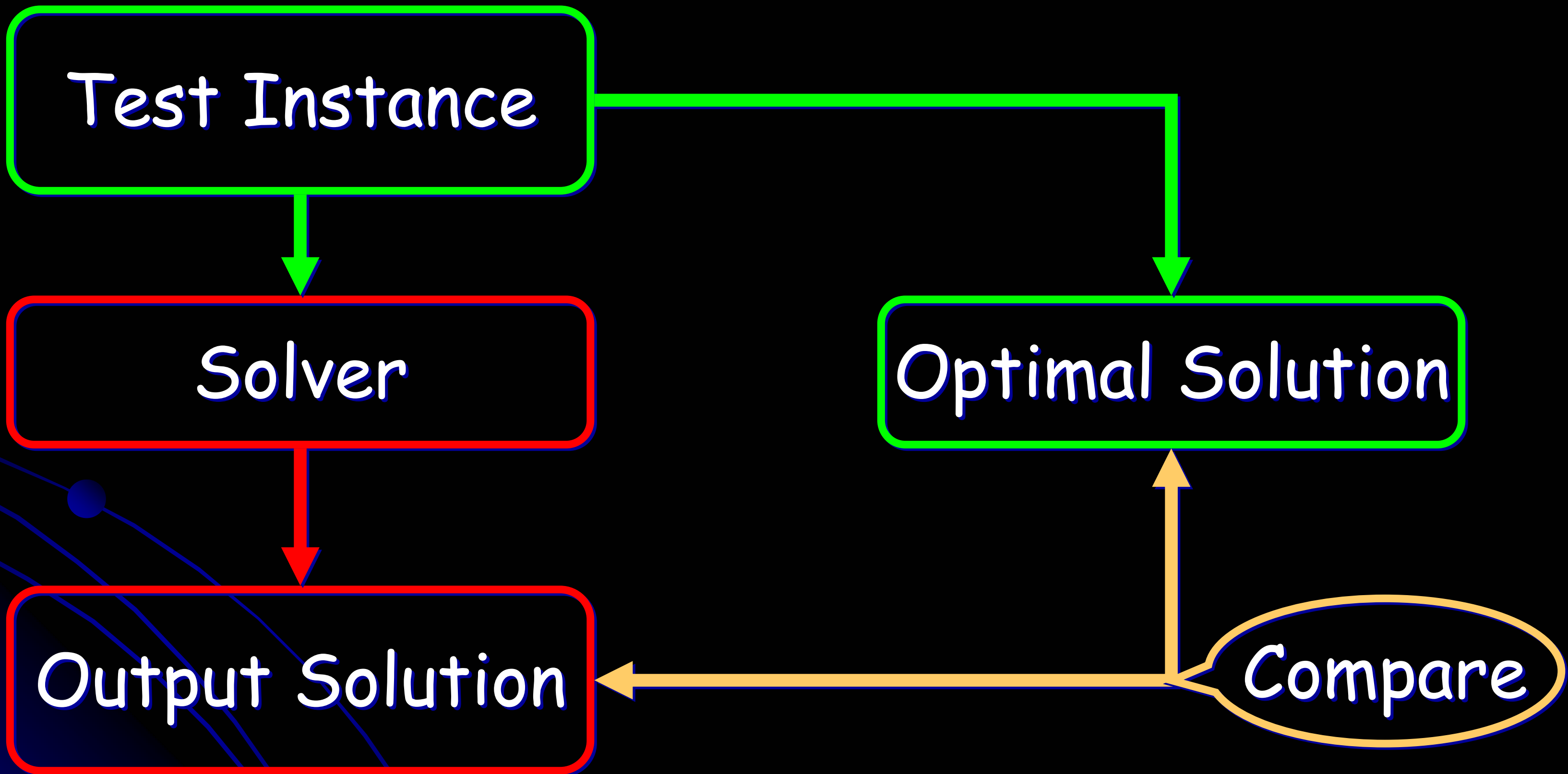
# Test Instance Generation for MAX 2SAT

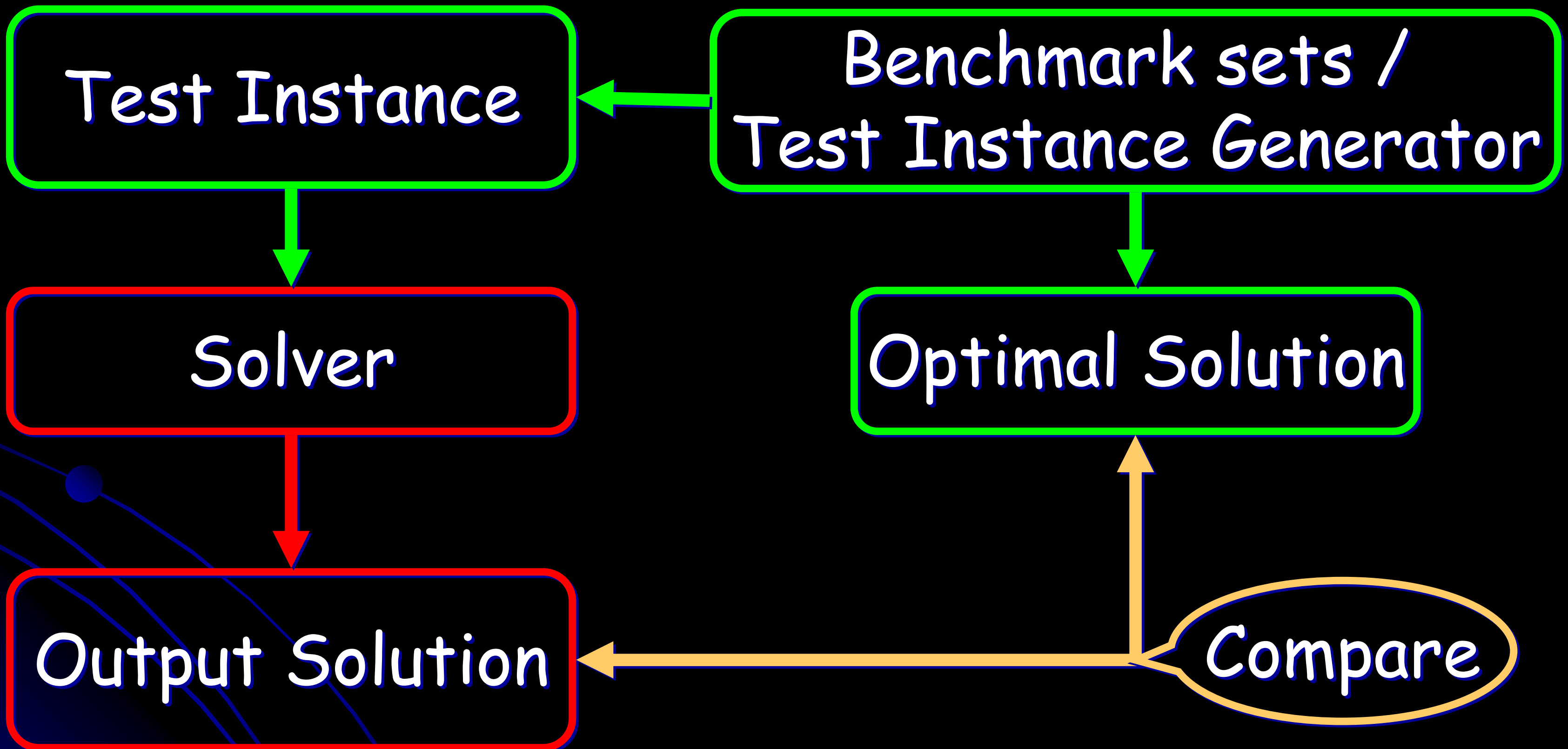
Mitsuo Motoki
Japan Advanced Inst. of Sci. and Tech. (JAIST)
JAPAN ADVANCED INSTITUTE OF SCIENCE AND TECHNOLOGY 1990

# Test Instance Generation problem

# Empirical study of solvers for combinatorial opt. problem



# Empirical study of solvers for combinatorial opt. problem

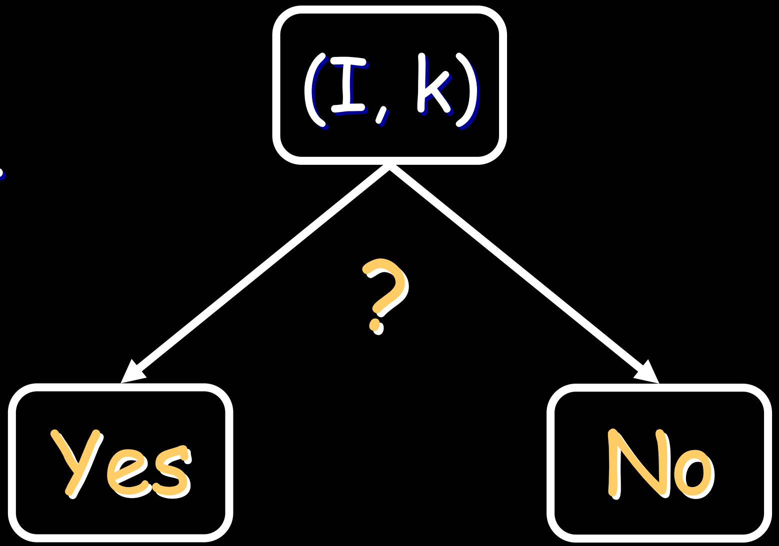


# Test instance generator


- Ideally…
  - Generate all instances with the opt. solution
  - Running time is polynomial in the length of output instance.

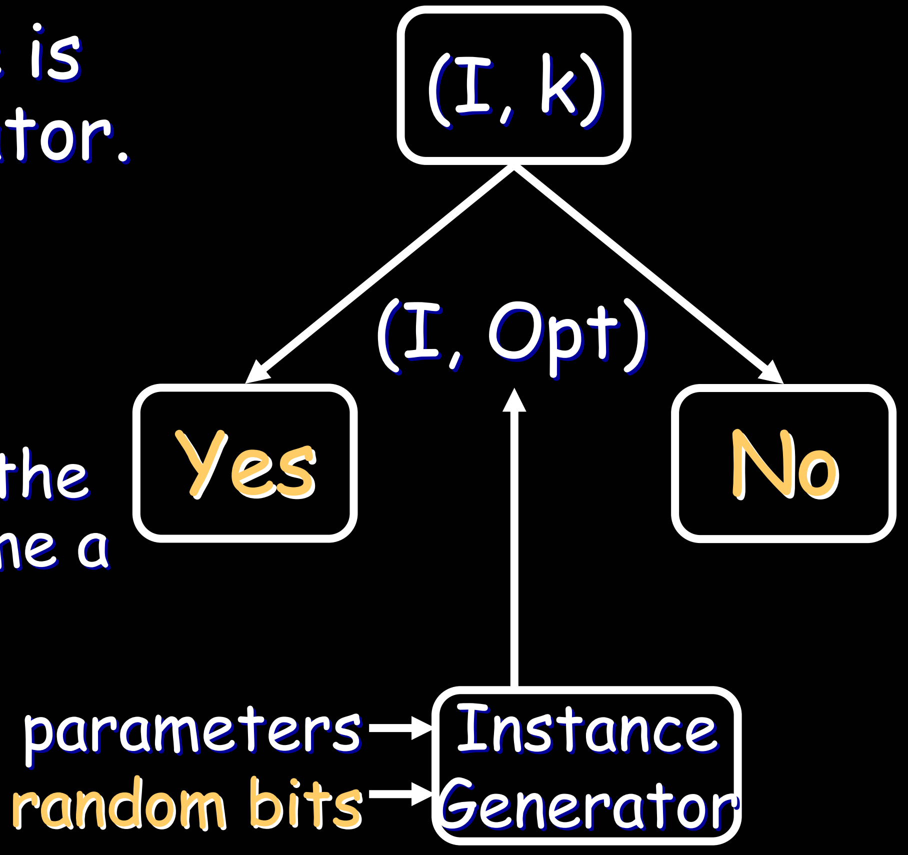
# for NP hard optimization problem

- Unless NP=co-NP, there is no ideal instance generator.
- Why?
  - Consider the decision version (NP hard)



# for NP hard optimization problem

- Unless NP=co-NP, there is no ideal instance generator.
- Why?
  - Consider the decision version (NP hard)
  - The random bits used in the instance generator become a witness for each "yes" instance,
  - and also for each "no" instances.



# What can we do?

- Relax some requirements for instance generator
  - Can generate instances from **some** subset of whole instance set
  - Outputs a **feasible solution** instead of the optimal solution

    (Outputs optimal solution with high prob.)
  - Running time is "**exponential**" instead of "polynomial"

# What can we do?

- Relax some requirements for instance generator
  - Can generate instances from **some** subset of whole instance set
  - Outputs a feasible solution instead of the optimal solution

    (Outputs optimal solution with high prob.)
  - Running time is “exponential” instead of “polynomial”

# Our approach

- Poly. time exact instance generator
  - The set of instance generated is a subset of the whole instance set
  - The generator always outputs a test instance with the optimal solution
  - The running time is polynomial in the length of the output instance

# New requirements

- The instances generated should be hard.
- How to guarantee the hardness?
  - Theoretical way
    - For any poly. time exact instance generator, the decision problem over the set of instance generated is NP $\cap$ co-NP (no more NP complete)
    - How hard to distinguish the instances generated?
  - (Empirical study)

# Poly. time exact instance generator for MAX 2SAT

# MAX 2SAT

- Input: 2CNF formula
  - Each clause consists of exactly 2 literals
  - Each variable appears at most once in a clause
  - Any clause can appear more than once
- Question: find a truth assignment s.t.
  - maximizes # of satisfied clauses,
  - i.e., minimizes # of unsatisfied clauses.

# How hard?

- MAX SNP complete
- Decision version (Is there an assignment that satisfies at least $k$ clauses?) is NP complete
- For satisfiable 2CNF formulas, poly. time solvable (2SAT is in P)
- Inapproximability upper bound:
  - $21/22 \approx 0.955$ [Håstad STOC'97]
  - 0.945 (under some unproven conjectures) [Khot, Kindler, Mossel, and O'Donnell FOCS'04]

# Related works

- Probabilistic generator for MAX *k*SAT [Dimitriou CP'03]
  - Unique optimal solution w.h.p., $O(n^k)$ clauses
- Exact/probabilistic generator for MAX 2SAT [Yamamoto '04]
  - To characterize opt. solution, requires an expander graph
  - They use an explicit expander graph construction algorithm / a random graph
- Probabilistic generator for MAX 3SAT [MM COCOON'01]

# Strategy of instance generator

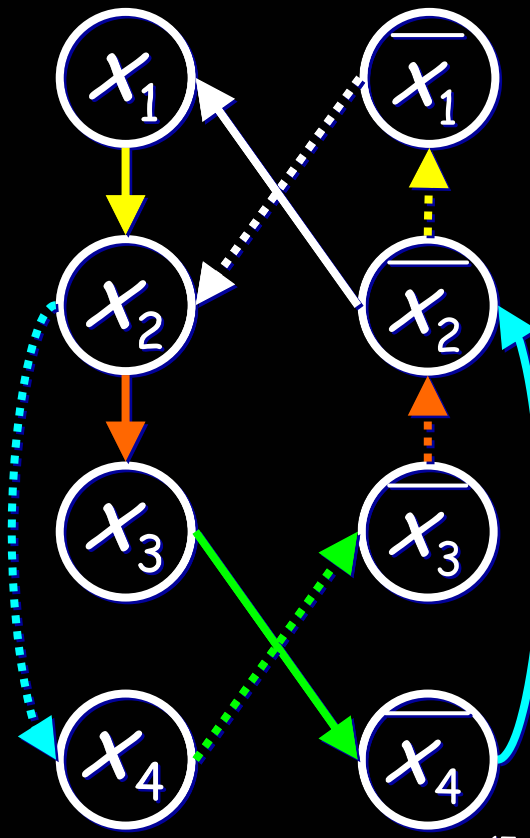
i. Choose $t \in \{0,1\}^n$ at random as the optimal solution
ii. Combine appropriate number of minimal unsat. 2CNFs that contains exactly 1 clause falsified by $t$
iii. Add several clauses satisfied by $t$

- There is no assignment that falsifies less # of clauses than # of 2CNFs in ii.
- Thus $t$ is an optimal solution

# Implication graph [Aspvall, Plass, Tarjan '79]

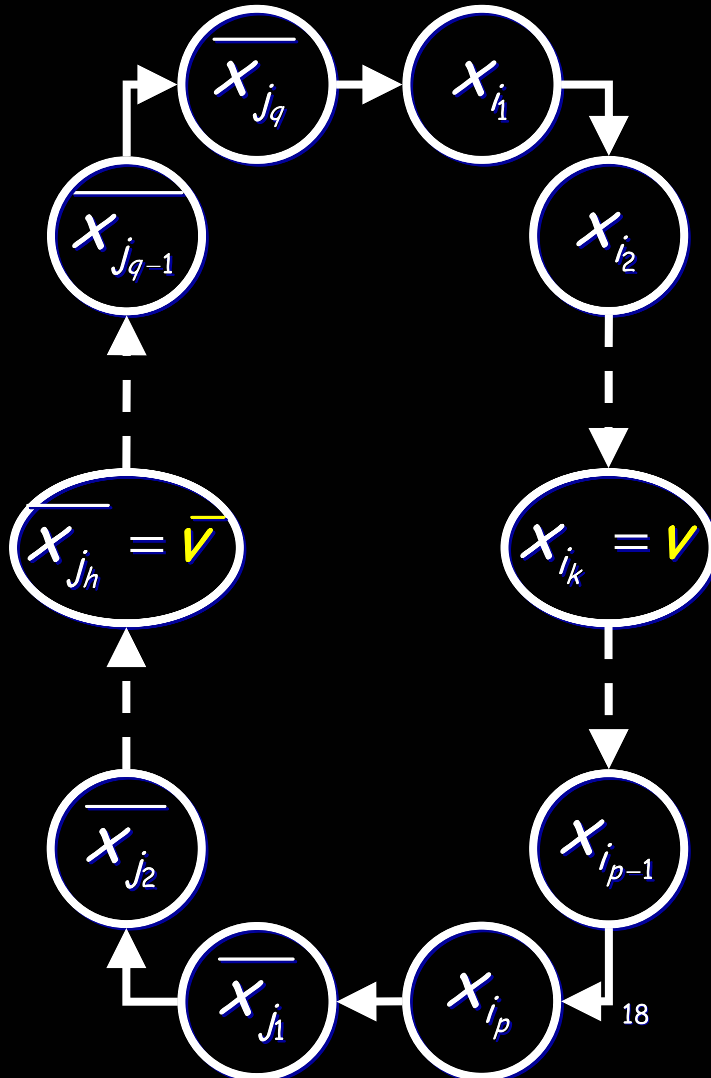
$(x_1 \vee x_2)(\overline{x_2} \vee x_3)(\overline{x_3} \vee \overline{x_4})$
$(x_4 \vee \overline{x_2})(\overline{x_1} \vee x_2)$

- Transform 2CNF $F$ into a digraph $G_F$
  - Each vertex corresponds to a literal
  - $F$ contains a clause $(a \vee b) \Leftrightarrow G_F$ has edge $\overline{a} \rightarrow b$ and $\overline{b} \rightarrow a$
- Contradictory bicycle
  - $F$ is unsat. $\Leftrightarrow G_F$ has a cycle (and its complement cycle) containing x and $\overline{x}$



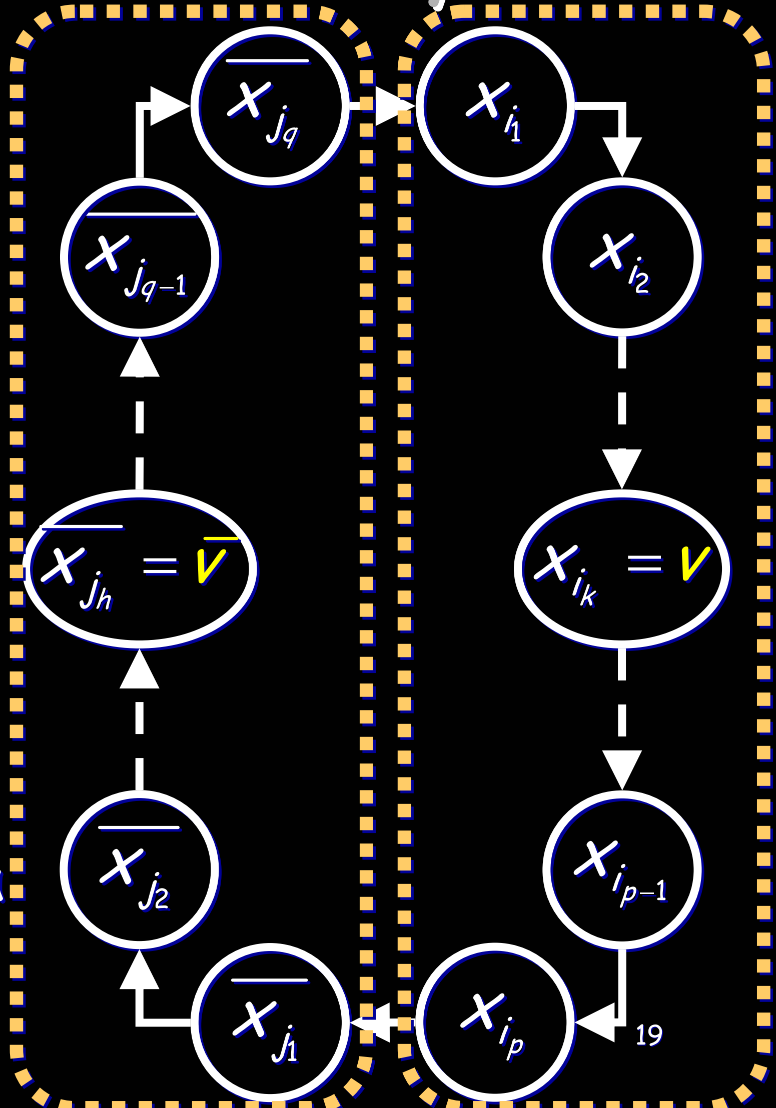
# Minimal unsat. 2CNF $F$ containing exactly 1 clause falsified by $t$

- W.l.o.g. assume $t = 1^n$
  - $G_F$ is a (simple) contradictory bicycle
  - $F$ has just 1 clause consisting of only negative literals
  - $\Rightarrow$ Cont. cycle has just 1 edge from positive literal to negative literal
  - Cont. cycle contains some variables as positive and negative literal
  - $\Rightarrow$ There is exactly 1 edge from negative literal to positive literal



# Minimal unsat. 2CNF $F$ containing exactly 1 clause falsified by $t$

- W.l.o.g. assume $t = 1^n$
  - $G_F$ is a (simple) contradictory bicycle
  - $F$ has just 1 clause consisting of only negative literals
  - $\Rightarrow$ Cont. cycle has just 1 edge from positive literal to negative literal
  - Cont. cycle contains some variables as positive and negative literal
  - $\Rightarrow$ There is exactly 1 edge from negative literal to positive literal
- We can divide a cont. cycle into a sequence of positive literals and a sequence of negative literals



# Instance generator algorithm

Input: # $n$ of vars.

i. Let $F$ be an empty formula
ii. Choose $t \in \{0,1\}^n$ at random
iii. Choose min. # $k$ ($\geq 0$) of unsat. clauses
iv. <u>for</u> $i=1$ <u>to</u> $k$ <u>do</u>
    Generate a 2CNF in $B_t$ at random and add to $F$
v. Add clauses in $C_t$ to $F$ at random

# The set $\mathbf{I}$ of instances generated

- $B_t$ : the set of minimal unsat. 2CNFs containing exactly 1 clause falsified by $t$
- $C_t$ : the set of clauses satisfied by $t$

- $\mathbf{I} = \{F \in 2\mathrm{CNF} \mid F$ consists of elements of $B_t$ and $C_t$ for some $t\}$
  - $F \in \mathbf{I} \Leftrightarrow$ min. # of unsat. clauses = max. # of cont. bicycles
  - Both $t$ and a partition into elements of $B_t$ and $C_t$ become a witness of $F \in \mathbf{I}$

# Hardness results

# Hardness of the instances generated

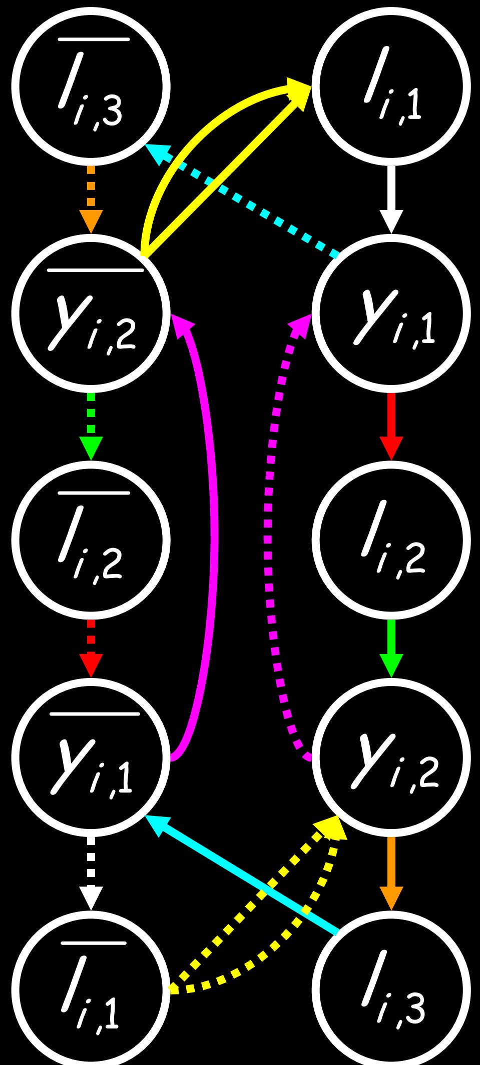
- To distinguish the set **I** of instances generated is NP complete
  - I.e., finding an opt. solution is at least as hard as finding a sat. assign. of satisfiable 3CNFs
- To approximately distinguish the set **I** of instances generated is also NP hard

# I is NP complete (1)

- Reduction from 3SAT
- For any 3CNF $F_{3CNF} = c_1 \wedge c_2 \wedge \ldots \wedge c_m$, transform each clause $c_i = (l_{i,1} \vee l_{i,2} \vee l_{i,3})$ into

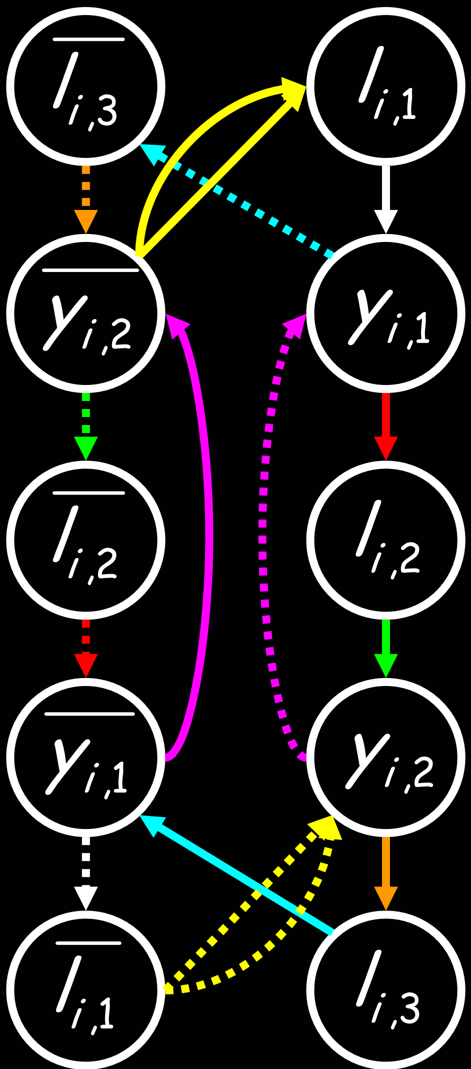
$$b_i = (\overline{l_{i,1}} \vee y_{i,1}) \wedge (\overline{y_{i,1}} \vee l_{i,2}) \wedge (\overline{l_{i,2}} \vee y_{i,2})$$
$$\wedge (\overline{y_{i,2}} \vee l_{i,3}) \wedge (\overline{l_{i,3}} \vee \overline{y_{i,1}}) \wedge (y_{i,1} \vee \overline{y_{i,2}})$$
$$\wedge (y_{i,2} \vee l_{i,1}) \wedge (y_{i,2} \vee l_{i,1})$$

  and let $F_{2CNF} = \wedge_i\, b_i$.
- Note that new variables $y_{i,1}$ and $y_{i,2}$ appear **only** in $b_i$



# I is NP complete (2)

| $l_{i,1}$ | $l_{i,2}$ | $l_{i,3}$ | $y_{i,1}$ | $y_{i,2}$ | # of unsat. |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 2 |
| 0 | 0 | 0 | 0 | 1 | 2 |
| 0 | 0 | 0 | 1 | 0 | 3 |
| 0 | 0 | 0 | 1 | 1 | 2 |
| 0 | 0 | 1 | 0 | 1 | 1 |
| 0 | 1 | 0 | 1 | 1 | 1 |
| 0 | 1 | 1 | * | 1 | 1 |
| 1 | 0 | 0 | * | 0 | 1 |
| 1 | 0 | 1 | 0 | 0 | 1 |
| 1 | 1 | 0 | 1 | * | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 |



# $\mathbf{I}$ is NP complete (3)

- If $F_{3CNF} = c_1 \wedge c_2 \wedge \ldots \wedge c_m$ is satisfiable
  - $F_{2CNF}$ has $m$ contradictory bicycle
  - Min. # of unsat. clauses in $F_{2CNF}$ is $m$
  - $F_{2CNF} \in \mathbf{I}$
- If $F_{3CNF}$ is unsatisfiable
  - $F_{2CNF}$ has $m$ contradictory bicycle
  - Min. # of unsat. clauses in $F_{2CNF}$ is at least $m+1$
  - $F_{2CNF} \notin \mathbf{I}$

# Hardness for approximation

- If $c_i \in F_{3CNF}$ is falsified by the opt. solution, 2 clauses in $b_i$ are falsified by corresponding assignment
  - If min. # of unsat. clauses in $F_{3CNF}$ is ***k***, min. # of unsat. clauses in $F_{2CNF}$ is ***m+k***

| $l_{i,1}$ | $l_{i,2}$ | $l_{i,3}$ | $y_{i,1}$ | $y_{i,2}$ | # of unsat. |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 2 |
| 0 | 0 | 0 | 0 | 1 | 2 |
| 0 | 0 | 0 | 1 | 0 | 3 |
| 0 | 0 | 0 | 1 | 1 | 2 |
| 0 | 0 | 1 | 0 | 1 | 1 |
| 0 | 1 | 0 | 1 | 1 | 1 |
| 0 | 1 | 1 | * | 1 | 1 |
| 1 | 0 | 0 | * | 0 | 1 |
| 1 | 0 | 1 | 0 | 0 | 1 |
| 1 | 1 | 0 | 1 | * | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 |

# Hardness for approximation

- The reduction from 3SAT is a gap reserving reduction
  - If $F_{3CNF}$ is satisfiable, min. # of unsatisfiable clauses in $F_{2CNF}$ is $m$
  - If 1/8 fraction of $F_{3CNF}$ is unsatisfiable, min. # of unsat. clauses in $F_{2CNF}$ is m+m/8 = 9m/8
- If we can approximate any member of **I** within $\frac{8m-9m/8}{8m-m} = {}^{55}\!/_{56}$, we can distinguish satisfiable 3CNFs and unsat. 3CNFs

# Future works

- Improve the hardness for approximation ($55/56 \approx 0.982$).
  - Cf. imapproximability: $21/22 \approx 0.955$, $0.945$
- Estimate appropriate values for parameters
  - Analyze the distribution/expectation of # of contradictory bicycles in random 2CNFs
- Instance generator for other problems