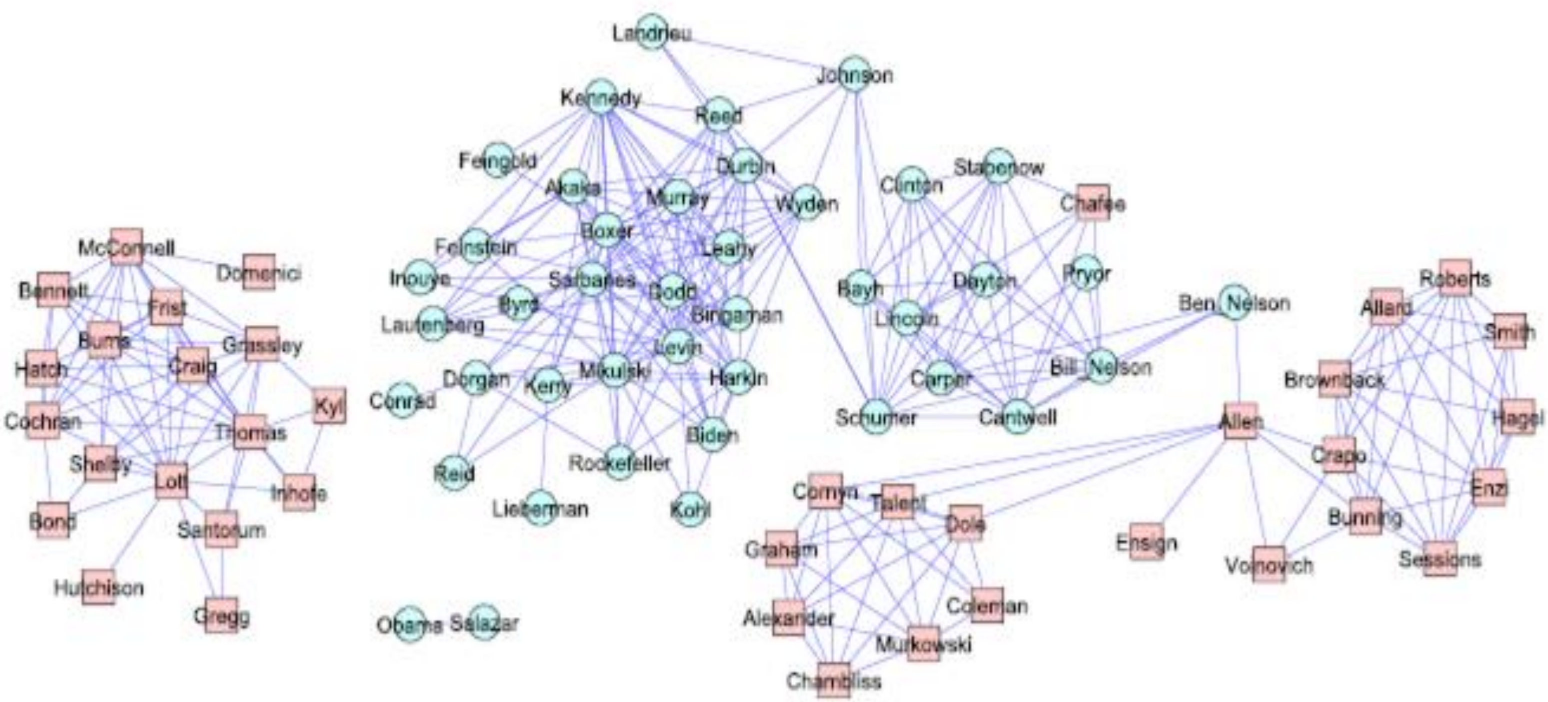
07 Jan 2010
ROSAEC workshop

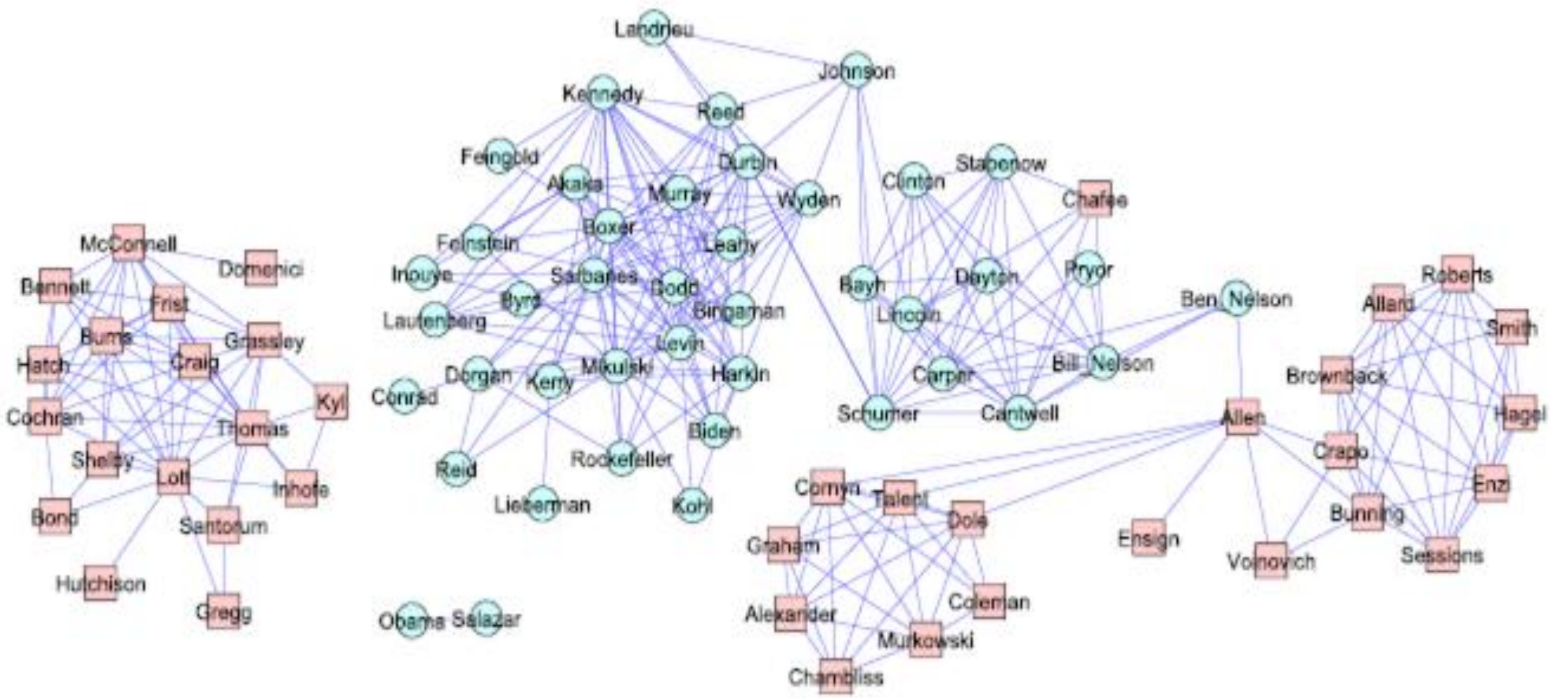
Learning Complex Networks, and Its Application to Software Verification

Kyomin Jung (KAIST)

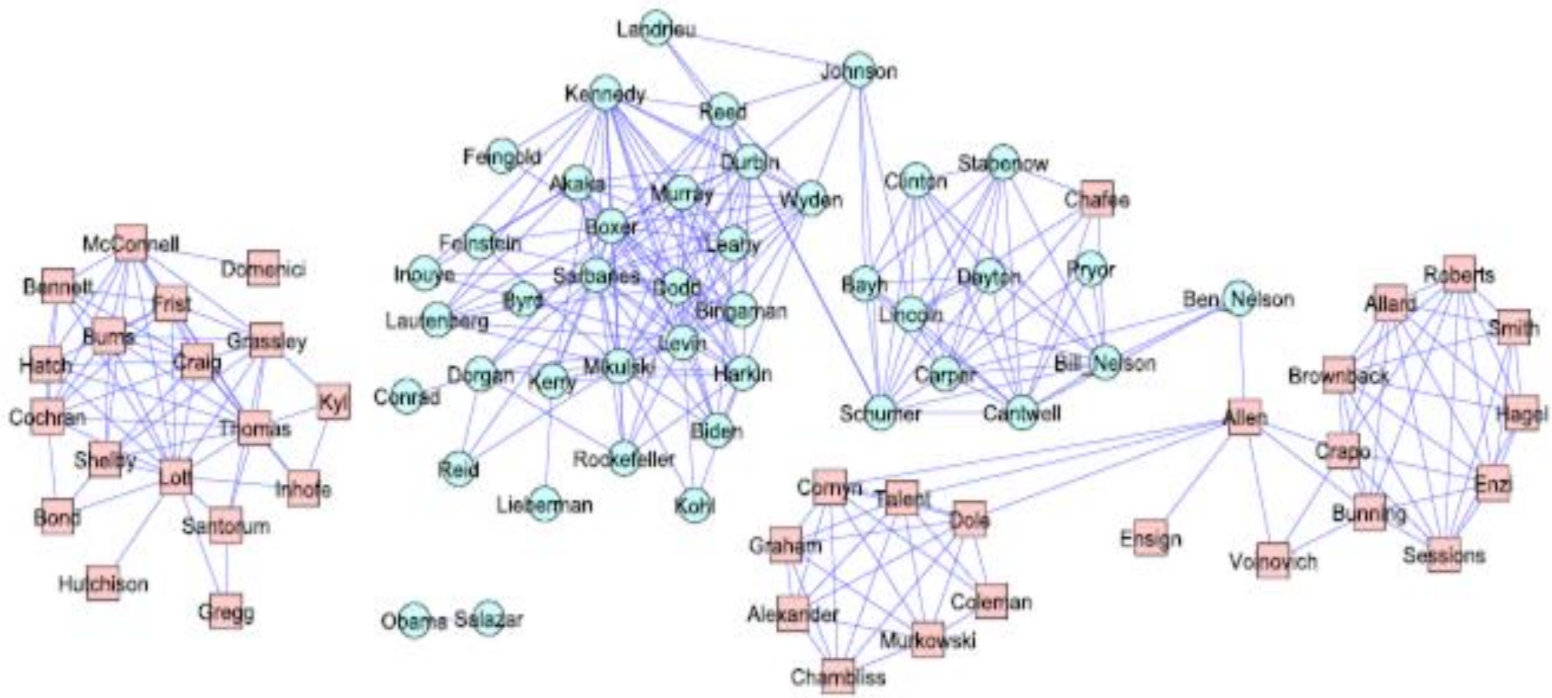
Complex Network



Complex Network

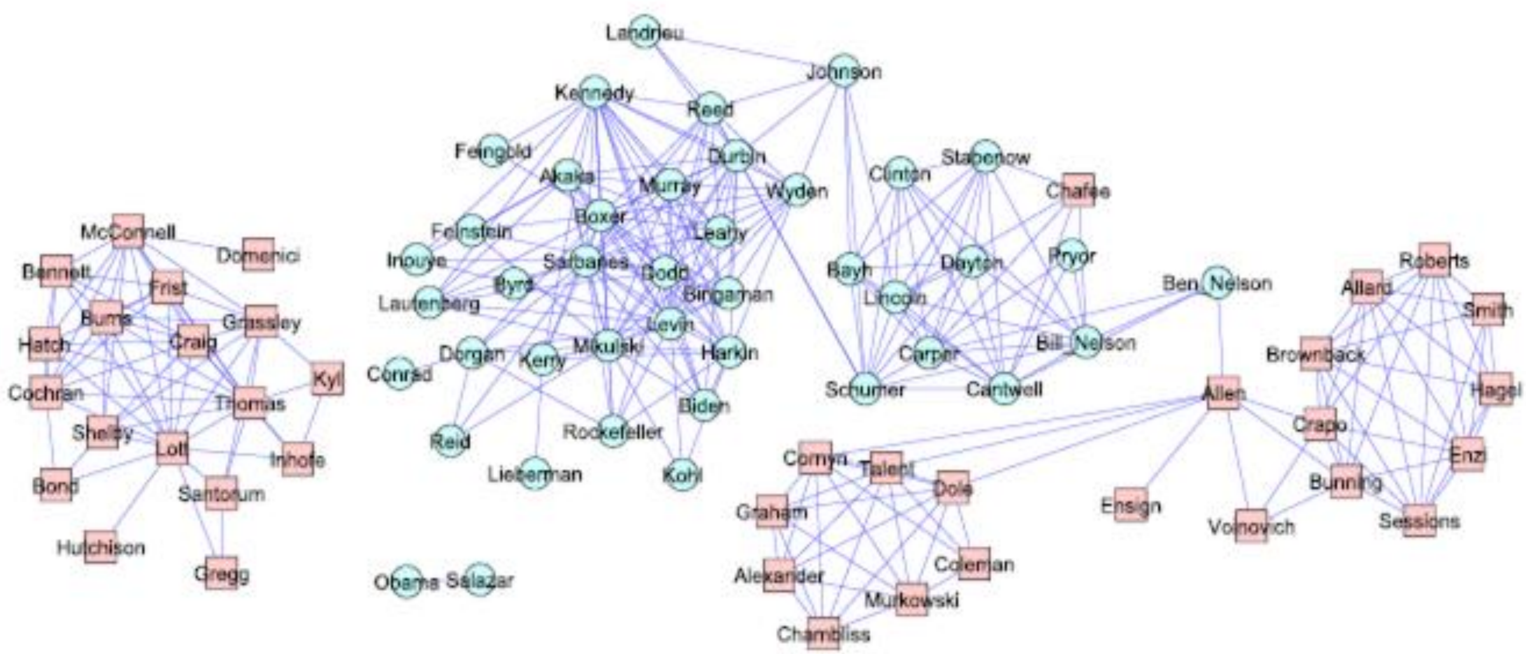


Complex Network



Complex Network

- Computational model for a given system
  - Learning the system w.r.t. the model
  - Inference (optimization/computation) based on the model
- The structure of a computer program/software itself can be understood as a complex network

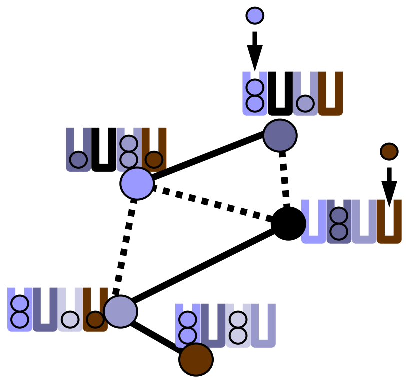


Outline : Part I

- **Randomized Learning of pseudo-Boolean functions**
- **Has application to Software Verification**
  - **Randomized property testing: test whether a given "program" has an error**

- Goal : learn the model with less function queries
- **Our results**
  - We provide an algorithm with **almost optimal number of function queries**
  - We prove **phase transition** of a random fitness model

Outline : Part II



- Communication networks
  - We provide algorithms for learning feasible allocation rate
  - Study adversarial networks
- Software Verification
  - K-SAT problem

Part I : Learning Pseudo-Boolean Functions

- Computational models
  - To understand and predict properties of complex system
- Example
  - Fitness of a species in a given environment
  - Widely used for property testing in software verification

Fitness of gene expression

- Each organism(genotype) is expressed by $x \in \{0,1\}^n$.

- Kauffman's model ['89] (a.k.a. NK model)

$$f(x) = \sum_{i=1}^{m} f_i(x_{i1}, x_{i2} \ldots x_{ik})$$

- $f_i \in R$ corresponds to a sub-characteristic of an organism

- Widely used in evolutionary biology and genetic algorithms
  - For evolutions of amino acid sequences (Macken et al '89)
  - Evolutions of protein or RNA sequences (Schuster et al '94)
  - For evaluating encoding schemes and genetic operators (Merz et al '98)

Learning Pseudo-Boolean function

$$f(x) = \sum_{i=1}^{m} f_i(x_{i1}, x_{i2} \ldots x_{ik})$$

- Our goal is to learn f by performing function queries.
  - function query = checking fitness of one gene expression
- Ex: In genetic engineering, the goal is to understand the species and design a “super organism”
- We provide algorithm to learn f with O(m log n) queries

Relevant Work

- Learning Boolean functions
  - KM alg. ['93], Jackson ['97], Bshouty, Jackson, and Tamon ['04]
  - Cannot be extended to pseudo-Boolean functions

- Learning Pseudo-Boolean functions
  - Kargupta and Park ['01] proposed a deterministic algorithm with $\theta(n^k)$ queries
  - Heckendorn and Wright ['03] proposed a randomized algorithm based on random perturbations, and show that it runs with $O(mn \log n)$ queries on average case. Choi, Jung, Moon ['08] show the same query complexity for the worst case.

Theorem (Choi, Jung, Kim)*

$$f(x) = \sum_{i=1}^{m} f_i(x_{i1}, x_{i2} \ldots x_{ik})$$

- We propose an adaptive, randomized algorithm that learns f with O(m log n) queries with failure probability $O(\frac{1}{n^{100}})$.

* Appeared in COLT '08

Phase Transition of Random Pseudo-Boolean function

$$f(x) = \sum_{i=1}^{n} f_i(x_i, x_{i2} \ldots x_{ik})$$

- To understand average properties of the species, a random fitness model was proposed by Gao and Culberson ['02]
- f_i 's are chosen randomly according to a parameter z
  - f_i takes maximum value for z many assignments among 2^k of them
- Problem: Is there a "super organism" ?
- When $k \leq 2$, the problem is easy. For k=3, it is NP-hard.

Phase Transition of Random Pseudo-Boolean function

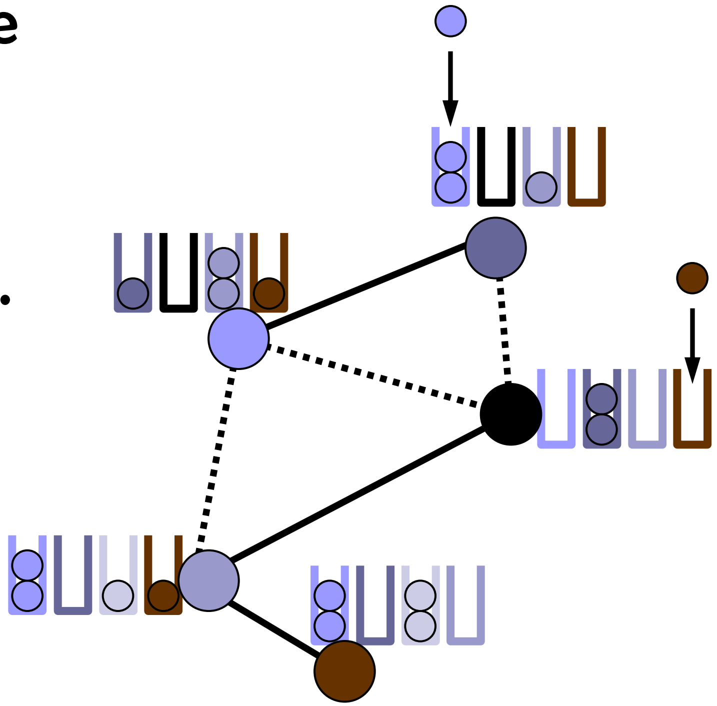
$$f(x) = \sum_{i=1}^{n} f_i(x_{i_1}, x_{i2} \ldots x_{ik})$$

- When k=3, Gao and Culberson ['02] proved that when z<5.163, "there is no super organism" with high probability.
- Theorem (Choi, Jung, Kim)*
  - When z>5.163, there is a super organism with positive probability.

* Appeared in GECCO '05 & Artificial Intelligence '08

Part II : Communication Networks

- Routing/scheduling in communication networks with queue
- Main problem
  - Whether the network is stable, i.e. queue size is bounded over time



Our Work on Communication Networks

- We design algorithms (Gummadi, Jung, Shah and Sreenivas)*
  - To learn whether a given arrival rate vector makes the network stable or not
  - Utilize structural properties of the network

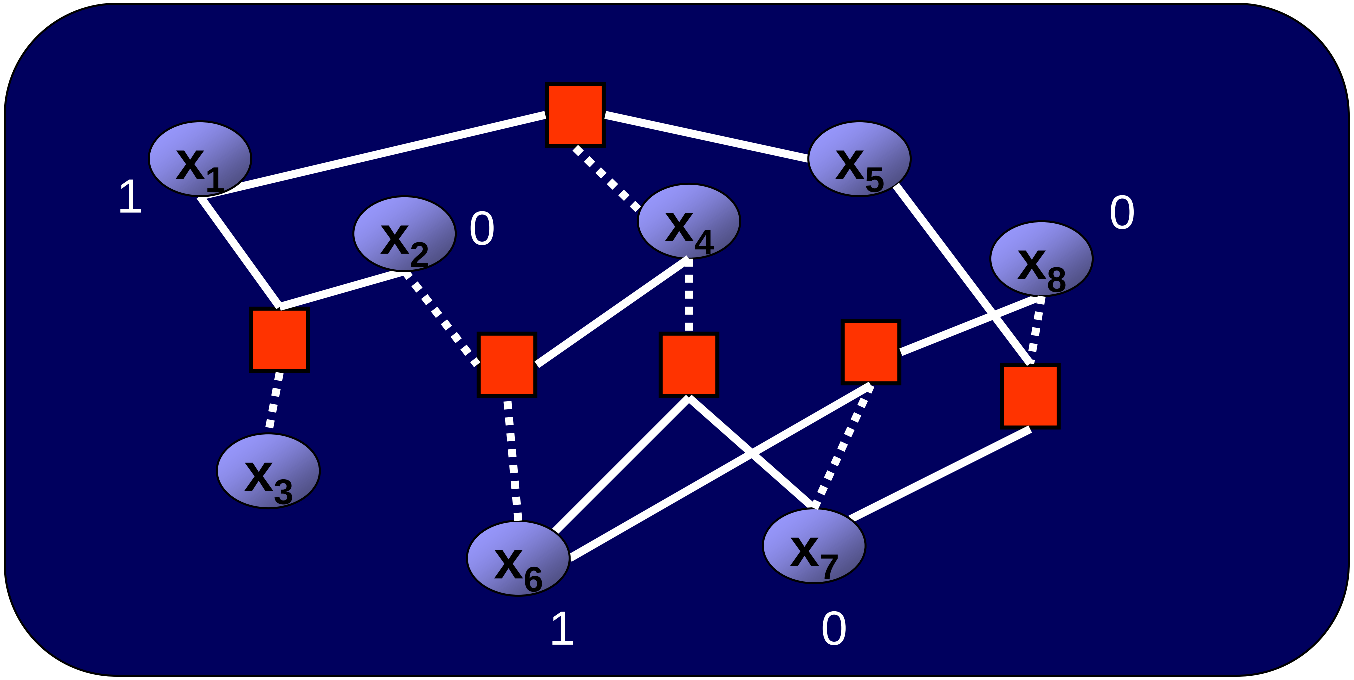
- Theorem (Andrews, Jung, Stolyar)**
  - Local optimizer, "Max-weight algorithm", makes the system stable under adversarial arrival process for dynamic graphs in the edge interference network.

* Appeared in INFOCOM '08, INFOCOM '09

** Appeared in STOC '07

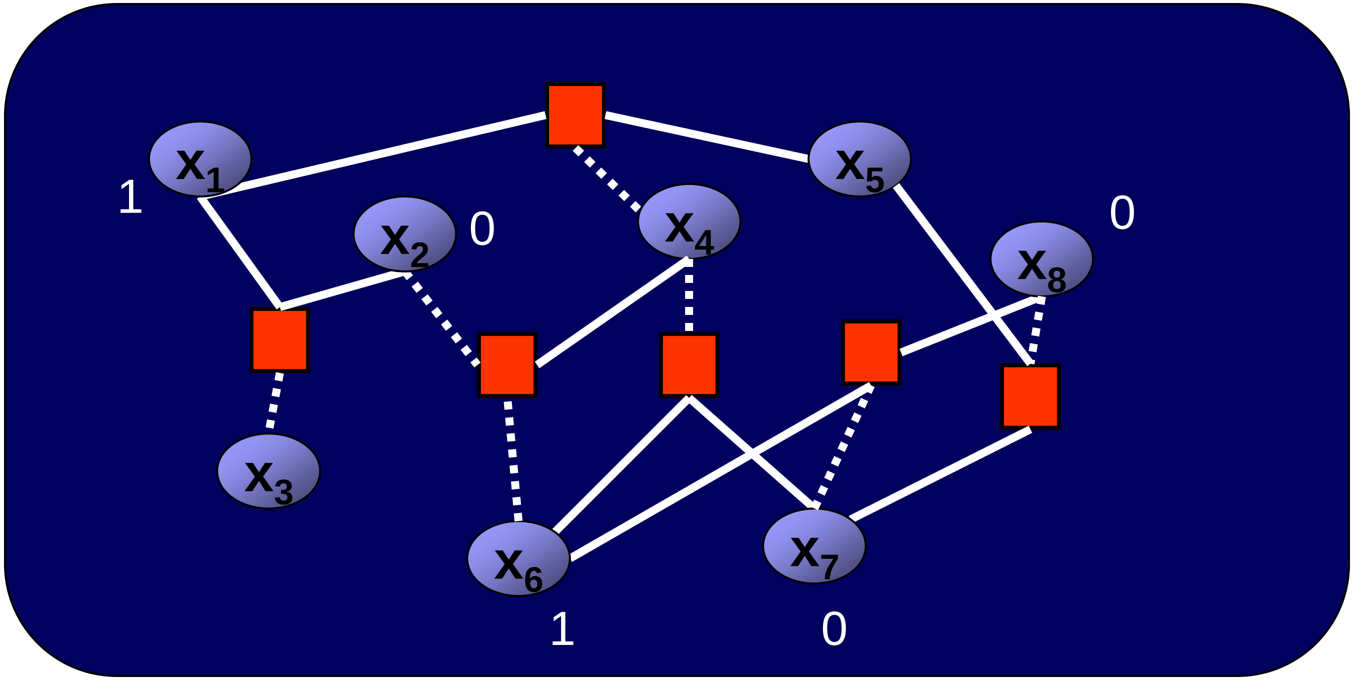
K-SAT problem

- Variables: $x_1, x_2, \ldots, x_n$ take values {TRUE, FALSE}
- Constraints: (x_1 or x_2 or not x_3) , (not x_2 or x_4 or not x_6), …
  $(x_1 \vee x_2 \vee x_3) \wedge (x_2 \vee x_4 \vee x_6) \wedge \ldots$
- Application to Software Verification



Designing K-SAT solver based on graph structures

- Currently we are working on designing a K-SAT solver based on structural properties of a k-SAT instance.
  - (joint work with Yungbum Jung and Suwon Jang)



Thank you.