graph of the $r + 1$ color classes
 - Edge $XY \in H$ if $\exists x \in X$ s.t. x has no neighbors in Y
 - x "witnesses" edge XY
 - V^- is the color class with $s - 1$ nodes
 - V^+ is the color class with $s + 1$ nodes
- Sets
 - A is the set of color classes that can reach V^-
 - B is the complement of A
 - A' is the set of terminal color classes
 - $W \in A'$ if $W \in A$ and deleting W from the digraph does not change the set $A - \{W\}$
 - B' set of color classes in B that can be reached by V^+

Data Structure

- L : $n \times r$ array, $L(v,i)$ is the i th neighbor of v in G and 0 otherwise
- L' : $n \times r$ array, $L'(v,i)$ is the i th neighbor of v in the current iteration of the algorithm
- F : n array, $F(v)$ is the color class to which v belongs
- C : $r + 1$ array, $C(X)$ is the list of vertices belonging to color class X
- H : $(r + 1) \times (r + 1)$ array, $H(X,Y)$ is the number of witnesses to the edge XY in the digraph
- N : $n \times (r + 1)$ array, $N(v,X)$ is the number of neighbors of v in color class X

Algorithm

- Start with empty graph G_0 : equitably-colored
- Iteratively add nodes to the G_{i-1} to make G_i
- If the new node u has a neighbor in its color class, move it to a color class without this conflict
- This step creates V^+ and V^- which we will resolve using the procedure on the next slide
- After applying the procedure, G_i is equitably colored and by induction we can equitably color $G_n = G$

Procedure

- Case 0: path exists from V^+ to V^-
 - Augment along this path, resolving disparity: trivial!
- Assume no path from V^+ to V^-
 - Case 1: A solo edge exists wy
 - Augment along the path WV^- and move y into W
 - Recursively call procedure on $G_i[B - y]$
 - Case 2: $|A'| \geq |B|$
 - Use First-Fit to construct a maximal independent set in B' starting with the nodes in V^+
 - As adding each node to set, mark its solo neighbors in A'
 - If a node w is marked twice by z_1 and z_2 , we have a way to resolve V^-
 - Move w to some class in B
 - Augment along path WV^- and V^+Z
 - Move z_1 to W
 - Recursively call procedure on $G_i[B + W - w' - w + z_1]$

Questions

- Proposed by paper: Does a polynomial time algorithm exist to equitably $(r + 1)$ -color any graph if:
 - $\forall xy \in E(G) d(x) + d(y) \leq 2r + 1$
- What applications can we come up with for equitable coloring?
 - e.g. Scheduling and Load-Balancing

Saving Space By Algebraization

Daniel Lokshtanov, Jesper Nederlof

Presented by Bo Kang, Hao Hu, Ke Chen

April 25, 2014

NP-hard Problems

Many problems are NP-hard, but we still need to solve them anyway, such as:

- SubsetSum
- Knapsack
- Unweighted Steiner Tree
- Traveling Salesman Problem
- Weighted Set Cover

Dynamic Programming

By finding the optimal structure of a NP-hard problem, we can use dynamic programming technique to design the algorithm to solve the hard problem efficiently.

For a SubsetSum problem, given integer $\{e_1, \dots, e_n\}$ and t in binary representation, count the number of subsets $S \subseteq \{1, \dots, n\}$ such that $\sum_{i \in S} e_i = t$.

DP gives an $O(nt)$ time and $O(t)$ space algorithm. In other word, DP-based algorithm solves the subsetsum problem in pseudo-polynomial time and space complexity.

Save space – Why

DP-based algorithm can be seen as a combination of a table and an algorithm computing table entries, usually formalized as a recurrence.

An inherent property of DP algorithms is that they require a relatively big amount of working memory to compute such table entries.

Thus, when the input size of a problem is large enough, it is prone that the DP-based algorithm might run out of memory instead of running out of time.

Save space – How – Think and Ask Question

If a DP-based algorithm can solve a problem, most of time, we only care about the result of such problem instead of intermediate results, which are stored in the table.

The question becomes how to transform general DP-based algorithm into another algorithmic representation so as to get avoid of creating table.

Based on the above question, we have to discern the dynamic programming technique. Then the second question is that what kinds of intrinsic properties dynamic programming have. What is the relationship between the optical structure recurrence and the computed table?

This paper's Idea

Let us focus on the algebraic property of dynamic programming technique. We should see the computing table as the combination of an algebraic structure with some operators to simulate the recurrence.

For an NP-hard problem, which can be solved using the DP-based algorithm. We can reformulate the algorithm as:

Given a set of input elements, we sequentially apply a countable number of operators to yield the final result without storing the intermediate results.

This paper's Idea – Continue

The authors creatively construct two algebraic systems, which form Boolean circuits to simulate the dynamic programming procedure. A boolean circuit is a non-uniform Turing machine.

Figure 4 shows a circuit computing the XOR function on two bits. (Refer to the book *Computational Complexity A Modern Approach* By Sanjeev Arora and Boaz Barak)

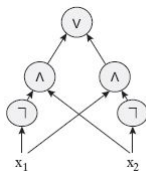


Figure: A circuit C computing the XOR function (i.e, $C(x_1, x_2) = 1 \text{ iff } x_1 \neq x_2$).

Part of formal definition of Circuit: for a set S and binary operators O_1, O_2 on S , a circuit C over $(S; \oplus, \otimes)$ is a directed acyclic graph, in which \oplus means the pointwise addition operator and \otimes means pointwise multiplication operator.

How could transform general DP-based algorithm into boolean circuit with basic operators? The authors propose the convolution operator. Why? Let us come back to the subsetsum problem.

This paper's Idea – Continue

Given integer $\{e_1, \dots, e_n\}$ and t in binary representation, count the number of subsets $S \subseteq \{1, \dots, n\}$ such that $\sum_{i \in S} e_i = t$.

Define $P(x) = (1 + x^{e_1})(1 + x^{e_2}) \dots (1 + x^{e_n})$.

Notice $P(x)$ can be evaluated using n additions and multiplications. Let p_i be the coefficients of $P(x)$, that is $P(x) = \sum_i p_i x^i$. Then p_t is exactly the number of subsets $S \subseteq \{1, \dots, n\}$ such that $\sum_{i \in S} e_i = t$.

Thus, we want to know a coefficient of a polynomial that we can evaluate efficiently. This can be done with the Discrete Fourier Transform. It states that if $p_j = 0$ for every $j > d$, then,

$$p_t = \frac{1}{d} \sum_{j=0}^{d-1} w^{-tj} P(w^j)$$

where w is s.t. $w^d = 1$ and $w^i \neq 1$ for $1 < i < N$.

Subset Sum

$$p_t = \frac{1}{d} \sum_{l=0}^{d-1} \omega^{-tl} P(\omega^l)$$

implies that whenever P can be evaluated in $\mathcal{O}^*(1)$ time, we can find the coefficient p_t in $\mathcal{O}^*(d)$ time and (1) space.

$$A[i, j] = \begin{cases} 0 & \text{if } i = 1, e_1 \neq j, \text{ and } e_1 \neq 0 \\ 1 & \text{if } i = 1 \text{ and } (e_1 = j \text{ or } j = 0) \\ A[i-1, j] + A[i-1, j - e_i] & \text{if } i > 1 \end{cases}$$

Define $A[i](x) = \sum_{j=0}^{d-1} A[i, j] x^j$, then

$$A[i](x) = \begin{cases} x^{e_1} + 1 & \text{if } i = 1 \\ A[i-1](x) + A[i-1](x)x^{e_i} & \text{if } i > 1 \end{cases}$$

Figure: This slide comes from the authors' slide

Convolution

- For which recurrences do we obtain a recurrence for an easy to compute polynomial?
- This is exactly when the original recurrence only uses addition and multiplication of polynomials in coefficient form, which is:

Definition (Convolution)

- Define the operator \otimes on vectors of size N to be

$$\mathbf{A} \otimes \mathbf{B} = \left(\sum_{i+j=k} A[i]B[j] \right)_{0 \leq k < N}$$

Figure: This slide comes from the authors' slide

The problem can be solved in $O(n^3 \log t)$ time and $O(n^2)$ space.

This paper's Idea – Continue

Definition

Let $(R; +, \cdot)$ be a field (or ring) and (G, \circ) be a (semi-)group. Then the (semi-)group ring/algebra $R[G]$ is defined to be all functions $f : G \rightarrow R$ with operations

$$(f + g)(x) = f(x) + g(x) \quad (f * g)(x) = \sum_{y \circ z = x} f(y)g(z)$$

- Elements of $R[G]$ should be seen as "generalized polynomials". Our previous recurrence should be interpreted as an expression over the group algebra $F[\mathbb{Z}_+^{nt}]$ (and we used $F = \mathbb{C}$).
- The question is, for which other kind of (semi-)groups can we use the same idea to save space?
- It appears we can embed any $a \in R[G]$ in the group of $|G| \times |G|$ matrices by defining $A[x, y] = \sum_{x \circ z = y} a(x)$.

Figure: This slide comes from the authors' slide

This paper's Idea – Continue

Beside DFT linear transform from convolution operator to \otimes pointwise multiplication operator, the authors also illustrate another approach by using mobius inversion to linear transform the subset convolution problem.

DFT based linear transformation can reduce the DP space complexity when the computable table is indexed by integers. Such mobius inversion linear transformation can further reduce the DP space complexity when the computable table is indexed by subsets.

By using the combination of DFT and mobius inversion, optimization problems such as TSP can be further space efficiently solved. The authors claim that the traveling salesman problem can be solved in $O(2^n d)$ time and polynomial space.

Conclusion

The authors present an interesting algebraic-oriented numerical analysis approach to analyze the space complexity of dynamic programming technique.

Open Questions

1. Are there more algebraic properties of DP algorithms that imply the possibility of saving space?
2. For a DP-based algorithm, which runs in pseudo-polynomial time complexity, what is the lower bound of its space complexity?

Thanks



Minimum Manhattan Network is NP-Complete

by Francis Y.L. Chin, Zeyu Guo, and He Sun

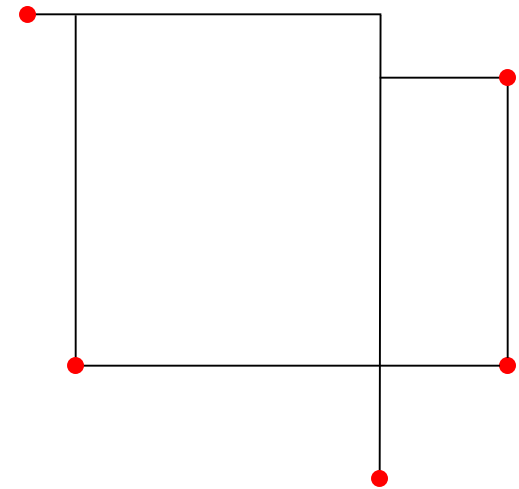
presented by:

R. Barmaki, A. Jackson, E.Havvaei, R. Coaguila

COT 6410 - Spring 2014
Fast Forward Presentation

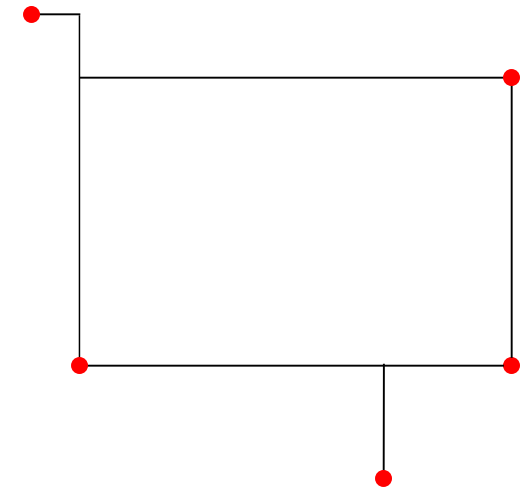
Minimum Manhattan Network

- A *Manhattan network* on a set T of points in \mathbb{R}^2 is a graph $G = (V, E)$ with the property that all its edges are vertical or horizontal line segments and all pairs of points in T are connected by a Manhattan path in E

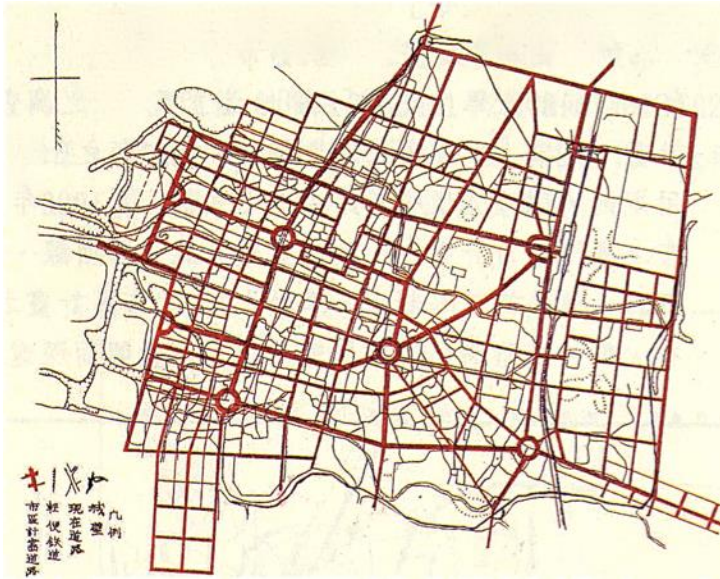


Minimum Manhattan Network

- A *Manhattan network* on a set T of points in \mathbb{R}^2 is a graph $G = (V, E)$ with the property that all its edges are vertical or horizontal line segments and all pairs of points in T are connected by a Manhattan path in E
- The *minimum Manhattan network* (MMN) is the one with the smallest total length.



Applications



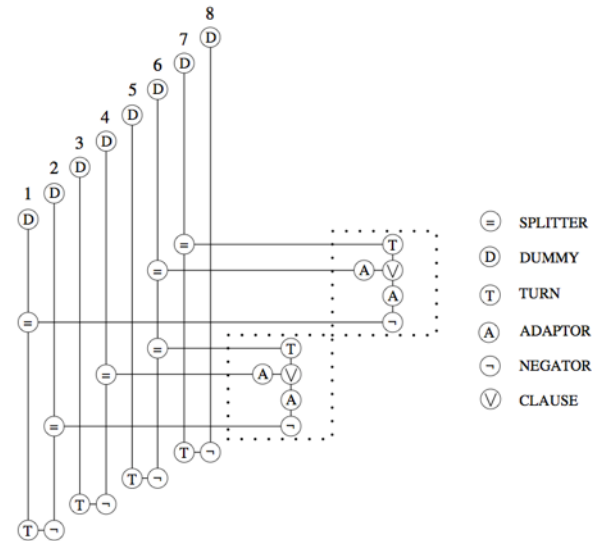
Urban Planning



VLSI

Our goal

$$(x_1 \vee \neg x_3 \vee x_4) \wedge (\neg x_1 \vee \neg x_2 \vee \neg x_3)$$

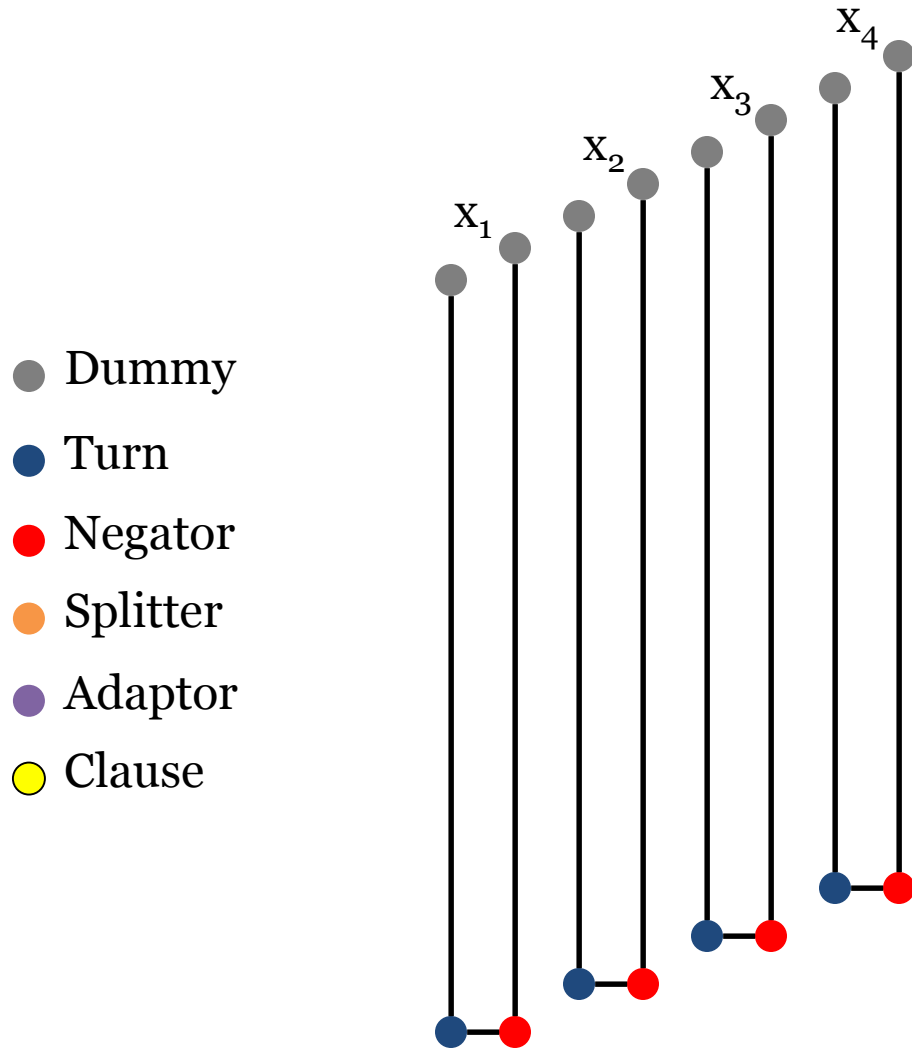


3-SAT

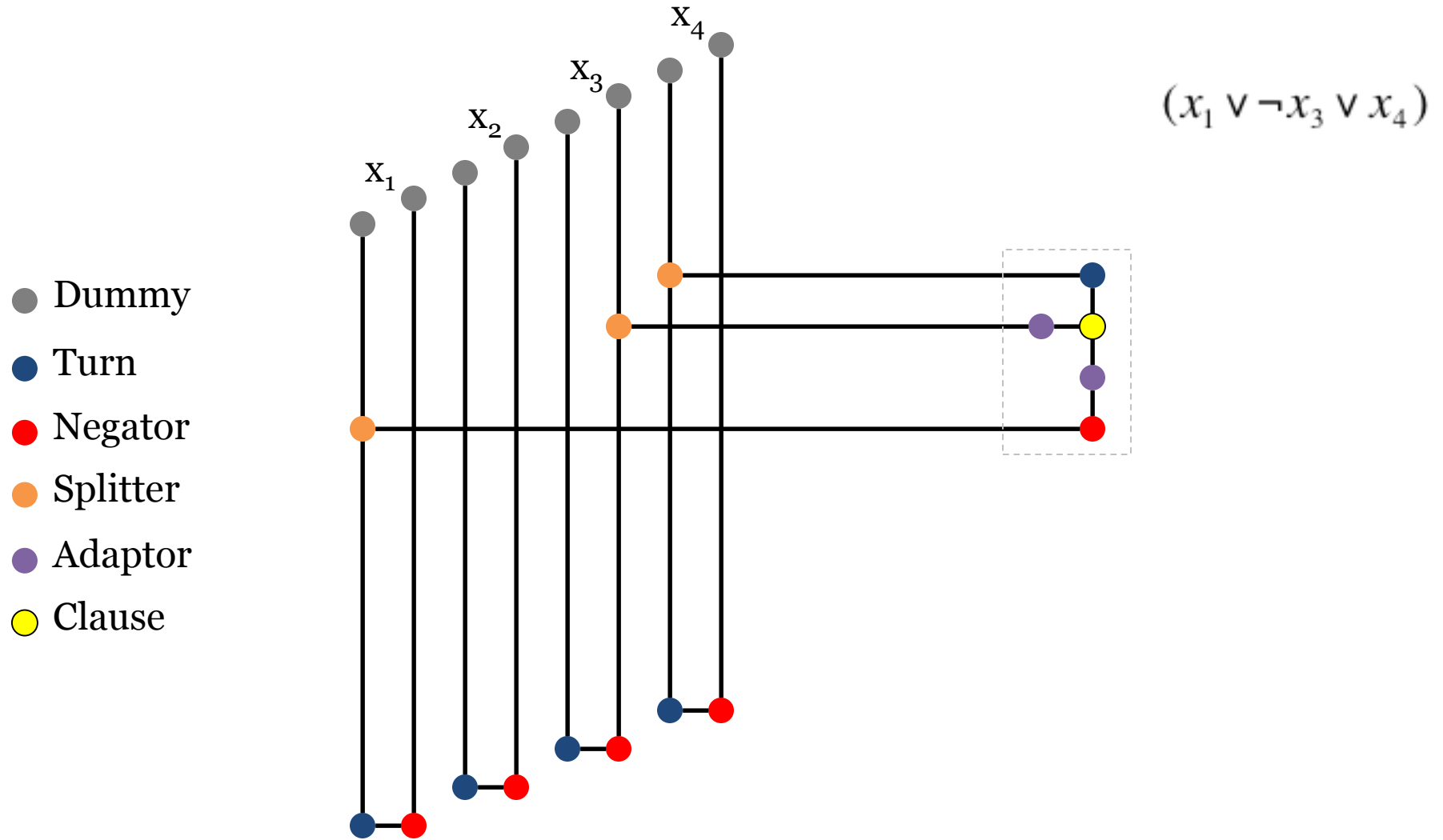
≤

Minimum Manhattan Network

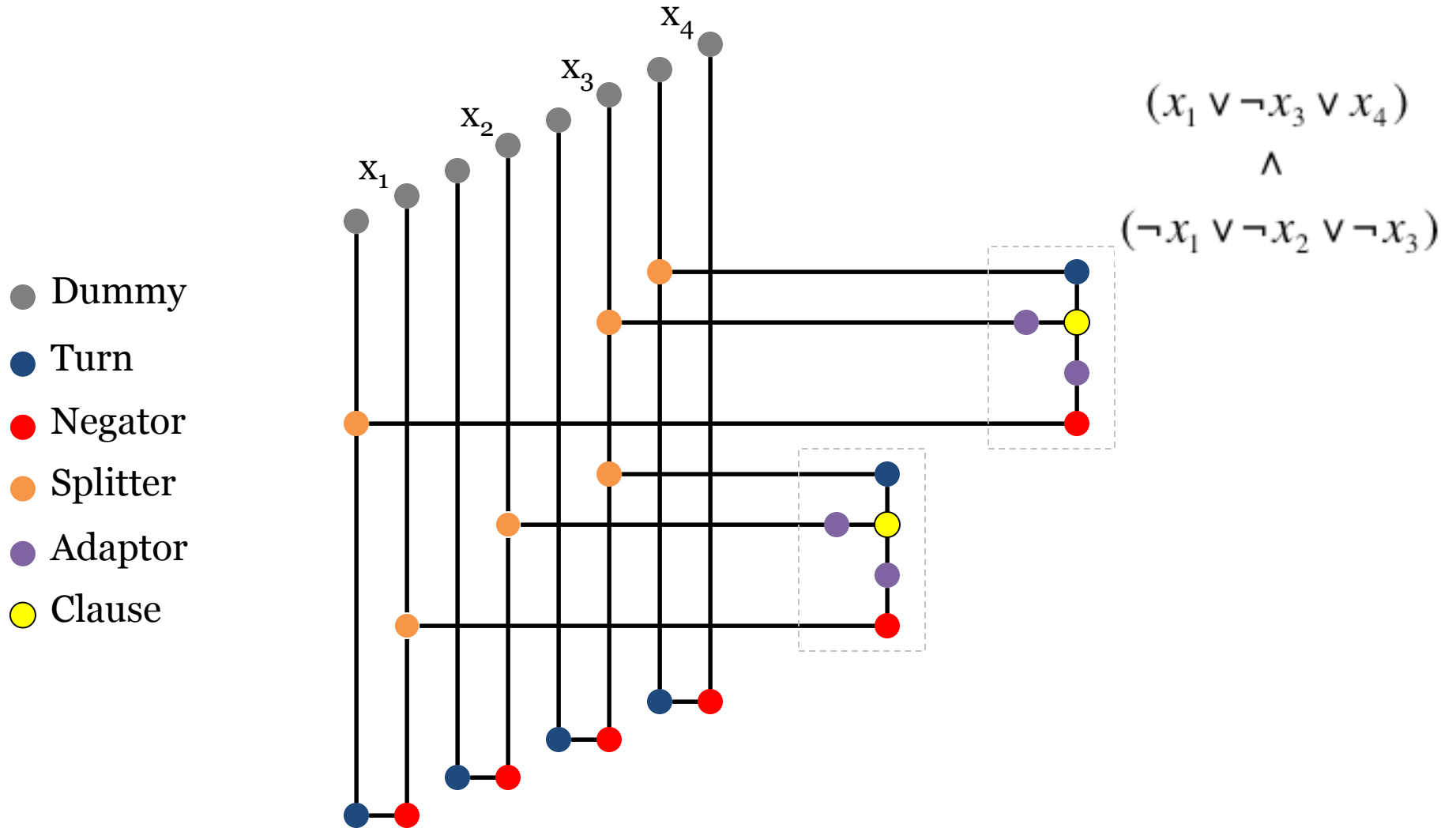
Reduction Summary



Reduction Summary



Reduction Summary



Conclusion

- 3SAT can be reduced to MMN.
- MMN is strongly NP-complete, and there does not exist an FPTAS for this problem unless $P = NP$.

Open Problems

- No approximation algorithms for **MMN in Higher dimensions** proposed with constant factor.
 - Best solution has the approximation factor of $O(\log^{d+1} n)$ where d is the dimension (DAS et al., 2012)

Open Problems

- No approximation algorithms for **MMN in Higher dimensions** proposed with constant factor.
 - Best solution has the approximation factor of $O(\log^{d+1} n)$ where d is the dimension (DAS et al., 2012)
- Is there any constant factor approximation algorithm for 2D-MMN?
 - Open Problem

Questions

- Question 1: Can you think of further application for which the MMN problem is relevant?
- Answer: Transport and communication networks or gene alignment in computational biology
- Question 2: Which Reduction method was used in this proof?
- Answer: Construction

Thanks

Testing Properties of Sparse Images

Soumyabrata Dey

Nasim Souly

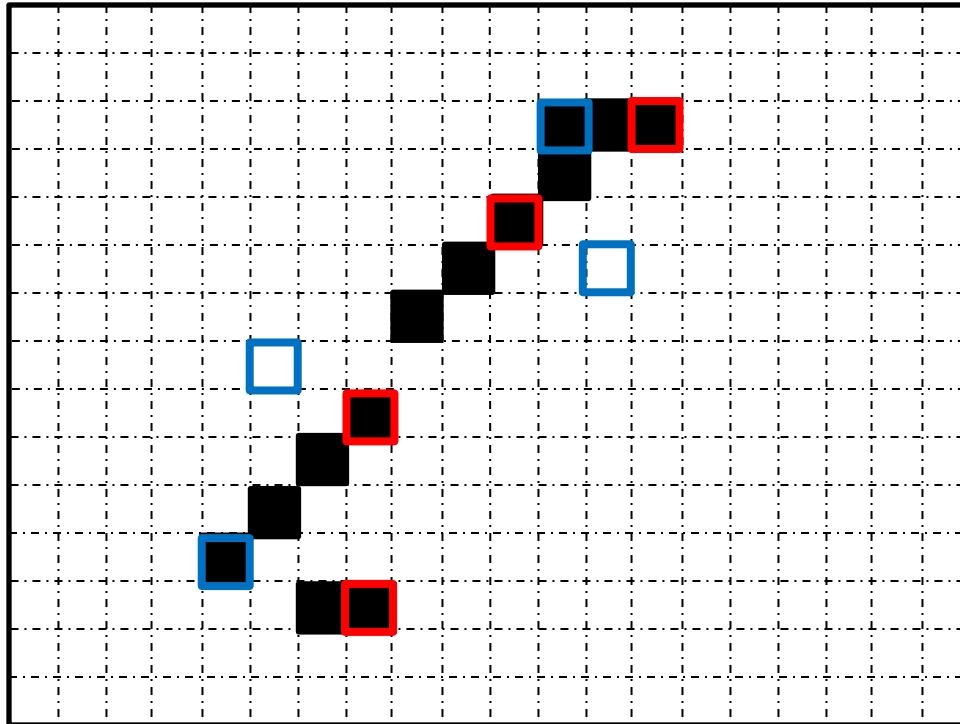
Dong Zhang

Waqas Sultani

Introduction

- M : a binary $\{0,1\}$ sparse image represented by an $n \times n$ matrix
- $w = w(M)$ number of 1-pixels
- Problems: given a distance parameter ϵ
 - Accept with high constant probability if M has the property P
 - Reject with high constant probability if more than ϵw pixels must be changed in M for it to have the property P .
- The algorithms presented in this paper require a constant factor approximation \hat{w} of w .

Image access



Uniform sampling of 1-pixels □

Random probing of pixels □

Images are need to store in an efficient data structure to help sampling of 1-pixel

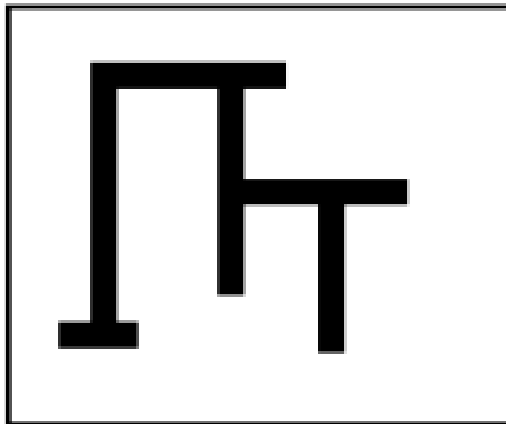
Properties to Test

- Connectivity
- A line
- Convexity
- Monotonicity

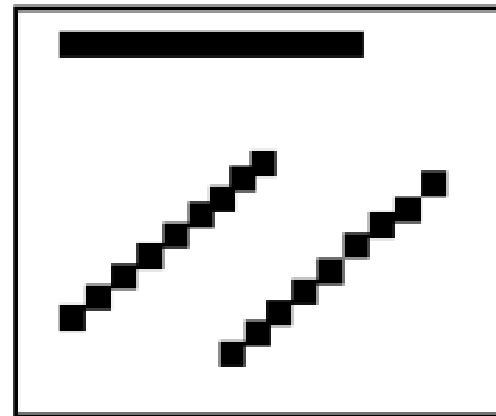
Connectivity

- Definition

- If the underlying graph induced by the neighborhood relation between 1-pixels is connected



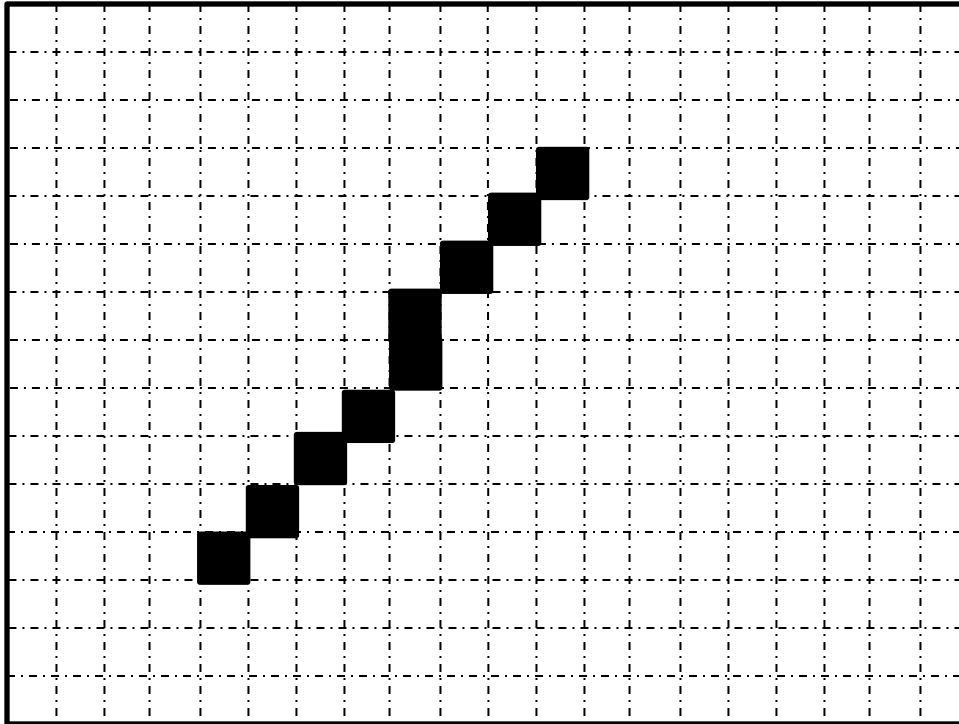
Connected



Not Connected

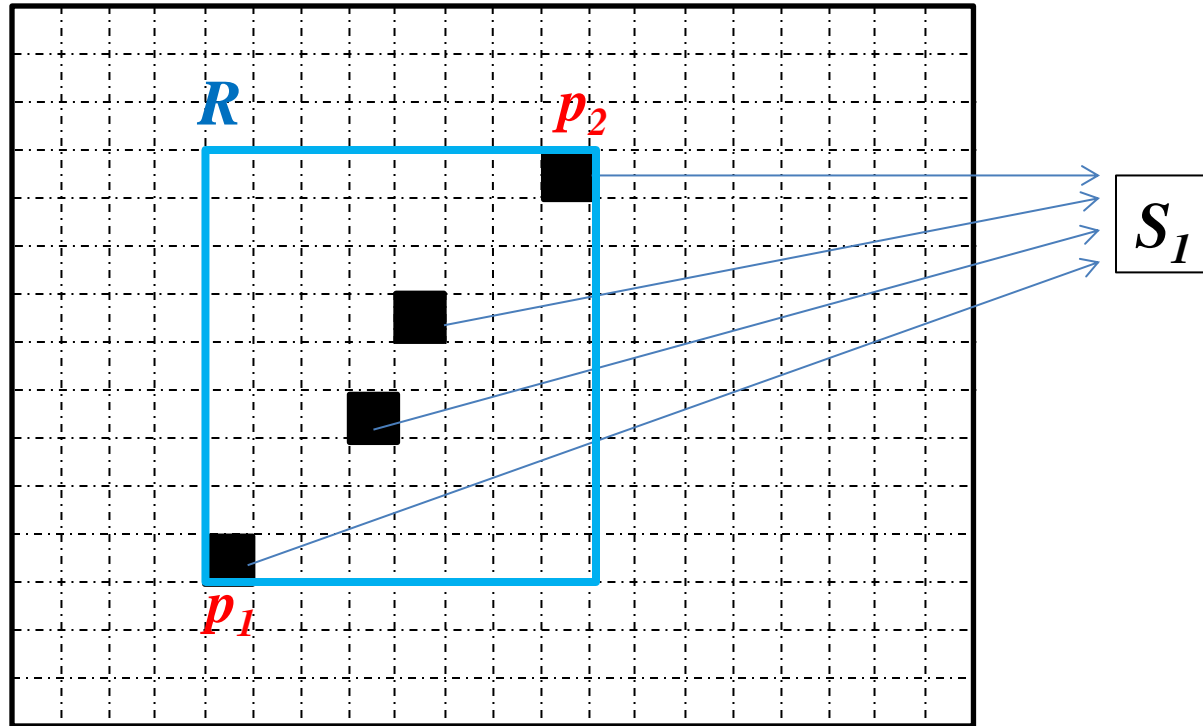
$$\tilde{O}(\min\{w(M)^{\frac{1}{2}}, n^2/w(M)\} \cdot \epsilon^{-2})$$

Line Imprint



Line Imprint: A line segment on real plane intersects a set of pixels of an image.

Line Imprint

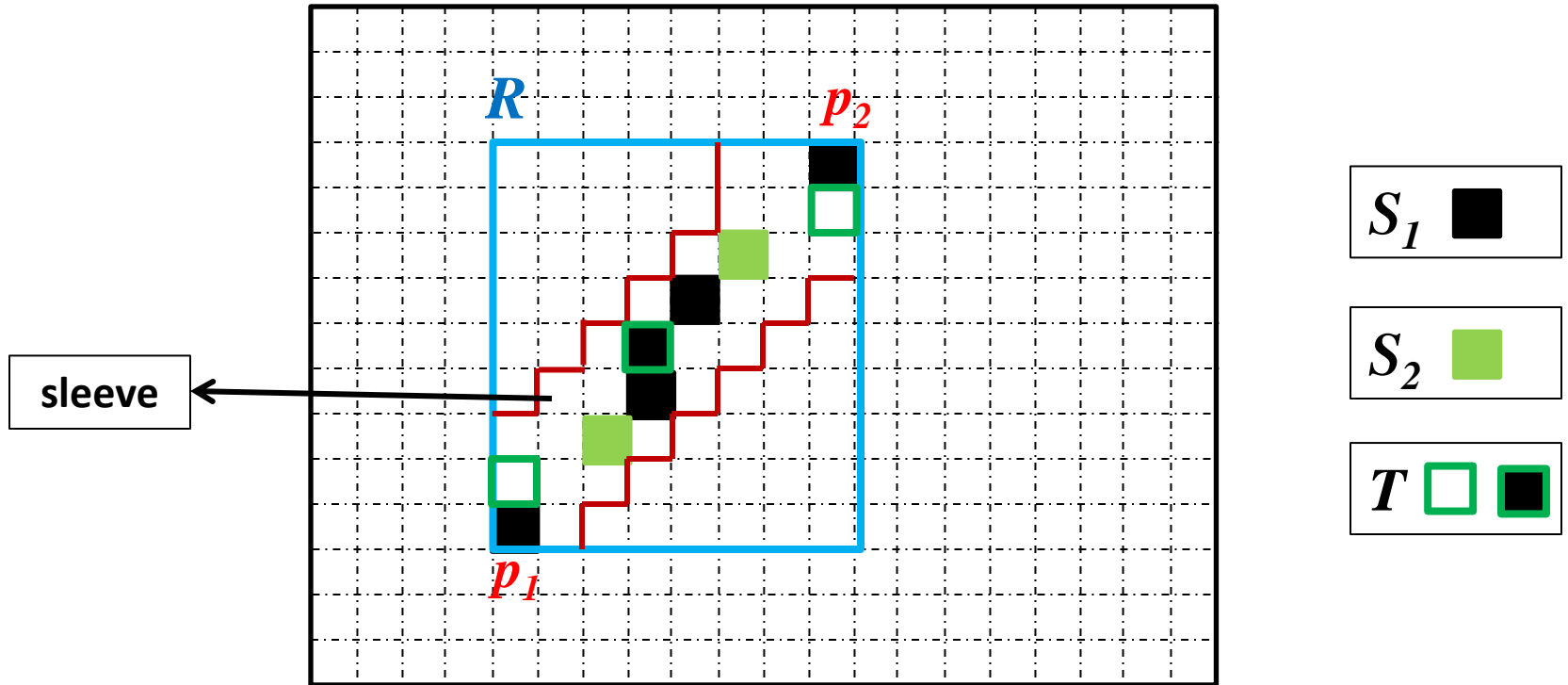


Select uniformly sampled 1-pixel set S_1

Farthest pixel pair p_1 and p_2

R is the minimal rectangle contains all the pixels of S_1

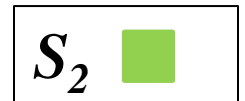
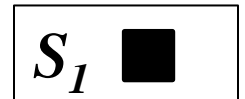
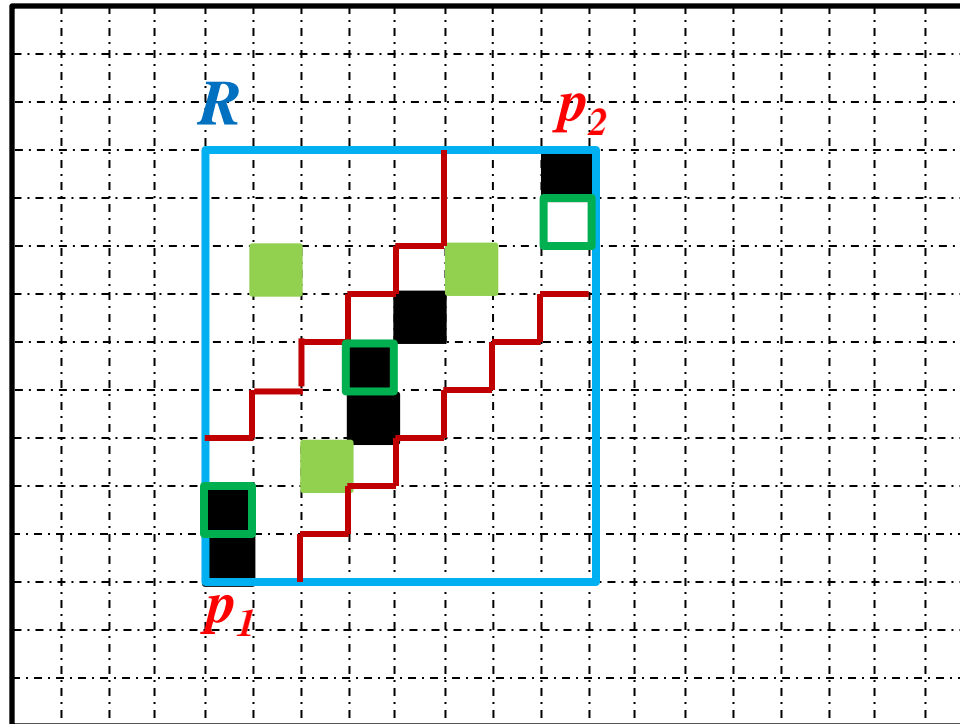
Line Imprint



Select uniformly sampled 1-pixel set S_2

Select uniformly sampled pixel set T from sleeve and query those pixels

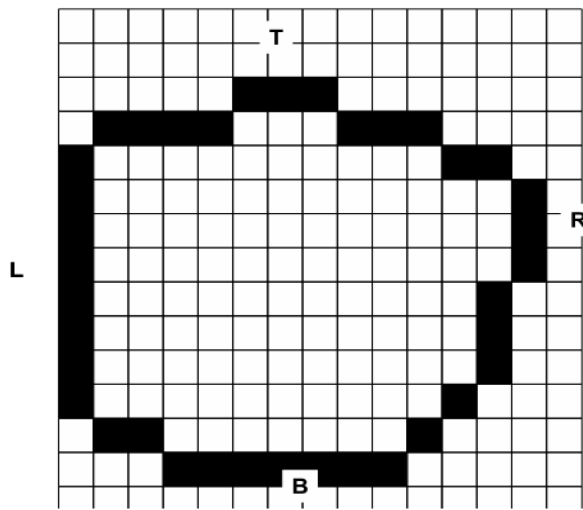
Line Imprint



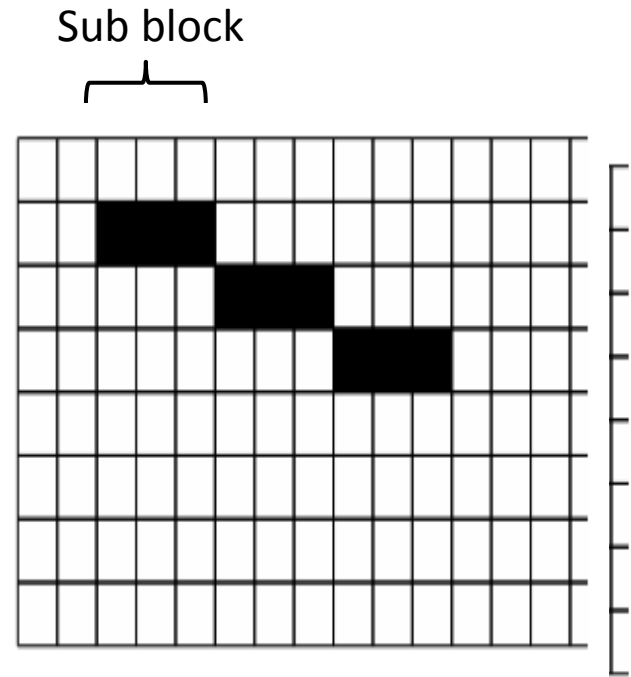
Cases when algorithm rejects

Algorithm complexity is $O(\log(1/\epsilon)/\epsilon)$

Testing Convexity



Convex Shape

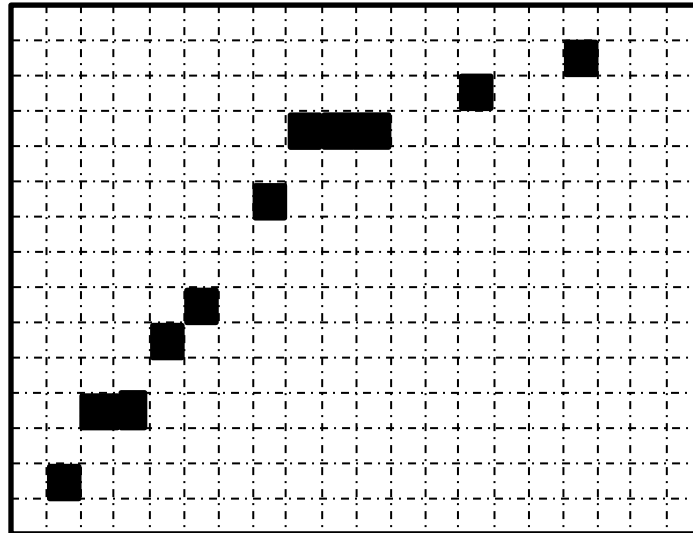


Horizontal Block

$$\text{Complexity} = \tilde{O}(w(M)^{\frac{1}{4}} \cdot \epsilon^{-2})$$

Testing Monotonicity

- If for every two entries (i_1, j_1) and (i_2, j_2) in a matrix, where $M[i_1, j_1]=1$ and $M[i_2, j_2]=1$, if $i_1 \leq i_2$ then $j_1 \leq j_2$, we will say that matrix M is *monotone*.



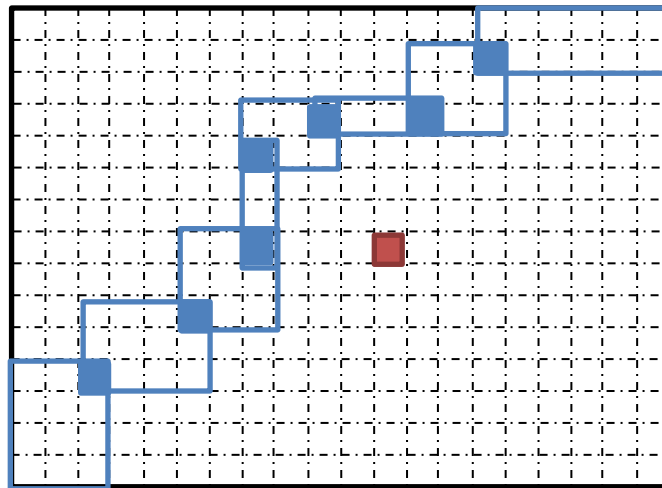
(i_0, j_0)

Violation Graph

- *Violation graph* is defined as $G_{viol}(M) = (V(M), E_{viol}(M))$ where $V(M) = \{(i, j): M[i, j] = 1\}$ and $E_{viol}(M)$ consists of all violating pairs in M .
- The distance of M to being monotone is the size of minimum vertex cover of $G_{viol}(M)$.
- The algorithm will reject with high constant probability, if M is ϵ -far from monotone.
- Complexity $\Theta((w(M))^{1/2})$
- Just sampling would not reduce the complexity
- Queries should be performed

Testing Monotonicity Algorithm

- Take a sample that with high probability
 - either contains evidence that the matrix is not monotone
 - or it can be used to determine a set of sub-matrices with the following properties
- complexity $\Omega(\min\{w(M)^{1/2}, n^{2/3} / w(M)^{1/3}\})$



Questions

- Combining a testing algorithm for connectivity and a testing algorithm for monotonicity, we get a testing algorithm for the combined property. Is there a more efficient algorithm for the combined property?
- Can you think of the application of this paper in digit recognition and Image matching?

Statistical Model Checking

Amit Goel and Taranjeet Singh Bhatia

04-29-2014

Overview

Objective

Introduction

Stochastic Model Checking

Statistical Model Checking

Future

Statistical Model
Checking

Amit Goel and
Taranjeet Singh
Bhatia

Objective

Introduction

Stochastic Model
Checking

Statistical Model
Checking

Future

Objective

Statistical Model
Checking

Amit Goel and
Taranjeet Singh
Bhatia

3 Objective

Introduction

Stochastic Model
Checking

Statistical Model
Checking

Future

This paper surveys the state of art statistical model checking and outlines an overview of some of common approaches.

Why must we verify?

Testing can only show the presence of errors, not their absence.

Statistical Model
Checking

Amit Goel and
Taranjeet Singh
Bhatia

Objective

4 Introduction

Stochastic Model
Checking

Statistical Model
Checking

Future

Why must we verify?

Statistical Model
Checking

Amit Goel and
Taranjeet Singh
Bhatia

Objective

4 Introduction

Stochastic Model
Checking

Statistical Model
Checking

Future

Testing can only show the presence of errors, not their absence.

To rule out errors one must consider all possible executions often not feasible mechanically!

So What?

- ▶ True, There are many large scale computer systems which can survive even after failure
 - ▶ Online banking, electronic commerce
 - ▶ Online libraries, Information services
 - ▶ Mobile phone networks

Statistical Model
Checking

Amit Goel and
Taranjeet Singh
Bhatia

Objective

5 Introduction

Stochastic Model
Checking

Statistical Model
Checking

Future

So What?

- ▶ True, There are many large scale computer systems which can survive even after failure
 - ▶ Online banking, electronic commerce
 - ▶ Online libraries, Information services
 - ▶ Mobile phone networks
- ▶ Still, there are many application domain with far greater complexities and higher expectations
 - ▶ Automotive
 - ▶ Medical sensors
 - ▶ Intelligent buildings and spaces

Statistical Model
Checking

Amit Goel and
Taranjeet Singh
Bhatia

Objective

5 Introduction

Stochastic Model
Checking

Statistical Model
Checking

Future

So What?

- ▶ True, There are many large scale computer systems which can survive even after failure
 - ▶ Online banking, electronic commerce
 - ▶ Online libraries, Information services
 - ▶ Mobile phone networks
- ▶ Still, there are many application domain with far greater complexities and higher expectations
 - ▶ Automotive
 - ▶ Medical sensors
 - ▶ Intelligent buildings and spaces
- ▶ And Learning from mistakes costly . . .

Statistical Model
Checking

Amit Goel and
Taranjeet Singh
Bhatia

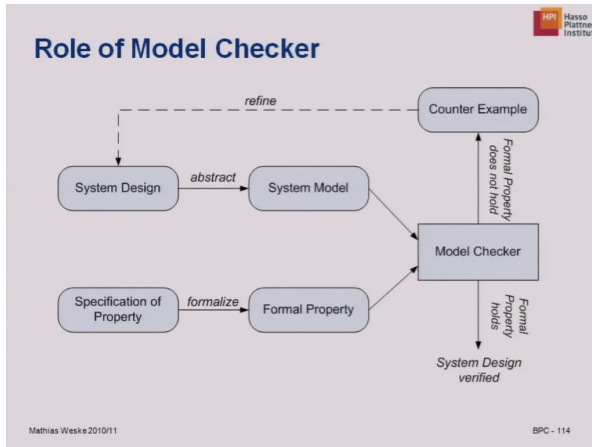
Objective

5 Introduction

Stochastic Model
Checking

Statistical Model
Checking

Future



Role of model checking

- ▶ Automated techniques for the assurance of
 - ▶ safety
 - ▶ security
 - ▶ performance
 - ▶ dependability

Statistical Model
Checking

Amit Goel and
Taranjeet Singh
Bhatia

Objective

7 Introduction

Stochastic Model
Checking

Statistical Model
Checking

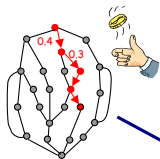
Future

Role of model checking

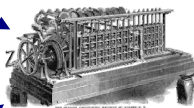
- ▶ Automated techniques for the assurance of
 - ▶ safety
 - ▶ security
 - ▶ performance
 - ▶ dependability
- ▶ This paper focus on Stochastic Model checking
 - ▶ to capture probability and resource usage
 - ▶ range of quantitative analyses

Stochastic Model Checking ...

in a nutshell



Probabilistic model



Probabilistic
Model Checker



or



or

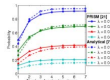
The probability

send $\rightarrow P_{\geq p}$ [F deliver]

Probabilistic temporal
logic specification

SFM-07:PE

State 5: 0.6789
State 6: 0.9789
State 7: 1.0
...
State 12: 0
State 13: 0.1245



12

Two Approaches

- ▶ Numerical Approach
 - ▶ These approaches compute the exact measure of paths satisfying relevant subformulas

Statistical Model
Checking

Amit Goel and
Taranjeet Singh
Bhatia

Objective

Introduction

9 Stochastic Model
Checking

Statistical Model
Checking

Future

Two Approaches

- ▶ Numerical Approach

- ▶ These approaches compute the exact measure of paths satisfying relevant subformulas
- ▶ works only for special system that have certain structural properties

Statistical Model
Checking

Amit Goel and
Taranjeet Singh
Bhatia

Objective

Introduction

9 Stochastic Model
Checking

Statistical Model
Checking

Future

Two Approaches

▶ Numerical Approach

- ▶ These approaches compute the exact measure of paths satisfying relevant subformulas
- ▶ works only for special system that have certain structural properties
- ▶ requires lot of time and space

Statistical Model
Checking

Amit Goel and
Taranjeet Singh
Bhatia

Objective

Introduction

9 Stochastic Model
Checking

Statistical Model
Checking

Future

Two Approaches

▶ Numerical Approach

- ▶ These approaches compute the exact measure of paths satisfying relevant subformulas
- ▶ works only for special system that have certain structural properties
- ▶ requires lot of time and space
- ▶ not popular among engineers because of logics are extensions of classical temporal logics

Statistical Model
Checking

Amit Goel and
Taranjeet Singh
Bhatia

Objective

Introduction

9 Stochastic Model
Checking

Statistical Model
Checking

Future

Two Approaches

- ▶ Simulation based approach
 - ▶ To deduce whether or not the system satisfies the property by observing some of its execution with a monitoring procedure and use hypothesis testing to infer whether the sample provides a statistical evidence for the satisfaction or violation of the specification

Statistical Model
Checking

Amit Goel and
Taranjeet Singh
Bhatia

Objective

Introduction

10 Stochastic Model
Checking

Statistical Model
Checking

Future

Two Approaches

- ▶ Simulation based approach
 - ▶ To deduce whether or not the system satisfies the property by observing some of its execution with a monitoring procedure and use hypothesis testing to infer whether the sample provides a statistical evidence for the satisfaction or violation of the specification
 - ▶ Statistical model checking is the most popular simulation based approach that we over-viewed

Statistical Model
Checking

Amit Goel and
Taranjeet Singh
Bhatia

Objective

Introduction

10 Stochastic Model
Checking

Statistical Model
Checking

Future

Advantages

- ▶ These algorithm only require that the system be executable. Thus, can be applied to larger class of systems including black-box systems and infinite systems

Statistical Model
Checking

Amit Goel and
Taranjeet Singh
Bhatia

Objective

Introduction

Stochastic Model
Checking

11 Statistical Model
Checking

Future

Advantages

- ▶ These algorithm only require that the system be executable. Thus, can be applied to larger class of systems including black-box systems and infinite systems
- ▶ take lesser memory and time to execute

Statistical Model
Checking

Amit Goel and
Taranjeet Singh
Bhatia

Objective

Introduction

Stochastic Model
Checking

11 Statistical Model
Checking

Future

Advantages

- ▶ These algorithm only require that the system be executable. Thus, can be applied to larger class of systems including black-box systems and infinite systems
- ▶ take lesser memory and time to execute
- ▶ allow us to bound the probability of making an error

Statistical Model
Checking

Amit Goel and
Taranjeet Singh
Bhatia

Objective

Introduction

Stochastic Model
Checking

11 Statistical Model
Checking

Future

Advantages

- ▶ These algorithm only require that the system be executable. Thus, can be applied to larger class of systems including black-box systems and infinite systems
- ▶ take lesser memory and time to execute
- ▶ allow us to bound the probability of making an error
- ▶ easy parallelization

Statistical Model
Checking

Amit Goel and
Taranjeet Singh
Bhatia

Objective

Introduction

Stochastic Model
Checking

11 Statistical Model
Checking

Future

Disadvantages

- ▶ only provides probabilistic guarantees about the correctness of solutions

Statistical Model
Checking

Amit Goel and
Taranjeet Singh
Bhatia

Objective

Introduction

Stochastic Model
Checking

12 Statistical Model
Checking

Future

Disadvantages

- ▶ only provides probabilistic guarantees about the correctness of solutions
- ▶ only work for pure stochastic systems

Statistical Model
Checking

Amit Goel and
Taranjeet Singh
Bhatia

Objective

Introduction

Stochastic Model
Checking

12 Statistical Model
Checking

Future

Disadvantages

- ▶ only provides probabilistic guarantees about the correctness of solutions
- ▶ only work for pure stochastic systems
- ▶ correctness of model checker is proportional to sample size

Statistical Model
Checking

Amit Goel and
Taranjeet Singh
Bhatia

Objective

Introduction

Stochastic Model
Checking

12 Statistical Model
Checking

Future

Open Questions

- ▶ Using statistical model checking for verification of Multi-Core systems

Statistical Model
Checking

Amit Goel and
Taranjeet Singh
Bhatia

Objective

Introduction

Stochastic Model
Checking

Statistical Model
Checking

13 Future

Open Questions

- ▶ Using statistical model checking for verification of Multi-Core systems
- ▶ Using statistical model checking for verification of systems combining stochastic and non-deterministic aspects

Statistical Model
Checking

Amit Goel and
Taranjeet Singh
Bhatia

Objective

Introduction

Stochastic Model
Checking

Statistical Model
Checking

13 Future

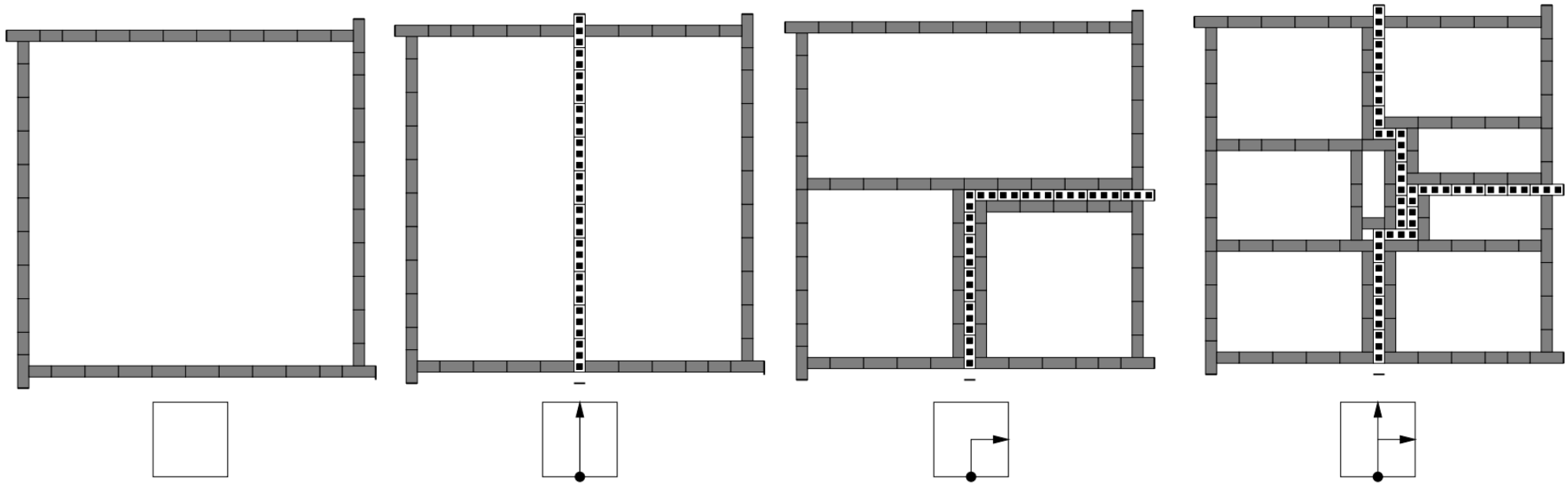
Rush Hour is PSPACE-
complete, or "Why you
should generously tip
parking lot attendants"

ANDREW HARN, NICHOLAS BUELICH, TRAVIS MEADE

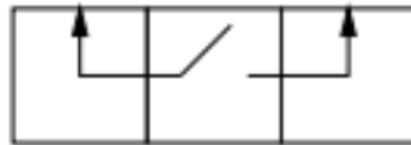
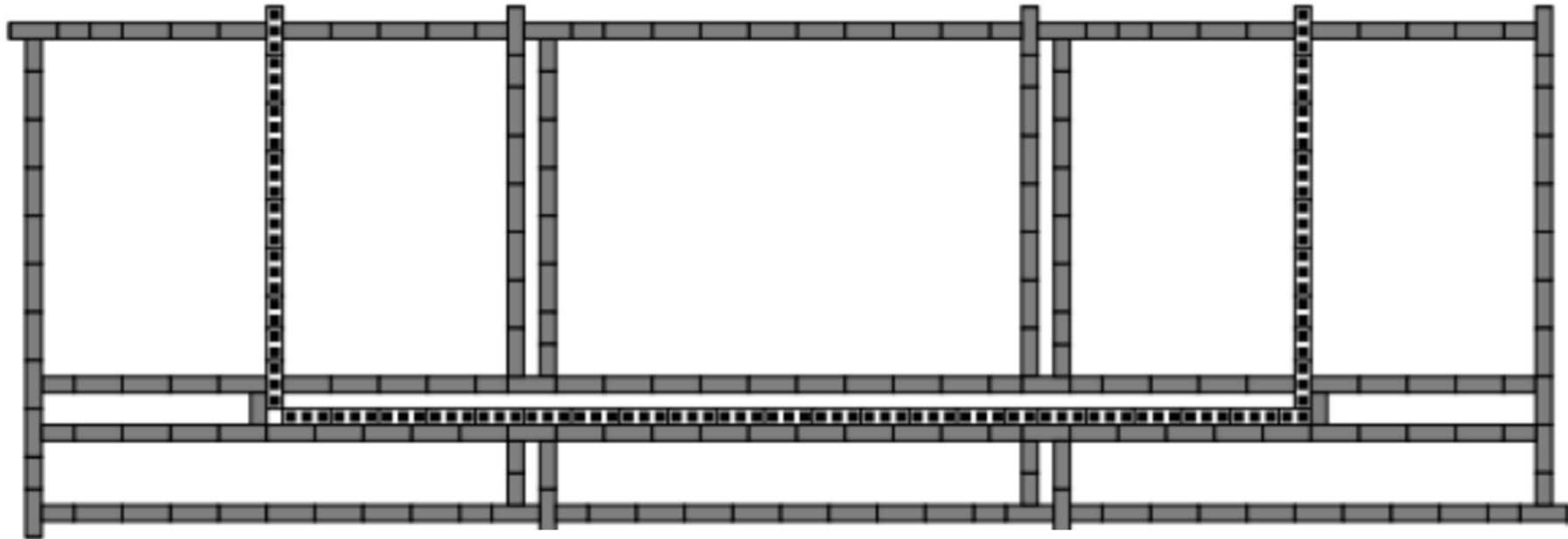
Generalized Rush Hour



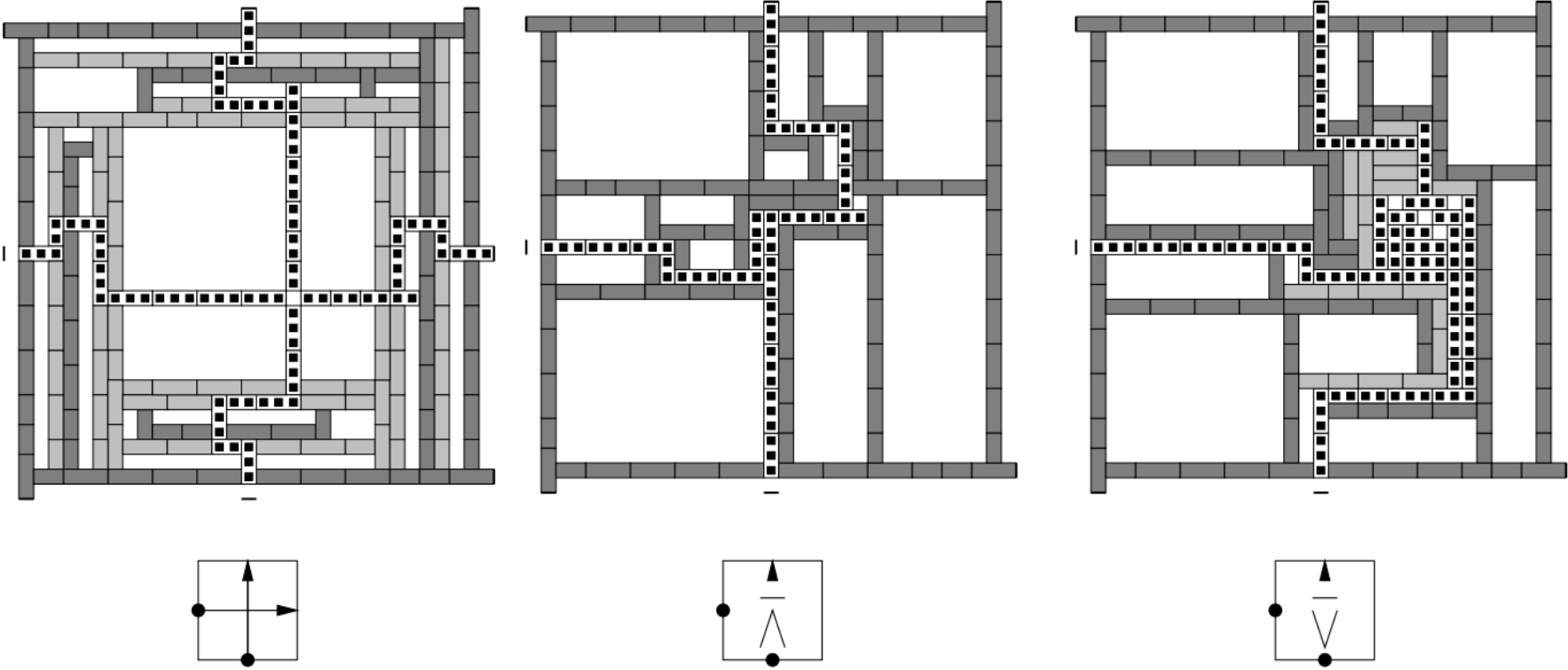
Empty, Pass, Turn, and Split Blocks



Switch



Intersection, Conjunctive Gate, and Disjunctive Gate



Constructing a NTM for GRH

The GRH state can be represented by a string of bits that is $\text{num_cars} * \log_2(1 + \text{Maximum}(N, M))$ in length, representing the current positions of the cars and their orientations

Given a fixed board size, the size of a GRH state is a constant factor times the number of cars.

The tape of our NTM will have encoded on it the current state of our GRH.

If the state is a winning state, the NTM will halt.

Otherwise it will transition to one of at most $2 * \text{cars}$ possible states (a car can move left/right or up/down one space) from the current configuration.

The NTM will make the minimum number of moves to solve the instance of GRH.

Thus our total space is only based upon our current state, thus a NTM can solve GRH in PSPACE.

Question

We have shown that Generalized Rush Hour is NP-Hard and in PSPACE. Why do we not say it is in NP?

Answer

We have shown that 3SAT can be polynomial reduced to GRH, but we did not reduce GRH to any known NP problem, meaning there is no proof that GRH is NP-Easy or NP-Equivalent (in NP).

Reference

Gary William Flake, Eric B. Baum: "Rush Hour is PSPACE-complete, or "Why you should generously tip parking lot attendants""

Paper Presentation

"Short lists with short programs in short time"

by Gurkan Solmaz and Sharif Hassan
COT 6410 - Computational Complexity - Spring 2014

Department of Electrical Engineering and Computer Science
University of Central Florida - Orlando, FL

April 25, 2014

Outline

1 Introduction

Outline

- 1 Introduction
- 2 On-line Matching Problem

Outline

- 1 Introduction
- 2 On-line Matching Problem
- 3 Construction of the graphs

Outline

- 1 Introduction
- 2 On-line Matching Problem
- 3 Construction of the graphs
- 4 References and Q/As

Introduction

• Definitions

- ▶ Kolmogorov complexity ($C(x)$): Length of the shortest program p computing x
- ▶ Finding the shortest program p (i.e. $|p| = C(x)$) is uncomputable
- ▶ Given U : universal TM, $U(p) = x$ (p is a program for x)
- ▶ p is a c -short program if $|p| = C(x) + c$
- ▶ Function f is list approximator if $\forall x, f(x)$ is the list containing the c -short program

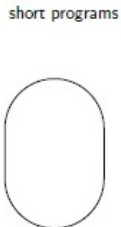
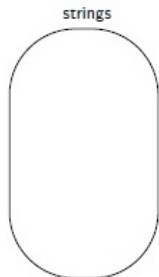
• Results

- ▶ (Thm 1) \exists computable f including $O(1)$ -short programs with size $O(n^2)$
- ▶ (Thm 2) \exists computable f in polynomial time including $O(\log n)$ -short p
- ▶ (Thm 3) Size is $\Omega(n^2/c^2)$ for any f if it is a list approximator

On-line matching problem

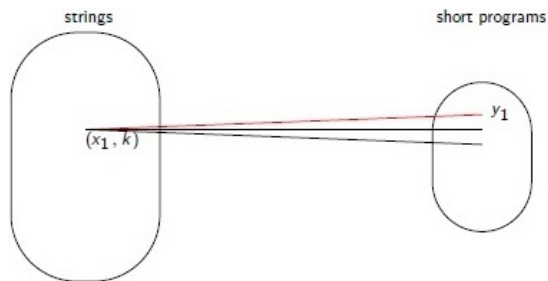
- Two equivalent problems:
 - ▶ P1. The problem of constructing list approximators
 - ▶ P2. OM: Constructing families of bipartite graphs of a certain type
- Bipartite graph: $G = (L, R, E \subset L \times R)$
- L and R consist of binary strings
- Off-line matching with overhead $c(n)$
 - ▶ For every set S of pairs (x, k)
 - ▶ $\forall x \in L$ and $k \in R, \exists (x, k) \in S$ pair $|p(x, k)| \leq k + c(|x|)$
 - ▶ $p(x_1, k_1) \neq p(x_2, k_2)$, where $x_1 \neq x_2$
- A bipartite graph has on-line matching if requests (x, k) appear one by one
 - ▶ We find $p(x, k)$ before the next request comes
 - ▶ A matching pair cannot be unmatched after assignment

Construction of the graphs



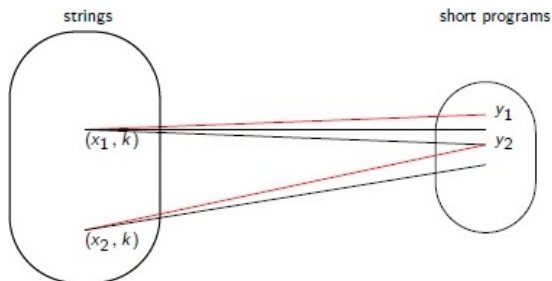
- Match $x \in L(\text{strings})$ with a free node $y \in R(\text{programs})$ if $|y| \leq (k + c)$

Construction of the graphs



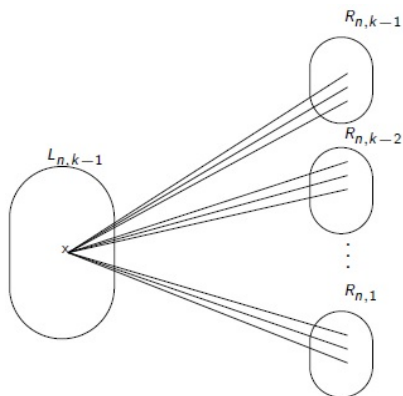
- $k \leq |x|$, number of requests $\leq 2^k, \forall k$

Construction of the graphs



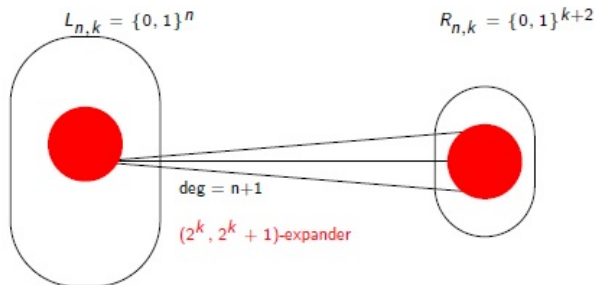
- Constraint: Requests must be responded online, before the next request comes

Polynomial size lists



- \exists polynomial size list approximator for $O(\log n)$ -short programs
- $L = \{0, 1\}^n, R = \{0, 1\}^{l-O(\log n)}, D(x) = \text{poly}(n)$

Basic building brick



- Satisfying all (x, k) requests for $|x| = n$ and a fixed k
- $D(x) = (k - 1)(n + 1) \Rightarrow O(n^2)$, overhead $c = 1$ (Theorem 1)

References

- (Main reference) B. Bauwens, A. Makhlin, N. Vereshchagin, M. Zimand, "Short Lists with Short Programs in Short Time", IEEE Conference on Computational Complexity (CCC), pp.98,108, 5-7 June 2013.
- Marius Zimand, "Short lists with short programs in short time - a short proof", CoRR, 2013.
- Jason Teutsch, "Short lists for shortest descriptions in short time", CoRR, 2012.
- L. I. Ming and P. Vitnyi. "Kolmogorov complexity and its applications." Handbook of Theoretical Computer Science: Algorithms and complexity 1 (1990): 187.
- Xixuan Feng. "Online Bipartite Matching: A survey and A New Problem" DOI 10.1.1.416.9903
- Benjamin Birnbaum and Claire Mathieu. 2008. "On-line bipartite matching made simple.", SIGACT News 39, 1, pp. 80-87, March 2008.

Questions

- Given an algorithm, can we prove if it is the shortest possible?
.
- How can we build an on-line matching bipartite graph?
.
- What are the applications of graphs with on-line matching?
.
- Is it possible to improve the suggested upper/lower bounds in future?

A Turing Complete Elementary Cellular Automaton

Steven Feldman

Jason Hochreiter

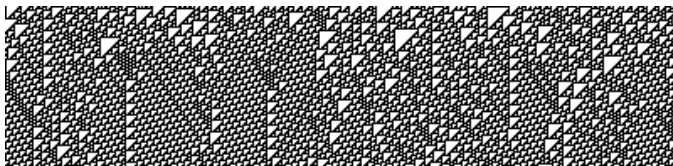
Pierre LaBorde

Kyle Martin

April 29, 2014

Rule 110

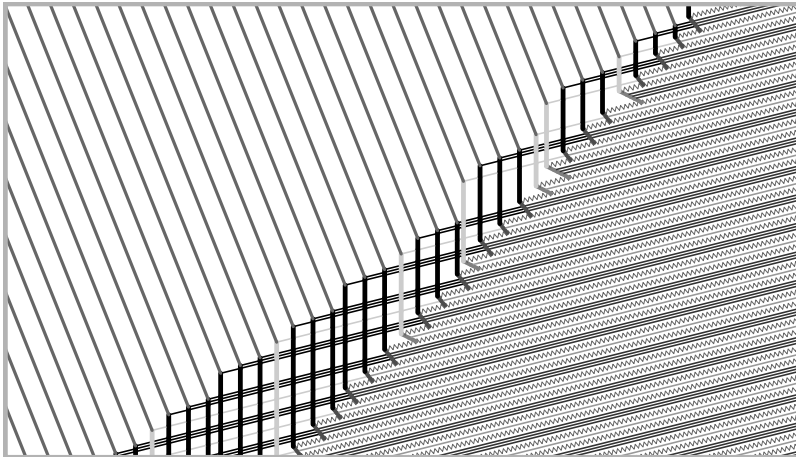
- ▶ A Class 4 cellular automaton – exhibits complex behavior
- ▶ Consists of a single infinitely long row of cells
- ▶ Each cell is either alive (1) or dead (0)
- ▶ Cells change based on neighbors:
 - ▶ A cell in state 0 enters state 1 if its right neighbor is in state 1 (life spreads leftward)
 - ▶ A cell in state 1 enters state 0 if both of its neighbors are in state 1 (overcrowding is deadly)



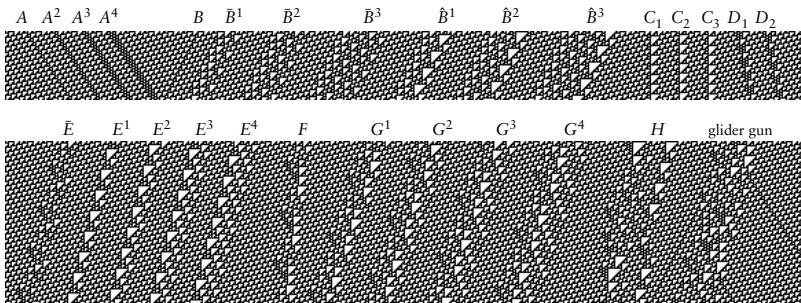
Proof Structure

- ▶ A complex proof structure is required to show the universality of Rule 110
- ▶ Relies on “nesting” universal machine emulators:
Turing Machine \Rightarrow Tag Systems \Rightarrow Cyclic Tag Systems \Rightarrow
Glider Systems \Rightarrow Rule 110
 - ▶ Tag system: read and consume a fixed number of symbols, appending a set of symbols (“tag”) to the end of the sequence based on the first symbol
 - ▶ Cyclic tag system: cycle through list of tags in order; append tag if currently read symbol is a Y and skip if symbol is an N
 - ▶ Glider systems: encode data using moving structures which collide to produce zero or more gliders

Glider Systems



Rule 110 Gliders



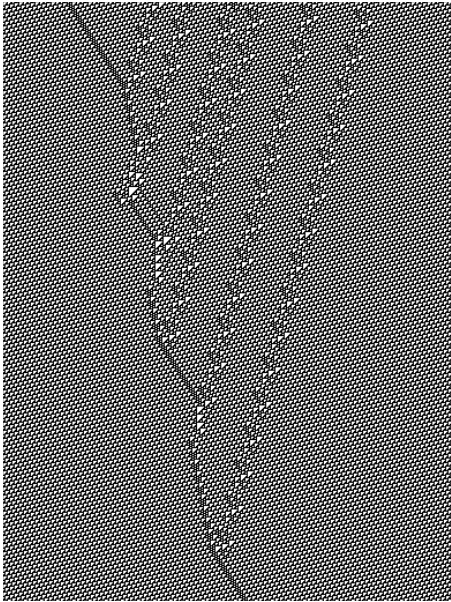
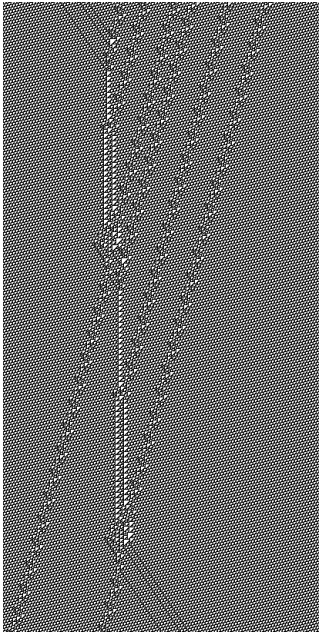
- ▶ Will primarily use A^n , C_2 and \bar{E} gliders
- ▶ Note the *ether* – background pattern of triangles

Data Encoding

- ▶ Collisions generate gliders based on the location and identity of the colliding gliders
- ▶ Important result – when C_2 s cross \bar{E} s, the newly created C_2 and \bar{E} gliders have the same relative positions as when they started
 - ▶ Since spacing is preserved, we can represent data in terms of spacings between gliders of the same type
- ▶ Collision with an A^4 can convert an \bar{E} into a C_2
 - ▶ Can convert data from one type to another
- ▶ Can precisely arrange gliders to preserve spacings and allow for collisions without interference

Implementing a Glider System in Rule 110

- ▶ Simulating a glider system:
 - ▶ \bar{E} s: moving data (left-moving “tag” glider)
 - ▶ C_2 s: tape data (stationary glider)
 - ▶ A^4 s: “ossifiers” that convert \bar{E} s into C_2 s (right-moving gliders)
 - ▶ Encode Y s and N s by varying spacing between C_2 s and \bar{E} s
 - ▶ Collisions with produced Y s and N s result in acceptance or deletion of data, respectively, allowing for emulation of a glider system within Rule 110



Undecidable Rule 110 Questions

- ▶ Will the system reach periodic behavior having some specified period?
- ▶ Will the system ever reach periodic behavior?
- ▶ Will the system ever produce a specific glider?
- ▶ Will the system ever produce a given sequence of bits?

Further Questions

- ▶ Are there any other Turing complete elementary cellular automata?
- ▶ Is Rule 110 still universal on a finite tape?
- ▶ Are there other gliders that could be used to prove universality of Rule 110 or other Class 4 CAs (like rules 30, 90, and 184)?
- ▶ Are all Class 4 elementary cellular automata universal?

Thanks!

A Turing Complete
Elementary Cellular Automaton

Steven Feldman

Pierre LaBorde

Jason Hochreiter

Kyle Martin

April 29, 2014

The complexity of UNO

Rui Hou, Yicong Tian

What's UNO?



Normal cards



Action cards



Game settings

- No action cards or draw pile
- Perfect information
- Rational play

Notations

- Color $X = \{1, 2, \dots, c\}$
- Number $Y = \{1, 2, \dots, b\}$
- A card: $t = (x, y)_i$
- $t' = (x', y')_{i'}$
- t' matches t
 $((x' == x) \vee (y' == y)) \wedge (i' == i + 1(\text{mod } p))$



Cooperative UNO

- Instance

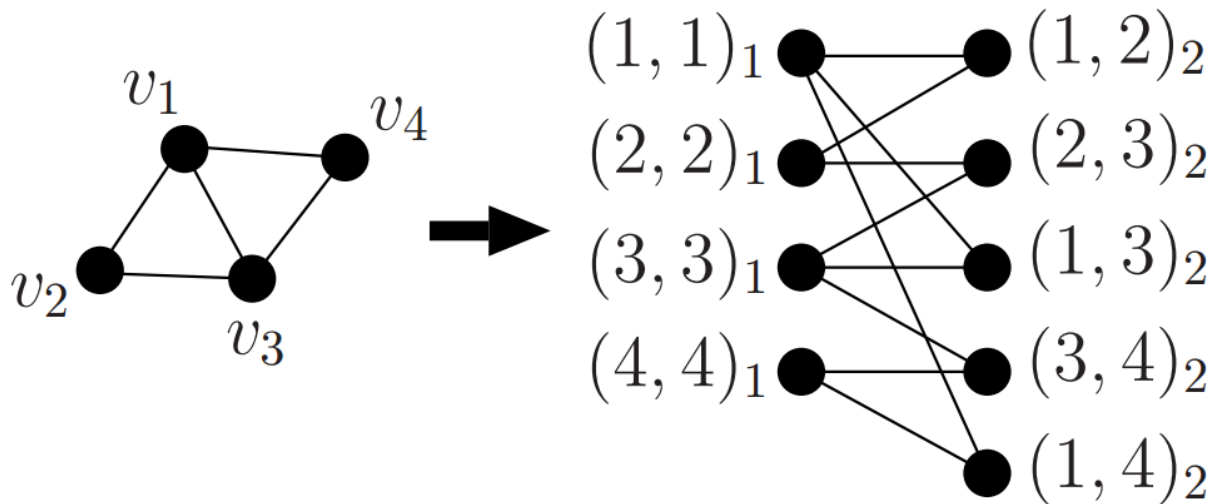
The number of players p , and player i 's card set C_i with c colors and b numbers.

- Question

Can all the players make player 1 win, i.e., make player 1's card set empty before any of the other players become finished?

Cooperative UNO-2

- Reduction from Hamiltonian Path to UNO-2
- UNO-2 is NP-complete



Uncooperative UNO

- Instance

The number of players p , and player i 's card set C_i with c colors and b numbers.

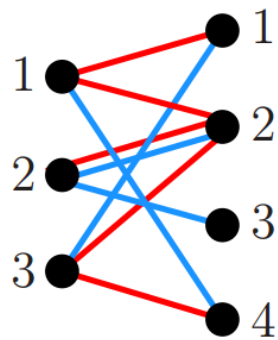
- Question

Determine the first loser; i.e., the player that cannot play one's card any more in spite that his/her hand is not empty.

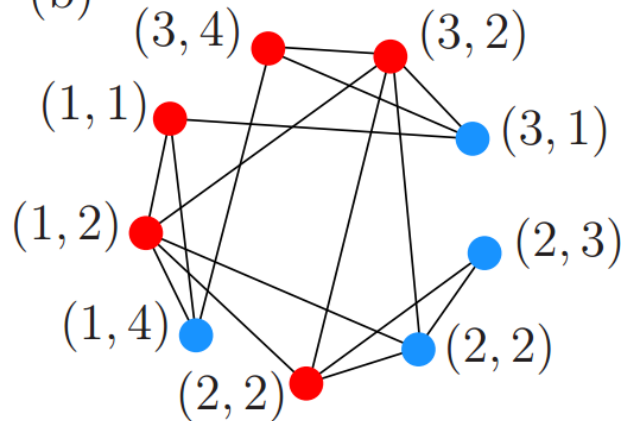
Uncooperative UNO-2

- Reduction from uncooperative UNO-2 to undirected vertex geography
- Uncooperative UNO-2 is in P

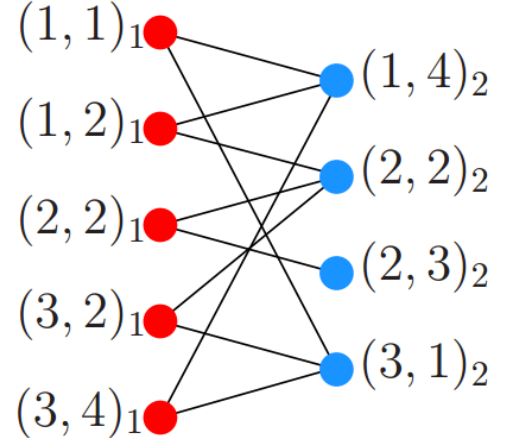
(a) color number



(b)



(c)



UNO-1

- Instance

A set C of n cards (x_i, y_i) ($i = 1, 2, \dots, n$), where $x_i \in \{1, 2, \dots, c\}$ and $y_i \in \{1, 2, \dots, b\}$.

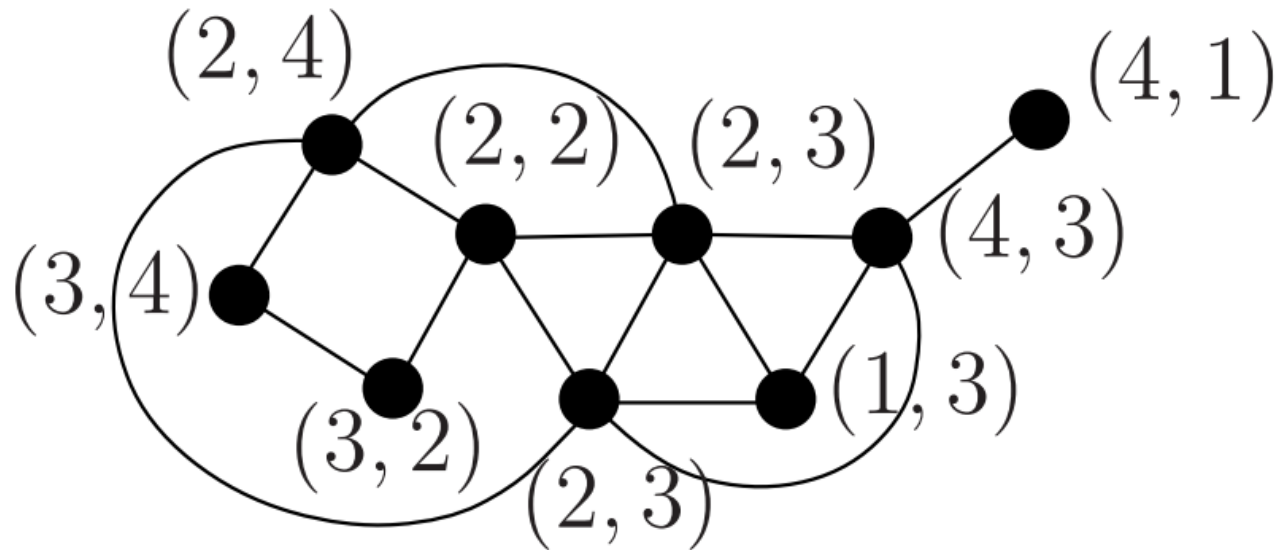
- Question

Determine whether the player can play all the cards.

- The cooperative and uncooperative versions of UNO become equivalent

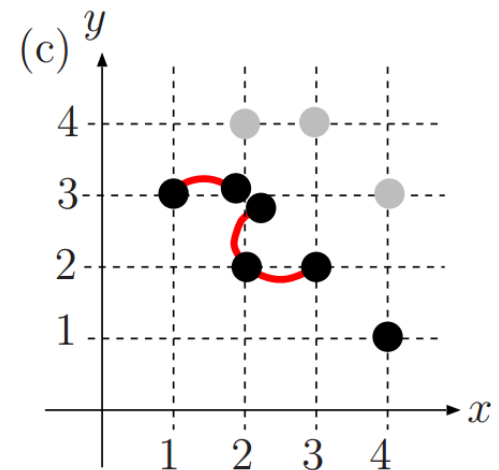
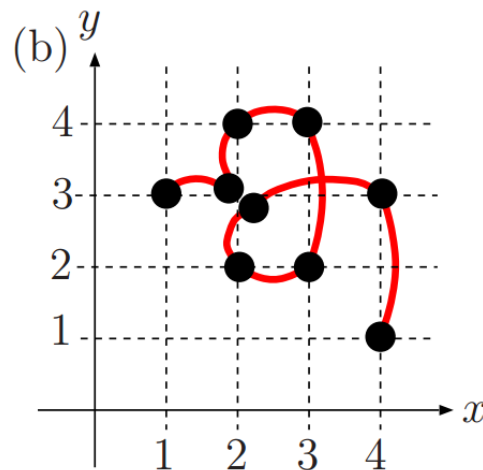
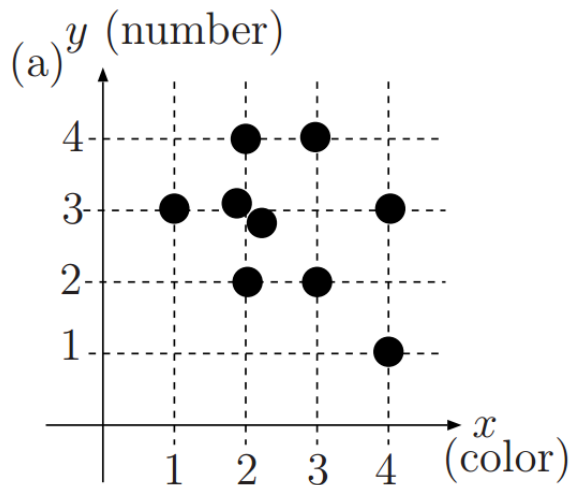
UNO-1 intractable case

- Reduction from Hamiltonian Path to UNO-1
- UNO-1 is NP-complete



UNO-1 tractable case

- Can be solved in P by dynamic programming (if b or c is a constant)



Question

- If we add action card 'skip' to the card set, how would the model and the complexity change?

Modeling Multiple-object Tracking as Constrained Flow Optimization Problem

Maryam Jaberi

Amara Tariq

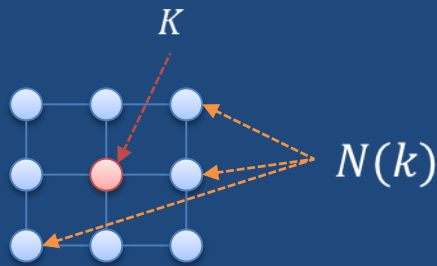
Muhammad Ali

Introduction

- Multiple-object tracking
 1. Detection Step: time-Independent
 2. Linking Step: connect detections into most likely trajectories: NP-Complete
- **Problem:** Linking detections into trajectories for multiple *visually-similar* objects
- **Solutions:**
 - Filtering, Greedy dynamic programming etc.: don't ensure global optimum
 - Integer Linear Programming (ILP): Ensures global optimum but NP-Complete

Multi-Object Tracking as Constrained Flow optimization

- Divide scene into k discrete locations.

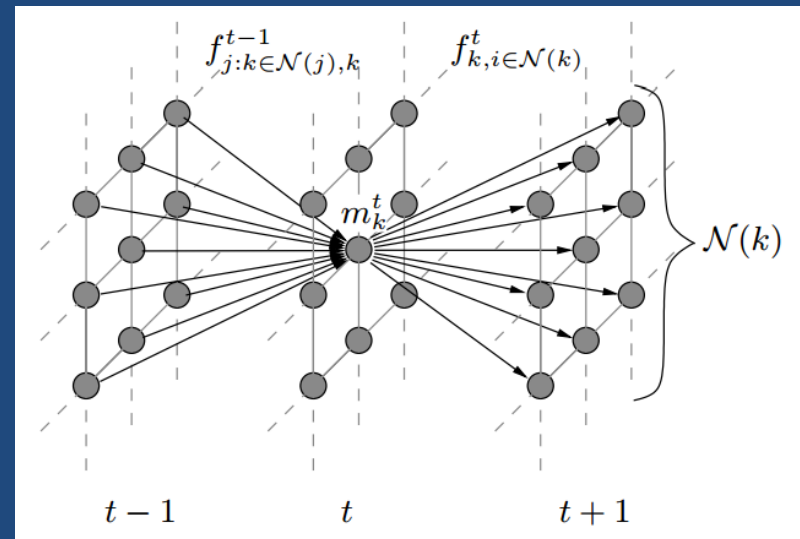


$$\forall j, t \quad \sum_{i: j \in N(i)} f_{i,j}^{t-1} = m_j^t = \sum_{k \in N(j)} f_{j,k}^t$$

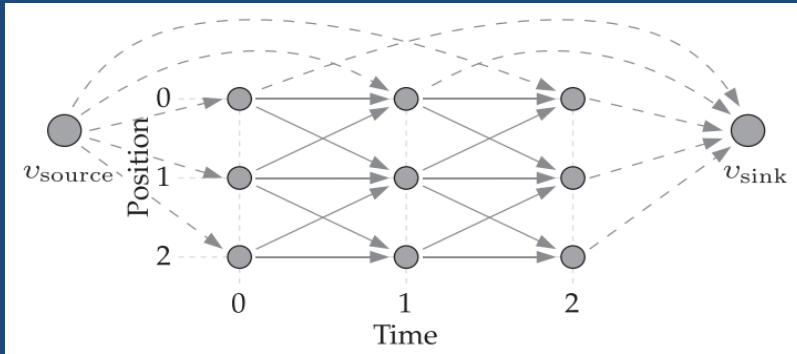
$$\forall k, t \quad \sum_{j \in N(k)} f_{k,j}^t \leq 1$$

$$\forall k, j, t \quad f_{k,j}^t \geq 0$$

- Model occupancy map over time using directed graph.



Multi-Object Tracking as Constrained Flow optimization



- Posterior probability of an object:
 - m : occupancy map (holding variables m_j^t)
 - \mathbb{F} : set of feasible maps m .
 - M_i^t : random variable presenting the true value of i in time t .
 - I^t : signal

$$\sum_{j \in N(v_{source})} f_{v_{source}, j}^t = \sum_{k: v_{sink} \in N(k)} f_{k, v_{sink}}^t$$

$$\rho_i^t = \hat{P}(M_i^t = 1 | I^t)$$

$$m^* = \arg \max_{m \in \mathbb{F}} \hat{P}(M = m | I^t)$$

- M_i^t as conditional independence variable

$$m^* = \arg \max_{m \in \mathbb{F}} \log \prod_{t, i} \hat{P}(M_i^t = m | I^t) = \arg \max_{m \in \mathbb{F}} \sum_{t, i} \log \hat{P}(M_i^t = m | I^t)$$

Integer Linear Programming (ILP) Formulation

- M_i^t as conditional independence variable

$$m^* = \arg \max_{m \in \mathbb{F}} \log \prod_{t,i} \hat{P}(M_i^t = m | I^t) = \arg \max_{m \in \mathbb{F}} \sum_{t,i} \left(\log \frac{\rho_i^t}{1 - \rho_i^t} \right) m_i^t$$

- **ILP System:**

- Maximize $\sum_{t,i} \left(\log \frac{\rho_i^t}{1 - \rho_i^t} \right) \sum_{j \in N(i)} f_{i,j}^t$

- Subject to $\forall i, j, t \quad f_{i,j}^t \geq 0$

$$\forall i, t \quad \sum_{j \in N(i)} f_{i,j}^t \leq 1$$

$$\forall i, t \quad \sum_{j \in N(i)} f_{i,j}^t - \sum_{k: i \in N(k)} f_{k,i}^{t-1} \leq 0$$

$$\sum_{k \in N(v_{source})} f_{j,k}^t - \sum_{i: j \in N(i)} f_{i,j}^{t-1} \leq 0$$

From Integer to Continuous Linear Program

- Integer LP solution: NP-complete problem
- Continuous LP: Polynomial time average complexity
- Relax the 'integer' condition to reduce complexity
- *Problem:* continuous LP does not usually converge to optimal solution of original ILP!!
- *Solution:* Total-unimodularity of constraint matrix

Total Unimodularity

- Total-Unimodular Matrix:
 - all square sub-matrices has determinants 1,0 or -1
- Theorem:
 - For every subset of rows $R \subseteq \{1,2,\dots,m\}$, there is a partition of rows such that $R = R_1 \cup R_2$, $R_1 \cap R_2 = \emptyset$
$$\forall_{j=1,2,\dots,n} (\sum_{i \in R_1} a_{ij} - \sum_{i \in R_2} a_{ij}) \in \{-1,0,1\}$$
- Ensures integer solution even for continuous LP
- Constraint matrix is Total-Unimodular

Total-Unimodularity of Constraint Matrix

- Only 3 rows can be non-zero ($\in \{-1,0,1\}$) for one column

| $\{a_{ij} i \in R_1\}$ | $\{a_{ij} i \in R_2\}$ | $\sum_{i \in R_1} a_{ij} - \sum_{i \in R_2} a_{ij}$ |
|--------------------------|--------------------------|---|
| $\{0, \dots, 0, 1\}$ | $\{0, \dots, 0\}$ | 1 |
| $\{0, \dots, 0, 1\}$ | $\{0, \dots, 0, 1\}$ | 0 |
| $\{0, \dots, 0, 1\}$ | $\{0, \dots, 0, -1\}$ | 2 |
| $\{0, \dots, 0, 1\}$ | $\{0, \dots, 0, 1, -1\}$ | 1 |
| $\{0, \dots, 0\}$ | $\{0, \dots, 0\}$ | 0 |
| $\{0, \dots, 0\}$ | $\{0, \dots, 0, 1\}$ | -1 |
| $\{0, \dots, 0\}$ | $\{0, \dots, 0, -1\}$ | 1 |
| $\{0, \dots, 0\}$ | $\{0, \dots, 0, 1, -1\}$ | 0 |

- Eight cases for partitions

$$R_1 = R \cap U_1, R_2 = R \cap U_2 \text{ for any } R \subseteq \{1, 2, \dots, m\}$$

- Every possibility satisfies total-unimodularity but 3rd – *Problem*
- *Solution*: Move non-zero row of R_1 to R_2 for 3rd case

K-shortest paths (KSP) formulation

- Relaxed ILP solution polynomial but practically not efficient, Need real time efficiency for practical problems
- KSP: Given a graph $G(V,E)$, compute a set of k shortest paths $\{p_1, p_2, \dots, p_k\}$ such that the total cost is minimum
- Path constraints:
 1. Node disjoint
 2. Node simple

ILP to KSP

- Maximizing flow \approx Minimizing cost function in
in ILP KSP
- Optimality of 'k': guaranteed
by convexity of the path
cost function
- Dijkstra's algorithm for
Path computation in each
iteration
- Complexity: $O(k(m+n\log n))$

$$c(e_{i,j}^t) = -\log\left(\frac{\rho_i^t}{1-\rho_i^t}\right)$$

$$f^* = \arg \min_{f \in \mathcal{F}} \sum_{t,i} c(e_{i,j}^t) \sum_{j \in \mathcal{N}(i)} f_{i,j}^t$$

$$\text{cost}(P_l) = \sum_{i=1}^l \text{cost}(p_i^*)$$

Applications

- 2D segmentation to 3D segmentation:
 - 2D: shortest path problem; Dijkstra's algorithm
 - 3D: minimal weight surface problem
 - can be presented as instance of ILP with totally unimodular constraint matrix
- Optimization by LP:
 - LP provides an upper/ lower bound for original ILP
 - Branch-and-bound: optimization of ILP through recursive LP solution.
- Min-Cost flow problems: efficient network routing
- Tracking multiple humans in surveillance videos

Question: How does Total-Unimodularity ensure Integer solution even for continuous LP?

- Quadratic system: $Cx = y$
- Cramer's rule: $x_j = |C_y^j| / |C|$
- $C_y^j = C$ with j^{th} column replaced with y , $(C_1, C_2, \dots, C_{j-1}, y, C_{j+1}, \dots, C_m)$
- $|C_y^j| = \sum_i (-1)^{i+j} y_i |C_{ij}|$
- $C_{ij} = C$ with i^{th} row, j^{th} column deleted
- If $|C| \in \{-1, 1\}$ and $C_{ij} \in \{-1, 0, 1\}$ for all i and j , then x is integer
- *Total Unimodularity \Rightarrow Integer solution*

Question: Why is the path cost function of KSP convex?

- Edge costs can be negative and total path cost function is summation of successive shortest path costs that are monotonically increasing!

$$\text{cost}(p_{i+1}^*) \geq \text{cost}(p_i^*) \quad \forall i$$

- Therefore, the path cost function over 'k' is convex

$$\text{cost}(P_l) = \sum_{i=1}^l \text{cost}(p_i^*)$$

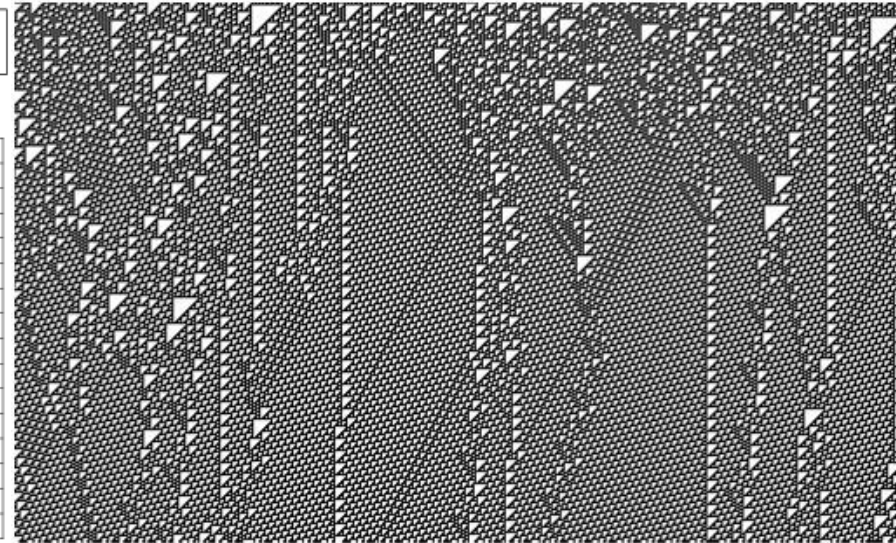
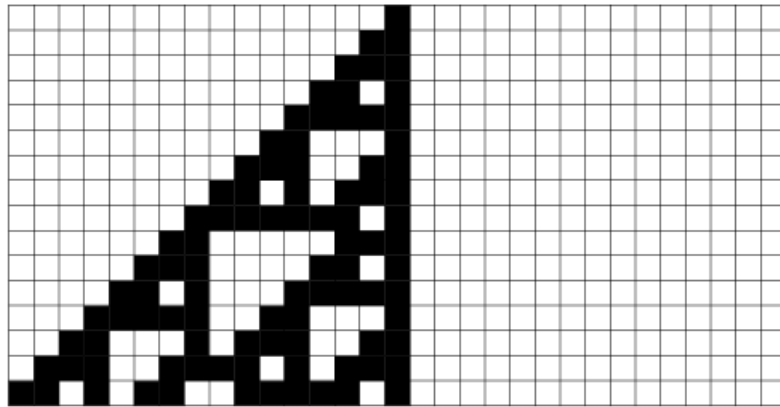
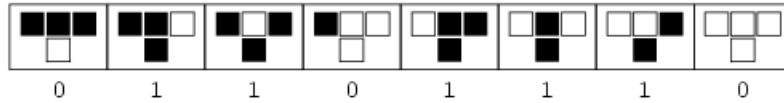
$$\text{cost}(P_{k^*-1}) \geq \text{cost}(P_{k^*}) \leq \text{cost}(P_{k^*+1})$$

Universality in Elementary Cellular Automata

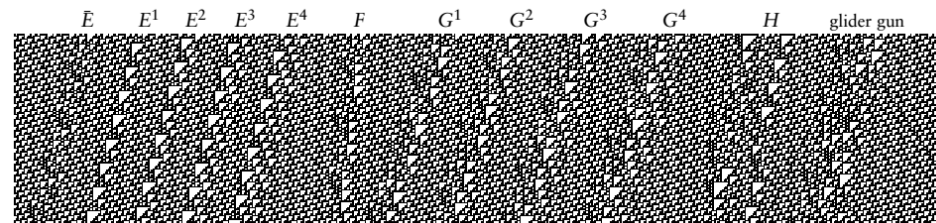
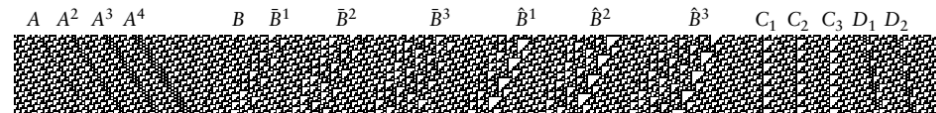
Kangsoo Kim, Myungho Lee, Sungchul Jung

Wolfram Rule 110

rule 110



- One of the elementary cellular automaton rules
- Turing Complete
- Extremely **simple** system (possibly naturally occurring), yet capable of **universality**
- <http://eli.fox-epste.in/rule110-full.html>



Tag System

Tape

| | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|--|--|--|--|--|--|--|--|--|
| A | C | D | A | B | B | E | | | | | | | | | | | | | |
| | | D | A | B | B | E | C | C | D | D | | | | | | | | | |
| | | | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | | | | | |

Lookup table

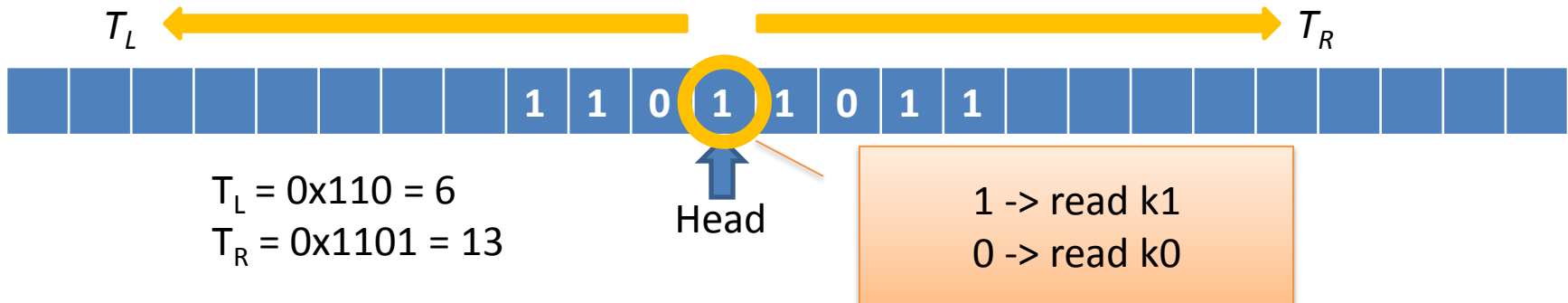
| | |
|-----|------|
| A | CCDD |
| ... | ... |
| D | ... |
| ... | ... |

TM to Tag System

To emulate a Turing machine with m states, we will use a tag system with $10m$ symbols:

$$\{L_k, L_{k0}, L_{k1}, R_k, R_{k0}, R_{k1}, R_{k*}, H_k, H_{k0}, H_{k1}\},$$

$k \in$ states of the Turing machine



Our tag system tape will represent these numbers in unary like this:

$$H_{k_1} H_{k_0} [L_{k_1} L_{k_0}]^{T_L} [R_{k_1} R_{k_0}]^{T_R}$$

TM to Tag System

$$\begin{array}{ll}
 H_{kz} & \rightsquigarrow [R_{k' * } R_{k' *}]^a [H_{k'}]^b H_{k'} [L_{k'} L_{k'}]^c \\
 L_{kz} & \rightsquigarrow L_{k'} [L_{k'} L_{k'} L_{k'}]^d \quad z \in \{0, 1\} \\
 R_{kz} & \rightsquigarrow R_{k'} [R_{k'} R_{k'} R_{k'}]^e \\
 R_{k * } & \rightsquigarrow R_k R_k \\
 H_k & \rightsquigarrow H_{k1} H_{k0} \\
 L_k & \rightsquigarrow L_{k1} L_{k0} \\
 R_k & \rightsquigarrow R_{k1} R_{k0}
 \end{array}$$

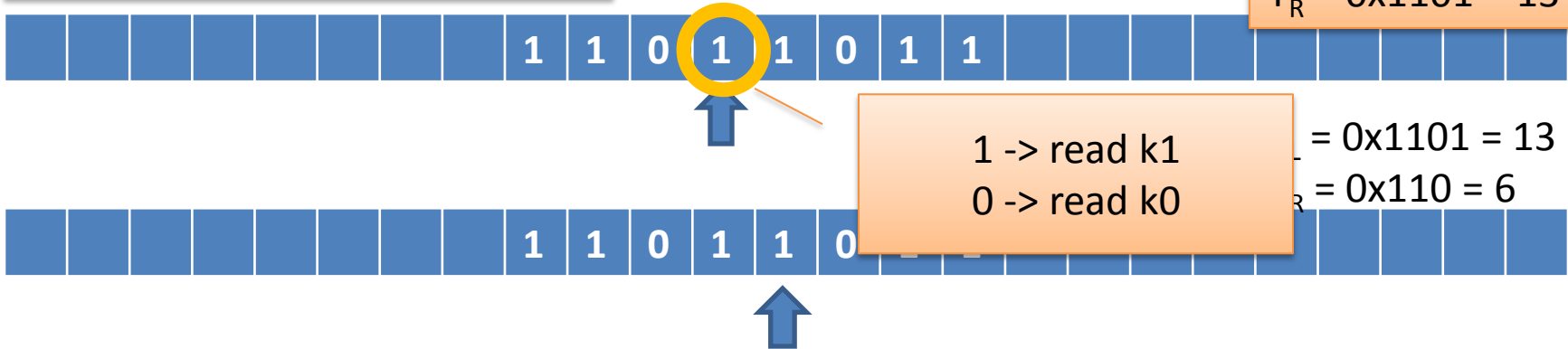
a : include when kz indicates a move to the left, writing a 1.
 b : include when z is 0.
 c : include when kz indicates a move to the right, writing a 1.
 d : include when kz indicates a move to the right.
 e : include when kz indicates a move to the left.

$$\begin{array}{c}
 H_{k1} H_{k0} [L_{k1} L_{k0}]^{T_L} [R_{k1} R_{k0}]^{T_R} \\
 \Downarrow \\
 [R_{k' * } R_{k' *}]^a [H_{k'}]^b H_{k'} [L_{k'} L_{k'}]^{(d?2:\frac{1}{2})T_L + (c?1:0)} [R_{k'} R_{k'}]^{(e?2:\frac{1}{2})T_R} \\
 \Downarrow \\
 H_{k'} [L_{k'} L_{k'}]^{(d?2:\frac{1}{2})T_L + (c?1:0)} [R_{k'} R_{k'}]^{(e?2:\frac{1}{2})T_R + (a?1:0)} \\
 \Downarrow \\
 H_{k'1} H_{k'0} [L_{k'1} L_{k'0}]^{T'_L} [R_{k'1} R_{k'0}]^{T'_R}
 \end{array}$$

TM to Tag System

Move to the right and writing 1

$T_L = 0x110 = 6$
 $T_R = 0x1101 = 13$



$H_{k1} H_{k0} L_{k1} L_{k0} L_{k1} L_{k0} L_{k1} L_{k0} L_{k1} L_{k0} L_{k1} L_{k0} L_{k1} L_{k0} L_{k1} L_{k0} R_{k1} R_{k0} R_{k1} R_{k0} R_{k1} R_{k0} R_{k1} R_{k0} R_{k1} R_{k0} R_{k1} R_{k0} R_{k1} R_{k0} R_{k1} R_{k0} R_{k1} R_{k0} R_{k1} R_{k0} R_{k1} R_{k0} R_{k1} R_{k0} R_{k1} R_{k0}$

$$z \in \{0, 1\}$$

- a : include when kz indicates a move to the left, writing a 1.
- b : include when z is 0.
- c : include when kz indicates a move to the right, writing a 1.
- d : include when kz indicates a move to the right.
- e : include when kz indicates a move to the left.

TM to Tag System

$R_{k1}R_{k0}R_{k1}R_{k0}R_{k1}R_{k0}R_{k1}R_{k0}R_{k1}R_{k0}R_{k1}R_{k0}R_{k1}R_{k0}R_{k1}R_{k0}H_kL_kR_kR_kR_kR_kR_k$

$R_{k1}R_{k0}R_{k1}R_{k0}R_{k1}R_{k0}R_{k1}R_{k0}R_{k1}R_{k0}R_{k1}R_{k0}R_{k1}R_{k0}H_kL_kR_kR_kR_kR_kR_k$

$R_{k1}R_{k0}R_{k1}R_{k0}R_{k1}R_{k0}R_{k1}R_{k0}R_{k1}R_{k0}R_{k1}R_{k0}H_kL_kR_kR_kR_kR_kR_k$

$R_{k1}R_{k0}R_{k1}R_{k0}R_{k1}R_{k0}R_{k1}R_{k0}R_{k1}R_{k0}H_kL_kR_kR_kR_kR_kR_k$

$R_{k1}R_{k0}R_{k1}R_{k0}R_{k1}R_{k0}R_{k1}R_{k0}H_kL_kR_kR_kR_kR_kR_k$

$R_{k1}R_{k0}R_{k1}R_{k0}R_{k1}R_{k0}H_kL_kR_kR_kR_kR_kR_k$

$R_{k1}R_{k0}R_{k1}R_{k0}H_kL_kR_kR_kR_kR_kR_kR_kR_kR_kR_kR_kR_kR_k$

$R_{k1}R_{k0}H_kL_kR_kR_kR_kR_kR_kR_kR_kR_kR_kR_kR_kR_k$

$H_kL_kR_kR_kR_kR_kR_kR_kR_kR_kR_kR_kR_kR_k$

$L_kR_kR_kR_kR_kR_kR_kR_kR_kR_kR_kR_kR_kR_kH_{k1}H_{k0}$

$L_kR_kR_kR_kR_kR_kR_kR_kR_kR_kR_kR_kR_kR_kH_{k1}H_{k0}L_{k1}L_{k0}$

$$\begin{aligned}
 H_k &\rightsquigarrow H_{k1}H_{k0} \\
 L_k &\rightsquigarrow L_{k1}L_{k0} \\
 R_k &\rightsquigarrow R_{k1}R_{k0}
 \end{aligned}$$

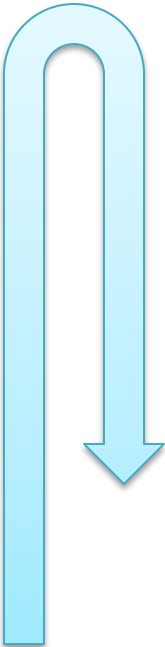
Tag to Cyclic Tag System

Symbol
Encoding



YNN...NNNNNN: 1st Symbol of k symbols
 NYN...NNNNNN: 2nd
 NNY...NNNNNN: 3rd
 ...
 NNN...NNNNNY: k_{th} Symbol

It's not a
lookup table.
Each step the
machine moves
on to the next
appendant in
the list
cyclically!



| | |
|-----------------|-----------------------------|
| 1 st | NNNYNNN...NNYNNN...NN |
| 2 nd | NN...Y...NNN...Y...NN |
| ... | ... |
| k | NNN..Y...NYN...NN |
| 1 | k appendants of length zero |
| 2 | |
| ... | |
| ... | |
| k | |

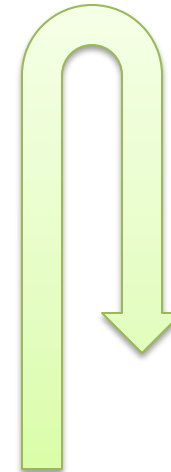
Tag to Cyclic Tag System

| | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|--|--|--|--|--|--|--|--|--|--|--|--|--|
| B | A | | | | | | | | | | | | | | | | | | |
| | | A | B | H | | | | | | | | | | | | | | | |
| | | | | H | B | B | | | | | | | | | | | | | |

| | |
|---|-----|
| A | YNN |
| B | NYN |
| H | NNY |

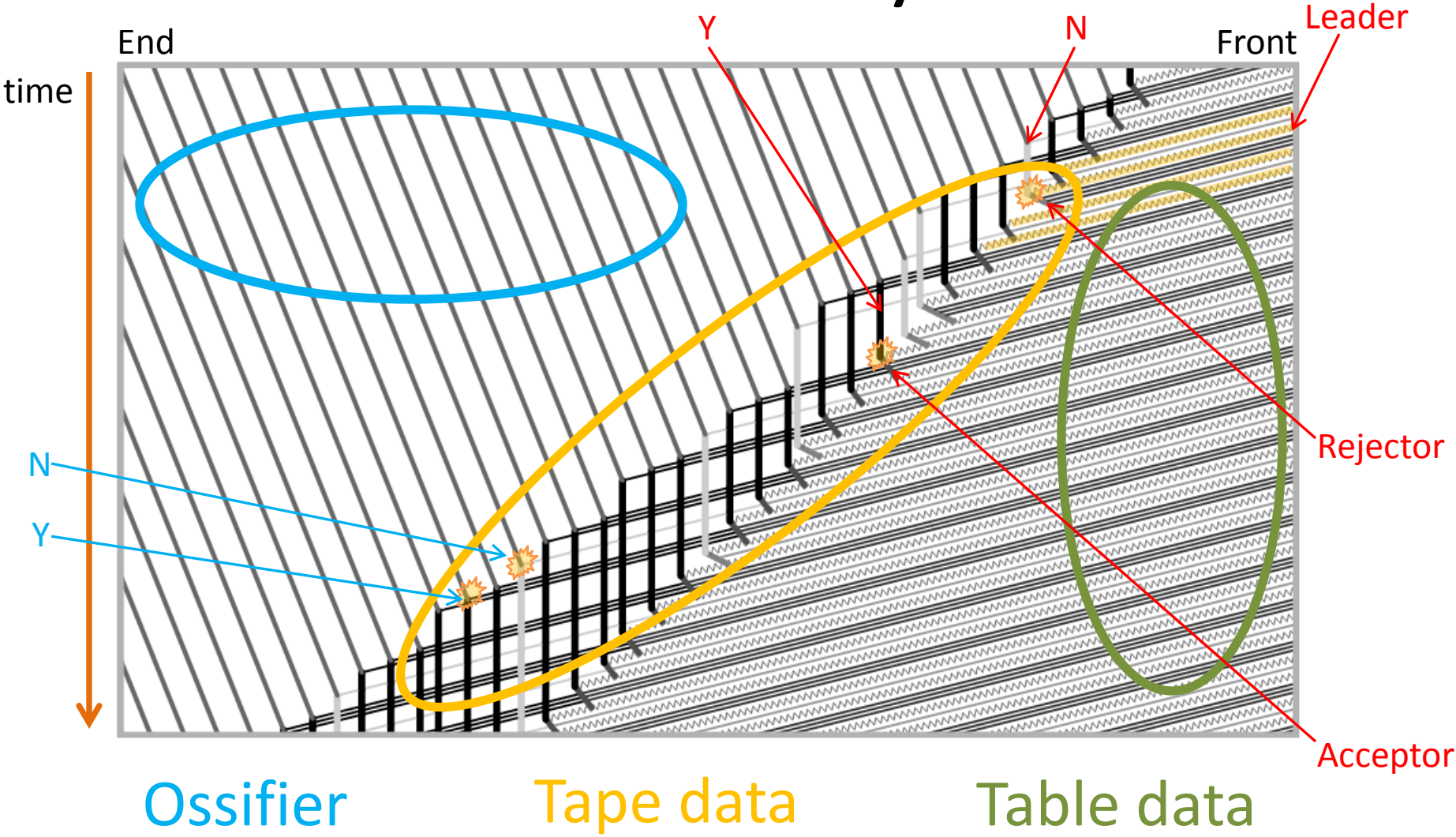
- 1 NYN YNN (=BA)
- 2 NYN YNN
- 3 NYN YNN YNN NYN NNY
- 4 NYN YNN YNN NYN NNY
- 5 NYN YNN YNN NYN NNY
- 6 NYN YNN YNN NYN NNY
- 1 NYN YNN YNN NYN NNY (=ABH)
- 2 NYN YNN YNN NYN NNY NYN NYN
- 3 NYN YNN YNN NYN NNY NYN NYN
- 4 NYN YNN YNN NYN NNY NYN NYN
- 5 NYN YNN YNN NYN NNY NYN NYN
- 6 NYN YNN YNN NYN NNY NYN NYN
- 1 NYN YNN YNN NYN NNY NYN NYN
- 2 NYN YNN YNN NYN NNY NYN NYN

...



| |
|-------------|
| NYN NYN |
| YNN NYN NNY |
| NNY |
| - |
| - |
| - |

CTS to Glider System

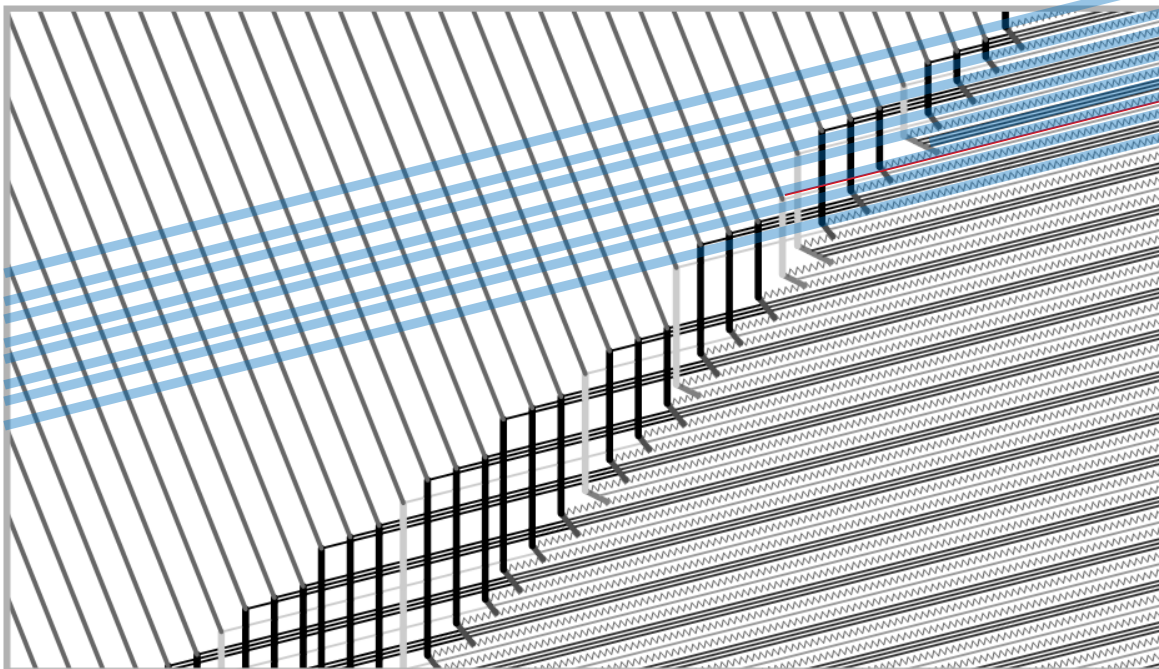


CTS to Glider System

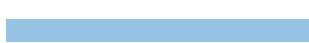
1 Y
2 YYY
1 YYYN
2 YYYNYYY
1 YYYNYYN
...



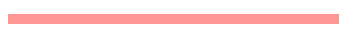
| |
|-----|
| YYY |
| N |



Y
YYY
YYN
YNYYY
NYYYN
...



NNN

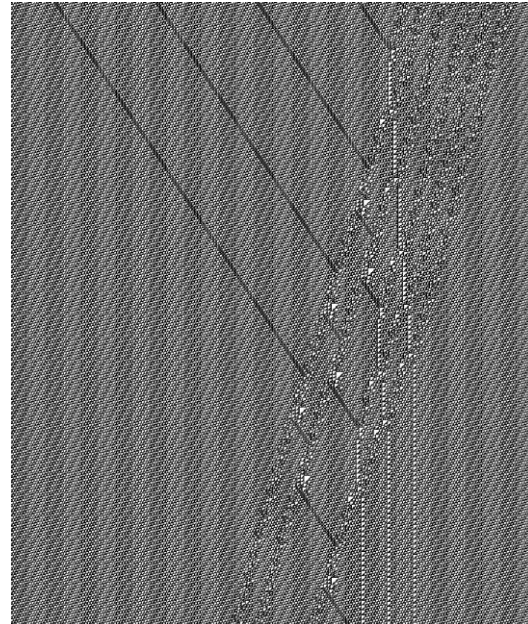
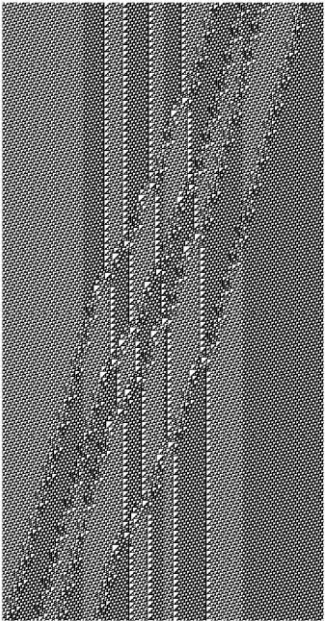


Y

Gliders in Rule 110

Spacing are preserved!

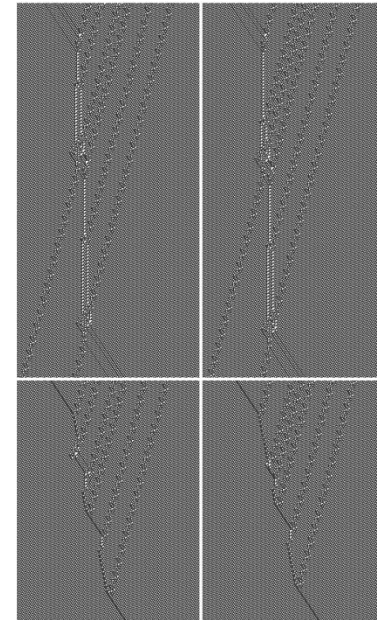
=> moving data crossing tape data



Converting moving data to tape data

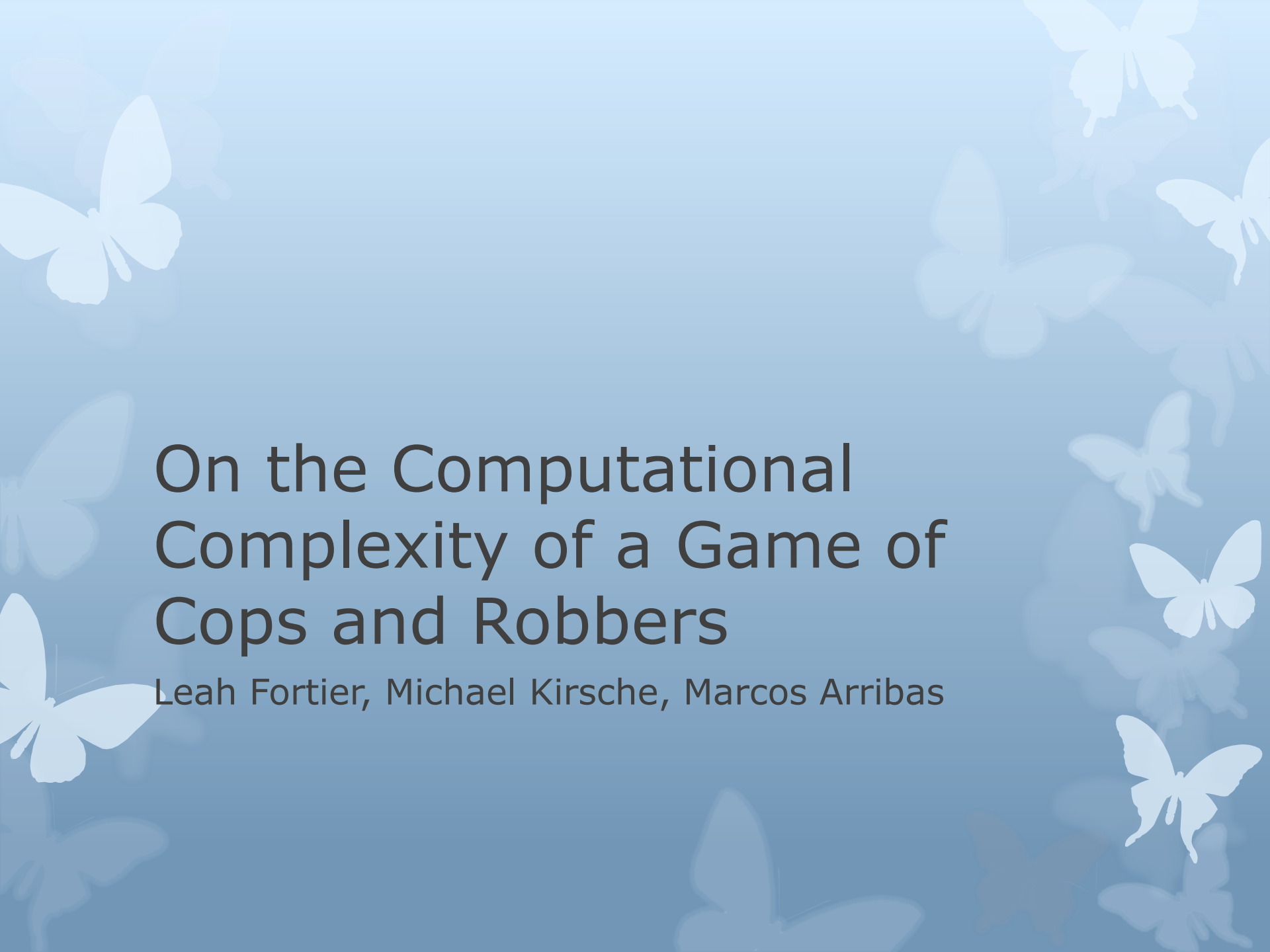
=> A⁴s : ossifiers

Components getting accepted(upper)
or rejected(lower)



Question!

- Will the behavior become periodic at all?



On the Computational Complexity of a Game of Cops and Robbers

Leah Fortier, Michael Kirsche, Marcos Arribas

Cops and Robbers (C&R)

- Undirected graph with n cops and one robber
- Game takes place in turns, cops move simultaneously then robber moves
- Cops win if they “catch” the robber by occupying the same vertex as the robber at the end of a turn

Cops and Robbers with Protection ($C\&R_p$)

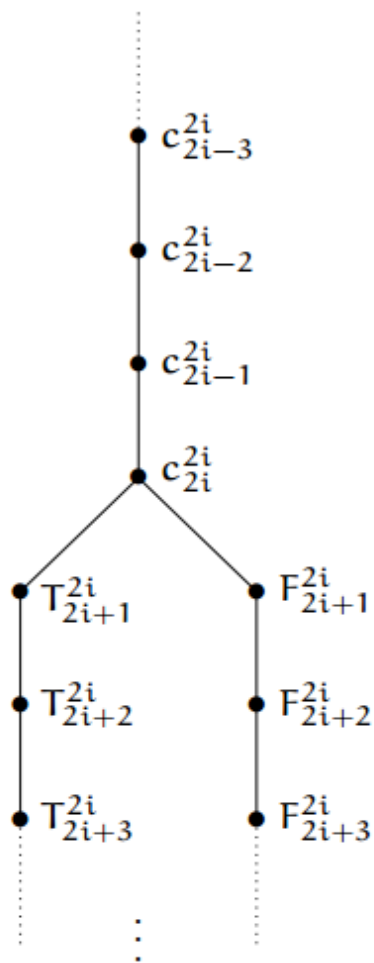
- Equivalent variation of original C&R problem
- Edges can be protected or unprotected
- Cops can only win by taking an unprotected edge to catch the robber

$$\text{QSAT} \leq_T \text{C\&R}_p^*$$

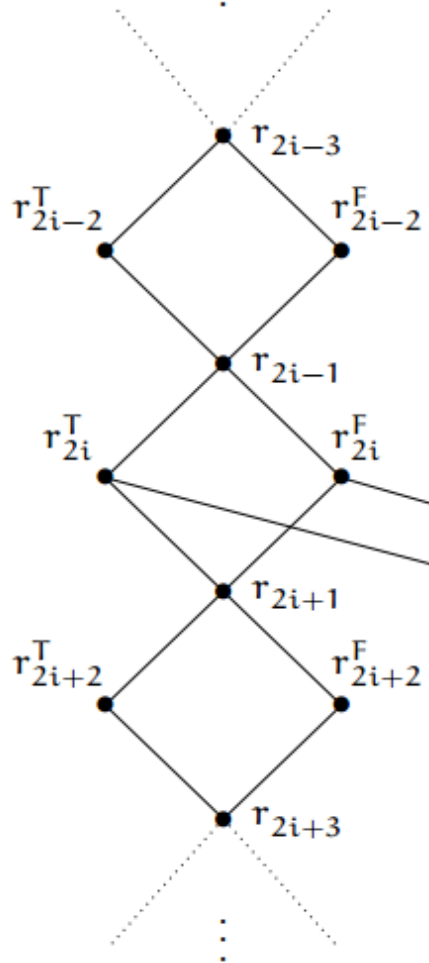
- C\&R_p^* is a variation with fixed starting points
- QSAT expression must contain an equal number of existential and universal quantifiers (trivial modification)

Gadgets

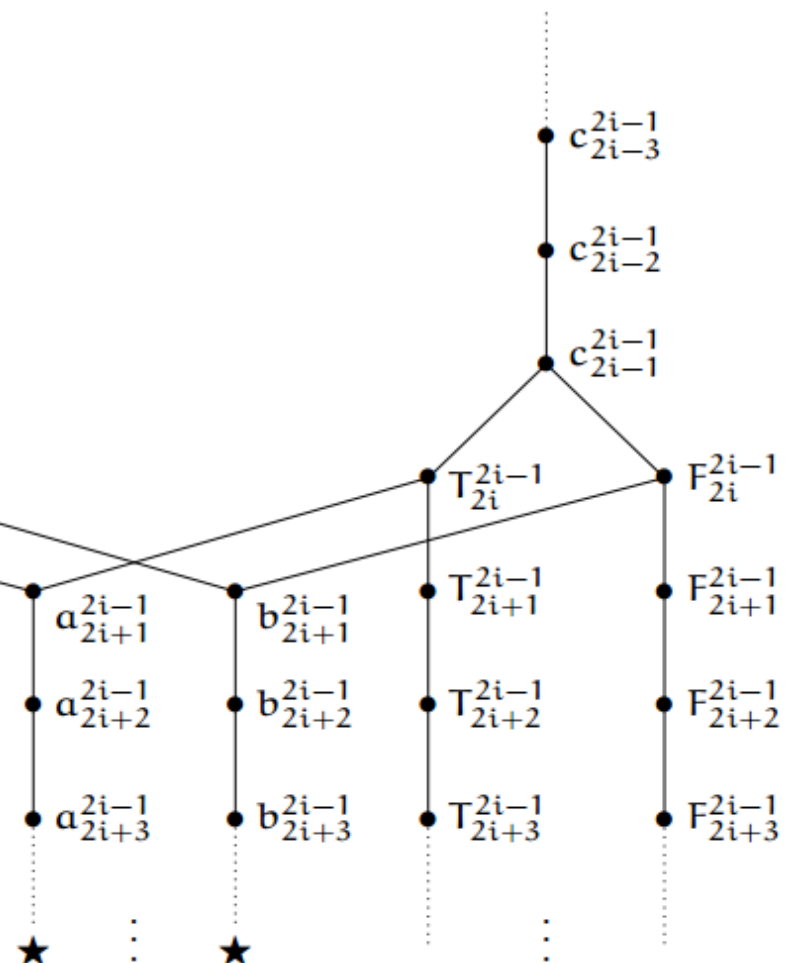
2i-th cop's track



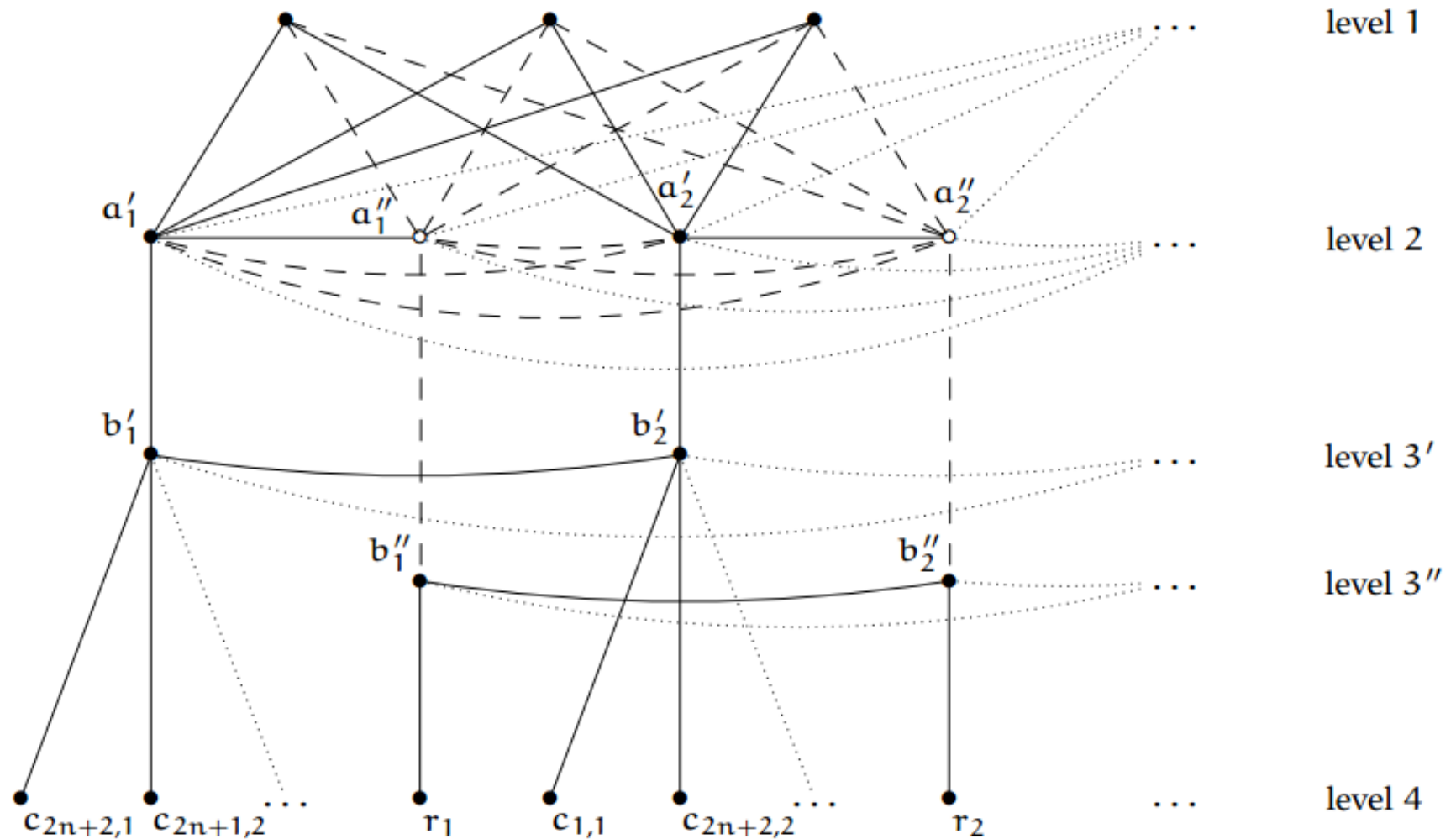
robber's track



(2i - 1)-th cop's track



QSAT \leq_T C&R_p



Open Questions

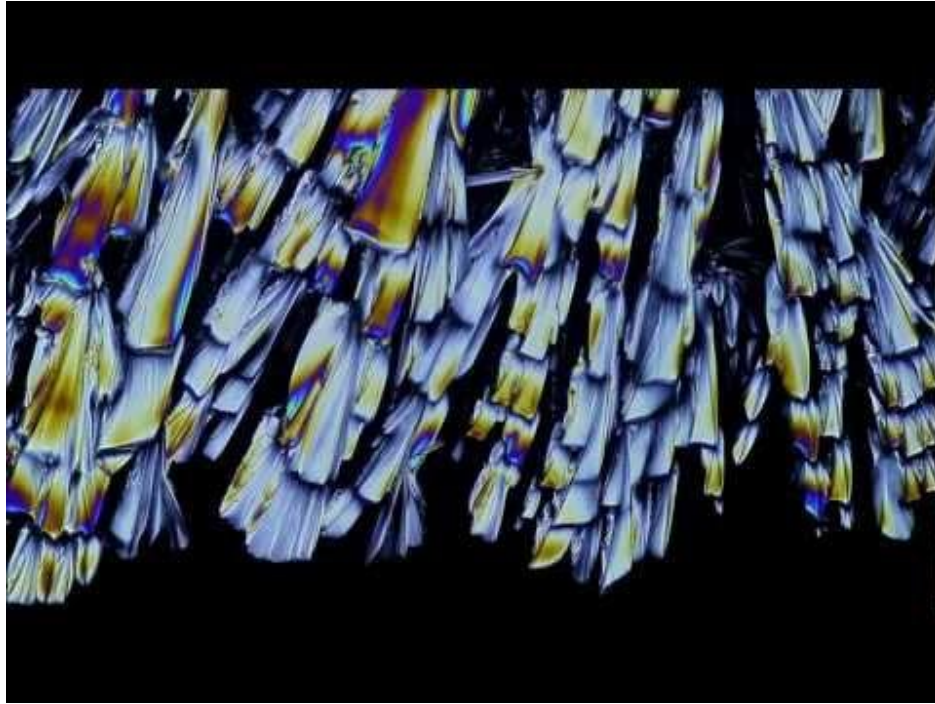
- Is there a tighter bound than PSPACE-hard on the complexity of C&R?
- Is $C\&R_p^*$ computationally equivalent to $C\&R_p$?

Theory of Algorithmic Self-Assembly

Gregory Morse and Lisa Soros

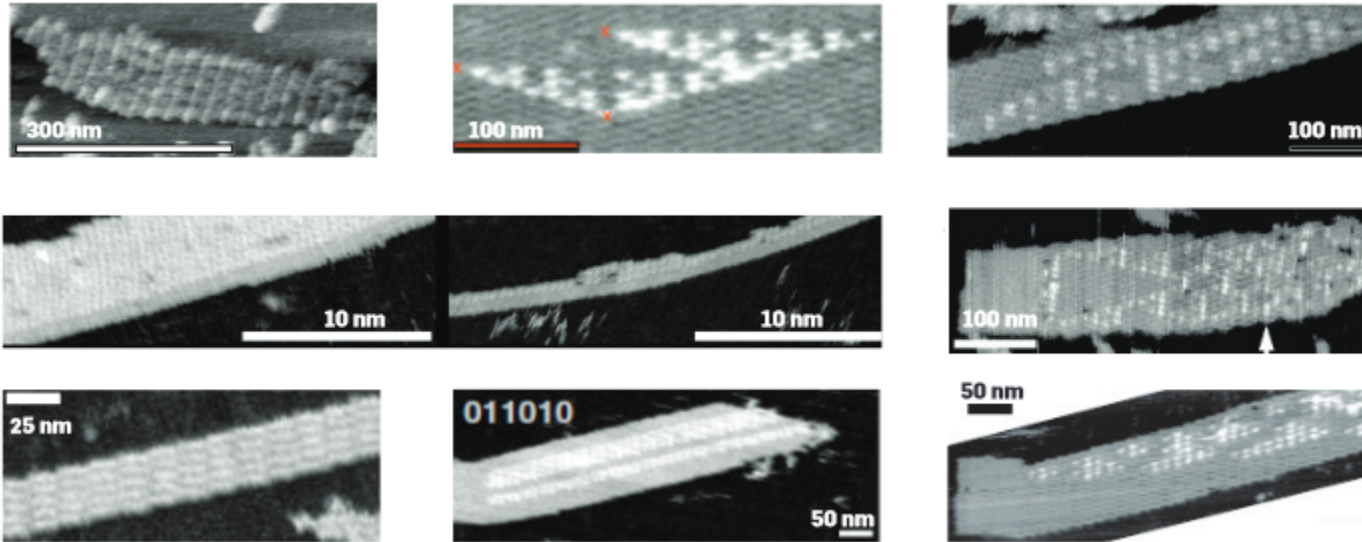
(Doty, David. "Theory of algorithmic self-assembly." *Communications of the ACM* 55.12 (2012): 78-88.)

Algorithmic Self-Assembly



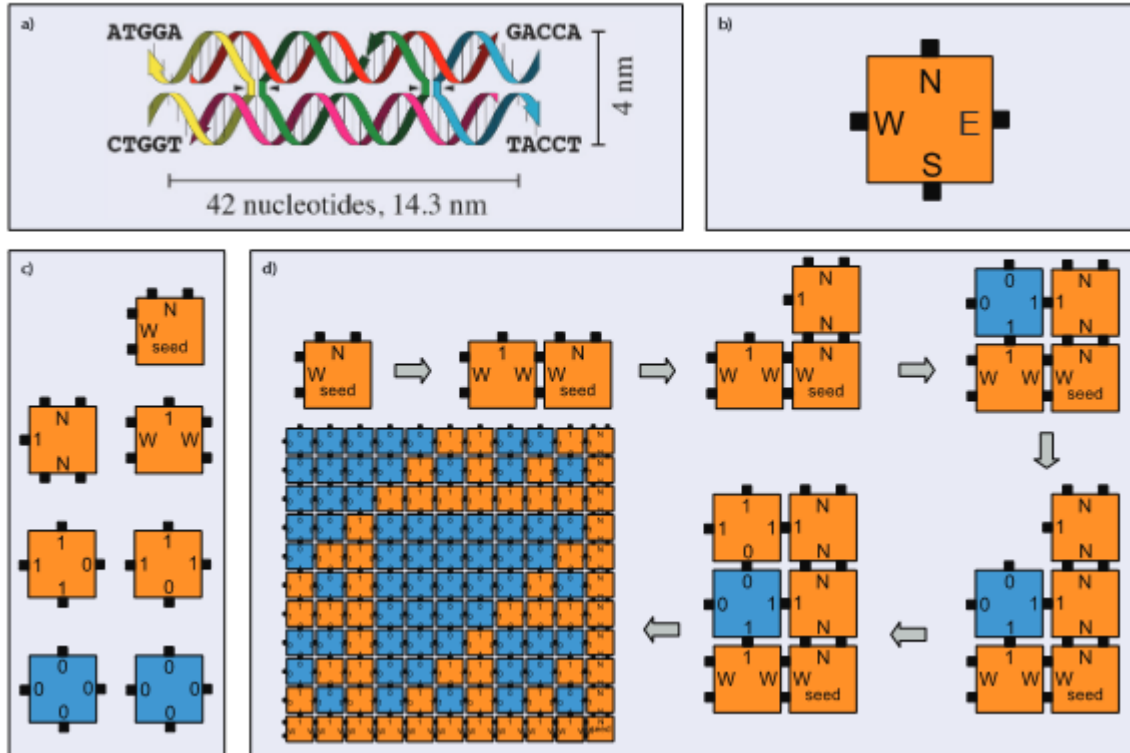
(ex. crystal growth)

Algorithmic Self-Assembly



(ex. DNA tiling)

abstract Tile Assembly Model (aTAM)



Why is this interesting?

Universal computation
+ Massive parallelization

Solution to NP-Complexity?



Open Questions

- Better error correction?
- Limits on parallelization?
- Optimization == pattern assembly?
- Power of cooperative binding?

etc. questions

1. Is algorithmic self-assembly using DNA molecules a universal model of computation?

(yes)

2. Can DNA tile be assembled in sublinear time? (no)

Classic Nintendo Games are
(Computationally) Hard

BY: Chris Ross and Corey
Pittman

The General Framework

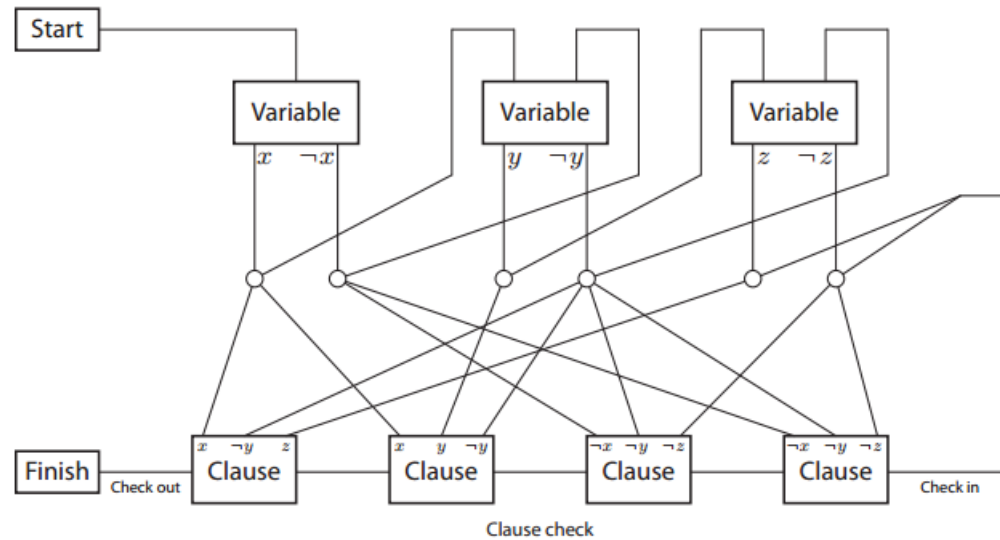
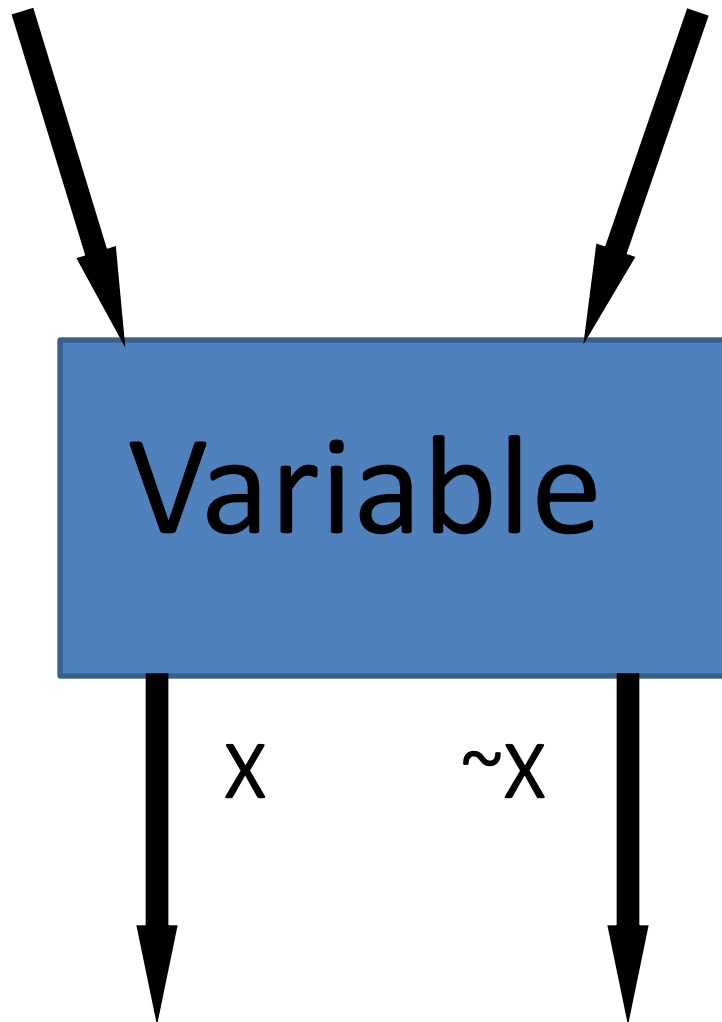
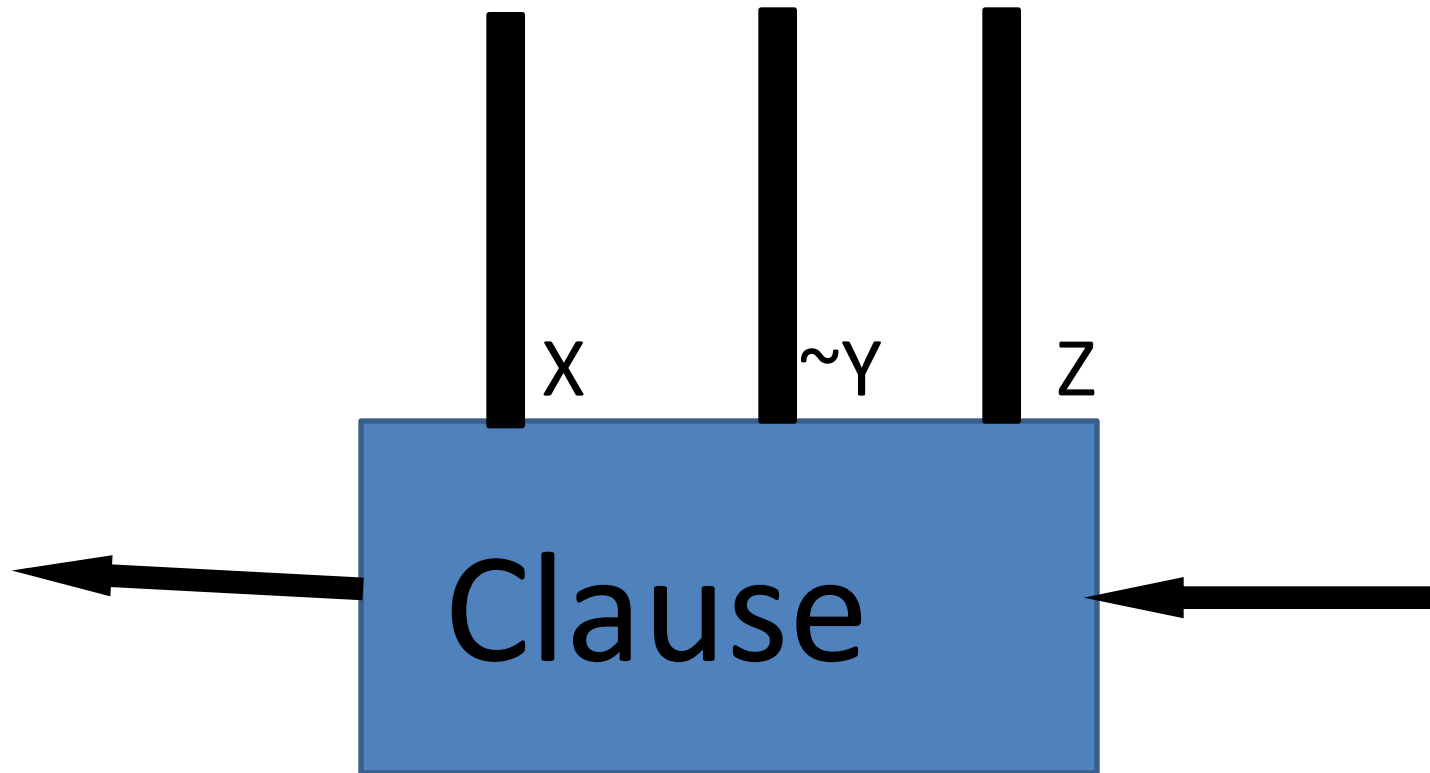


Figure 1: General framework for NP-hardness

The Gadgets



The Gadgets



The General Framework

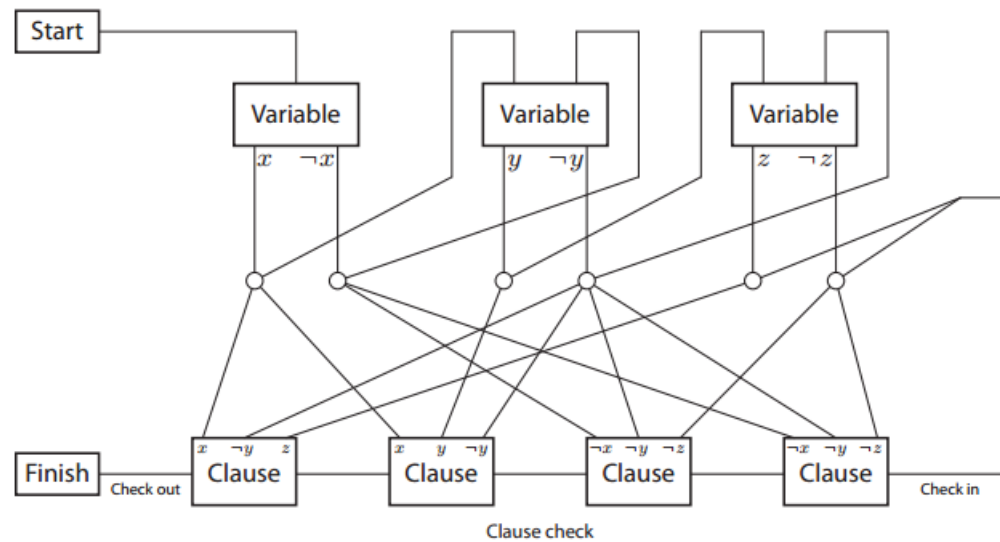


Figure 1: General framework for NP-hardness

In the Game (An example)

- Super Mario Bros.
- No glitches (Ideal game)
 - This can be addressed

In the Game (An example)

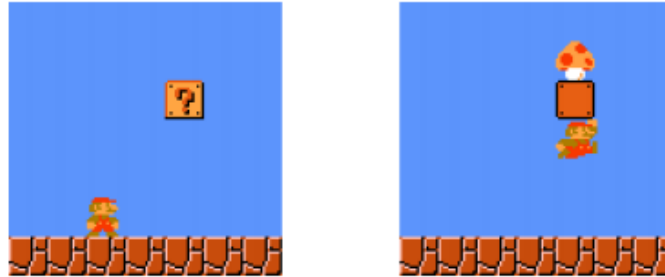


Figure 8: Left: Start gadget for Super Mario Bros. Right: The item block contains a Super Mushroom

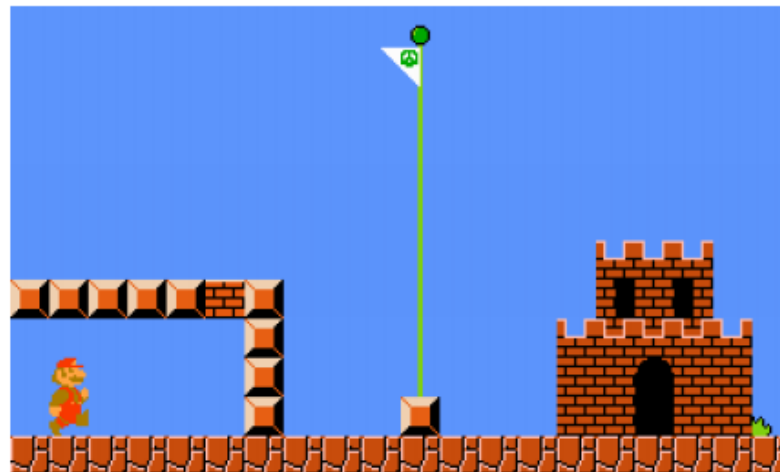


Figure 9: Finish gadget for Super Mario Bros.

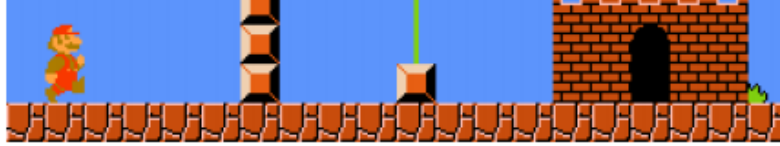


Figure 9: Finish gadget for Super Mario Bros.

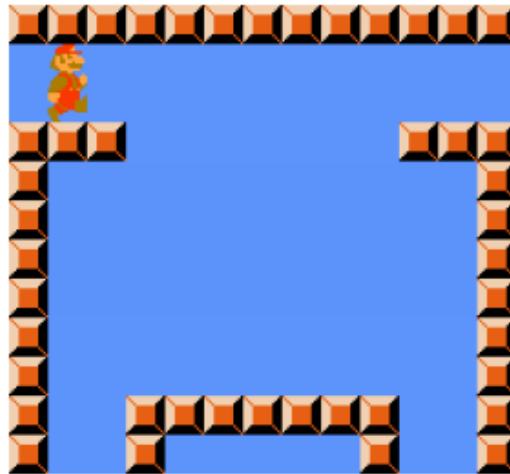


Figure 10: Variable gadget for Super Mario Bros.

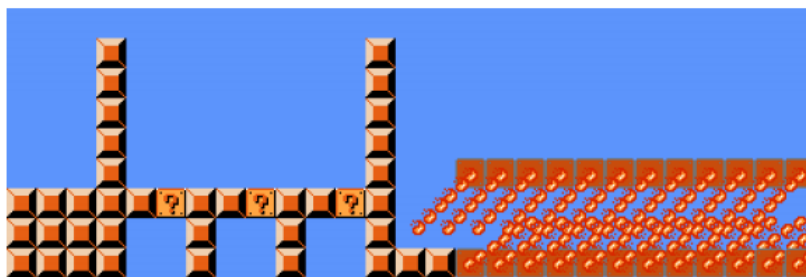


Figure 11: Clause gadget for Super Mario Bros.

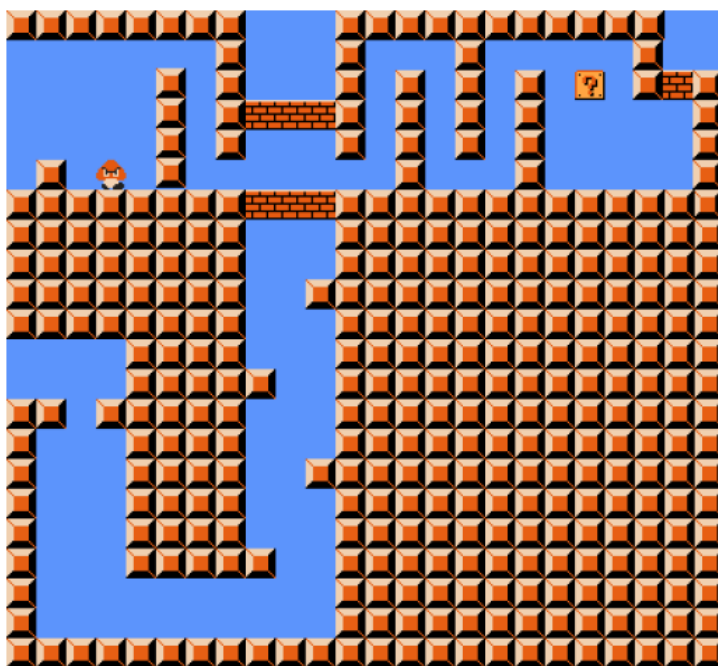


Figure 12: Crossover gadget for Super Mario Bros.

The General Framework

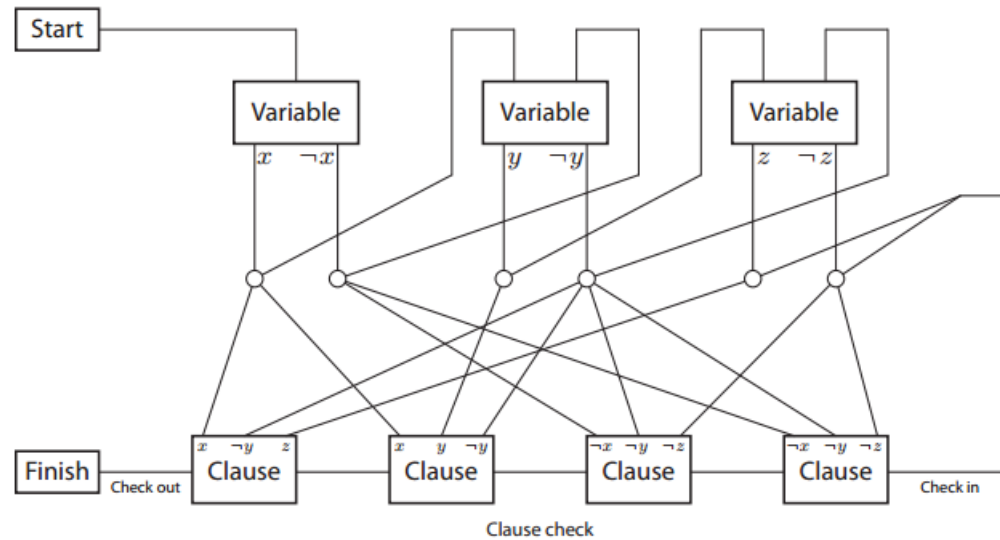


Figure 1: General framework for NP-hardness

Other Games

- Legend of Zelda
- Pokémon
- Donkey Kong Country
- Metroid
- Etc.

Open Questions:

- More Games?
- Something of a novelty
- What else can be done by humans that is (Computationally) hard
- Limited utility in modern games
 - Turing-complete scripting

Closing Complexity Gaps for Coloring Problems on H -Free Graphs

MARJANEH SAFAEI

VILDAN ATALAY

POOYAN BALOUCHIAN

Outline

1. Terminology
2. Goal
3. Classifying Precoloring Extension & 3-List Coloring
4. List Coloring for Complete Graphs Minus a Matching
5. List 4-Coloring for P_6 -Free Graphs
6. Conclusion
7. Questions

1. Terminology (Continued)

- **Graph Coloring:**

- A mapping $c : V \rightarrow \{1, 2, \dots\}$
- No two adjacent vertices of a graph $G=(V, E)$ have the same color; i.e., $c(u) \neq c(v)$ if $uv \in E$; if $|c(V)| \leq k$ then c is a k -coloring

- **H -free Graph:**

- A graph G contains no subgraph isomorphic to some graph H

- **P_r :**

- The path on r vertices in a graph G

- **List Assignment:**

- A function L that assigns a list $L(u)$ of admissible colors to each vertex in G

- **k -List Assignment:**

- $L(u) \subseteq \{1, \dots, k\}$ for each vertex. A coloring c respects L if $c(u) \in L(u)$ for each $u \in V$

1. Terminology (Continued)

- **List Coloring:**
 - Decide whether a given graph allows a coloring, such that every vertex u receives a color from some given set $L(u)$
- **ℓ -list Coloring:**
 - An upper bound ℓ on the size of each $L(u)$
- **List k -coloring:**
 - Decide whether G has a coloring that respects L , when given graph G with a k -list assignment L
- **Precoloring Extension:**
 - Decide for some integer k , whether a partial k -coloring of a graph can be extended to a k -coloring of the whole graph
- **k -Precoloring Extension:**
 - Decide whether we can extend c_W to a k -coloring of G , where
 - $c_W : W \rightarrow \{1, 2, \dots, k\}$ for some integer k
 - $W \subseteq V$ of G

1. Terminology

- k -Coloring as a special case of k -Precoloring Extension
 - $W = \emptyset$
- k -Precoloring Extension as a special case of List k -Coloring
 - $L(u) = \{cw(u)\}$ if $u \in W$ and $L(u) = \{1, \dots, k\}$ if $u \in W \setminus V$.
- List k -Coloring as a special case of k -List Coloring.
- k -Coloring is NP-Complete for G
 - \rightarrow k -Precoloring Extension is NP-Complete for G
 - \rightarrow List k -Coloring is NP-Complete
 - \rightarrow k -List Coloring is NP-Complete



2. Goal

1. Classify the Precoloring Extension problem and the ℓ -List Coloring problem for H -free graphs
2. 3-List Coloring is NP-Complete for n -vertex graphs of minimum degree $n-2$
3. List 4-Coloring is NP-Complete for P_6 -Free graphs

3. Classifying Precoloring Extension & 3-List Coloring (Continued)

- **Theorem 1.** *Let H be a fixed graph. If H is a (not necessarily proper) induced sub-graph of P_4 or of P_1+P_3 , then Coloring can be solved in polynomial time for **H-free** graphs; otherwise, it is NP-Complete for **H-free** graphs.*
 - The disjoint union of two graphs G and H is denoted $G+H$.
- **Lemma 1.** List Coloring can be solved in $\mathbf{O}((n+k)^{5/2})$ time on n -vertex complete graphs with a **k-list assignment**.

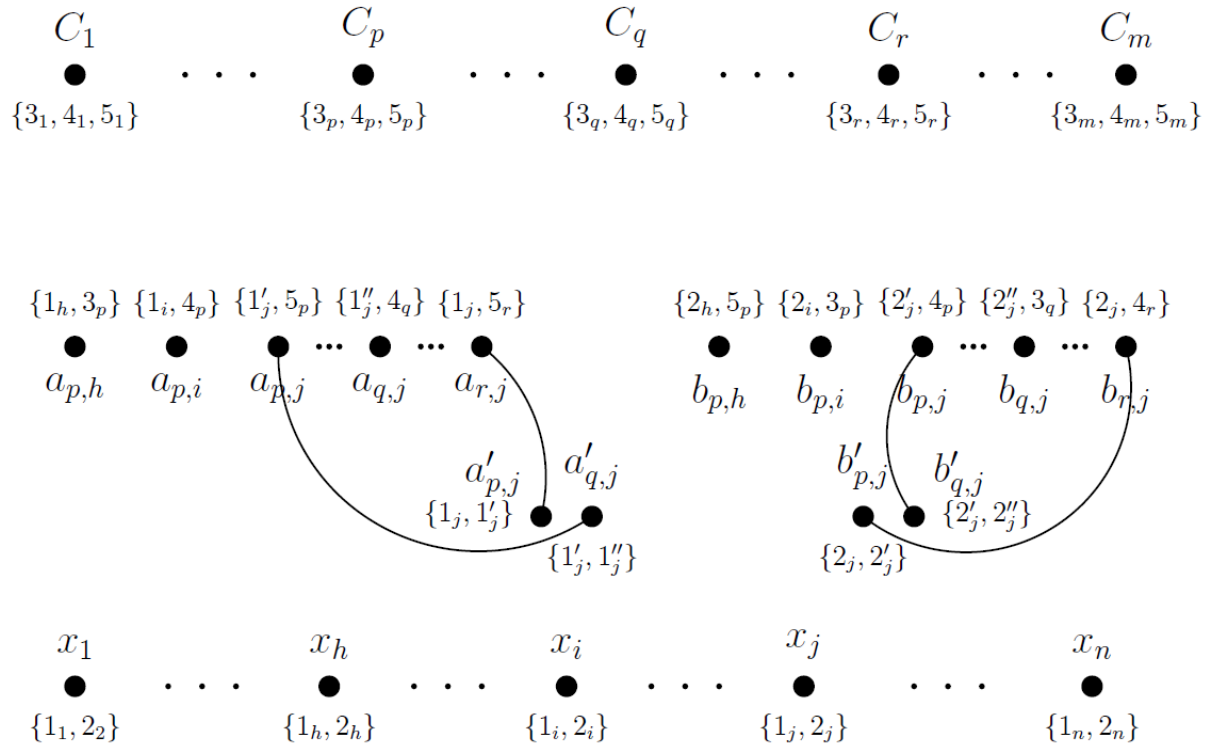
3. Classifying Precoloring Extension & 3-List Coloring

- **Theorem 2.** *Let ℓ be a fixed integer, and let H be a fixed graph. If $\ell \leq 2$ or H is a (not necessarily proper) induced sub-graph of P_3 , then **ℓ -List Coloring** is polynomial time solvable on **H-free** graphs; otherwise, **ℓ -List Coloring** is NP-Complete for **H-free** graphs.*
- **Theorem 3.** *Let H be a fixed graph. If H is a (not necessarily proper) induced sub-graph of P_4 or of $P_1 + P_3$, then **Precoloring Extension** can be solved in polynomial time for **H-free** graphs; otherwise it is NP-Complete for **H-free** graphs.*

4. List Coloring for Complete Graphs Minus a Matching (Continued)

- **Using Reduction: Theorem 4.** The **3-List Coloring** problem is NP-Complete for complete graphs minus a matching.
 - A **REDUCTION** from NOT-ALL-EQUAL(≤ 3 , $2/3$)-SATISFIABILITY.
 - **NOT-ALL-EQUAL-3-SATISFIABILITY:** At least one TRUE and one FALSE literal in each clause
 - **NOT-ALL-EQUAL(≤ 3 , $2/3$)-SATISFIABILITY:** Input is an instance I with the following properties:
 - each C_j contains 2 or 3 literals which are all positive,
 - each literal occurs in at most three clauses.
- **Using Lemma 1: Theorem 5.** The **List Coloring** problem can be solved in $O(2^p(n+k)^{5/2})$ time on pairs (G, L) where G is an n -vertex graph with p pairs of nonadjacent vertices and L is a k -list assignment.

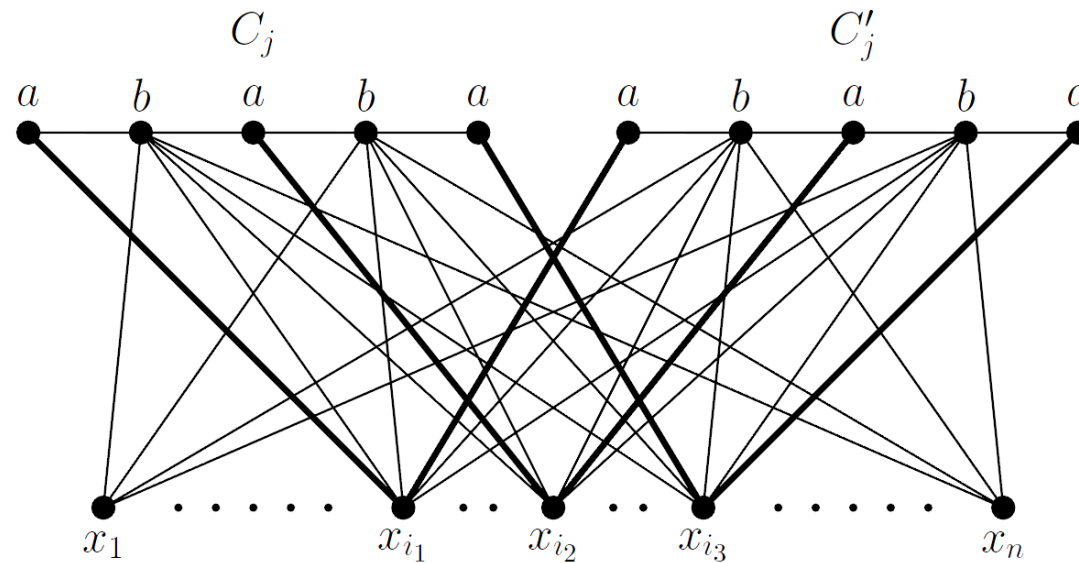
4. List Coloring for Complete Graphs Minus a Matching



An example of a graph $\overline{G_I}$ in which a clause C_p and a variable x_j are highlighted. Note that in this example C_p is a clause with ordered variables x_h, x_i, x_j , and that x_j is a variable contained in ordered clauses C_p, C_q and C_r .

5. List 4-Coloring for P_6 -Free Graphs

- **Using Reduction: Theorem 6.** The List 4-Coloring problem is NP-complete for P_6 -free graphs.
 - Reduce from NOT-ALL-EQUAL-3-SATISFIABILITY with positive literals
 - Given an instance I of NOT-ALL-EQUAL-3-SATISFIABILITY, build a graph G_I with 4-list assignment L .
 - G_I is shown to be P_6 -free and has a coloring that respects L if and only if I has a satisfying truth assignment.



The graph G_I for the clause $C_j = \{x_{i_1}, x_{i_2}, x_{i_3}\}$

6. Conclusion

- **k-Coloring, k-Precoloring Extension** and **List k-Coloring** behave similarly on P_6 -free graphs
- The NP-Completeness result obtained on List 4-Coloring for P_6 -free graphs indicates:
 - 4-Coloring for P_6 -free graphs is NP-Complete
 - New proof techniques not based on subroutines that solve List 4-Coloring are required for proving polynomial-time solvability.

7. Questions

Determine the computational complexity of:

- a) coloring for AT-free graphs
- b) List 4-Coloring for $(P_2 + P_3)$ -free graphs, for $2P_3$ -free graphs and for $(P_2 + P_4)$ -free graphs?
- c) The open cases marked “?” in Table 1

| r | k -COLORING | | | | k -PRECOLORING EXTENSION | | | | LIST k -COLORING | | | |
|------------|---------------|---------|---------|------------|----------------------------|---------|---------|------------|--------------------|-------------|---------|------------|
| | $k = 3$ | $k = 4$ | $k = 5$ | $k \geq 6$ | $k = 3$ | $k = 4$ | $k = 5$ | $k \geq 6$ | $k = 3$ | $k = 4$ | $k = 5$ | $k \geq 6$ |
| $r \leq 5$ | P | P | P | P | P | P | P | P | P | P | P | P |
| $r = 6$ | P | ? | ? | ? | P | ? | NP-c | NP-c | P | NP-c | NP-c | NP-c |
| $r = 7$ | ? | ? | ? | NP-c | ? | NP-c | NP-c | NP-c | ? | NP-c | NP-c | NP-c |
| $r \geq 8$ | ? | NP-c | NP-c | NP-c | ? | NP-c | NP-c | NP-c | ? | NP-c | NP-c | NP-c |

Table 1. The complexity of k -COLORING, k -PRECOLORING EXTENSION and LIST k -COLORING on P_r -free graphs for fixed k and r . The bold entry is our new result.

Deterministic Function Computation with Chemical Reaction Networks

Ho-Lin Chen, David Doty, David Soloveichik

Fast Forward Presentation by

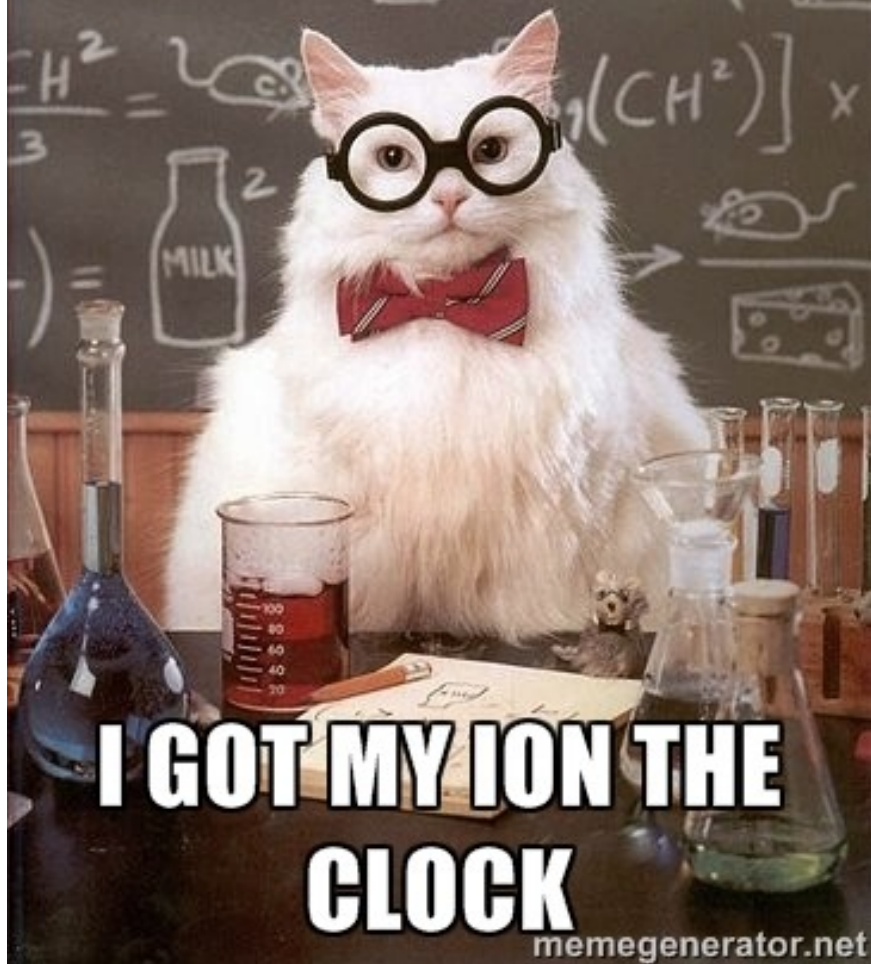
Brent Miller

&

Emily Sassano

CHEMICAL REACTION NETWORKS IN 5

MIN



**I GOT MY ION THE
CLOCK**

Chemical Reaction Networks (CRN): Motivation

- A cell's method for computation and memory
- Widely used to model signaling and regulatory networks
 - The computation that is achievable by CRNs is not well understood
- We need a better understanding to engineer controllers for artificial biochemical systems

Chemical Reaction Network Model

- Finite set of species {A, B, C..}. Each with a state (non-negative integer) representing the molecular count of each species.
- Finite set of reactions.



- This reaction decrease both A and B's molecular count by 1 and increases C's by 1.
- Each reaction also has an associated rate constant k in units of liters x molecules⁻¹ x sec⁻¹

Chemical Reaction Network Model

| Reaction j | Probability a_j of the reaction in time instance dt |
|---------------------|--|
| $A \rightarrow$ | $k \cdot \#A$ |
| $A + B \rightarrow$ | $k/v \cdot \#A \cdot \#B$ |
| $A + A \rightarrow$ | $k/v \cdot \#A \cdot (\#A-1)/2$ |

V = volume of solution
 K = rate constant
 $\#$ = the number of molecules

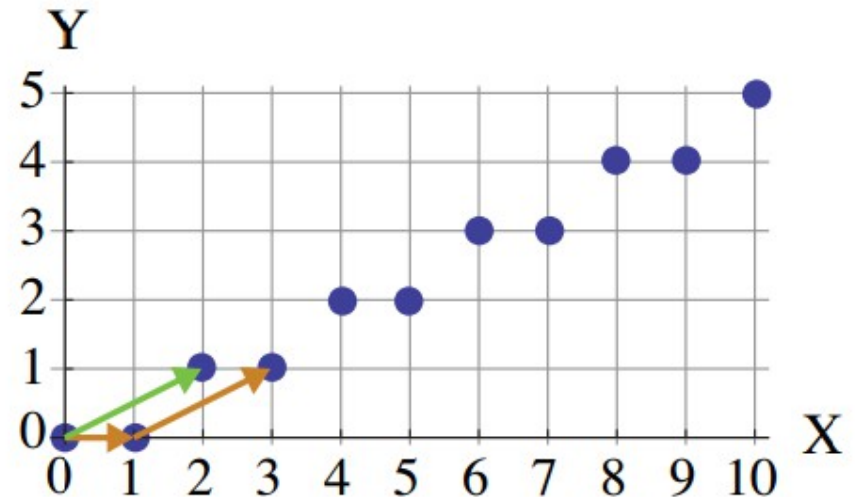
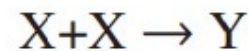
CRCs and Semilinear Functions

- Every function that is stably computed by a CRC (chemical reaction computer) is semilinear because we can stably decide the graph of f with a CRD (chemical reaction decider).

$$f(x) = \lfloor x/2 \rfloor$$

start with: (input) X

output: Y

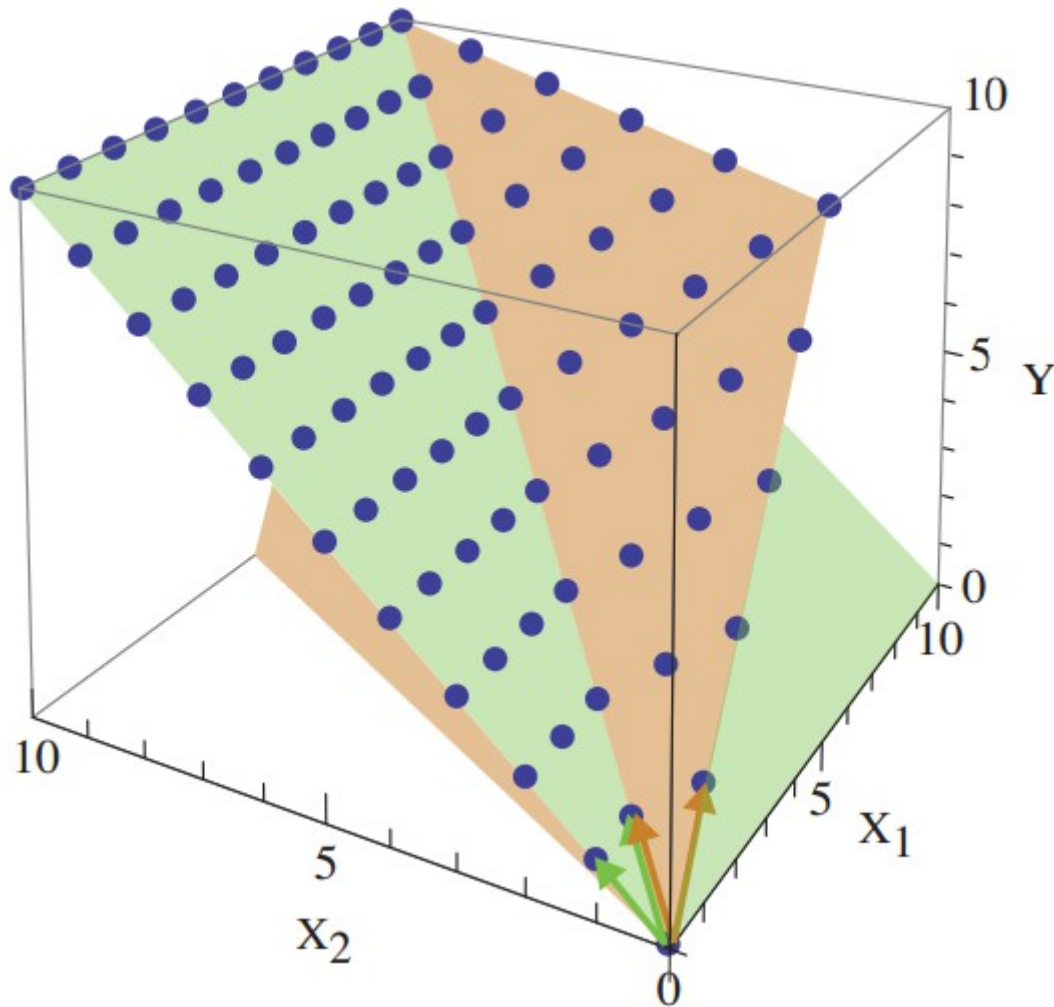


$$\{n_1 \cdot (2, 1) \mid n_1 \in \mathbb{N}\} \cup$$

$$\{(1, 0) + n_1 \cdot (2, 1) \mid n_1 \in \mathbb{N}\}$$

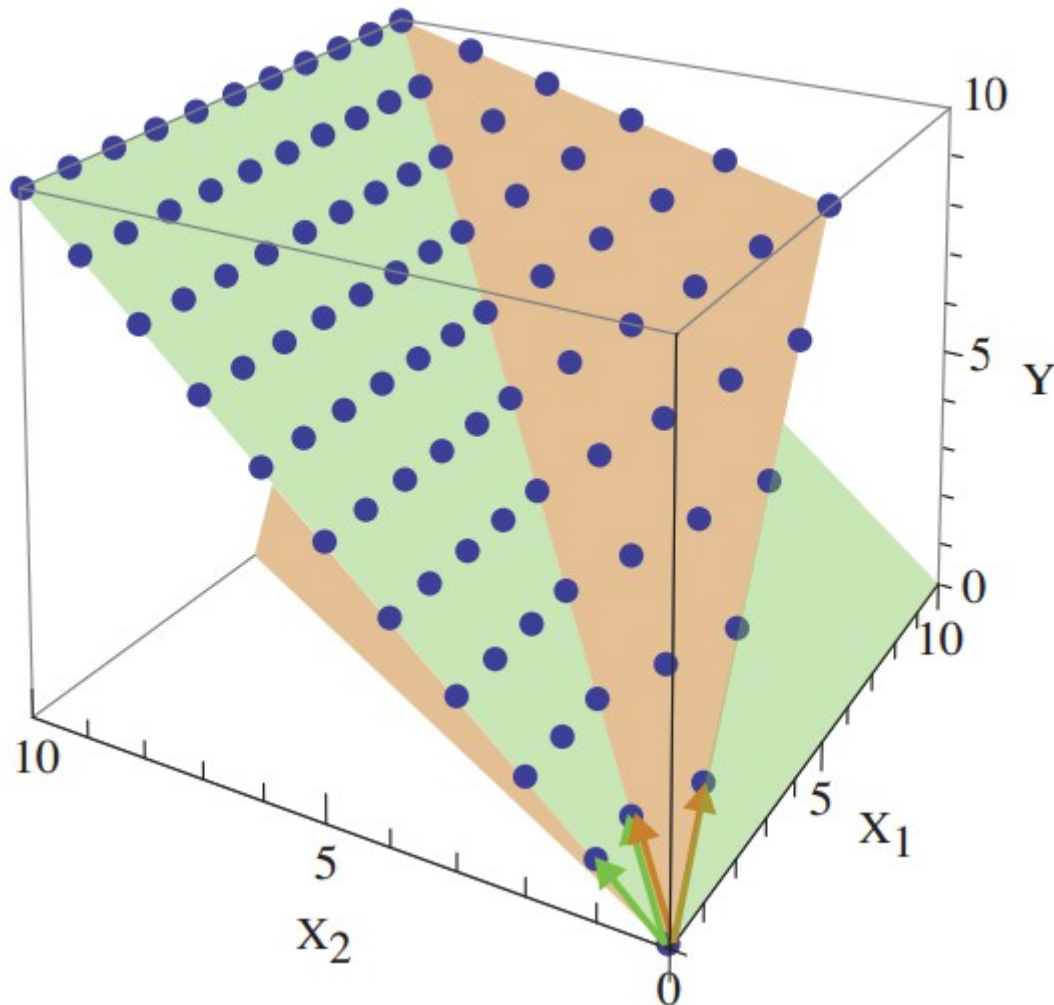
Semilinear Functions as Piecewise Linear Functions

$$f(x_1, x_2) = \begin{cases} g(x_1, x_2) = 2x_1 - x_2 & \text{if } x_1 > x_2 \\ h(x_1, x_2) = x_2 & \text{otherwise} \end{cases}$$



Semilinear Functions as Piecewise Linear Functions

$$f(x_1, x_2) = \begin{cases} g(x_1, x_2) = 2x_1 - x_2 & \text{if } x_1 > x_2 \\ h(x_1, x_2) = x_2 & \text{otherwise} \end{cases}$$



- Every semilinear function can be piecewise defined by linear functions with decisions as semilinear predicates

Deciding a Semilinear Function

- Given this simple piecewise definition,

$$f(x_1, x_2) = \begin{cases} g(x_1, x_2) = 2x_1 - x_2 & \text{if } x_1 > x_2 \\ h(x_1, x_2) = x_2 & \text{otherwise} \end{cases}$$

the following computation is done in parallel.

Deciding a Semilinear Function

- Given this simple piecewise definition,

$$f(x_1, x_2) = \begin{cases} g(x_1, x_2) = 2x_1 - x_2 & \text{if } x_1 > x_2 \\ h(x_1, x_2) = x_2 & \text{otherwise} \end{cases}$$

the following computation is done in parallel.

- Each of the m partial functions f_i is computed by a CRC

Deciding a Semilinear Function

- Given this simple piecewise definition,

$$f(x_1, x_2) = \begin{cases} g(x_1, x_2) = 2x_1 - x_2 & \text{if } x_1 > x_2 \\ h(x_1, x_2) = x_2 & \text{otherwise} \end{cases}$$

the following computation is done in parallel.

- Each of the m partial functions f_i is computed by a CRC
- Each semilinear predicate is computed by a CRD

Deciding a Semilinear Function

- Given this simple piecewise definition,

$$f(x_1, x_2) = \begin{cases} g(x_1, x_2) = 2x_1 - x_2 & \text{if } x_1 > x_2 \\ \hline h(x_1, x_2) = x_2 & \text{otherwise} \end{cases}$$



the following computation is done in parallel.

- Each of the m partial functions f_i is computed by a CRC
- Each semilinear predicate is computed by a CRD
- A single predicate will stabilize to true, and it will act as an “activator” for the output of the corresponding CRC

Exciting Questions!

- Why are the behaviors used in biology \ll then the behaviors of all “syntactically correct” CRNs?
 - Why does biology use this limited set of behaviors?
 - In an artificial biological system are we also constrained to using this set of behaviors?

Exploring The Turing and Super-Turing Computational Power of Neural Networks

A collection of papers by:
Hava Siegelmann et. al.

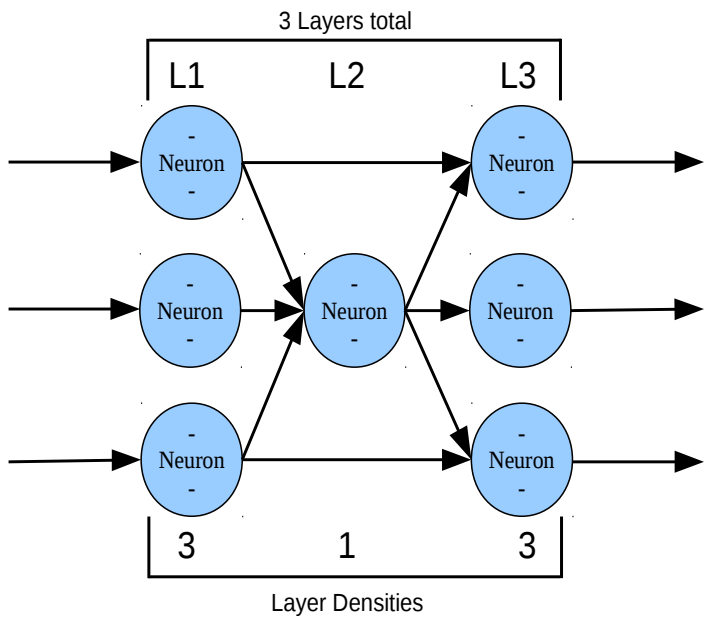
Presented by:
Gene Sher
Joshua Kuxhausen

Overview

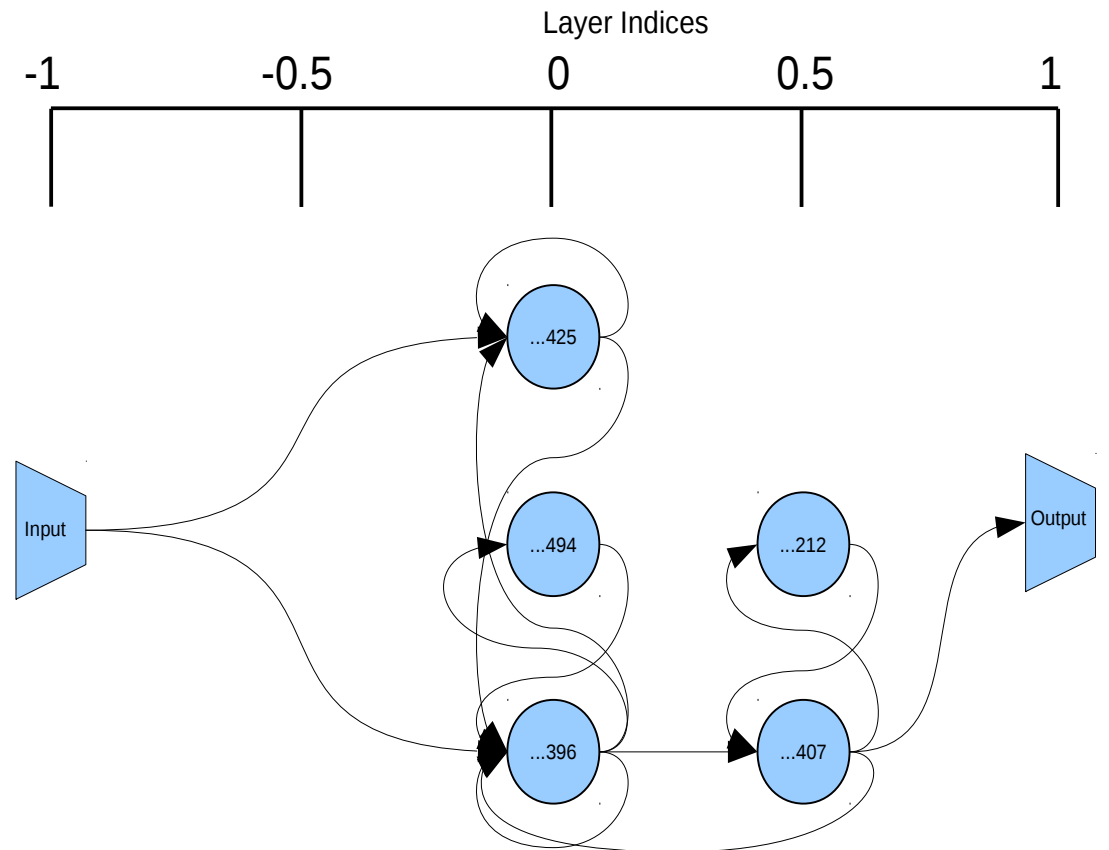
- What's a Neural Network?
- How Powerful Are Neural Networks?
- Rational Valued Recurrent Neural Networks
- Real Valued Recurrent Neural Networks
- Adaptive Rational Valued Neural Networks
- Adaptive Real Valued Neural Networks
- Physical Implementation
- What does it all mean?

Neural Network

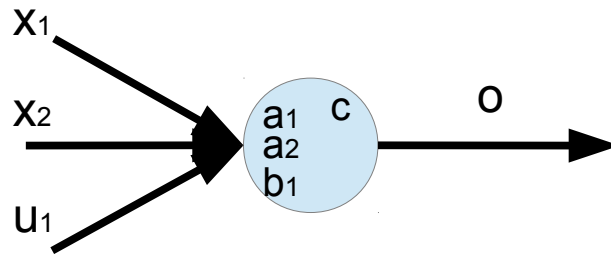
Feedforward



Recurrent



Neural Process using a saturated-linear function

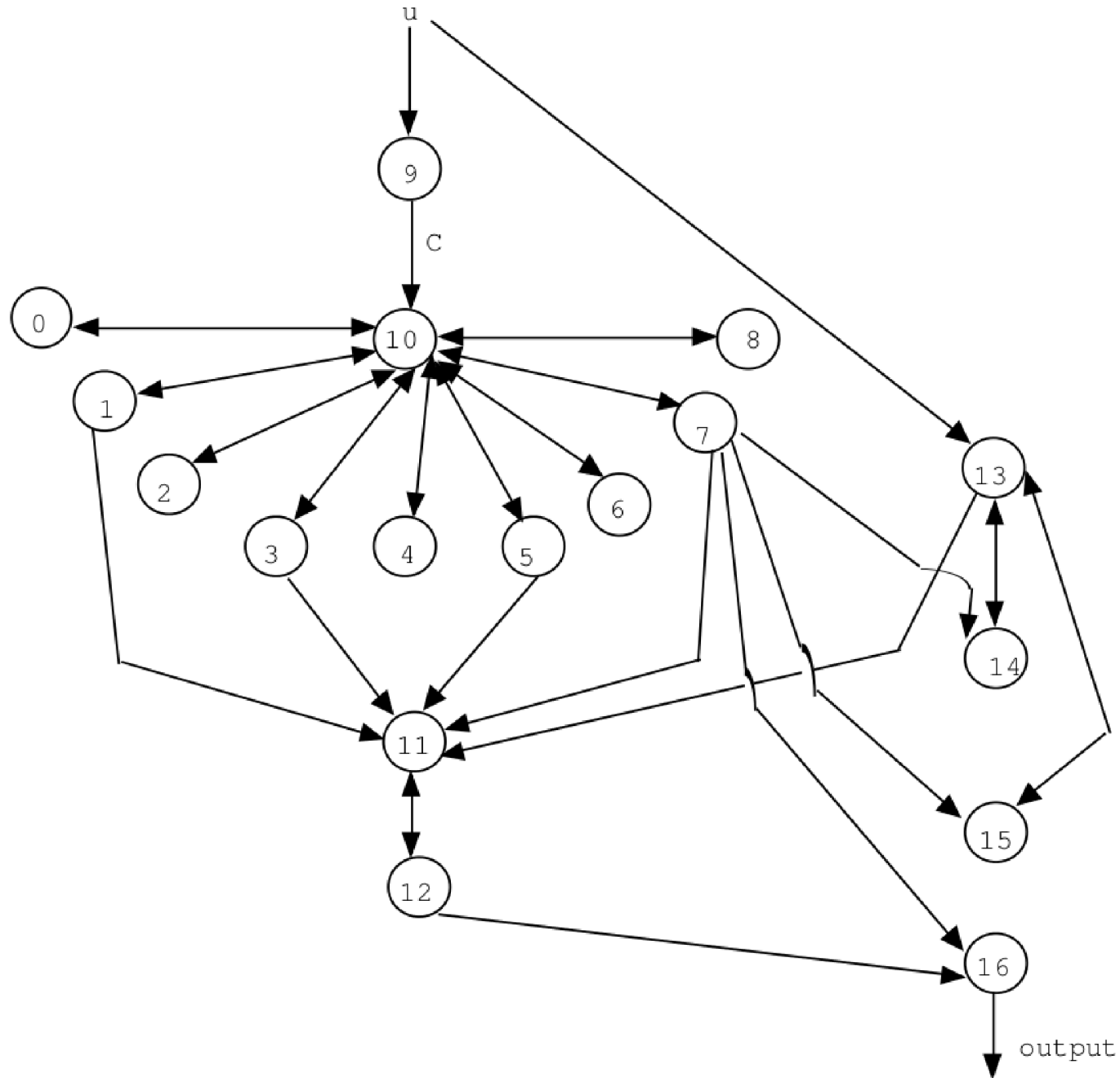


$$x_i(t + 1) = \sigma \left(\sum_{j=1}^N a_{ij} \cdot x_j(t) + \sum_{j=1}^M b_{ij} \cdot u_j(t) + c_i \right),$$

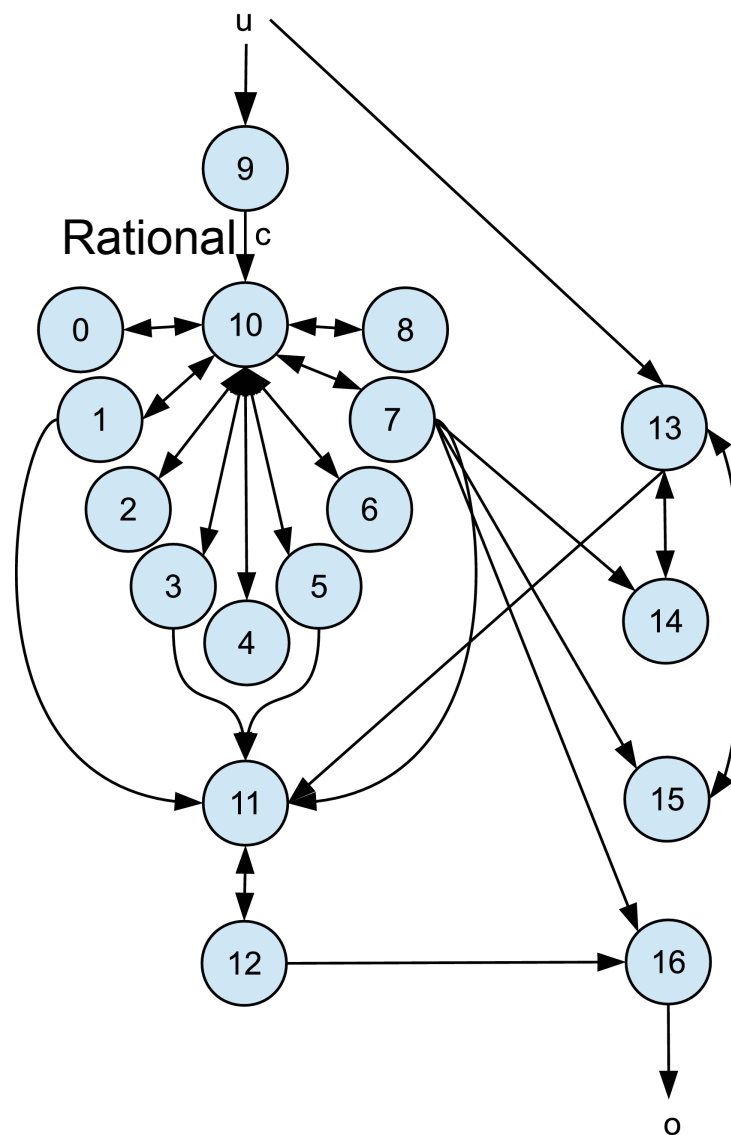
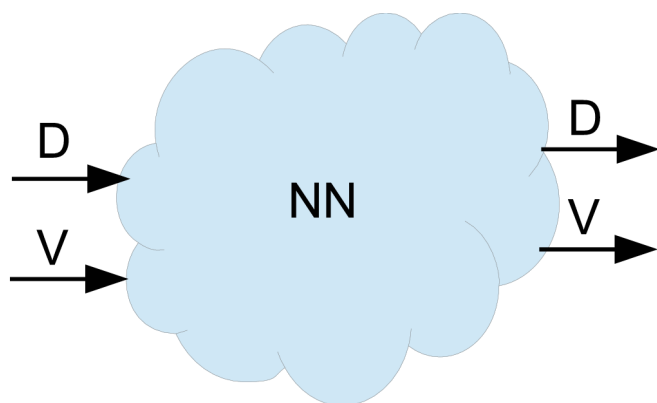
$i = 1, \dots, N$

$$\sigma(x) = \begin{cases} 0 & \text{if } x < 0, \\ x & \text{if } 0 \leq x \leq 1, \\ 1 & \text{if } x > 1. \end{cases}$$

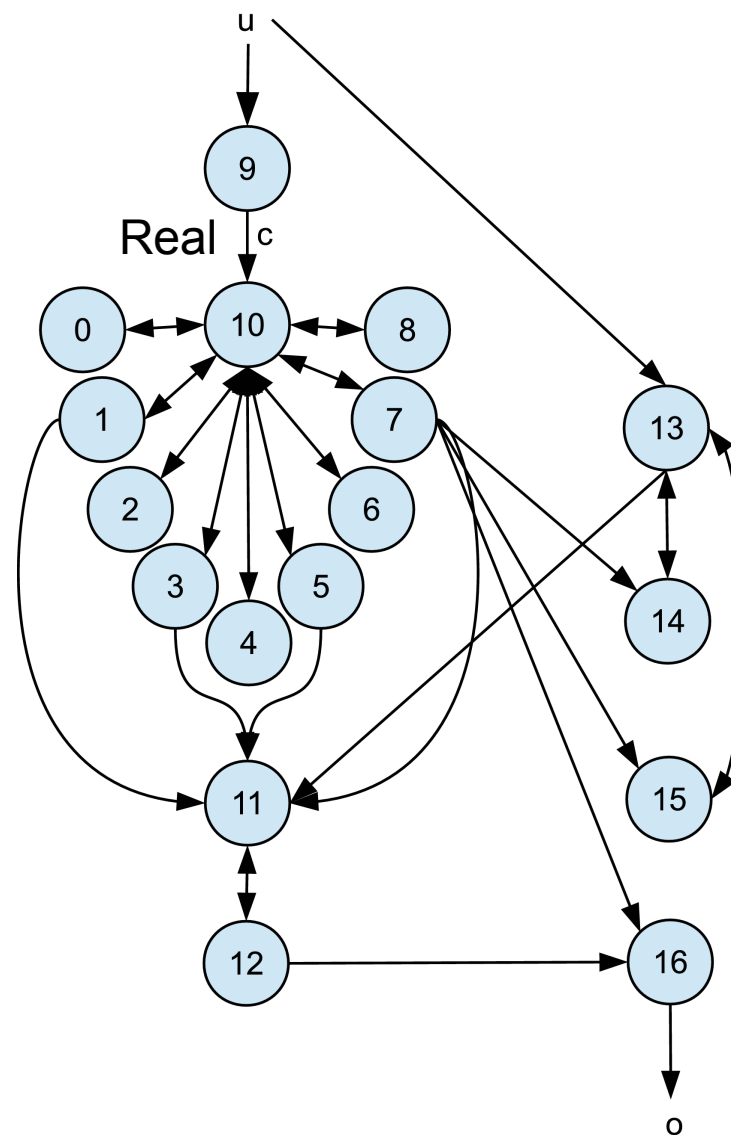
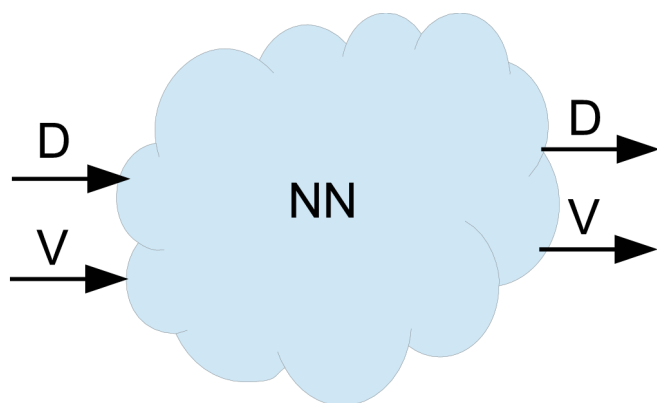
Retrieval Network



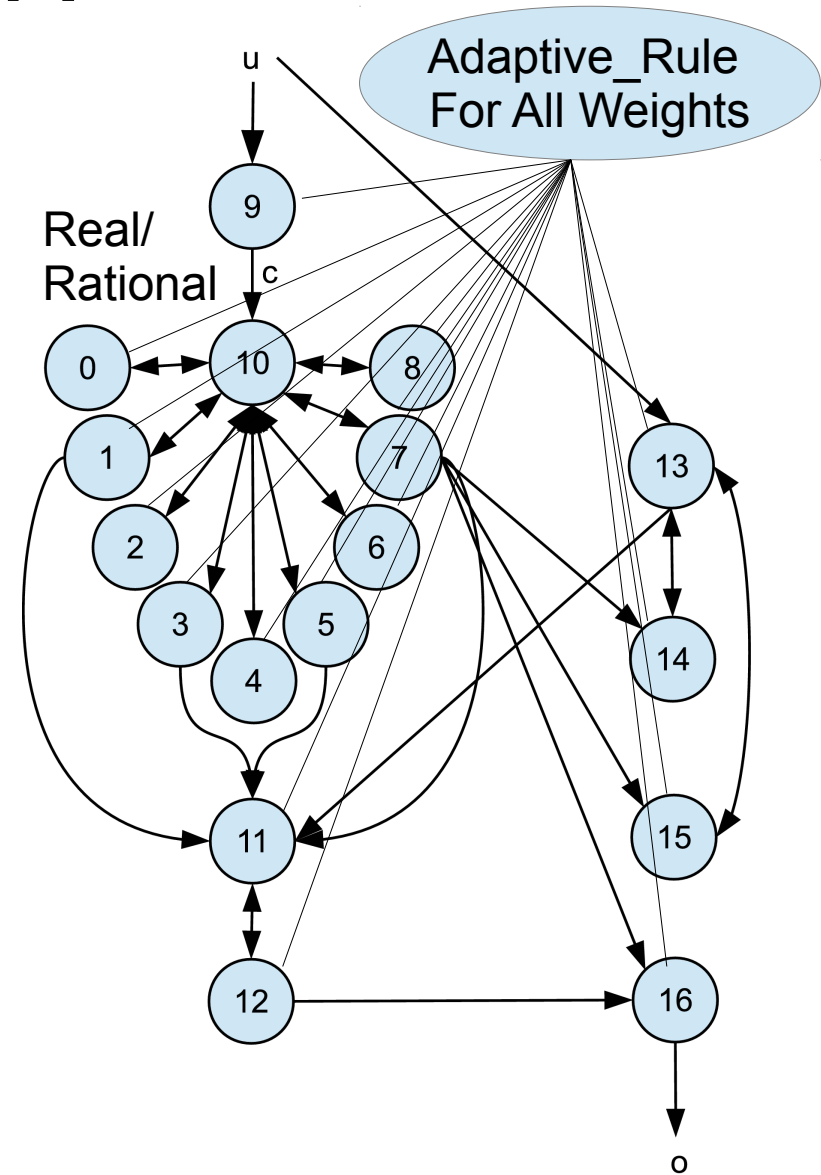
Rational Valued RNN



Real Valued RNN



Evolving Rational & Real Valued RNN



Both rational and real valued evolving/adaptive RNNs are super-turing, and equivalently as powerful.

Are any of these really realizable?

- *Thorough analysis of this system was performed by, and flaws found by, Douglass Keith:
Douglas, Keith. "Super-Turing Computation: A Case Study Analysis." Unpublished MS Thesis, Carnegie Mellon University (2003).*
- *"Since this weight can be an arbitrary real numbered value, it can do its trick by simply having the non-Turing computable arrangement of gates stored in it as a weight. (This makes it not too surprising that the Siegelmann networks can do one thing super-Turing: whether they can compute other super-Turing functions is not discussed.)"*
- *"She does not give the proof of the above result herself, but instead appeals to a volume by Balcázar, Díaz and Cabarró 1995. However, this volume does not support her claim as fully stated."*
- *"Note the difficulty even in computing a constant function. Since the weights of each node in a Siegelmann network are of infinite precision, outputting their value directly is impossible by the protocol described."*
- *"Siegelmann's networks require infinite sensitivity in order to make use of infinitely large "registers" (interpreting them in the usual way as containing a number between 0 and 1). Siegelmann uses Cantor set style encoding (in order to minimize the difficulties in recognition between two close register values). However, this trades one problem for another. How do the Cantor sets in turn get represented?"*
- *How exactly is it super turing "The class of languages decided by rational and real Ev-RNN's in polynomial time corresponds precisely to the complexity class P/poly"? Publications spanning the past 20 years, but details in all papers are vague.*
 - *Journal of computer and system sciences (1991)*
 - *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions (1997)*
 - *Applied Mathematics Letters*
 - *Science 268.5210 (1995)*
 - *The 2011 International Joint Conference on Neural Networks (IJCNN). IEEE, 2011.*

The End

Questions:

1. What do these results mean for the field of Computational Intelligence?
2. Are such systems realizable?

A Comprehensive Study of Self-Assembly



Presented by:

Anahita Davoudi
Ruijun Wang

Date: April 29 2014

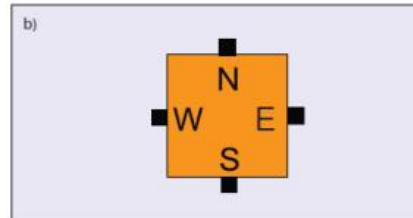
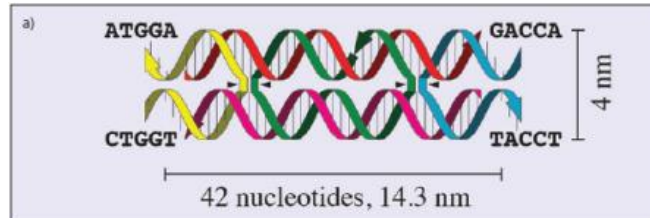
Outline

- Introduction about Self-Assembly
- A popular example of self-Assembly
 - DNA Tile self-Assembly model
 - Error correction methods
- Complexity
- Types
- Applications
- Examples

Self-Assembly

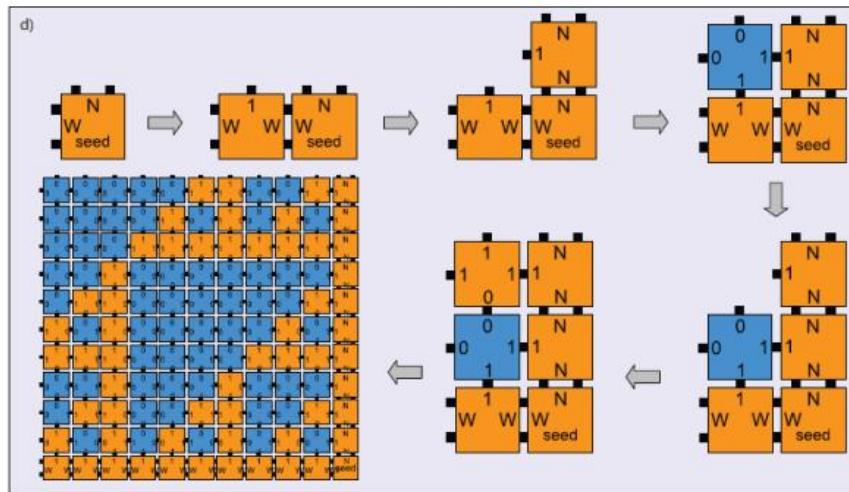
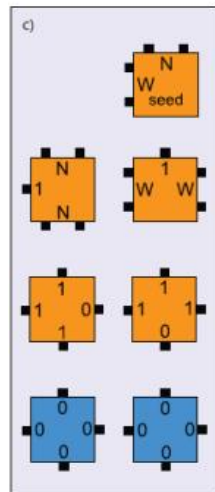
- Theory of Algorithmic Self-Assembly
- A practical implementation of this method was conducted using DNA as the main basis.
- Computing and Computational Universality

Abstract Tile Assembly Model



(a) Double-crossover tile with four sticky end

(b).Representation of a tile as a square with sides labeled by string “glues”

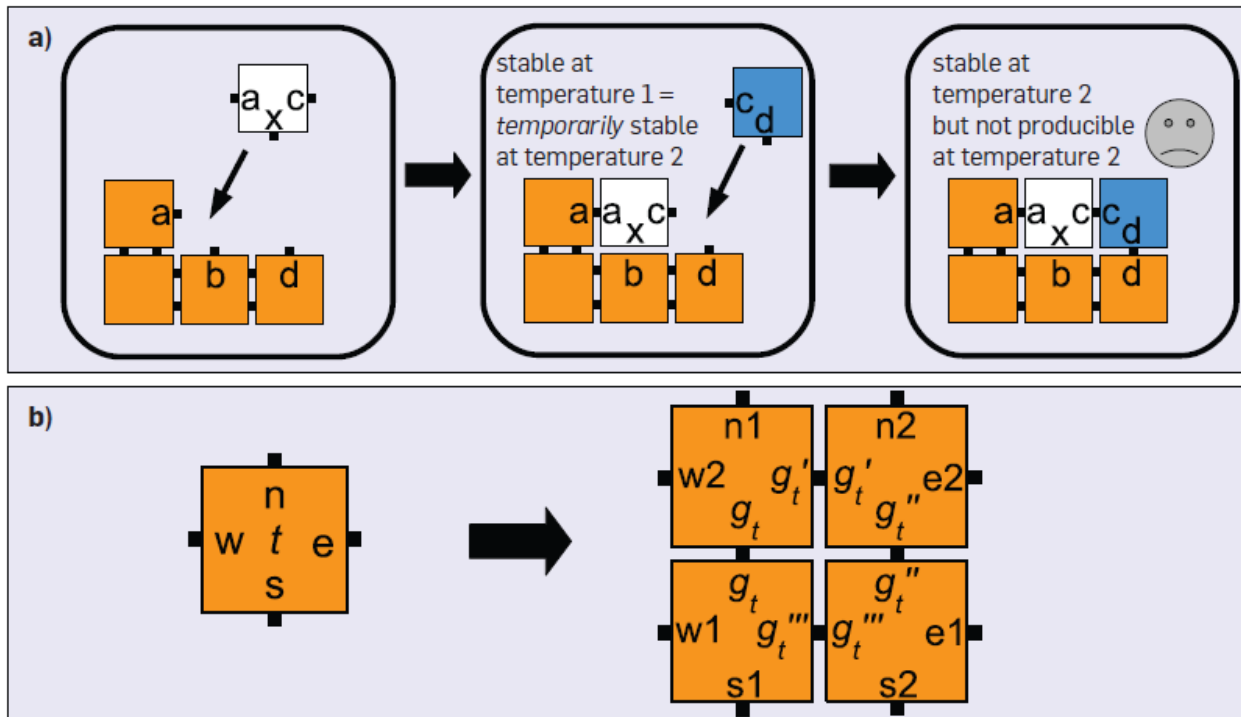


(c). Seven tile types. Bond strengths indicated by the number of small black squares on a side: total strength 2 is required to attach a tile to a partially formed assembly of tiles. One tile type is designated as the end, from which growth is assumed to nucleate.

(d). Growth of the tiles into an assembly with the discrete Sierpinski triangle pattern.

[1] D. Doty, “Theory of Algorithmic self-assembly”, Communications of the ACM, vol. 55, Issue 12, pp. 78-88, Dec 2012.

Error correction



(a). Growth error

(b). 2×2 proofreading. Each tile type t is replaced by a $k \times k$ block of tile types, with glues internal to the block unique to t .

[1] D. Doty, "Theory of Algorithmic self-assembly", Communications of the ACM, vol. 55, Issue 12, pp. 78-88, Dec 2012.

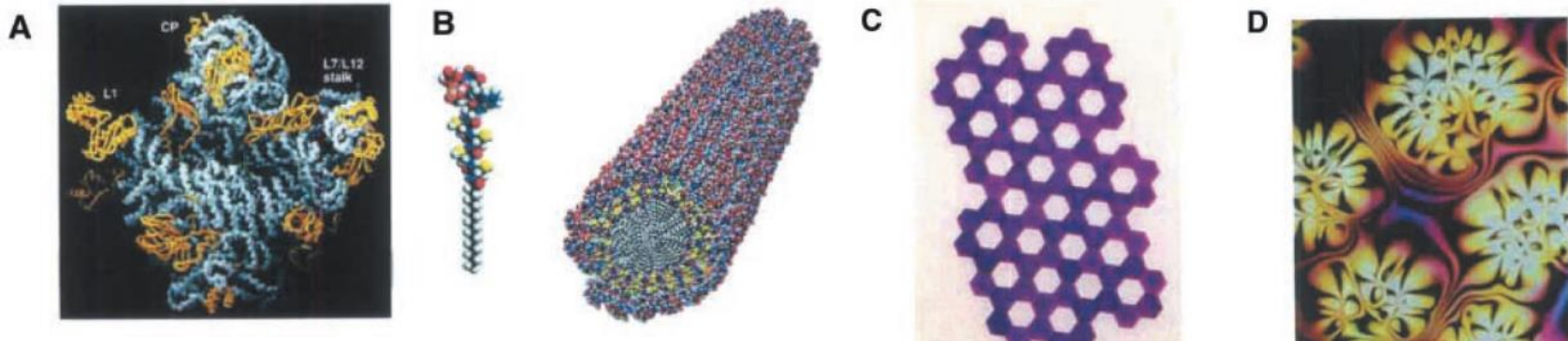
Complexity

- Tile Complexity
- Computational Complexity
 - NP-Complete
- Assembly Time Complexity
 - Linear for a shape of size N

Self-Assembly

Types of Self-Assembly

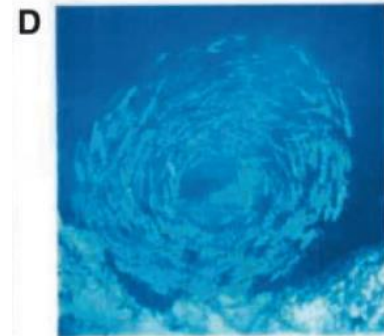
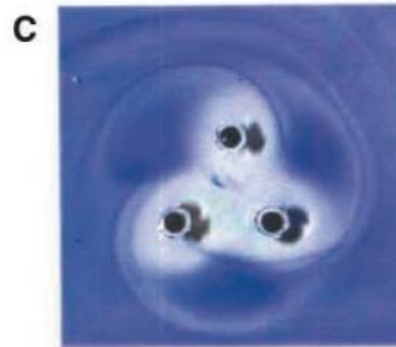
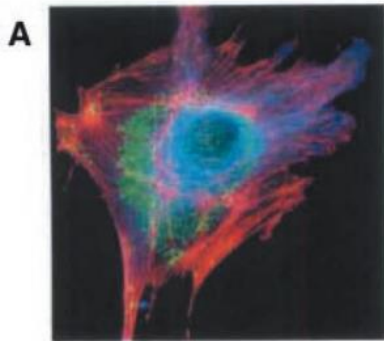
- **Static:** involves systems that are at global or local equilibrium and do not dissipate energy.



Self-Assembly

Types of Self-Assembly

- **Dynamic:** the interactions responsible for the formation of structures or patterns between components only occur if the system is dissipating energy.



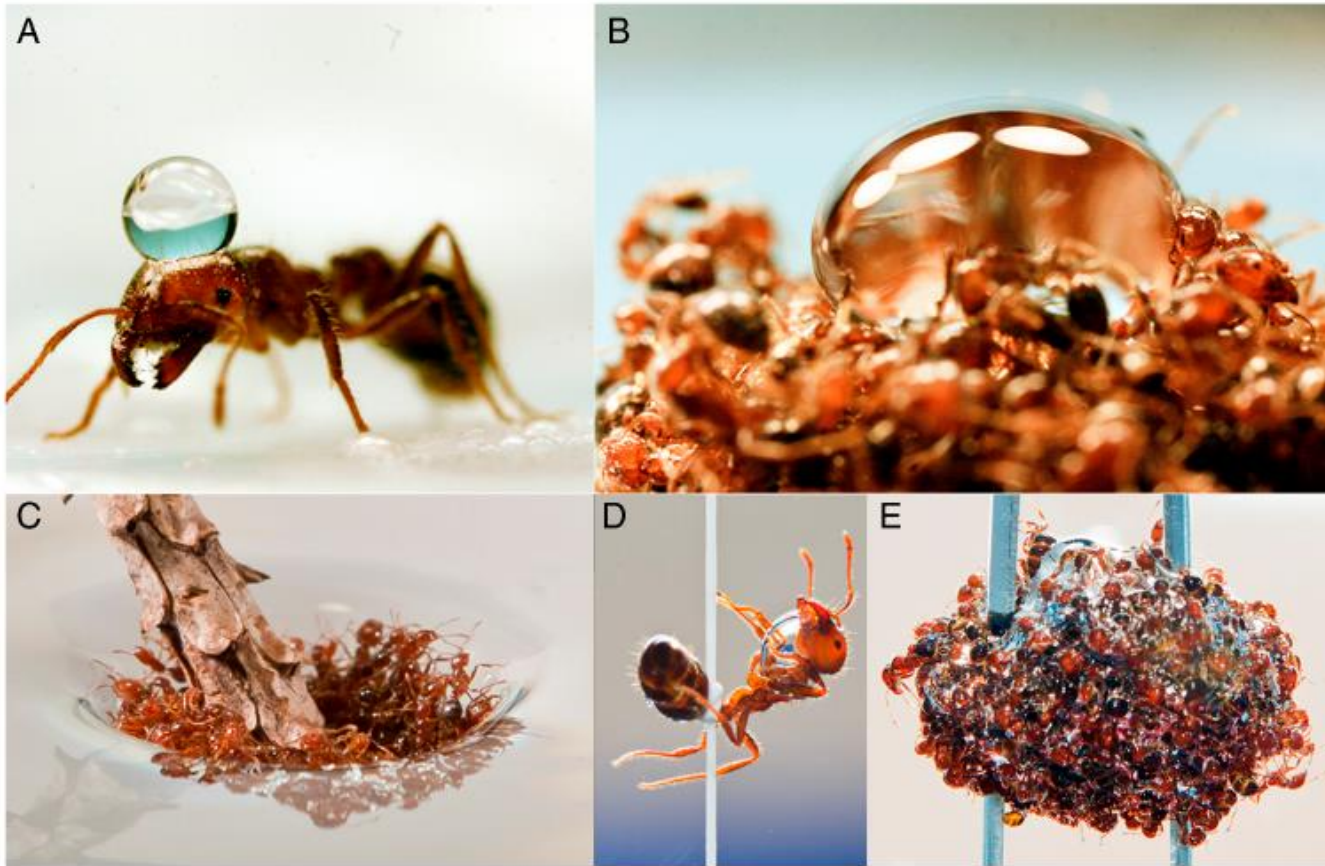
Self-Assembly

Present and Future Application

- Crystallization at All Scales
- Robotics and Manufacturing
- Nano-science and Technology
- Microelectronics
- Netted Systems

Although self-assembly originated in the study of molecules, it is a strategy that is, in principle, applicable at all scales.

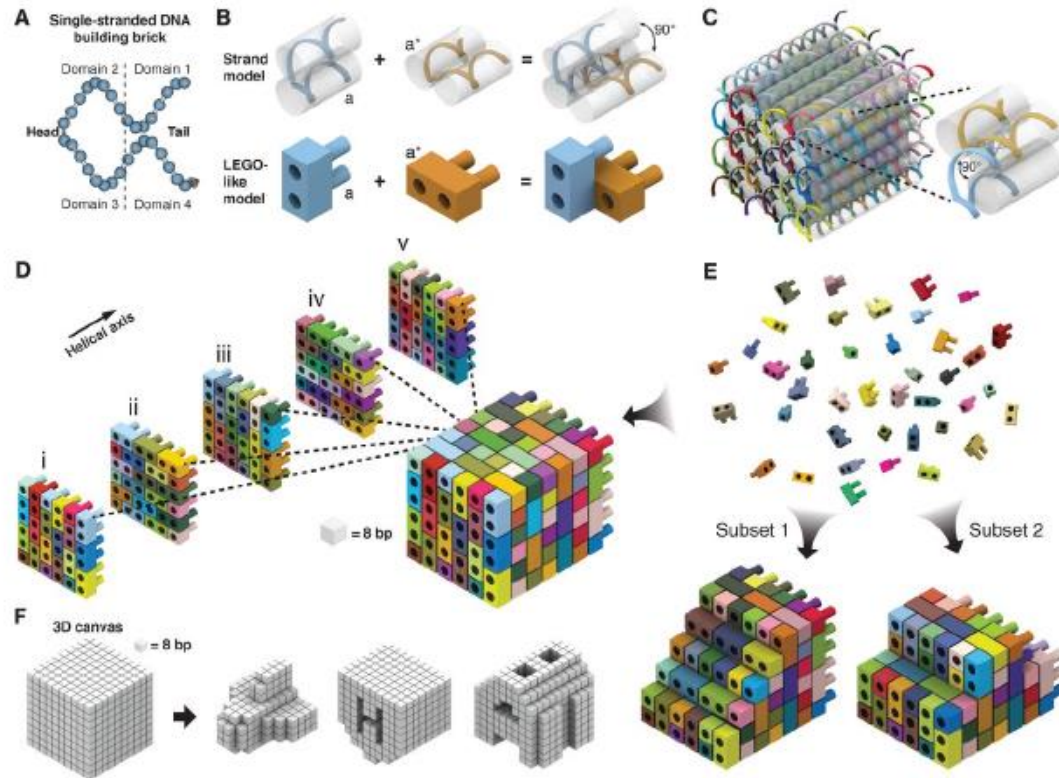
Fire ants self-assemble into waterproof rafts to survive floods



University of Central Florida

Self-Assembly

DNA Brick Structure Similar to LEGO Brick Structure

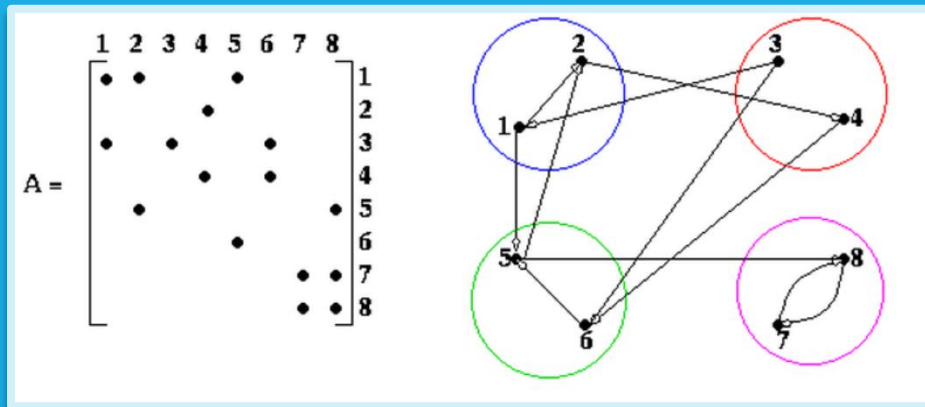


Reference

- [1] D. Doty, “Theory of Algorithmic self-assembly”, *Communications of the ACM*, vol. 55, Issue 12, pp. 78-88, Dec 2012.
- [2] M. Cook, Y. Fu, and R. Schweller, “Temperature 1 self-assembly: deterministic assembly in 3D and probabilistic assembly in 2D”, *SIAM: Proceedings of the twenty-second annual ACM-SIAM symposium on Discrete Algorithms (SODA’11)*, Jan 2011.
- [3] N. Bryans, E. Chiniforooshan, D. Doty, and L. Kari, “The Power of Nondeterminism in Self-Assembly”, *SIAM: Proceedings of the twenty-second annual ACM-SIAM symposium on Discrete Algorithms (SODA’11)*, Jan 2011.
- [4] George M. Whitesides and Bartosz Grzybowski, “Self-Assembly at All Scales”, *Science*, Vol. 295, Issue 2418, (2002).
- [5] Maik Hadorn, Eva Boenzli, Kristian T. Sørensen, Harold Fellermann, Peter Eggenberger Hotz, and Martin M. Hanczyc, “Specific and reversible DNA-directed of oil-in-water emulsion droplets”, *PNSA*, Vol. 109, Issue 50, (2012).
- [6] Nathan J. Mlot, Craig A. Tovey and David L. Hu, “Fire ants self-assemble into waterproof rafts to survive floods”, *PNSA.*, (2011).

Question!

The tile self-assembly is build on two assumptions, the one is tiles never detach from an assembly, the other is tiles only attach when their binding strength exceed the temperature threshold. These are not the truth, how can we improve these assumptions to make the model more realistic?



ON OPTIMAL AND BALANCED SPARSE MATRIX PARTITIONING PROBLEMS

Fast Forward
 by Kyle Burnett, Brian Woods,
 John Edison

THE PROBLEM - DEFINITION

- A constrained version of the parallel sparse matrix-vector multiplication problem can be reduced to a form of the hypergraph partitioning problem
- Hypergraph models have proved to be convenient abstractions in formulating the partitioning problems
- The hypergraph partitioning problems involves dividing a given hypergraph into two or more parts while satisfying some balancing condition

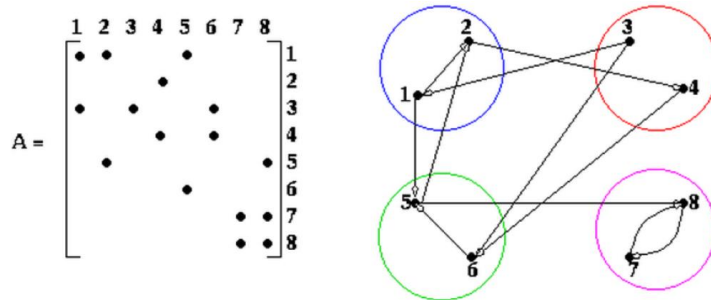


Figure 1: Partitioning a Sparse Matrix [2]

THE PROBLEM - APPLICATIONS

- Sparse matrix partitioning is an important problem arising in many applications including
 - sparse matrix ordering
 - parallelization of sparse matrix vector multiplication operations for distributed shared memory systems
 - crypto-system analysis
- All these applications, although different in nature, have two distinct partitioning constraints and three distinct partitioning objective functions [3]

THE PROBLEM - COMPLEXITY

- The standard one dimensional (1D) partitioning scheme of sparse matrices partitions either the rows or the columns of the matrix
- The standard 1D sparse matrix partitioning problem which is equivalent to the hypergraph partitioning problem is NP-hard
- In this work, we consider a variant of the standard 1D partitioning problem, called order restricted 1D partitioning problem
- A modified version of the algorithm presented by B. W. Kernighan is used to solve the restricted problem [3]

CUT-NET METRIC

- The first metric that the authors cover is the cut-net metric. We begin with the same collection of vertices V and no edges. For each net n_i in H , we add an edge from the first vertex in n_i to the last vertex in n_i with weight 1. If the vertex already exists, we add 1 to the weight. In the figure on the right, we start from the hypergraph in the top left. For h_1 , we add an edge from 1 to 4 with weight one. For h_2 , we add an edge from 1 to 3 with weight one. Since there is already an edge from 1 to 4, for h_3 , we simply increase the weight of that edge by one.

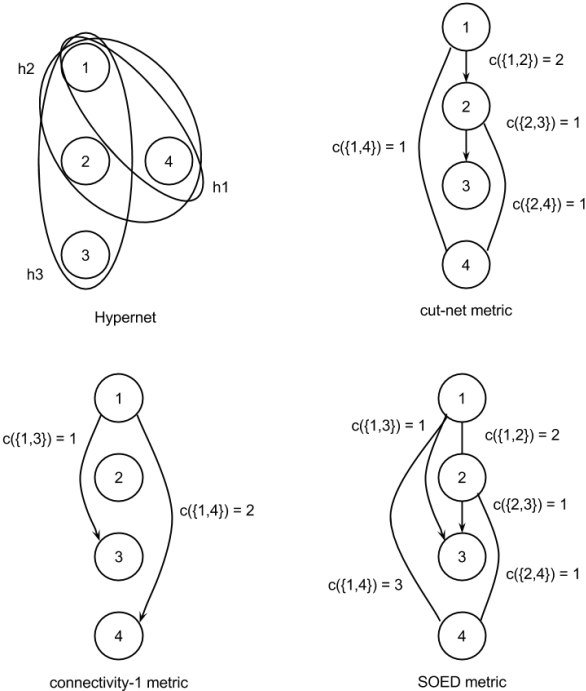


Figure 2: Original hypergraph and its Cut-net, connectivity-1, and SOED equivalents [3]

CONNECTIVITY-1 AND SOED METRICS

- The connectivity-1 metric is also straightforward. We begin in the same fashion as before — with a collection of unconnected vertices. For each consecutive pair in each net n_i in H , we add an edge from the first vertex in the pair to the last vertex in the pair with weight 1. If it already exists we add 1 to the weight of the edge.
- The sum of the edge weights from cut-net metric and the connectivity-1 metric constructs the SOED metric

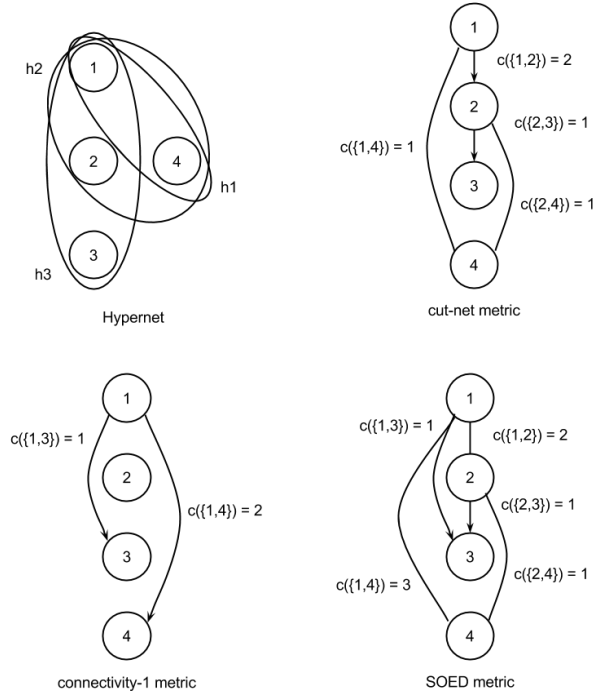


Figure 2: Original hypergraph and its Cut-net, connectivity-1, and SOED equivalents [3]

KERNIGHAN'S DYNAMIC PROGRAMMING ALGORITHM

- Input: A weighted graph $G = (V, E)$ whose vertices are ordered, and an upper bound U
- Output: A set of Breakpoints z_1, z_2, \dots, z_k in descending order

```
T(1) ← 0
R(1) ← 0
for x from 2 to M + 1 do
  Y ← {y | dy,x-1 ≤ U}
  T(x) ← miny ∈ Y (T(y) + C(x,y))
  R(x) ← argminy ∈ Y (T(y) + C(x,y))
  ► Break ties by choosing minimum y
z ← M + 1
while z > 1 do
  z ← R(z)
OUTPUT z
```

PROPOSED ALGORITHM

- Input: A weighted graph $G = (V, E)$ whose vertices are ordered, a number of parts $K \in \mathbb{N}$, and upper and lower bounds L, U on part sizes
- Output: A set of Breakpoints z_1, z_2, \dots, z_K in descending order

```
Find  $x_{\min}$  such that  $L \leq d_{y, x_{\min} - 1}$  and  $L > d_{y, x_{\min} - 2}$ 
 $T(x, 0) \leftarrow \inf$  for all  $x \leq M$ 
for  $x$  from 0 to  $x_{\min} - 1$  do
     $T(x, k) \leftarrow \inf$  for all  $k \in 0, \dots, K$ 
 $T(x_{\min}, 1) \leftarrow d_{1, x_{\min}}$ 
 $R(1, 0) \leftarrow 0$ 
for  $x$  from  $x_{\min}$  to  $M + 1$  do
    for  $k$  from 1 to  $K$  do
         $Y \leftarrow \{y | L \leq d_{y, x-1} \leq U\}$ 
         $T(x, k) \leftarrow \min_{y \in Y} (T(y, k - 1) + C(x, y))$ 
         $R(x, k) \leftarrow \operatorname{argmin}_{y \in Y} (T(y, k - 1) + C(x, y))$ 
        ► Break ties by choosing the minimum  $y$ 
 $z \leftarrow M + 1$ 
 $k \leftarrow K$ 
while  $z > 1$  do
     $z \leftarrow R(z, k)$ 
     $k \leftarrow k - 1$ 
OUTPUT  $z$ 
```

RESULTS

- Using a collection of sparse matrices from the University of Florida, the authors tested their algorithm's performance compared with PaToH, an established hypergraph partitioner
- PaToH uses a multilevel framework that can be broken down into three phases
 - Coarsening – hierarchical and agglomerative clustering
 - initial partitioning – greedy hypergraph growing algorithm
 - Uncoarsening – boundary FM hypergraph bisectioning algorithm
- Setting K to 16 quickly indicated that PaToH is best suited for lower values of K , and the dynamic programming algorithm is not
- The dynamic programming solution performed twice as fast as PaToH when K was large, tried raising K to 256

QUESTIONS

- Can the algorithm be improved to run as fast as PaToH even for partitioning input into a small number of large parts?
- Are there any row-ordering heuristics that could be used in the initialization stage of the algorithm to speed up the average run time?
- Does sequential partitioning seem like a good approach?
- Could a k-nearest neighbor clustering algorithm be used to find the optimal partitions?
- What about agent based simulation?

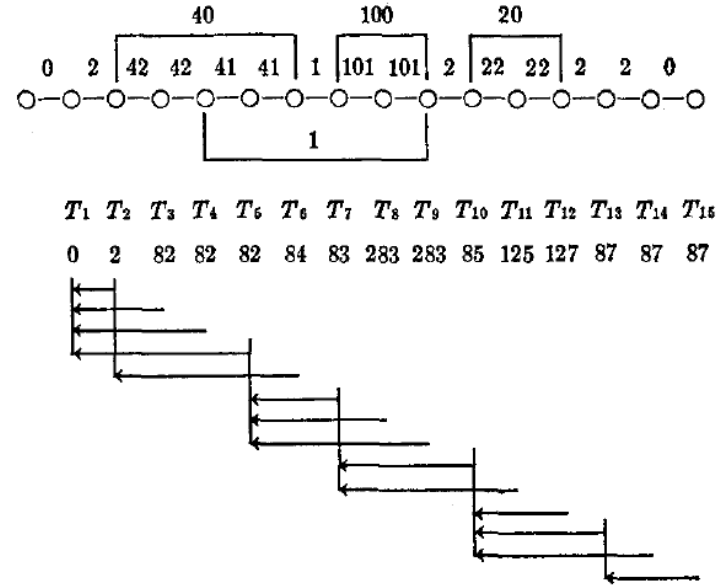


Figure 3: An optimal partitioning procedure by Kernighan's dynamic programming algorithm [1]

WORKS CITED

- [1] B. W. Kernighan, "Optimal sequential partitions of graphs," *J. ACM*, vol. 18, no. 1, pp. 34–40, Jan. 1971.
- [2] Demmel, James. "Sources of Parallelism and Locality in Simulation." *CS267: Notes for Lecture 16, Mar 9, 1995*. University of California at Berkeley, n.d. Web. 25 Apr. 2014.
- [3] Grandjean, A., Langguth, J., & Ucar, B. (2012). On Optimal and Balanced Sparse Matrix Partitioning Problems. *2012 IEEE International Conference On Cluster Computing*, 257. doi:10.1109/CLUSTER.2012.77

An energy complexity model for algorithms

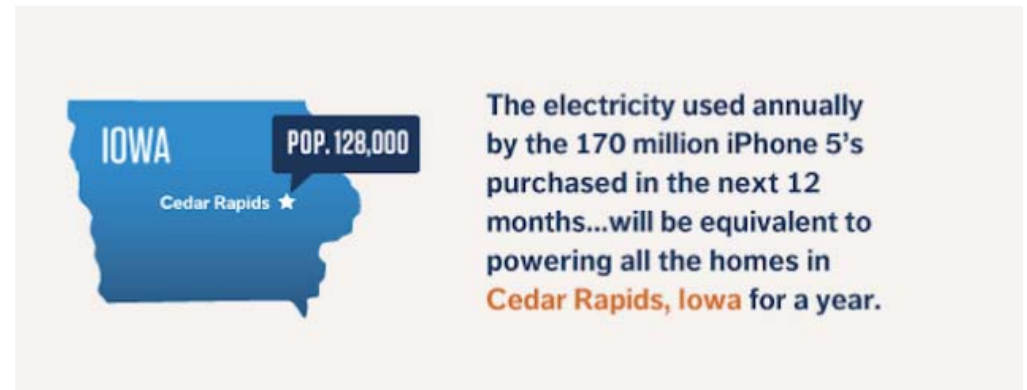
Roy, Swapnoneel, Atri Rudra, and Akshat Verma. *Proceedings of the 4th conference on Innovations in Theoretical Computer Science*. ACM, 2013.

Presented by Bo Yang & Yiyu Zheng

04/29/2014

Why focus on reducing energy consumption

- The ever-increasing energy consumption of data centers
- Explosive growth of personal computing devices
 - Smart phones, handhelds, notebooks...
 - Energy conservation means **INCREASING RUNTIME** of your device



Google's server farm in Douglas County, Iowa

<http://www.dailymail.co.uk/sciencetech/article-2219188/Inside-Google-pictures-gives-look-8-vast-data-centres.html>

Energy management across the computing desk

- Computer architecture (hardware design)
- Virtual Machine Hypervisors
- Operating System
- System Software



- But Applications and Algorithms ???

Two Energy models

- Work-oblivious
 - Server consumes same power irrespective of the working load
 - E.g., 2004 IBM Power5
- Work-proportional
 - A linear relationship between computational complexity and energy usage of an algorithm on a server running at a fixed frequency
 - For parallel algorithms

BUT experiment proves that energy consumption is neither oblivious nor proportional

Energy model for an algorithm A

- A weighted linear combination of
 - The work complexity of A
 - Time complexity in the traditional RAM model
 - The number of parallel accesses to the memory
 - A simple variation of the parallel disk I/O model
- Only consider power drawn by processors and memory. Storage power is beyond this scope.

The first work naturally combines the work complexity and parallel I/O complexity

Processor Power

- Leakage power drawn by the processor clock
- The power drawn by the processing elements, which is determined by the number of operations performed by the processor

$$E_{CPU}(\mathcal{A}) = P_{CLK}T(\mathcal{A}) + P_W W(\mathcal{A})$$



Memory Power

Memory Power

Leakage Power

Incremental cost for banks to be activated and waiting for command

$$E_{MEM}(\mathcal{A}) = P_{CKET}T(\mathcal{A}) + P_{STBY}T_{ACT}(\mathcal{A})ACT(\mathcal{A}) + E_{ACT}ACT(\mathcal{A}) + (RD(\mathcal{A}) + WR(\mathcal{A})) \times T_{RDWR}P_{RDWR}.$$

Incremental cost of various commands

A “complex” energy complexity model

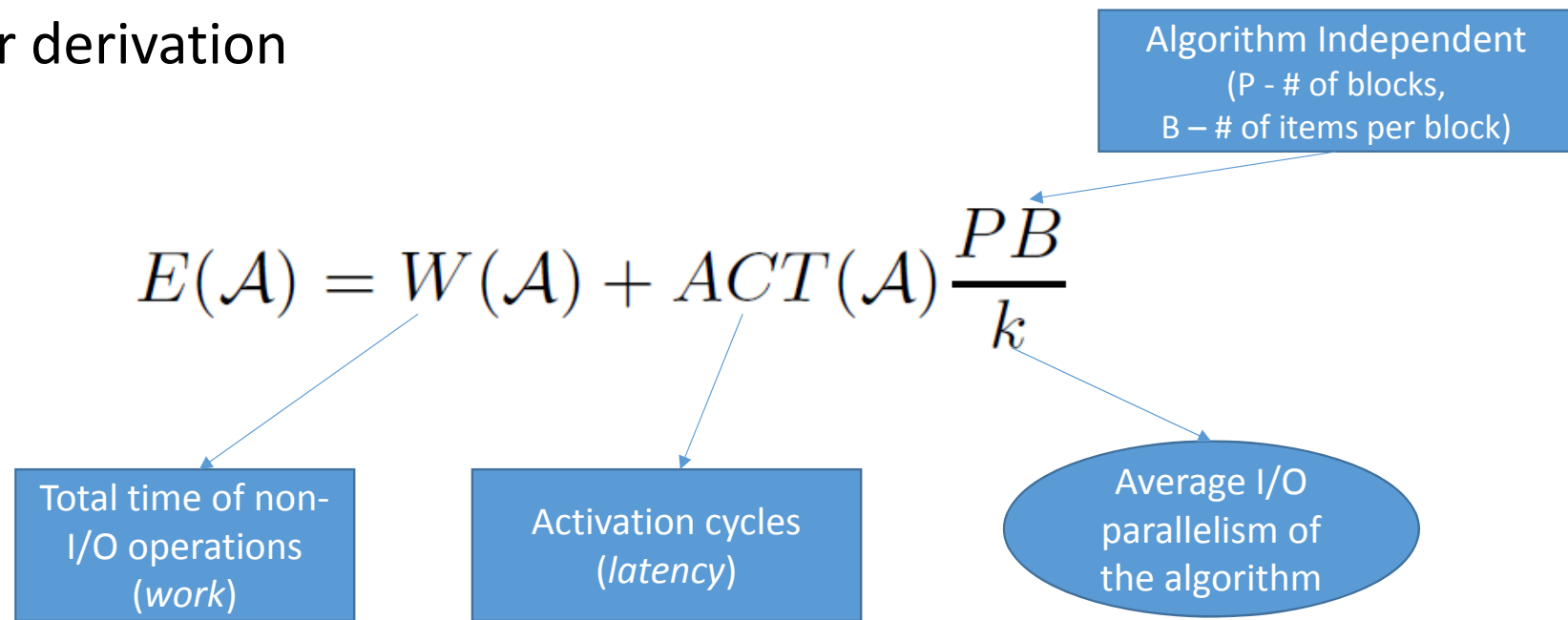
- Combine processor and memory power

$$E(\mathcal{A}) = P_{CLK}T(\mathcal{A}) + P_W W(\mathcal{A}) + P_{CKE}T(\mathcal{A}) + \\ P_{STBY}T_{ACT}(\mathcal{A})ACT(\mathcal{A}) + E_{ACT}ACT(\mathcal{A}) + \\ (RD(\mathcal{A}) + WR(\mathcal{A}))P_{RDWR}T_{RDWR}.$$

Do not worry!

Final Energy Complexity Model

- After derivation



Other Contributions

- Simulations of broad classes of algorithms
 - Usually do not change the work complexities but mildly blowup the parallel I/O complexity parameter
- Energy complexity analysis of classic algorithms sorting, matrix transpose, matrix multiplication and matrix vector multiplication.

Simulations of some certain class of algorithms

- Simulations of broad classes of algorithms
 - Usually do not change the work complexities but mildly blowup the parallel I/O complexity parameter
- Definition of k -stripped algorithm and bounded algorithm
- Two sufficient conditions for simulation

THEOREM 4.4. Let $1 \leq k, \ell \leq P$ be integers. Let \mathcal{A} be a k -striped algorithm with lookahead ℓ . Then there exists a (randomized) algorithm \mathcal{A}' with the same functional behavior as \mathcal{A} (and uses up at most three times as much space as \mathcal{A} does) such that for every x , we have $|\mathcal{I}(\mathcal{A}', x)| \leq O(|\mathcal{I}(\mathcal{A}, x)|)$, and $\mathcal{I}(\mathcal{A}', x)$ is (ℓ, τ) -parallelizable, where

$$\mathbb{E}[\tau] \leq O(1). \quad (12)$$

In particular, we have

$$\mathbb{E}[T(\mathcal{A}', x)] \leq O\left(\frac{|\mathcal{I}(\mathcal{A}, x)|}{\ell}\right). \quad (13)$$

Finally,

$$W(\mathcal{A}', x) \leq O(W(\mathcal{A}, x)). \quad (14)$$

THEOREM 4.5. Let $1 \leq \ell \leq P/2$ be an integer. Let \mathcal{A} be a bounded algorithm with lookahead ℓ that uses space $S(x)$ for input x . Then there exists a (randomized) algorithm \mathcal{A}' with the same functional behavior as \mathcal{A} (and uses three times as much space as \mathcal{A} does) such that for every x we have $|\mathcal{I}(\mathcal{A}', x)| \leq O(|\mathcal{I}(\mathcal{A}, x)|)$, and $\mathcal{I}(\mathcal{A}', x)$ is (ℓ, τ) -parallelizable, where

$$\mathbb{E}[\tau] \leq O\left(\frac{\log P}{\log \log P}\right). \quad (16)$$

In particular, we have

$$\mathbb{E}[T(\mathcal{A}', x)] \leq O\left(\frac{|\mathcal{I}(\mathcal{A}, x)| \cdot \log P}{\ell \cdot \log \log P}\right). \quad (17)$$

Finally,

$$W(\mathcal{A}', x) \leq O(W(\mathcal{A}, x)). \quad (18)$$

Corollaries for Specific Problems

- Inner Product of two vectors of length N
 - Work complexity: $O(N)$ I/O complexity: $O(N/B)$
 - Energy complexity: $\Theta(N)$
- Sorting
 - Work complexity: $O(N \log N)$ I/O complexity: $O(\frac{N}{B} \log_P(N/B))$
 - Energy complexity: $\Theta(N \log N)$
- Permuting a given vector of N items according to permutation
 - Work complexity: $O(N)$ I/O complexity: $\Omega(\min(N, \frac{N}{B} \log_M(N/B)))$
 - Energy complexity of $\Theta(N \log_P(N/B))$
- Matrix Transpose
 - Apply recursive Matrix Transpose algorithm which is $(P/2, 1)$ parallelizable
 - Energy complexity of $\Theta(N \log_{1+P}(B/P))$



- What are the similarities and differences between the energy complexity model and cache oblivious model?

Frigo, Matteo, et al. "Cache-oblivious algorithms." *Foundations of Computer Science, 1999. 40th Annual Symposium on.* IEEE, 1999.

- How to efficiently calculate the energy complexity of an algorithm?