アルゴリズムとプログラミング実践講座

http://akashi.ci.i.u-tokyo.ac.jp/mary/lectures/algorithm/

火曜　13:00 -- 14:30
I-REF 棟 6階　ヒロビー

稲葉真理　with 浅井大史・手塚宏史

今学期の振り返り

目標・ターゲット

他学部からの進学生など

アルゴリズム（など）の入り口（証明抜き）の紹介

ツールの使い方（研究室で先輩に教わるような事）

難易度について

受講者がバラエティーに富んでいた

「ツールのインストールが大変」 → 後半減らした

「簡単すぎるという批判」 → 難しくしたら提出激減

今学期の振り返り

感じてほしかった内容

大きいデータを扱う時代が来た結果の変化

→　厳密でなくても良いこともある

→　（繰り返し出た）Randomized Algorithm

（しつこくやった）おまけについて

大学時代は、教科書を読む時間が確保できる。

OS でも、アーキテクチャーでも、興味が持てたら、
ぜひ、１冊　読み通してほしい（願望）

７月は　研究の話を少し

7/1 SAT solver について

7/8 Can Computers Think?

by Prof. David Avis

7/15　出張のため休講

7/8（火）の講義

Can computers think?

Prof. David Avis @ 京大 ＋ McGill Univ.

We will discuss this question by beginning with Alan Turing's historic paper in 1950 that started the field of ariticial intelligence (AI).

This paper includes a test, now called the Turing test, in which a computer tries to convince humans that it is a human rather than a computer.

We will discuss this test and related competitions, including the famous Loebner prize, and claims that Eugene Goostman, a chatbot, passed the test on June 7, 2014. We will attempt our own Turing test during the last part of the lecture.

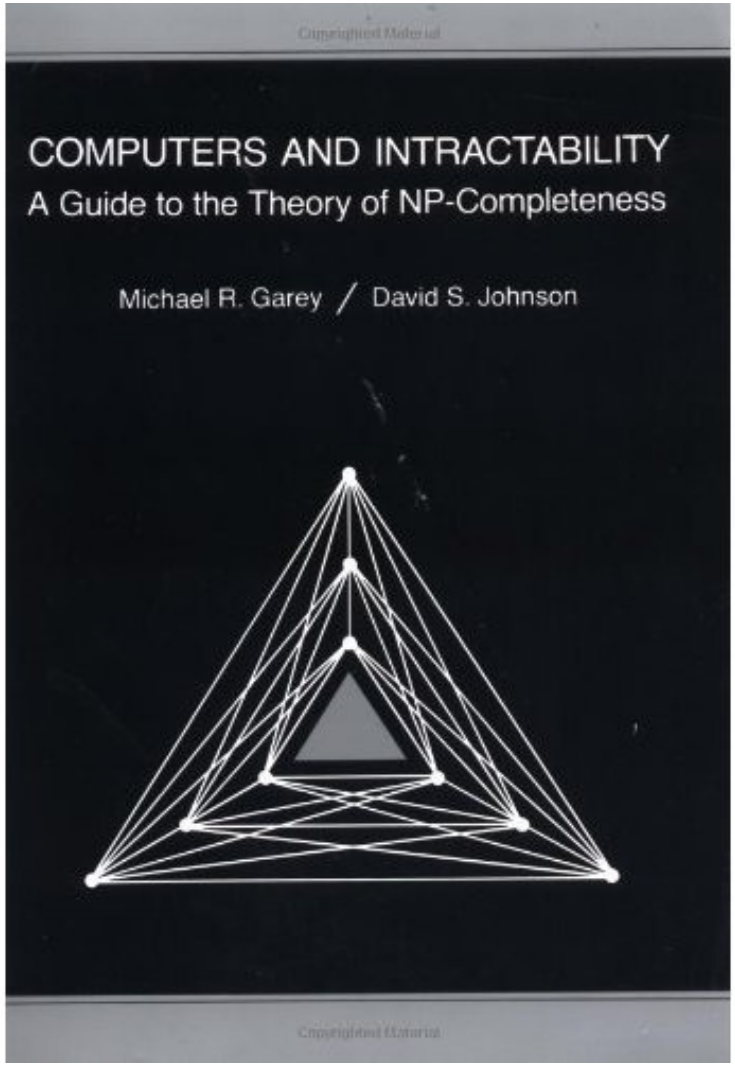
第十二回　課題

レポートについて

- レポートのファイルを添付し、メールで提出のこと
  - E-mail: algorithm2014@edu.jar.jp
  - サブジェクト 「アルゴリズムとプログラム実践講座・レポート」
  - 学生証番号と名前は、 メールの本文にも書いてください。

- 推奨環境など
  - 言語
    - 日本語 あるいは 英語

- **Prof. David Avis の講演の感想を A4 一枚程度にまとめて提出すること。**



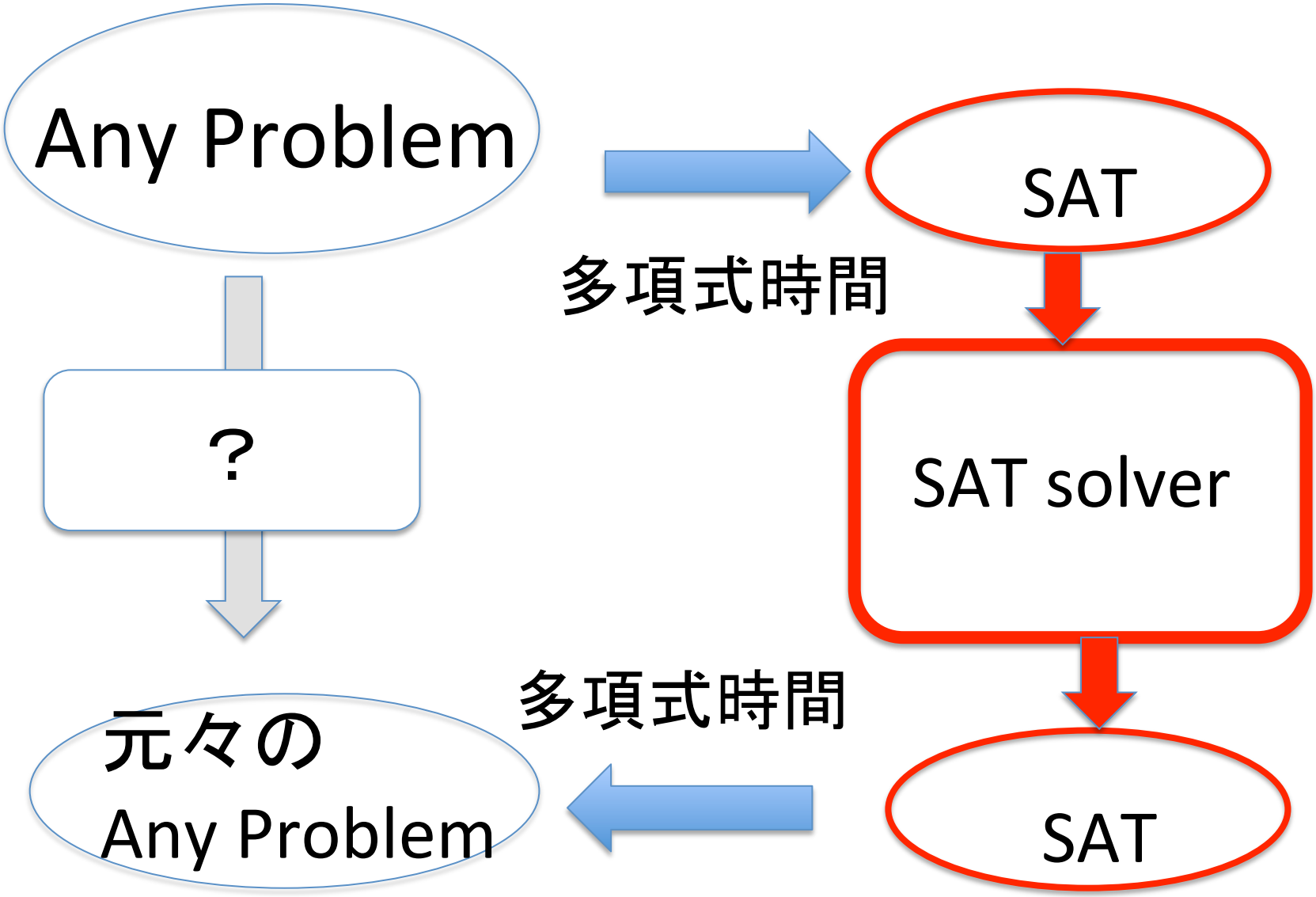
1979
Computers and Intractability
by Michael R. Garey
David S. Johnson

Cook の定理
[1971　Stephen Cook]

- **多項式時間変換**
- 充足可能性問題 (SATisfiability Problem(SAT)) は NP完全
- 判定問題 （注：最適化問題ではない）

NP完全問題とは クラスNPに属する問題でかつクラスNPのすべての問題から多項式時間帰着可能な問題

SAT solver



SAT solver

実際に、（お金が絡むところにも）使われている。

e.x. 回路検証

チップ一つ　tapeout するまでに

何億円、何百億円の世界

現在の実用的な SAT solver

- ノウハウのかたまり
- サイエンスじゃないと愚痴った先生あり

SAT問題とは

充足可能性問題:

与えられた論理式を満たす変数割当が存在するか？

$(a \vee b \vee c) \wedge (\neg b \vee d) \wedge (\neg a \vee e)$

a,b, c, d, e: 変数

¬ : 否定

a, ¬a: リテラル（否定込みの変数）

各クローズ最低一つ変数を「真」にできるか？

SAT 問題

小さい問題 → しらみつぶしでいい

大きい問題 → 変数が n 個とすれば
割当は 2^n 通り

SAT Solvers
(と SAT competition)

- 充足可能な場合は変数割当を答え
  不能ならUNSATと答える
- 毎年、SAT conference で、コンペティション
- 3 部門に分かれて競う (Application/Random/Crafted)
- **“Application 部門”の特徴**
  - 現実の問題から作られた問題
  - 問題に構造がある可能性あり
  - 構造があれば、速く解ける可能性あり
- アプリケーションの例: 回路検証, 定理証明器 などなど.

薗部知大（卒業生）

SAT Competition 2011

SAT ２位　MINISAT Hack １位

古典的な研究ねた

わからないことが沢山

どの変数が大事？ (VSIDS, CIR)

どのクローズが大事？（どれを捨て、選ぶ（並列）？）

分散も集中も大事。では良いバランスは？

経験的にわかってることが沢山

簡約も大事 (pre-process)

前の結果の利用は良いことが多い。（キャッシュも当たる）

(phase saving)

研究ネタ
最近のはやり並列化

- 情報共有をどうするか？
- 通信をどうするか？

- 普通の計算と違うのは、「途中の計算結果の情報共有しなくても、（遅くなる以外の）問題は発生しない」

SAT solver で
何変数くらい解けそう？

SAT Solver の進歩

1962	**DPLL**	**#variables ~ 10**
1986	BDDs	#variables ~ 100
1992	GSAT	#variables ~ 300
2001	**Chaff**	**#variables ~ 10,000 (CDCL)**
2004	MiniSAT	#variables ~ 100,000
Currently		#variables ~ 1,000,000

SAT Solver -- 技術の積み重ね (先達はあらまほしき事なり）

Preprocess
2004 pre-process

Phase Saving
2006 うまくいった （cache hit)

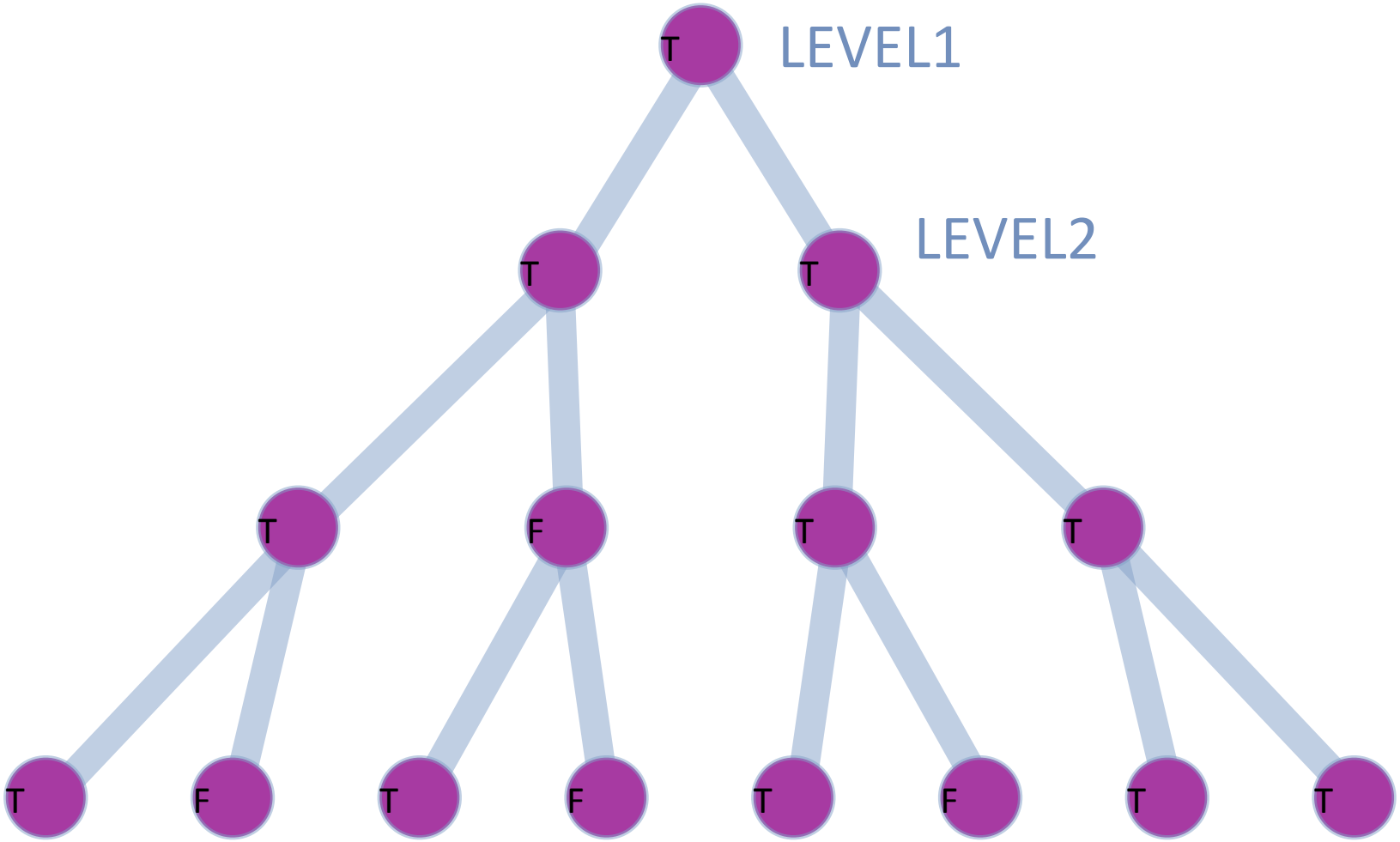
Restart
2007 Luby Restart (静的）
2008 Dynamic Restart (動的 ← 学習）

代表的な手法

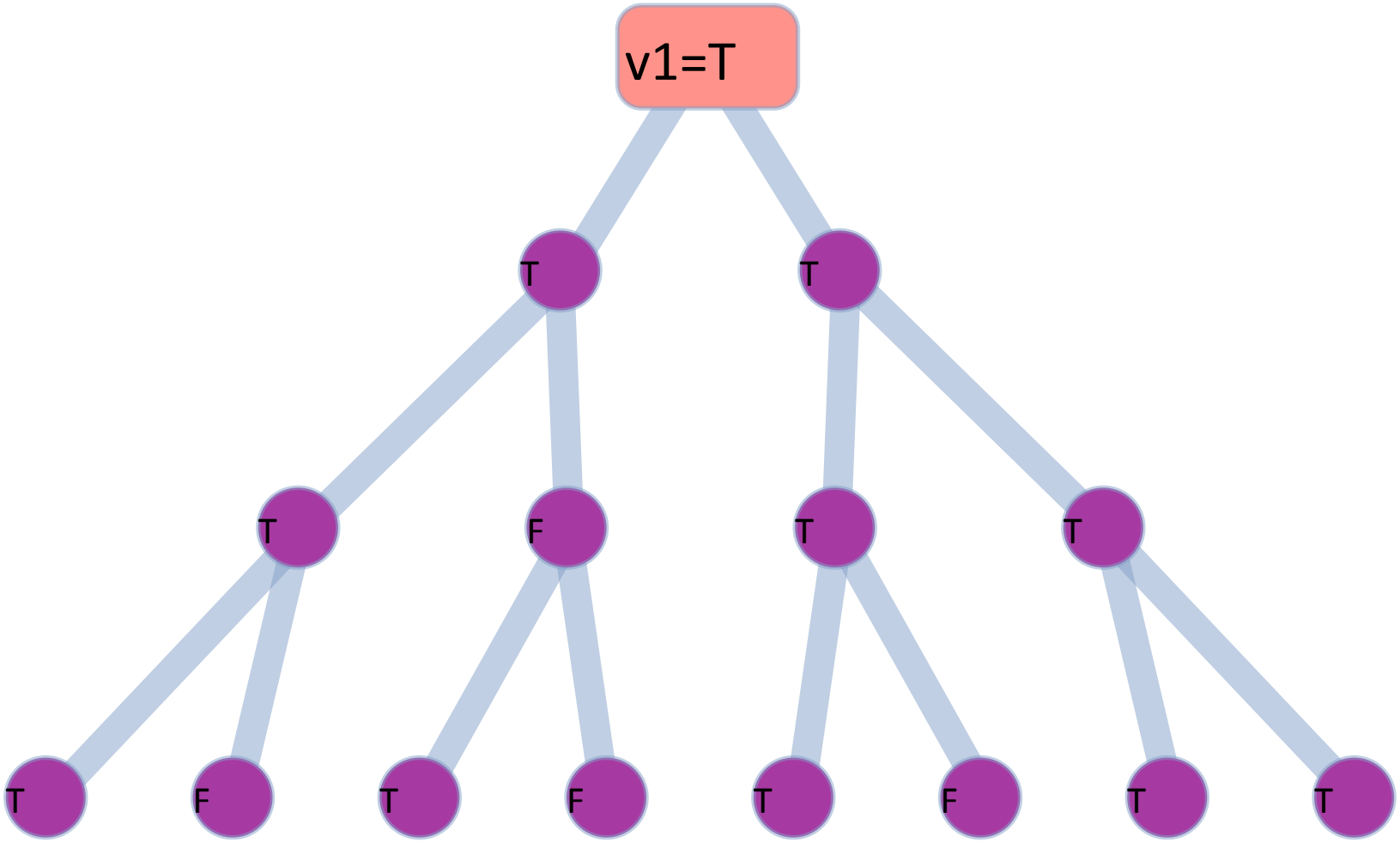
- **DPLL (Davis-Putnam-Logemann-Loveland, 1962)**
  - Decision, Propagation, Backtrack

- **CDCL (Conflict Driven Clause Learning, 2001)**
  - **Clause learning ：implication graph を利用**
  - Decision Heuristics -- **VSIDS**
  - Backtrack & Restart

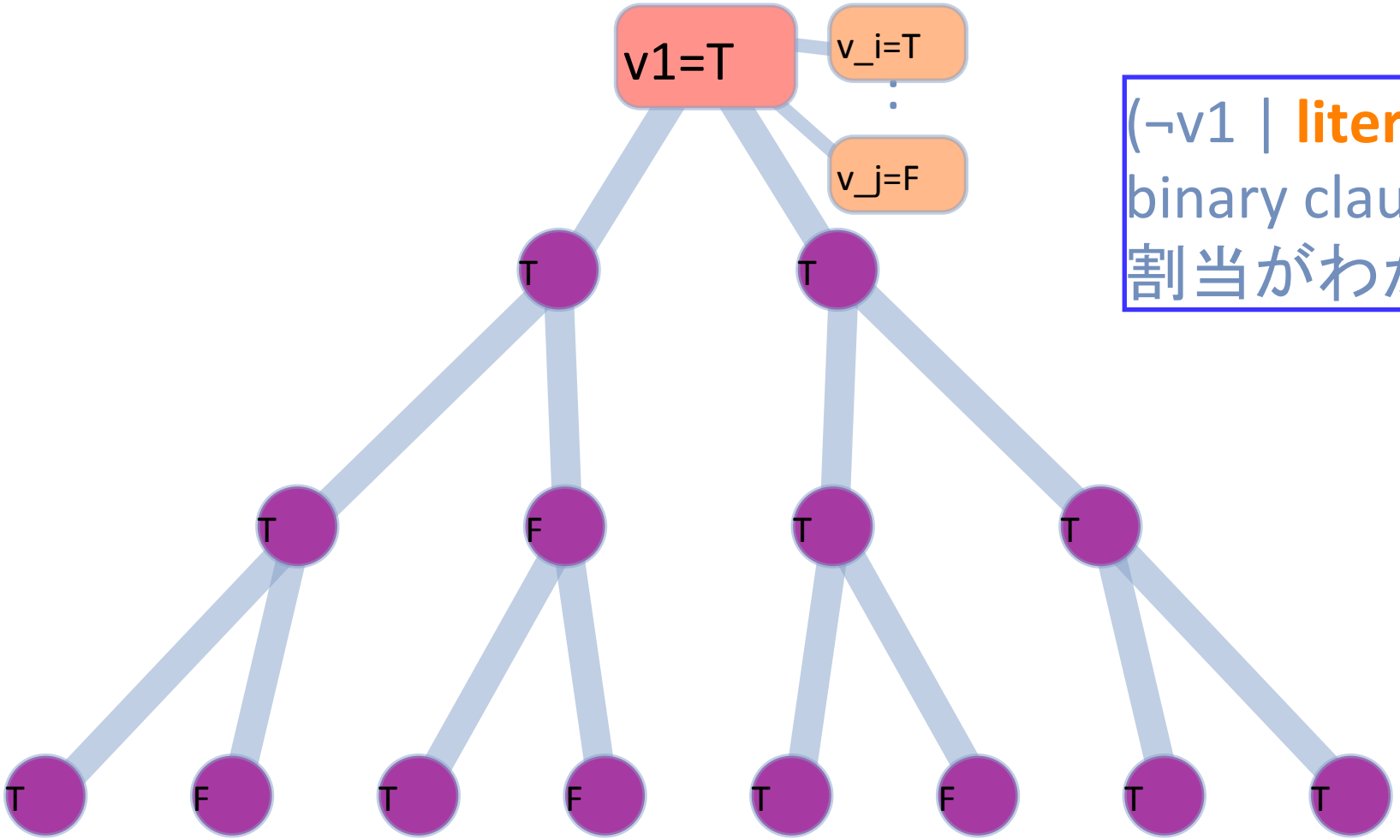
Decision LEVEL



Decision

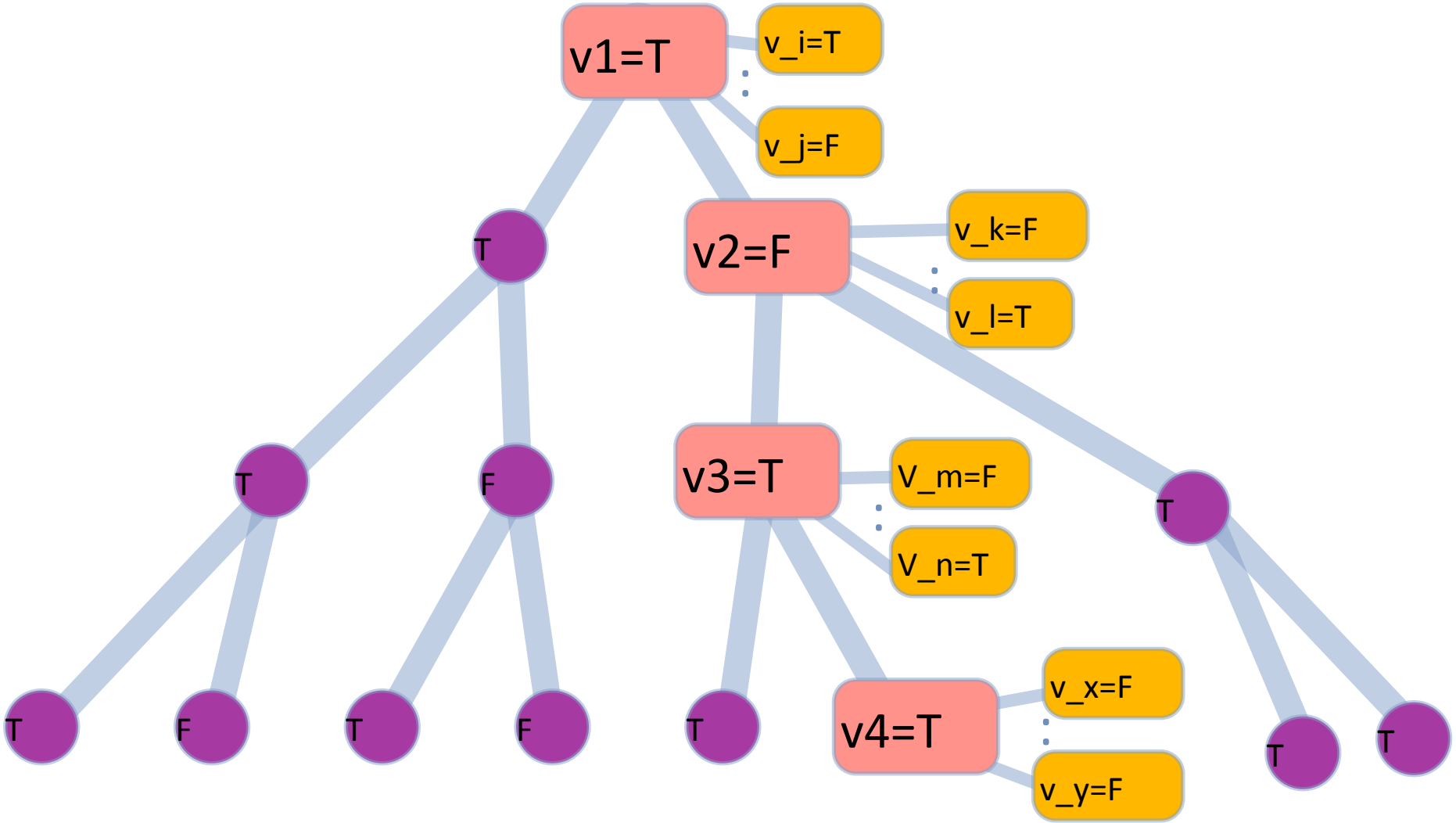


Unit Propagation

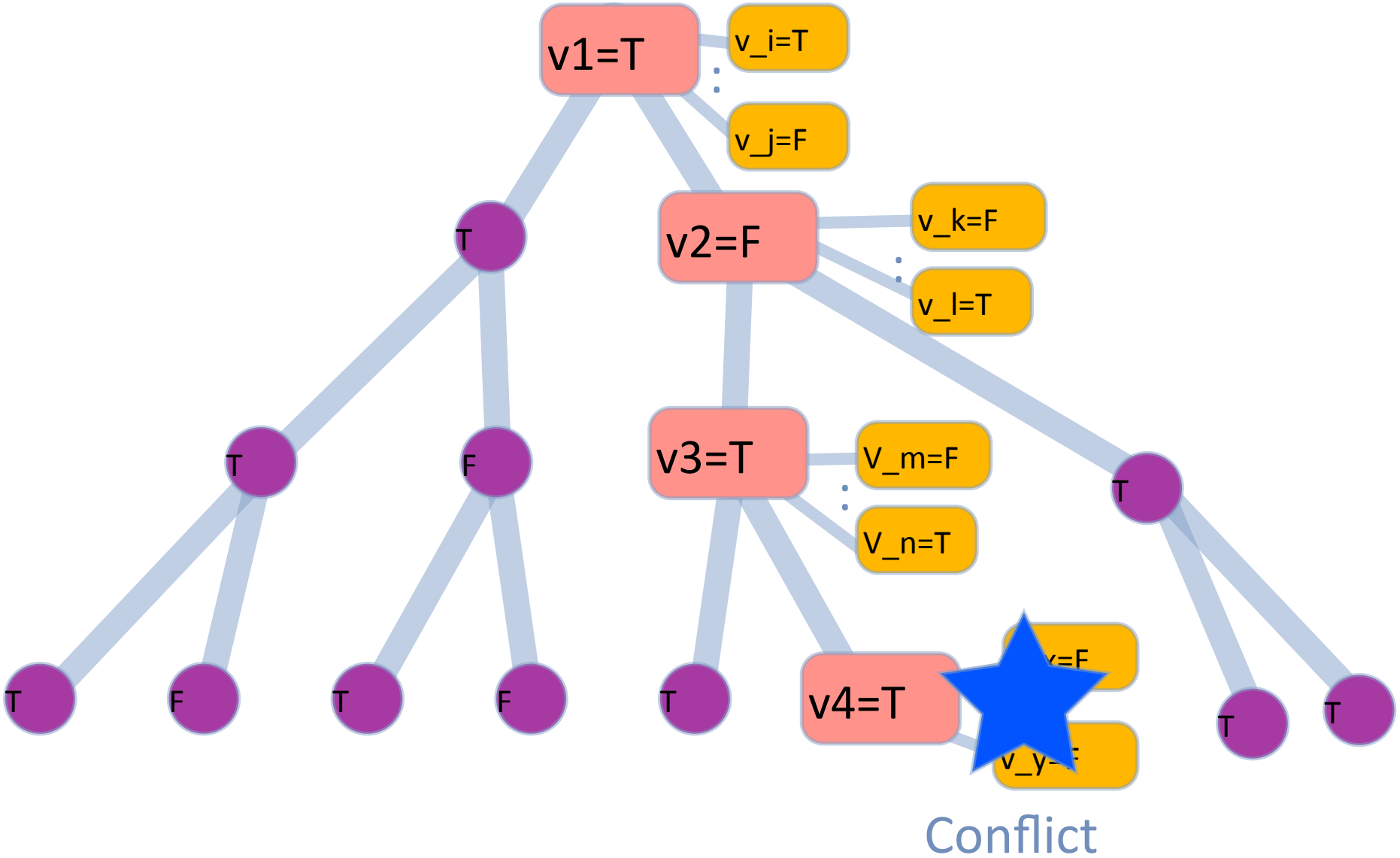


(¬v1 | **literal**)
binary clause
割当がわかる

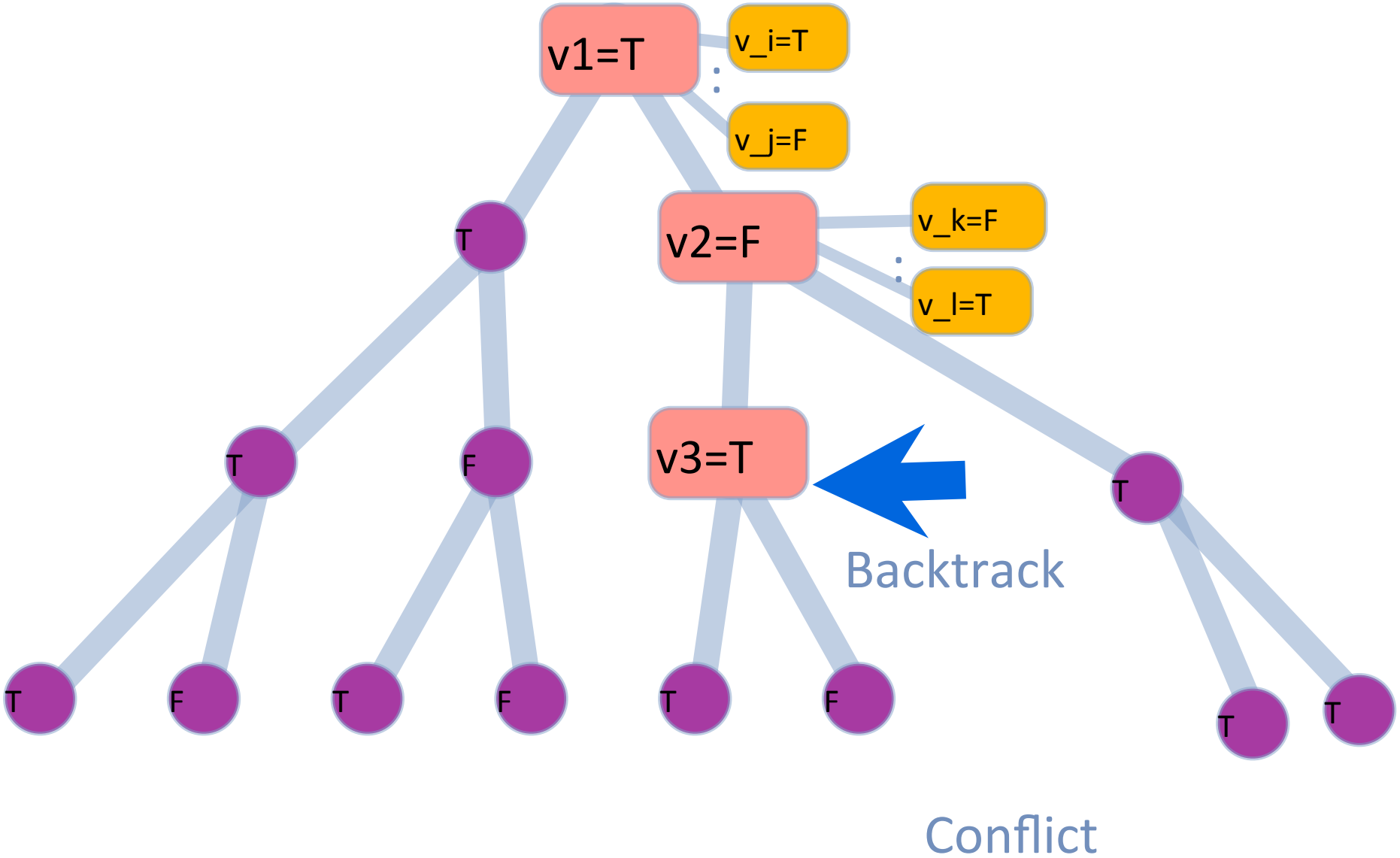
Decision & Propagation



Conflict



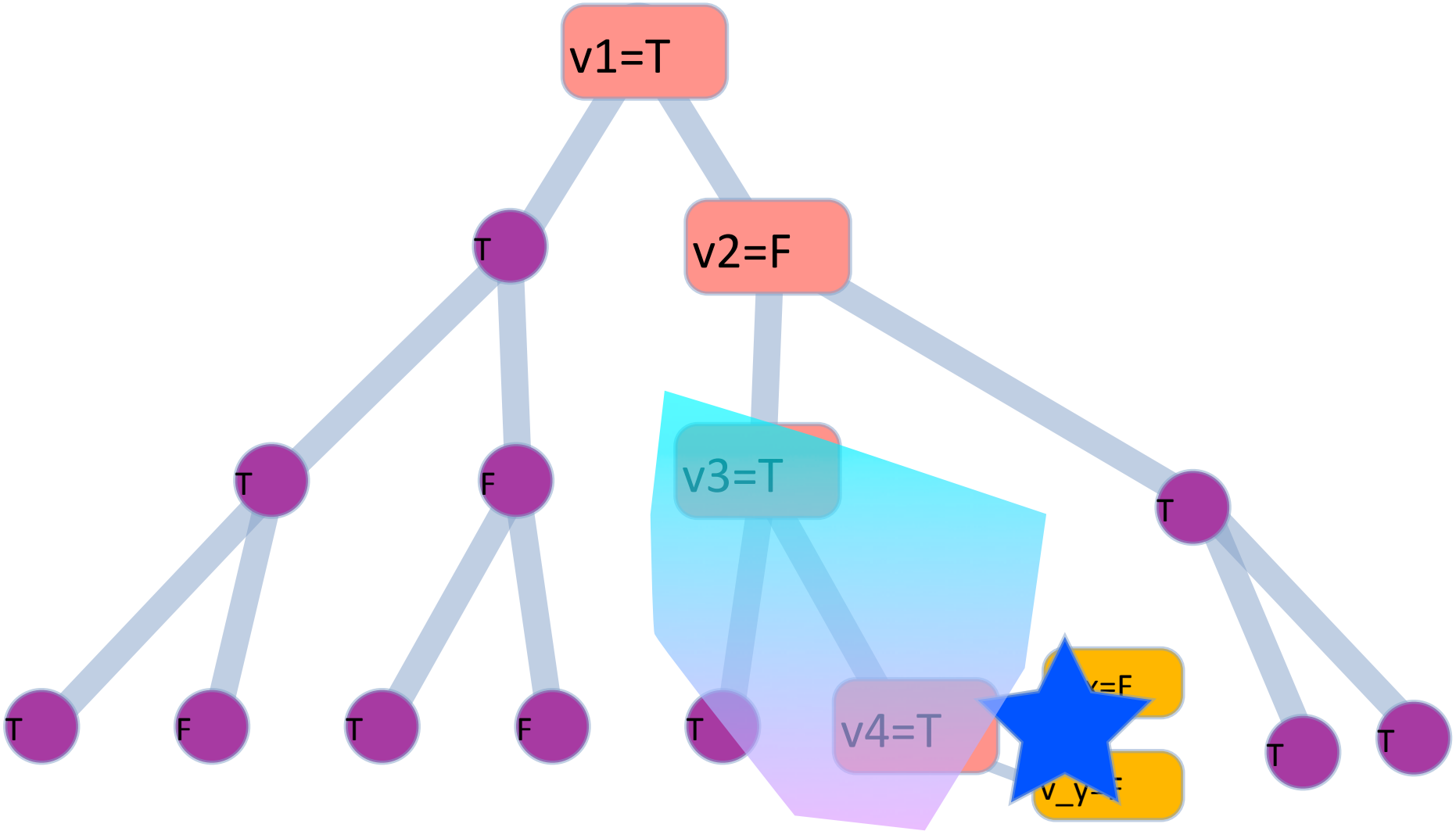
Conflict



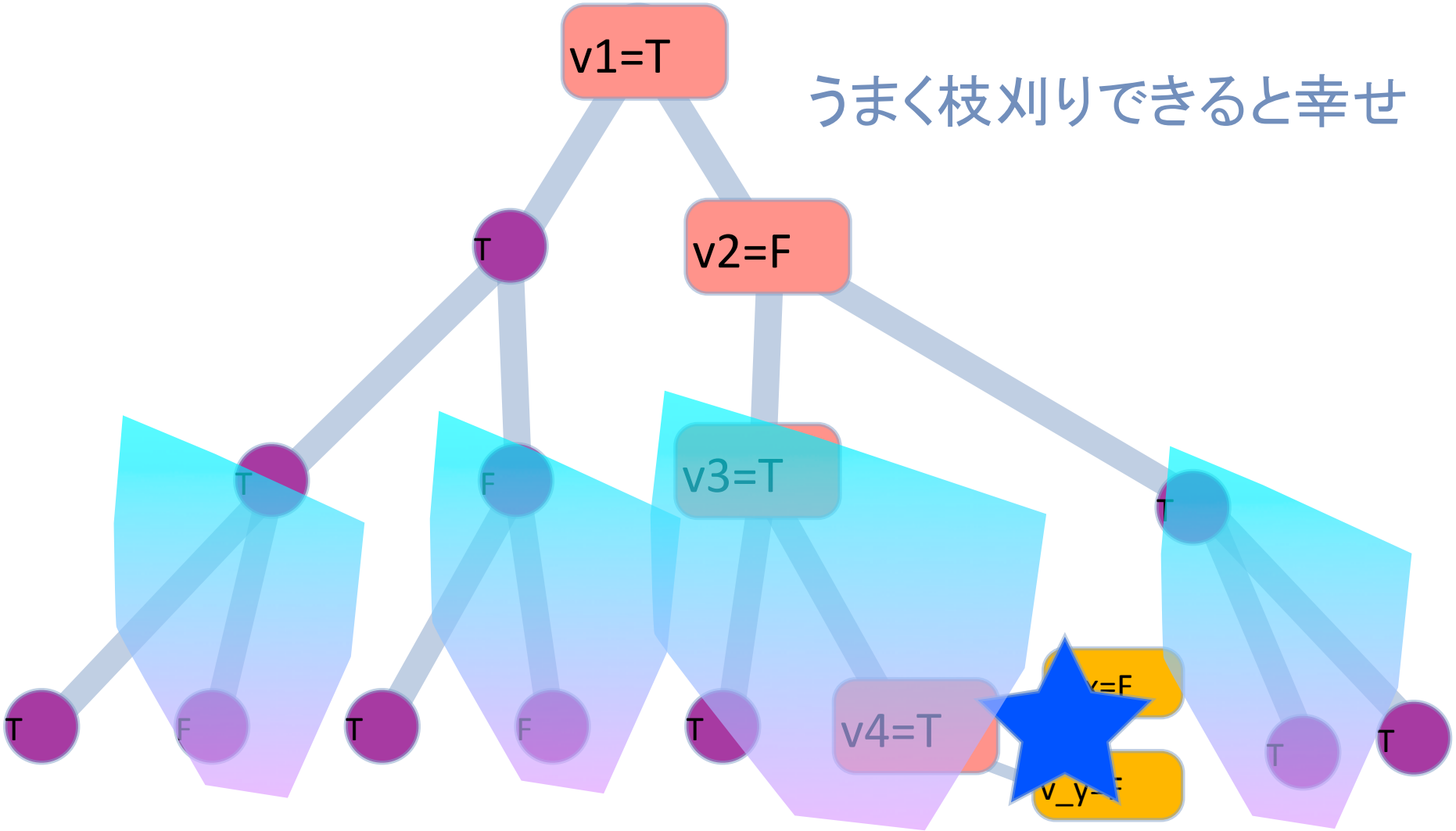
SAT Solvers の歴史

1962	**DPLL**	**#variable ~ 10**
1986	BDDs	#variable ~ 100
1992	GSAT	#variable ~ 300
2001	**Chaff**	**#variable ~ 10,000 (CDCL)**
2004	MiniSAT	#variable ~ 100,000
Currently		#variable ~ 1,000,000

Clause Learning



Clause Learning



Implication Graph

clause C = (x1| x2| ...| xn | y)

IF x?=F then C = **(F | F | ... | F | y =>TRUE)**

Denote exp(y,C)={x1, x2, ..., xn}, imp(C)=y

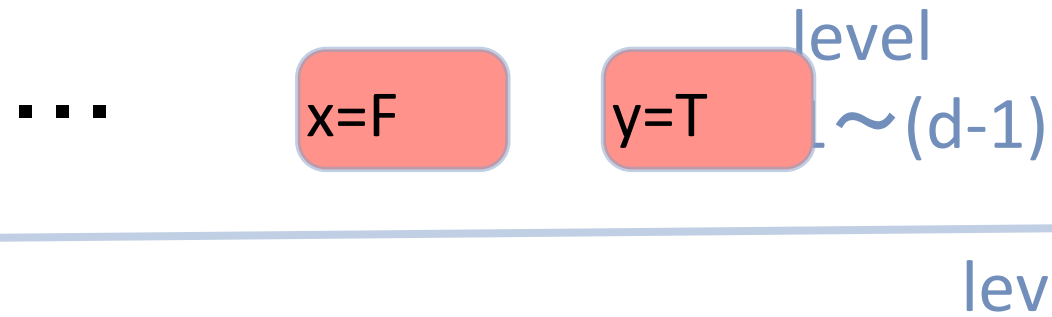
あるクローズで、最後にアサインされた変数 y
クローズの残りは、全部「偽」だったはず。
その関係を グラフGの 有向辺にする。

Implication graph G=(V,A)

V = {v |decision かimplication で割当済の変数}

A = {(x, y)|x ∈ exp(y,C), y = imp(C), ∀C }

例
レベルd のインプリケーショングラフ



¬x | a | b
y| c | d
:
a | c
¬a |b
¬a | ¬b |c
x |¬y | c | ¬d
c | d

例
レベルd のインプリケーショングラフ

level
1〜(d-1)

… x=F y=T

level d

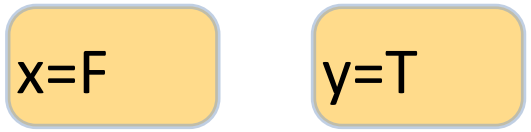
Decision a=T

a=T

~~¬x | a | b~~
~~y| c | d~~
:
a | c
¬a |b
¬b |¬c
~~x~~ |~~¬y~~ | c | ¬d
c | d

例
レベルd のインプリケーショングラフ

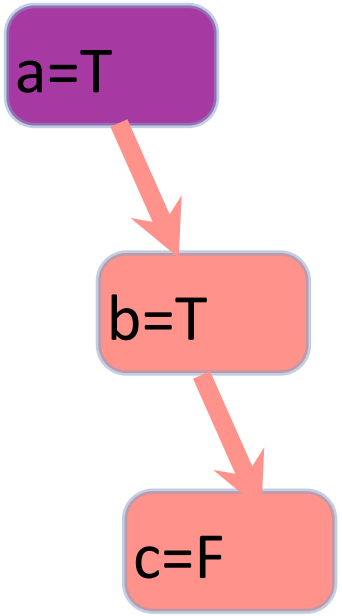
…



level
1〜(d-1)

level d

~~¬x | a | b~~
~~y| c | d~~
:
~~a | c~~
~~¬a~~ |~~b~~
~~¬b~~ |¬c
~~x~~ | ~~¬y~~ | c | ¬d
c | d

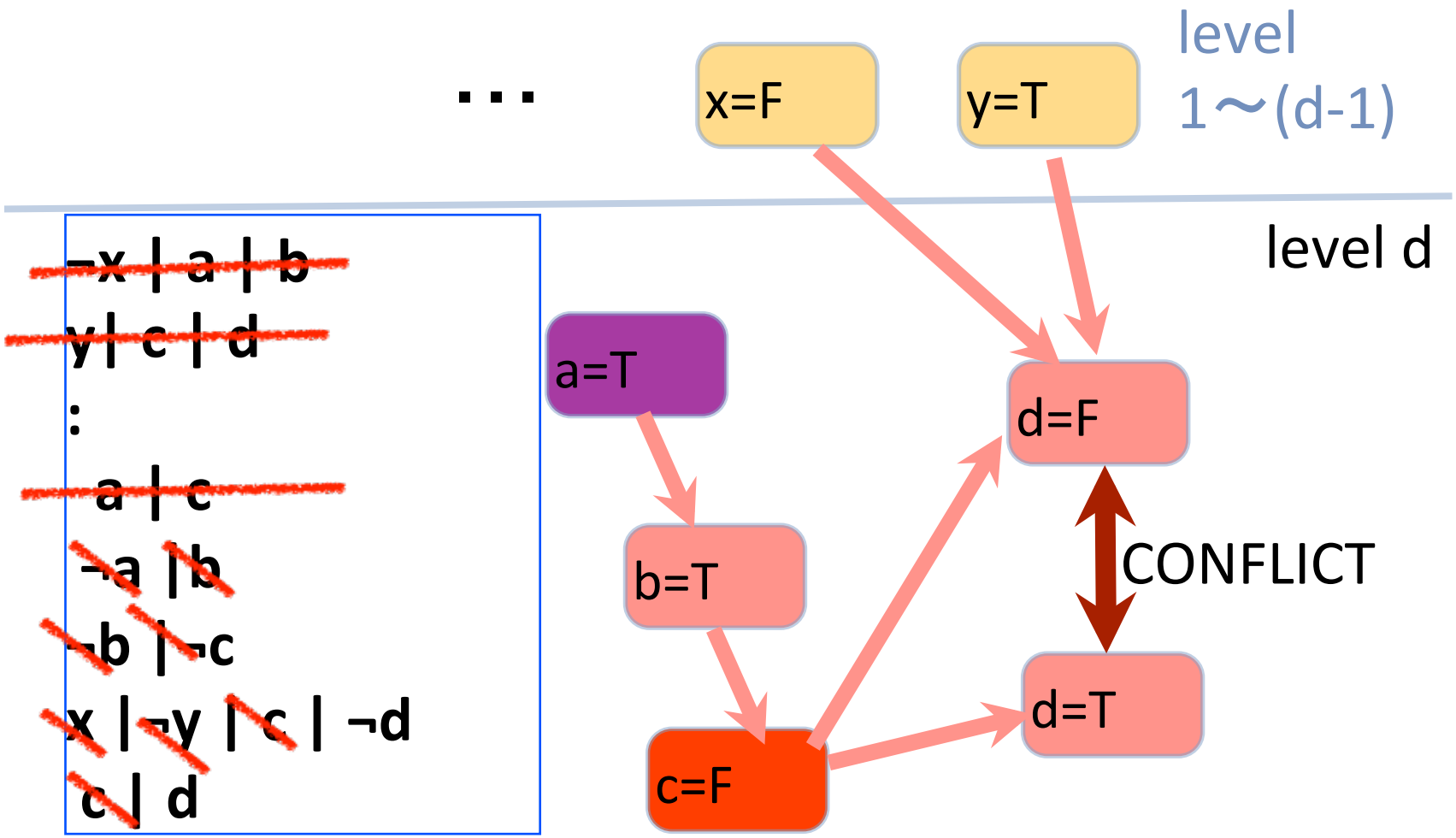


a=T

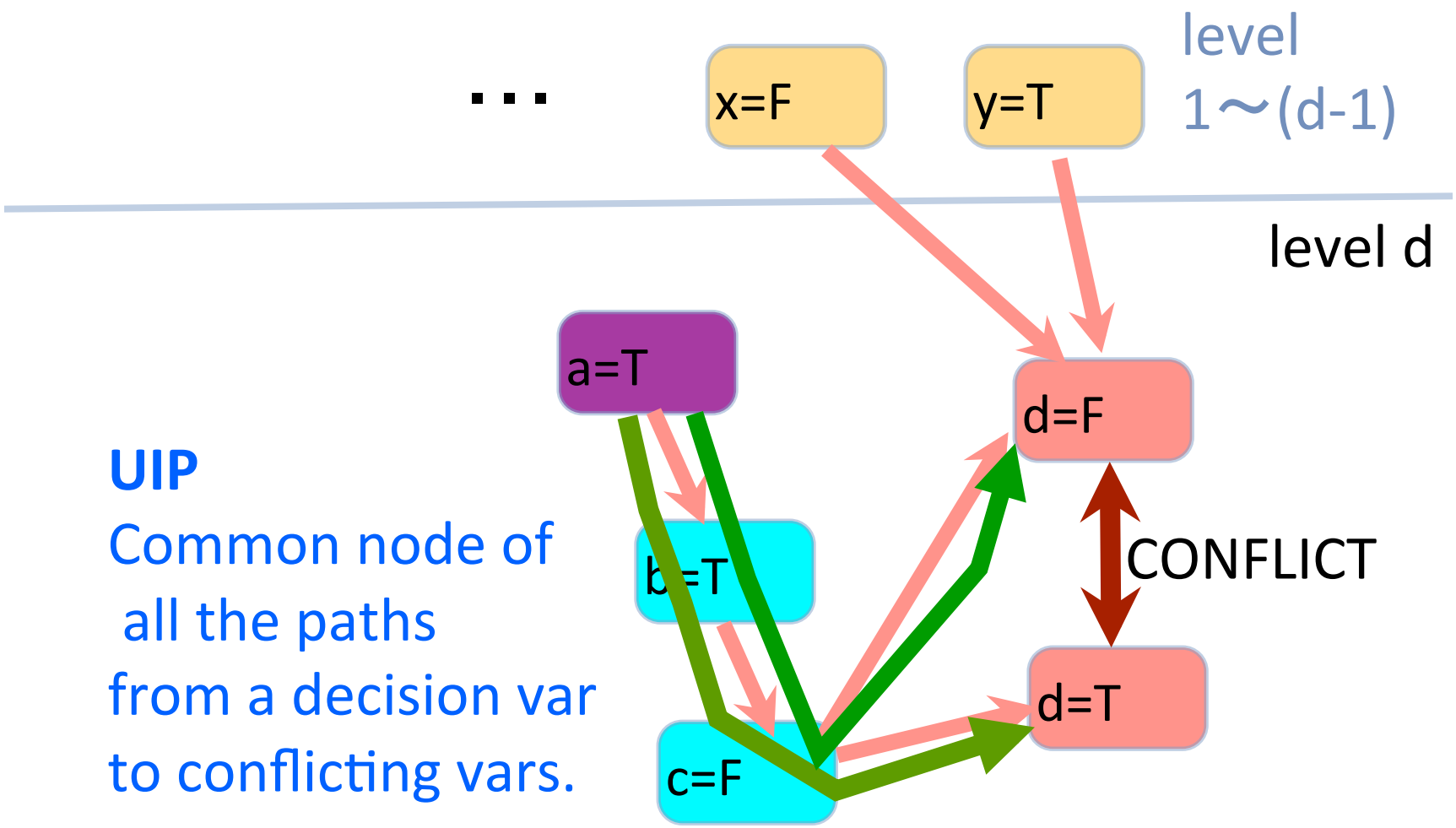
b= T

Unit Propagation c= F

Learning scheme: 1UIP cut [M. Moskewicz, et al., 2001]

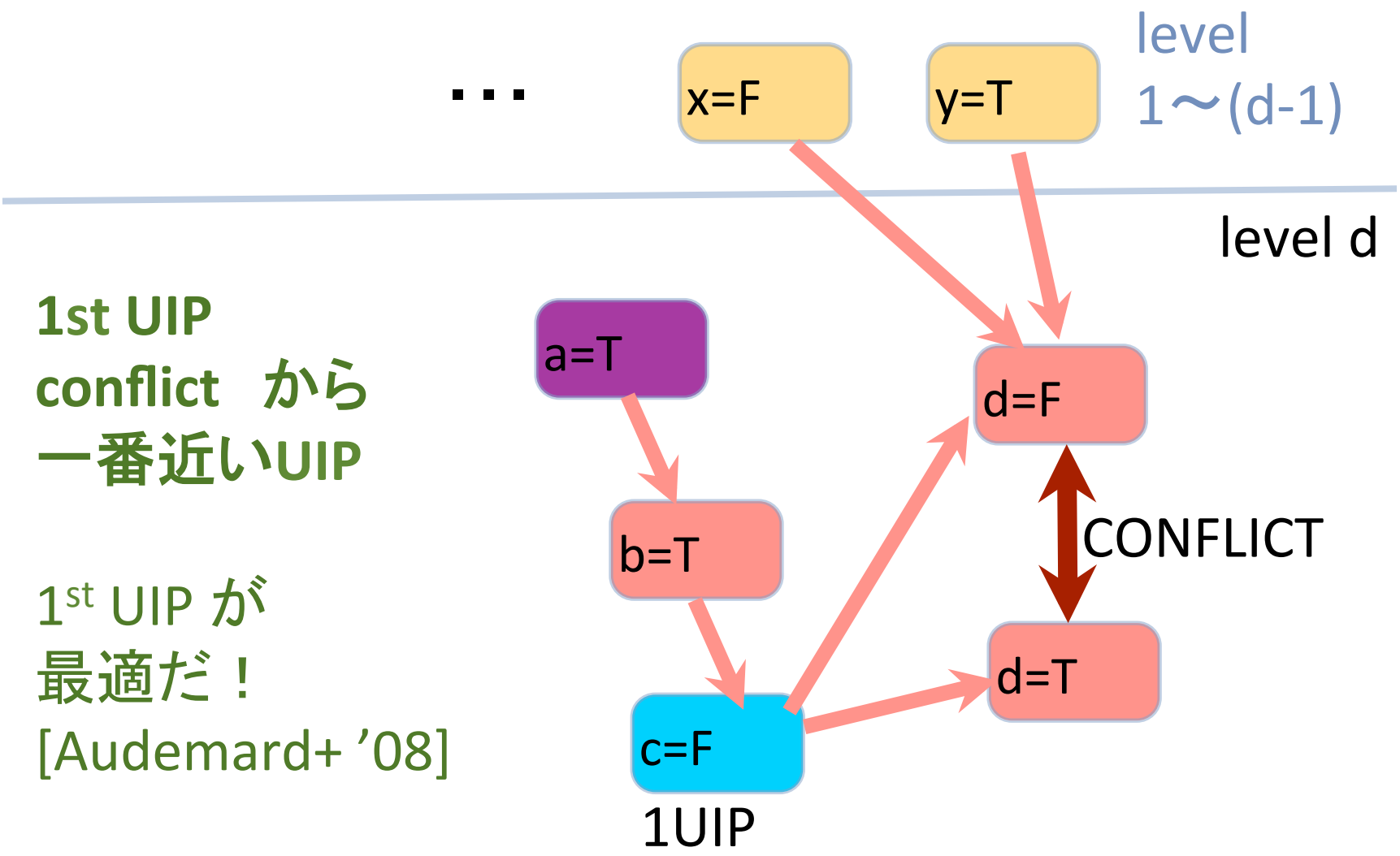


Unique Implication Point



UIP
Common node of
all the paths
from a decision var
to conflicting vars.

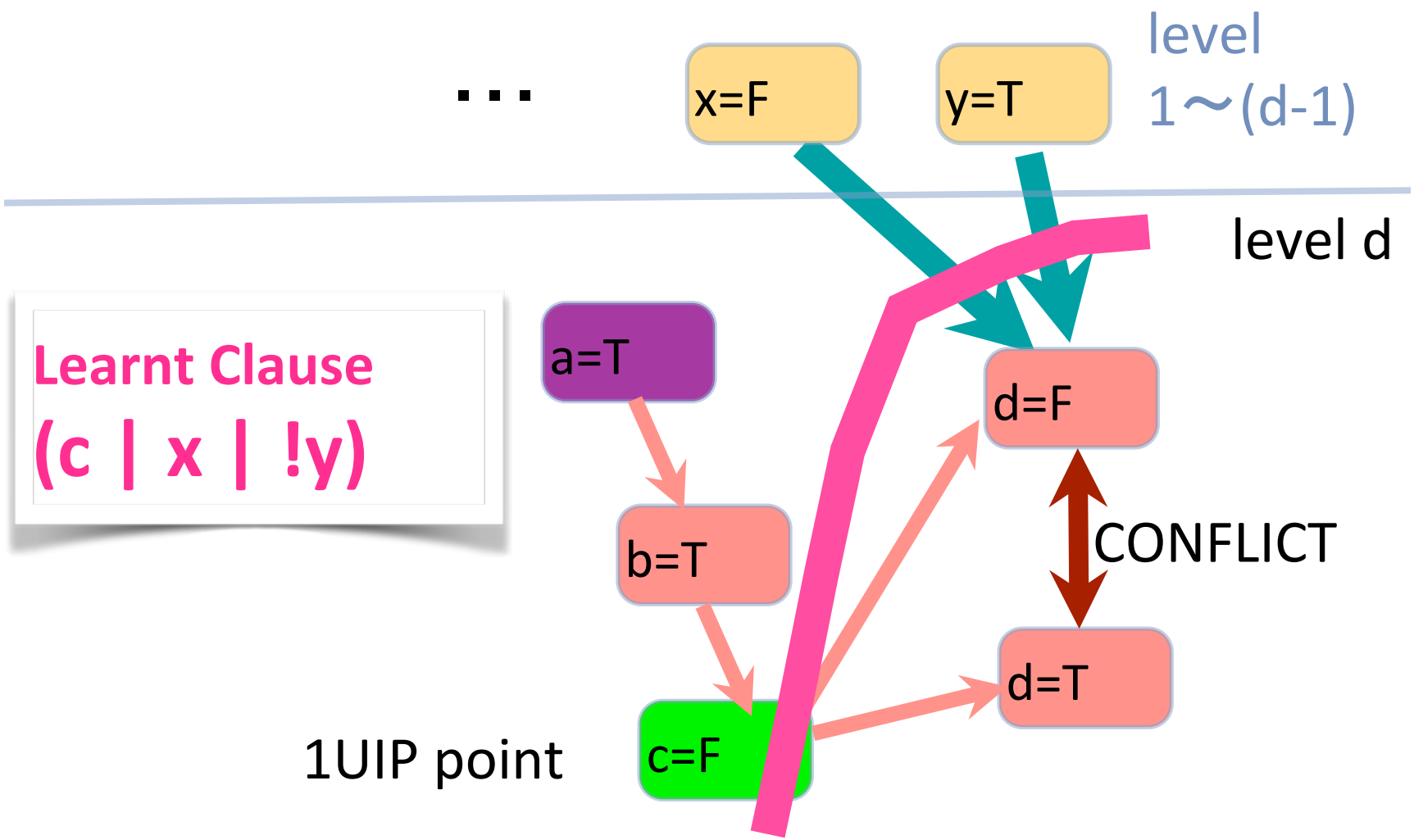
1st UIP



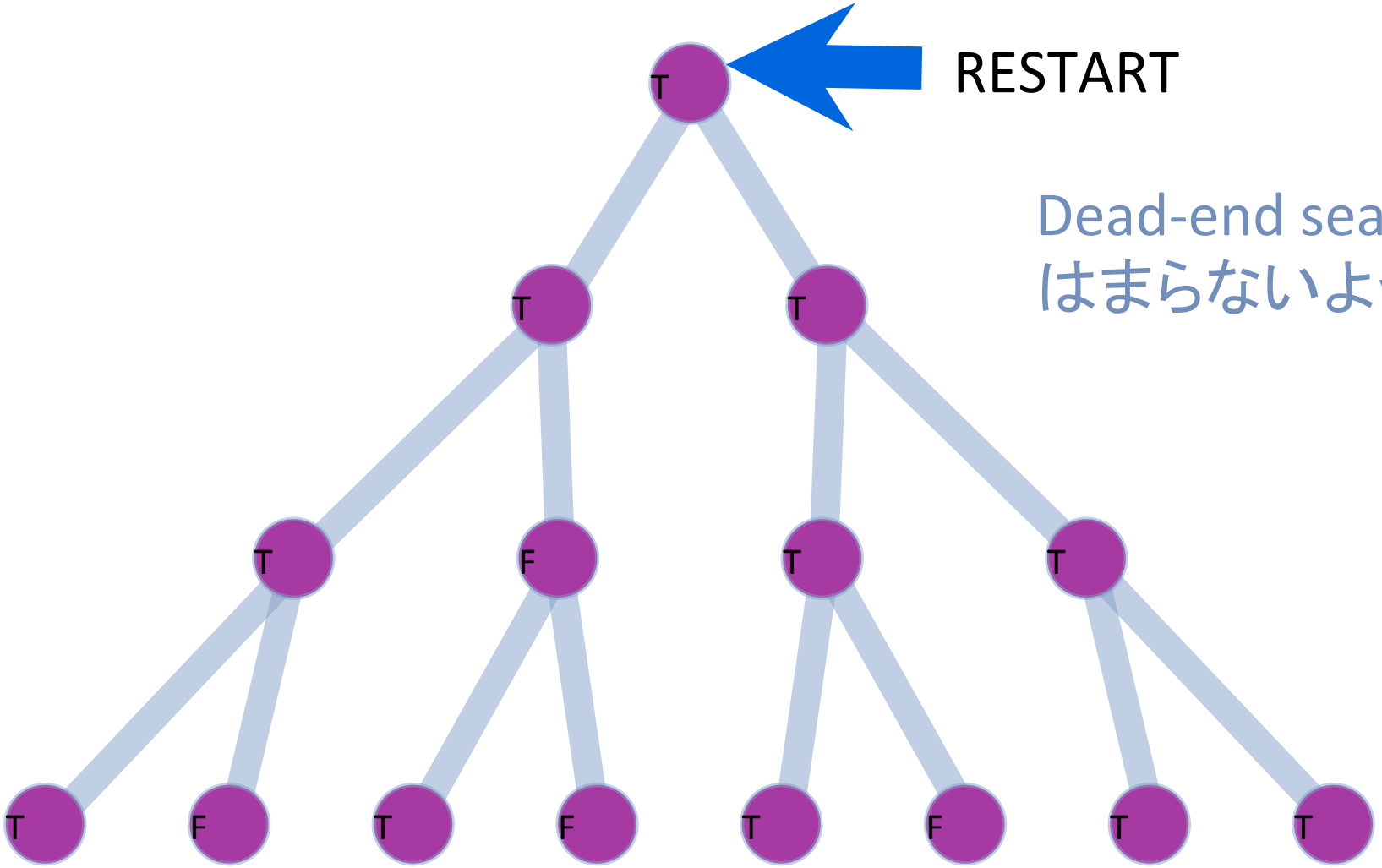
1st UIP
conflict から
一番近いUIP

1st UIP が
最適だ！
[Audemard+ ’08]

Learning scheme: 1st UIP cut [M. Moskewicz, et al., 2001]



Restart,（Backtrackではなく）



Dead-end searchに
はまらないように

VSIDS (Variable State Independent Decaying Sum)

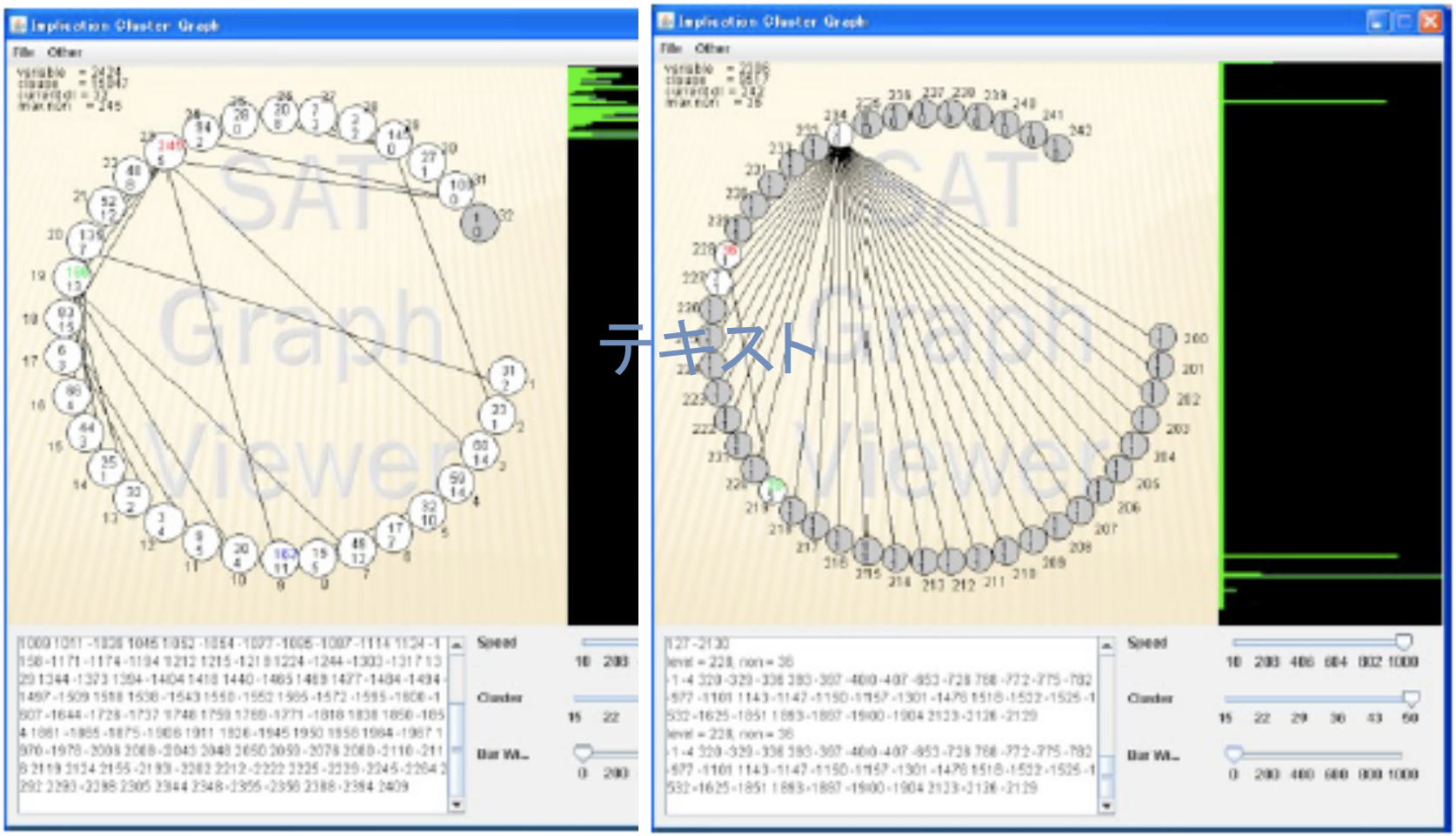
- スコアづけして **decision variable** を選ぶ.
  - **学習クローズにでた変数のスコアをあげる**
  - **時間がたつと減らす**

- スコアが大きい方が大事
  - 良くでてくる、かつ短いクローズのスコアが大きくなりがち

研究の話
可視化と観察

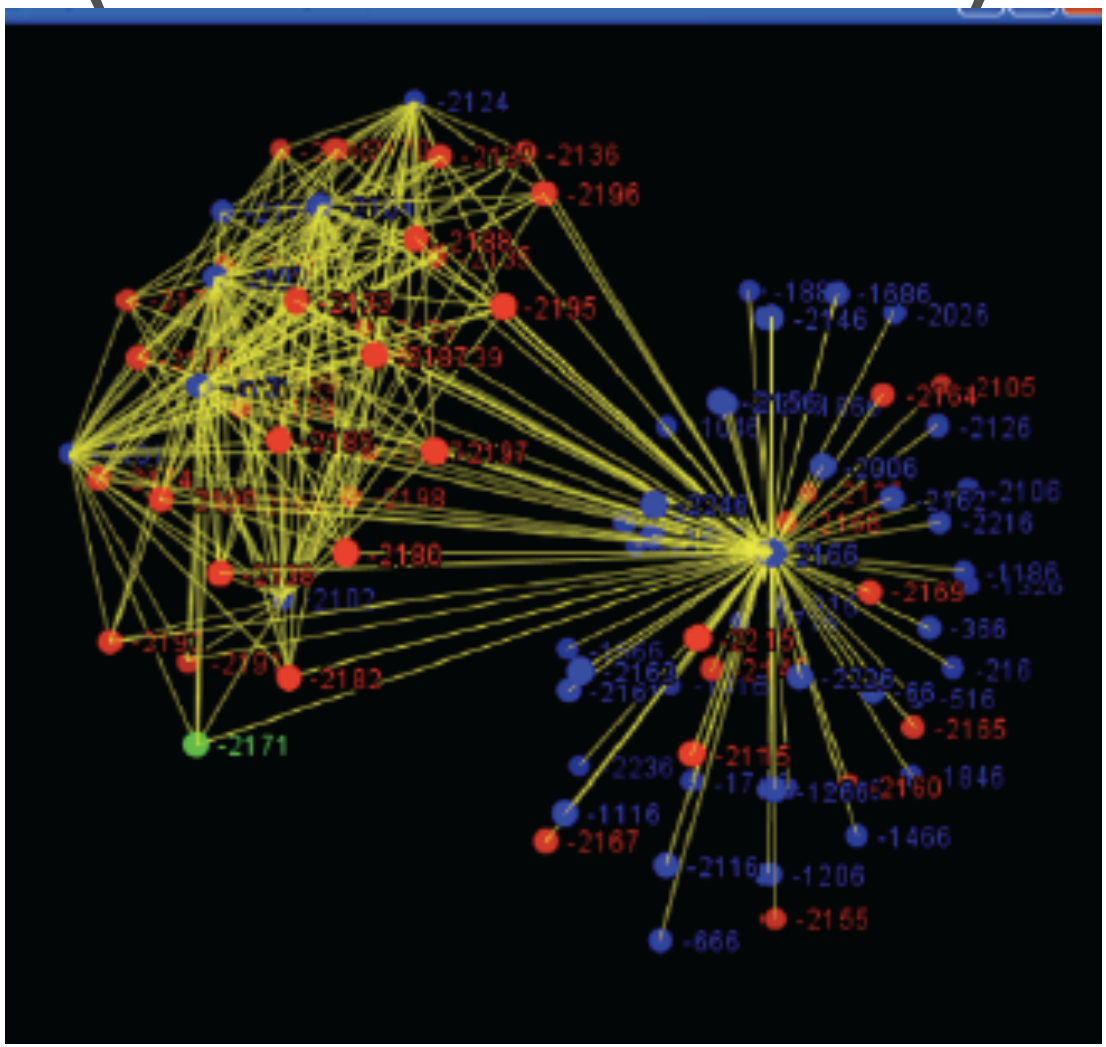
面白い例を見つけた
（薗部さん）

Contracted Implication Graph
（スナップショット）

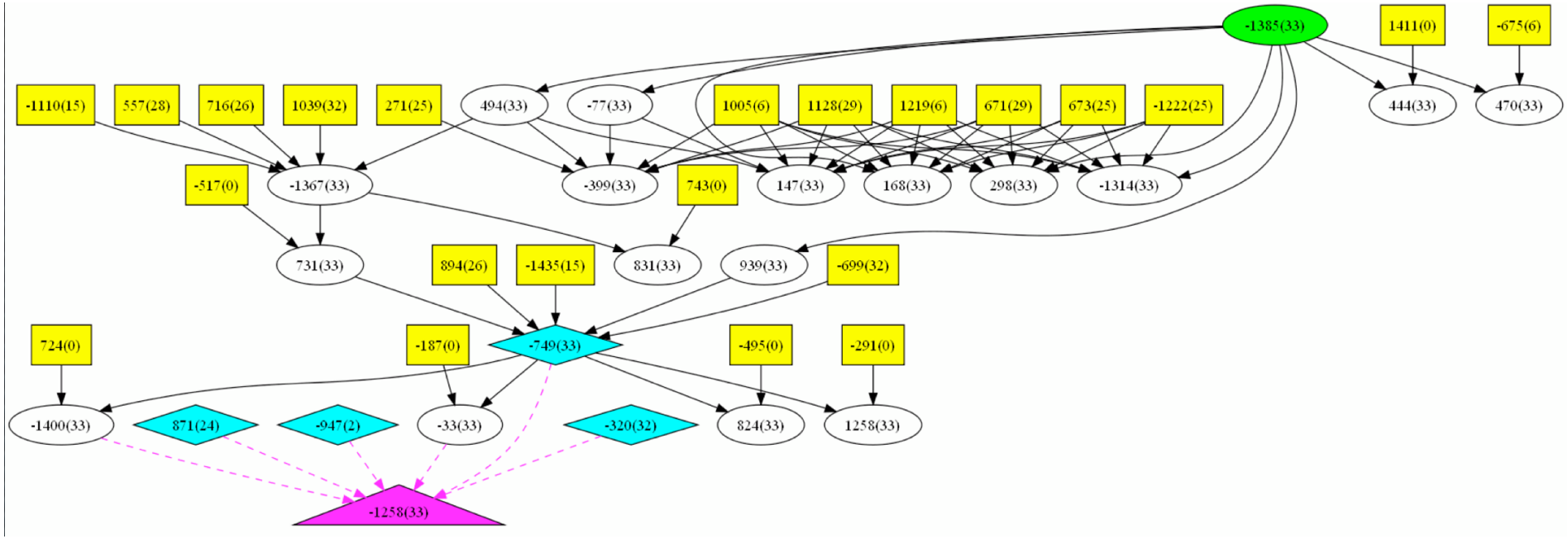


あるノードに大量のエッジ

Implication Graph (スナップショット)



Implication Graph & Learnt Clause



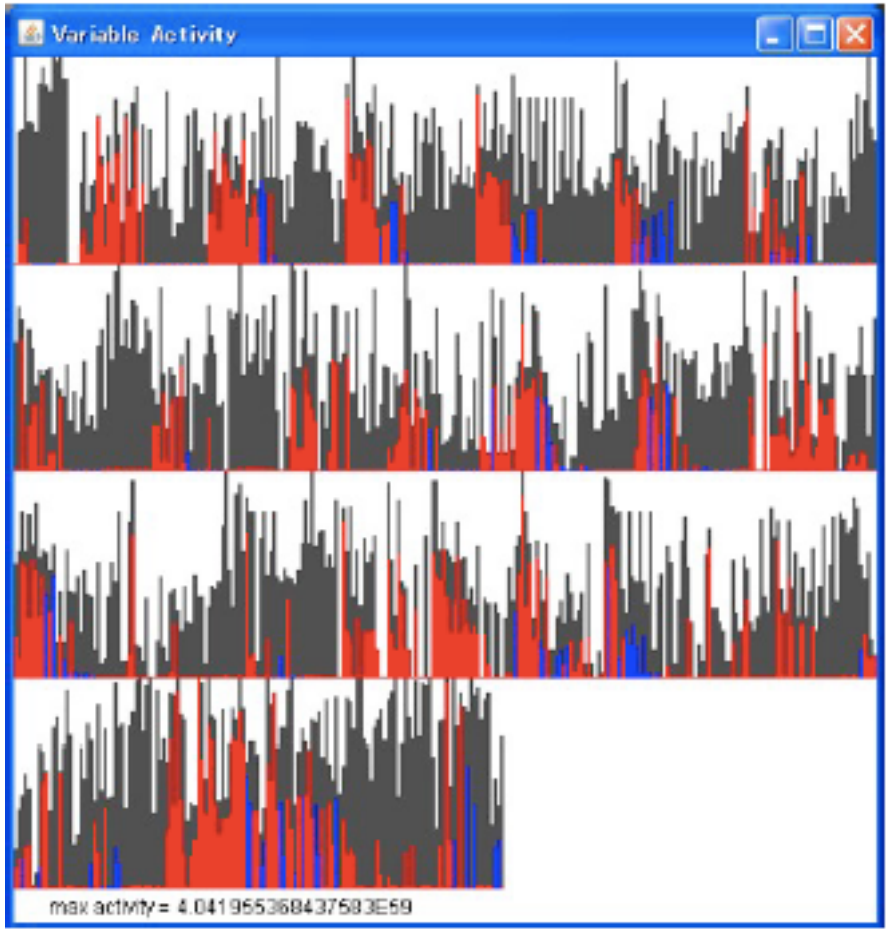
assigned at previous level

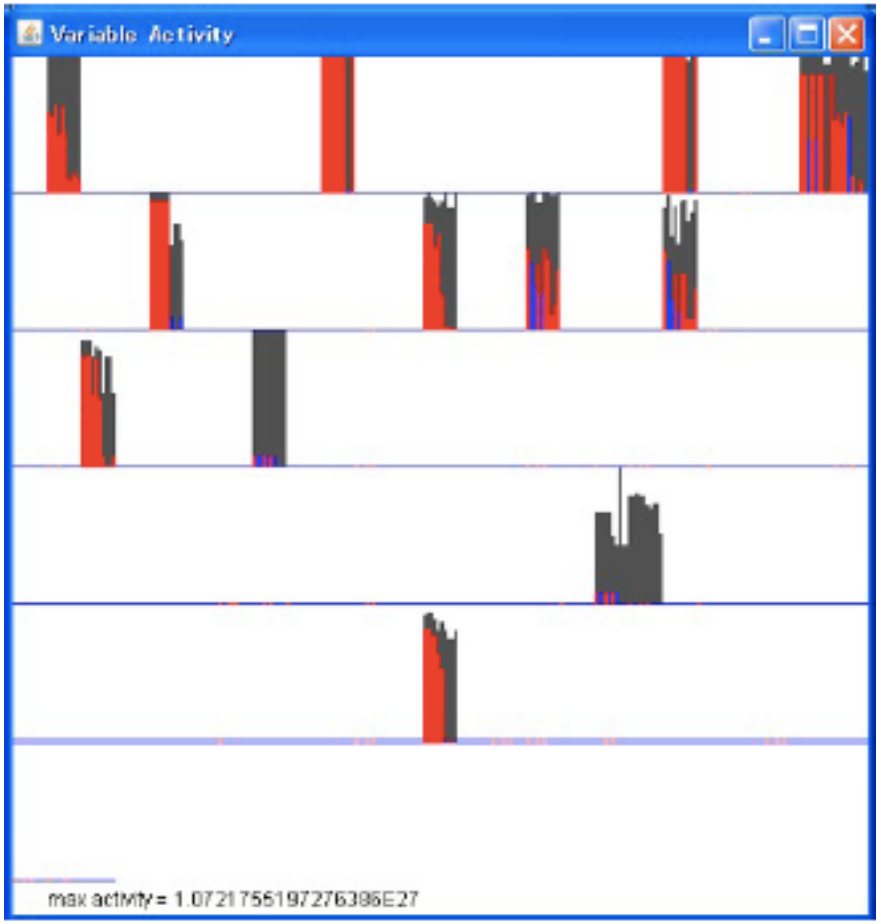
decision at current level

implied current level

variables of learnt clause

VSIDS scores of variables





Black : max VSIDS
Red : current VSIDS
横軸が　variable (6段)

でてこない変数が大量にある

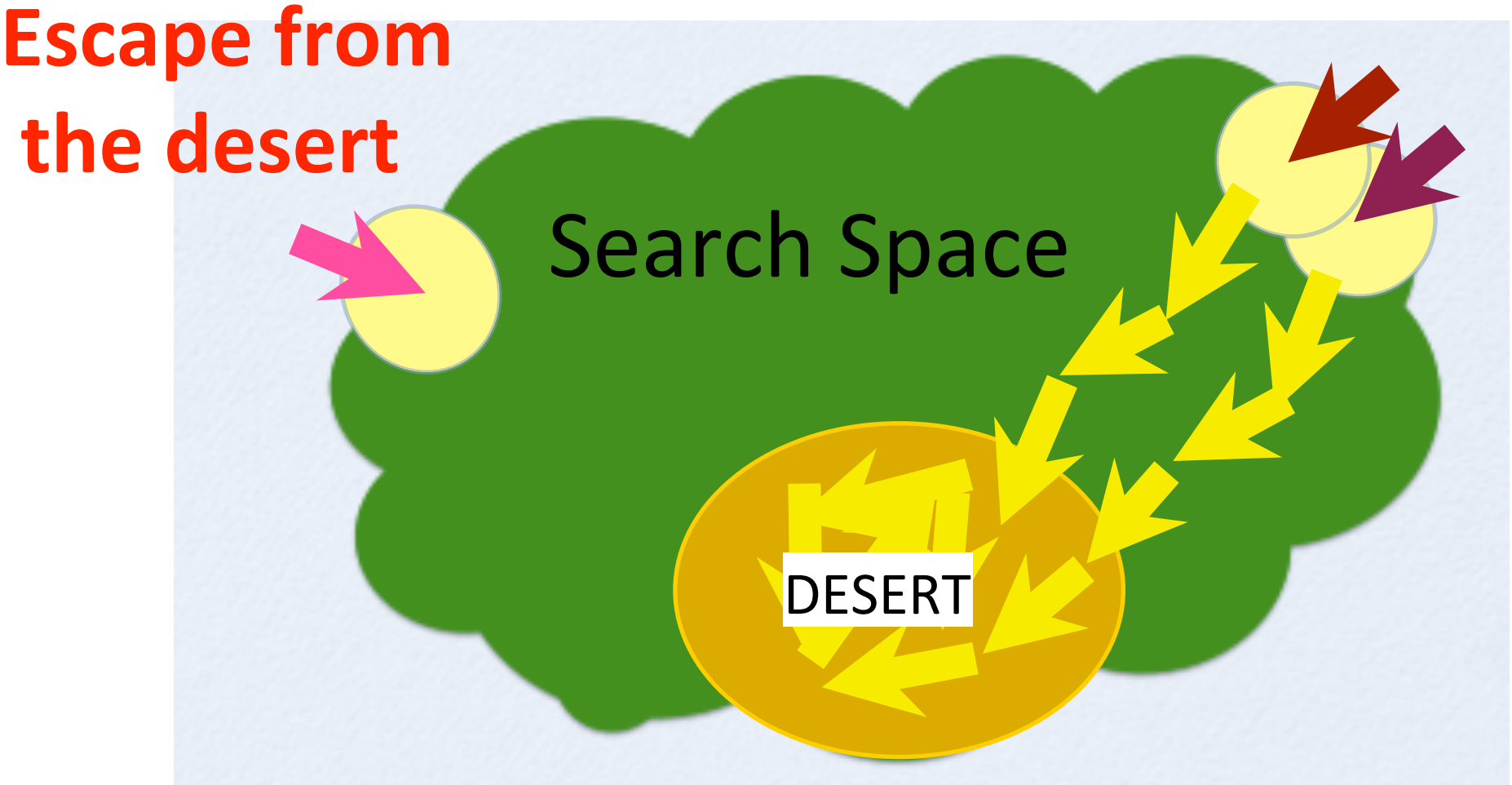
Observation

- Implication Graph は大きい。
- 難しい問題では
  - **同じ変数に、大量の in-edge が繰り返される**
  - **全く探索されない変数がありうる。**

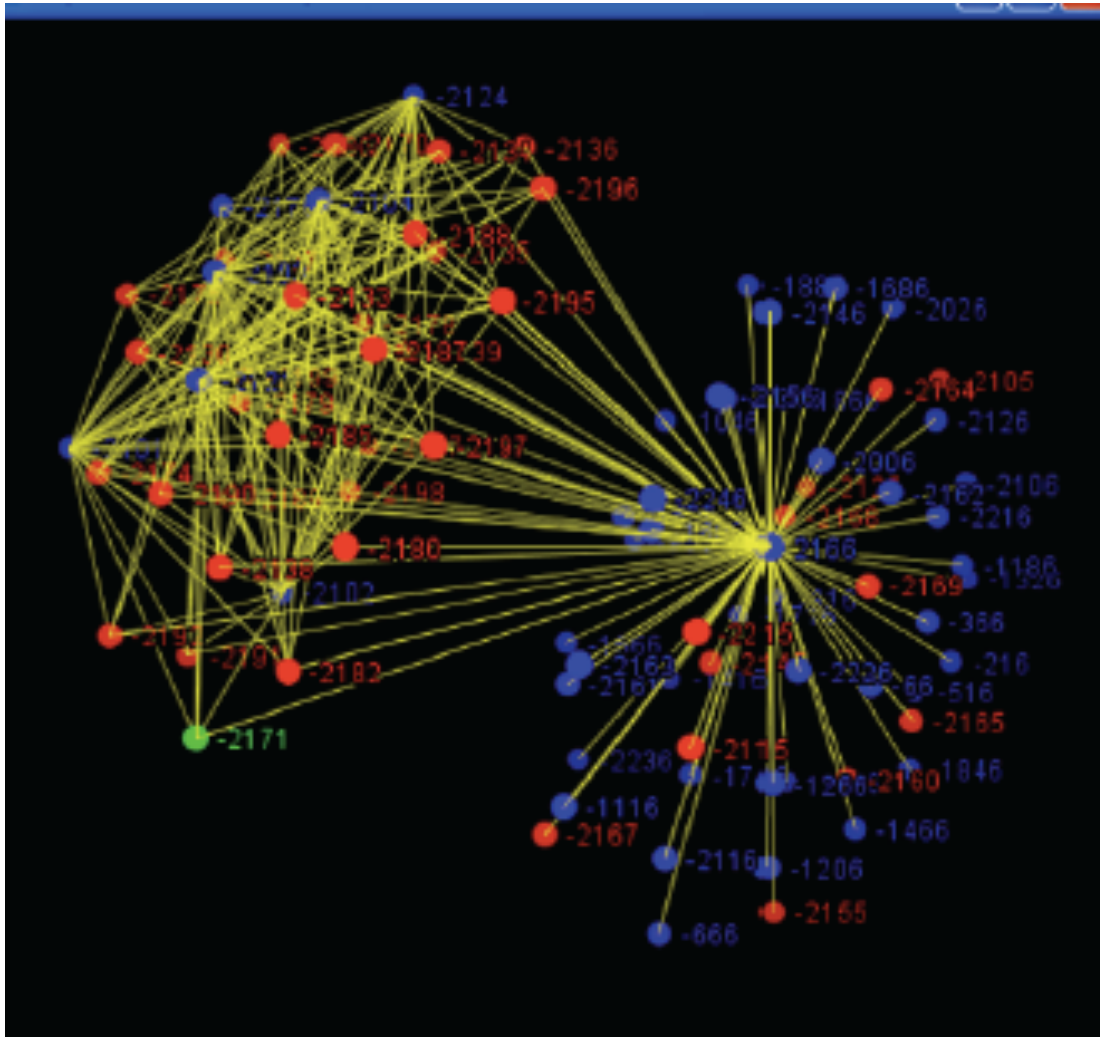
    より、多様性をもたせたい

Escape from Desert
(difficult problem)

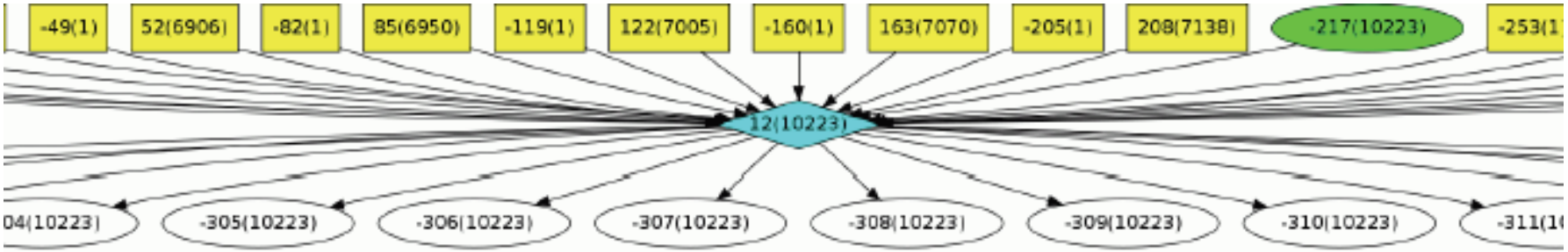
Search Space Diversity



Huge number of in-edges to a variable, (i.e., |exp(C)| is huge), appears repeatedly in implication graph



Proposal -- CIR
Counter Implication Restart



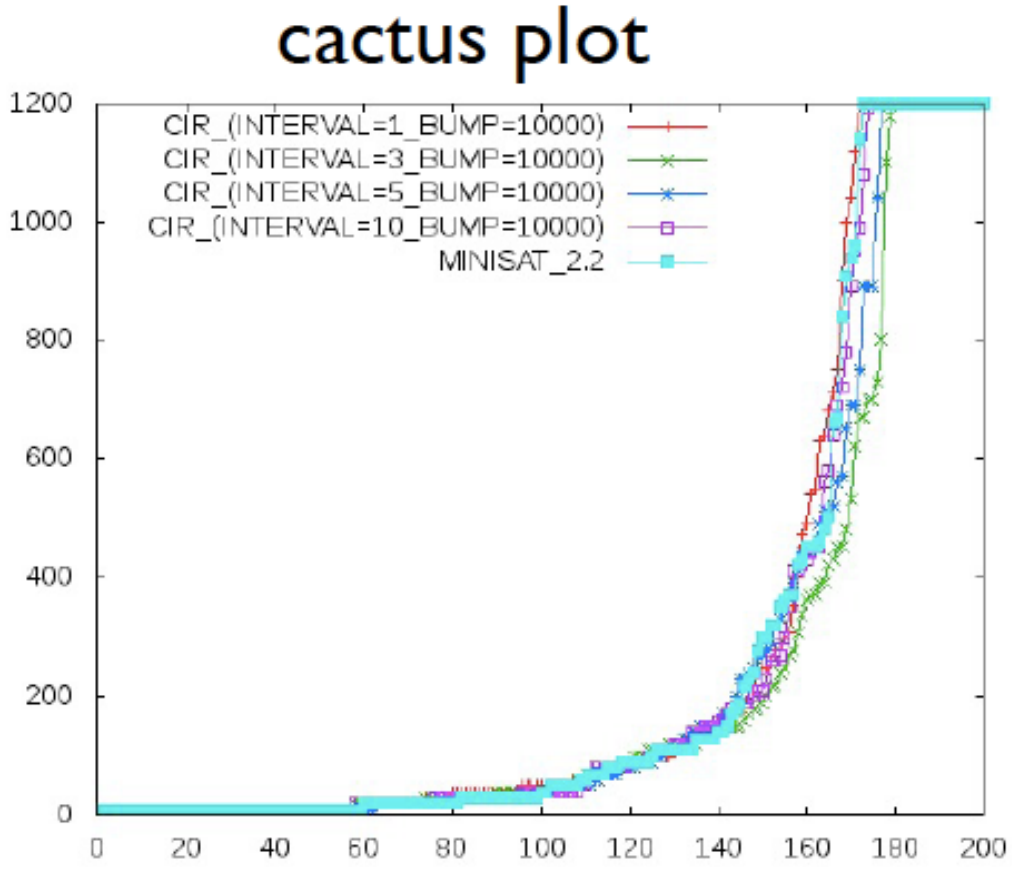
- **in-degree** が大きい変数を重要とみなす
- 長いクローズの場合ほど、大きく設定する

(F|F|F|F|F|F|F|F|F|F|F|F|F|F|F|F|F|v)

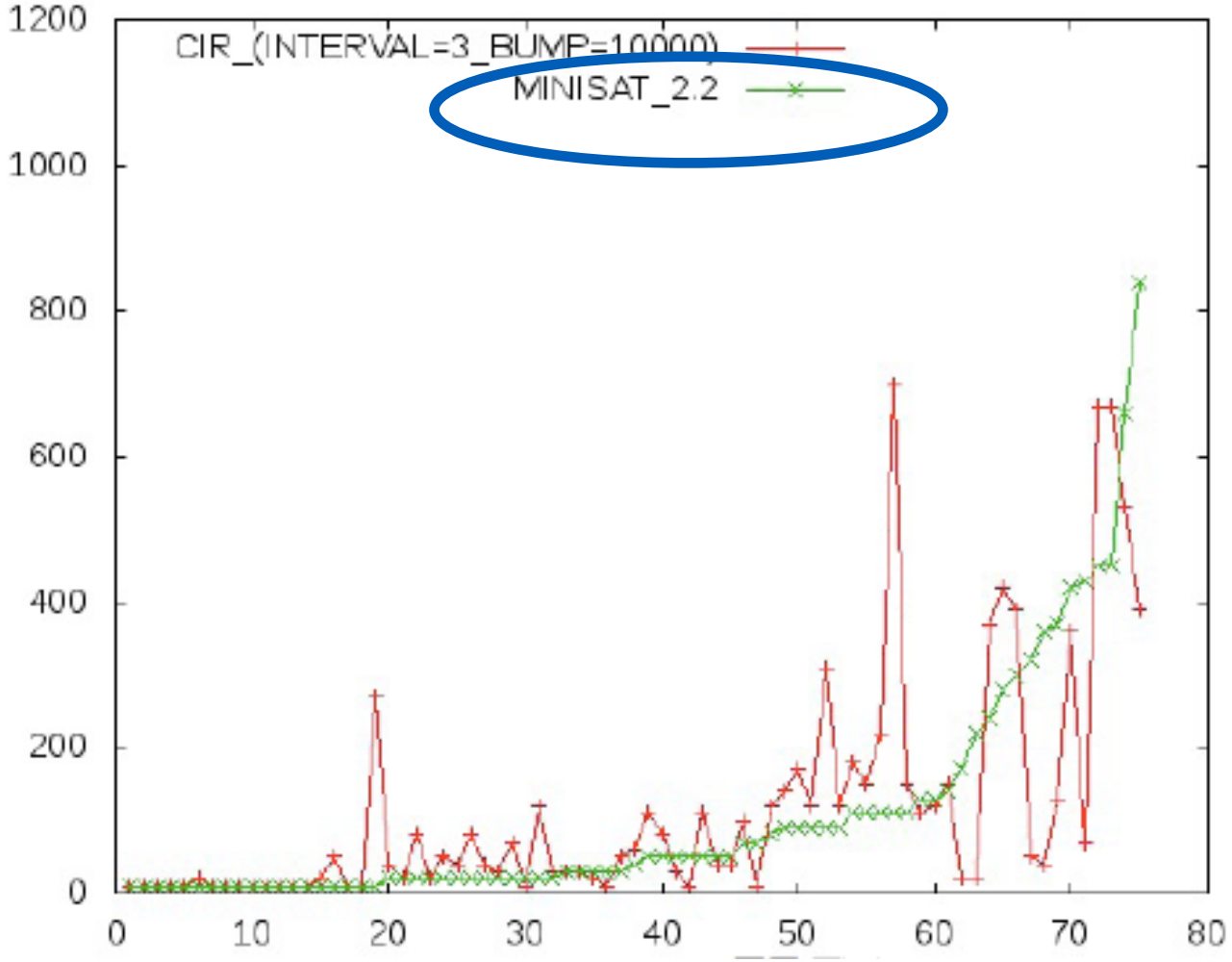
 # 直感に反するけれど

Cactus Plot

Time

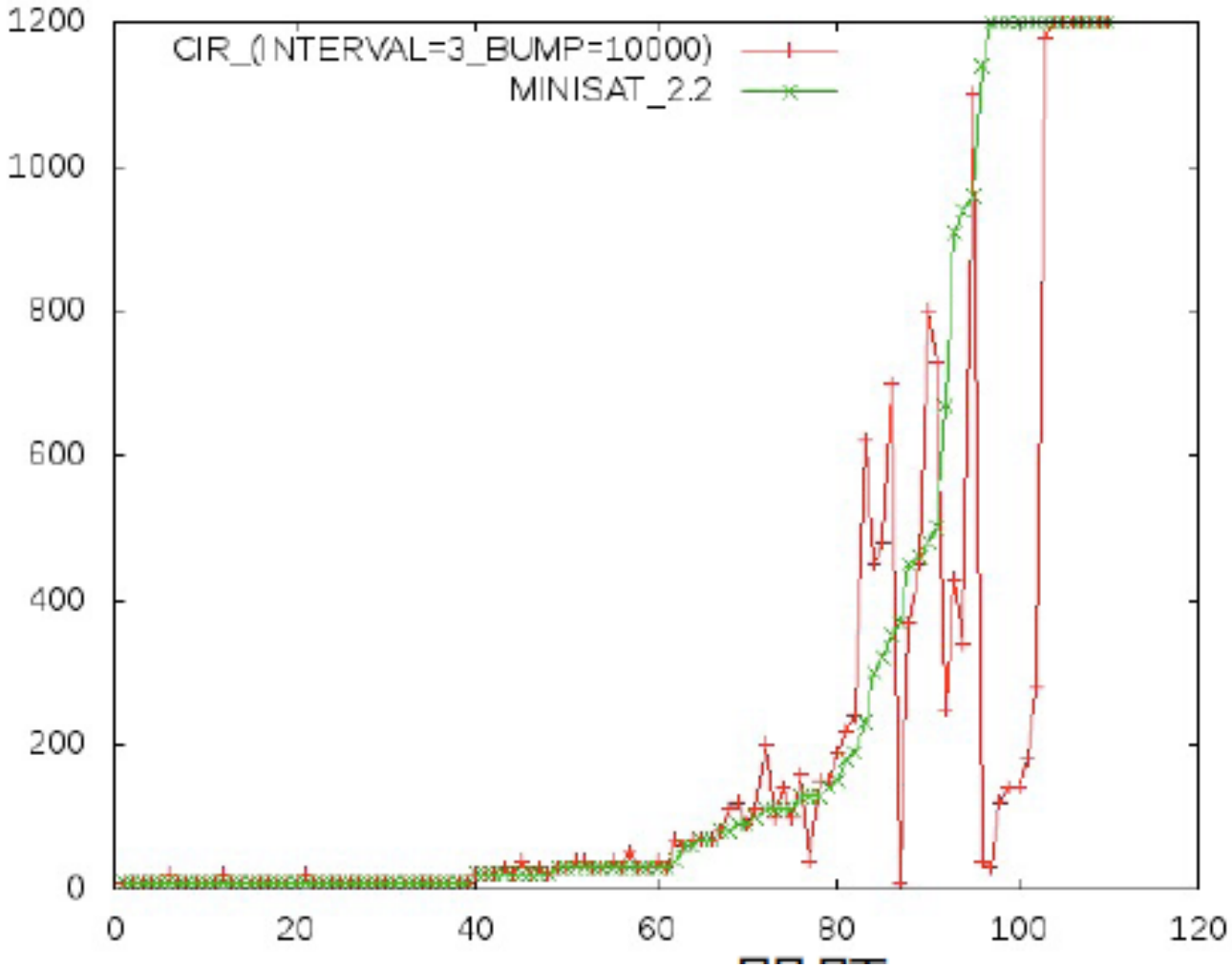


SAT problems



Problems are sorted by the performance of MINISAT 2.2

UNSAT problems



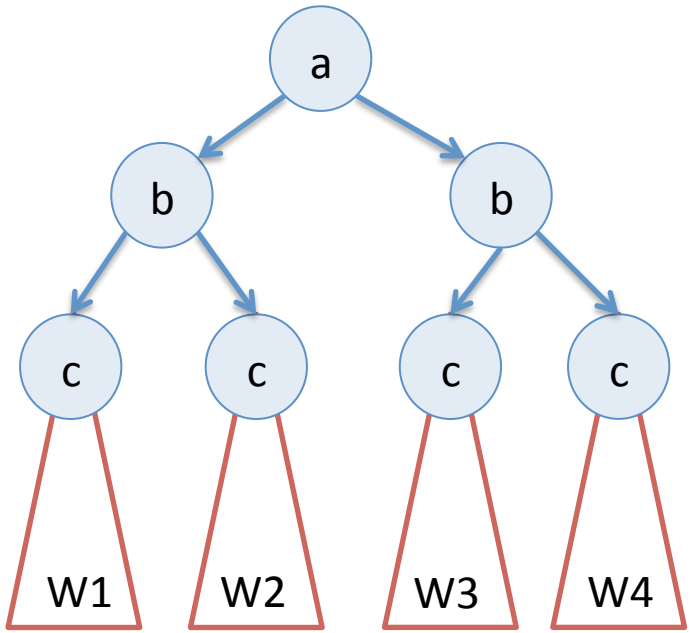
パラメータチューニング
経験的にマジックナンバーを探す

	SAT	UNSAT	total
MINISAT2.2	75	96	171
CIR(INT=1, BUMP=10000)	70	100	170
CIR(INT=3, BUMP=10000)	75	103	178
CIR(INT=5, BUMP=10000)	73	102	175
CIR(INT=10, BUMP=10000)	72	101	173

並列化

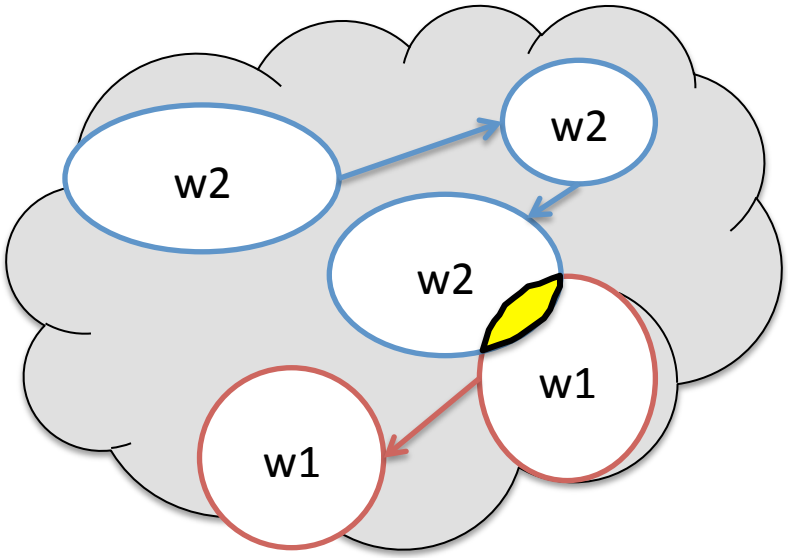
並列SATソルバ

Divide and Conquer



ロードバランス

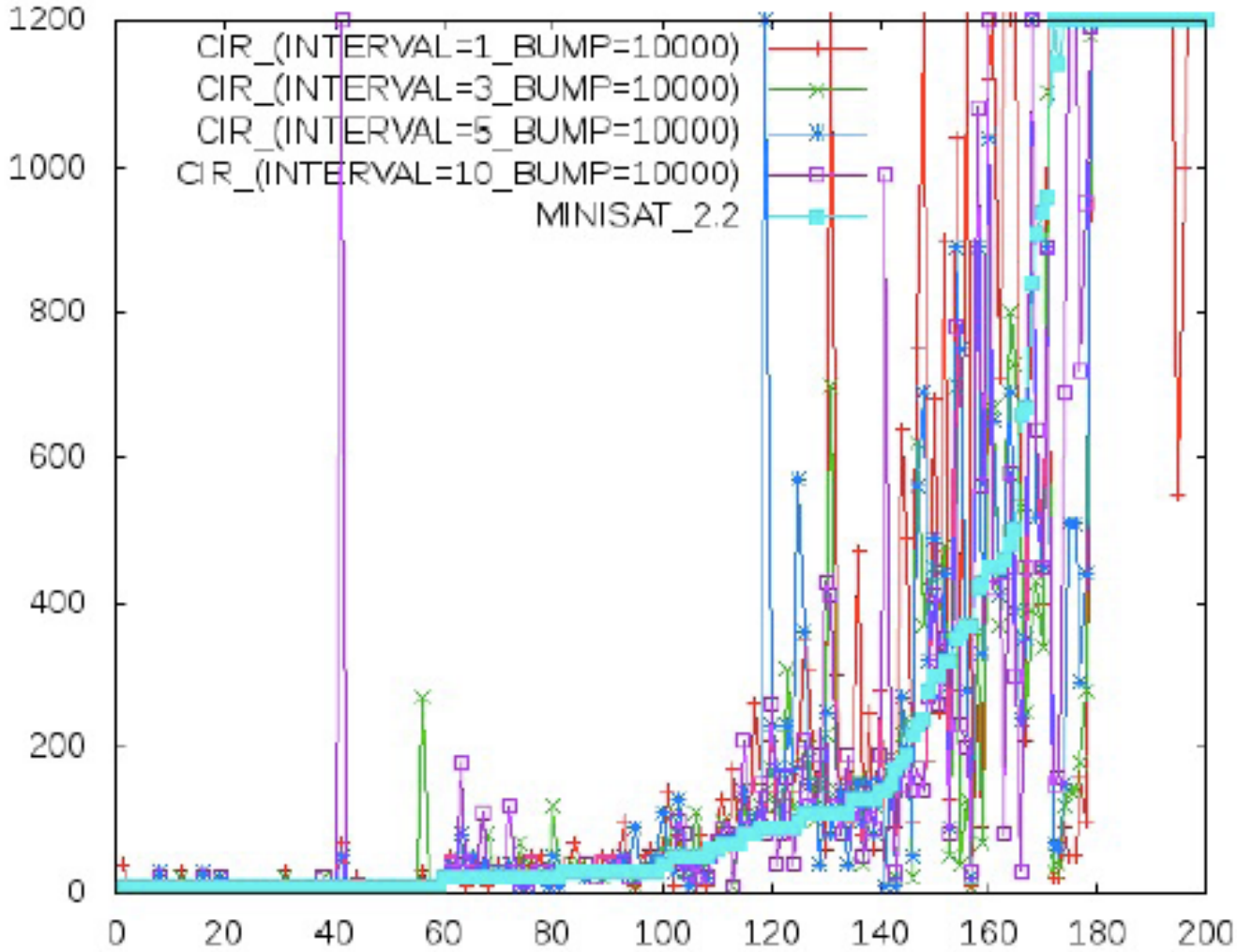
Portfolio



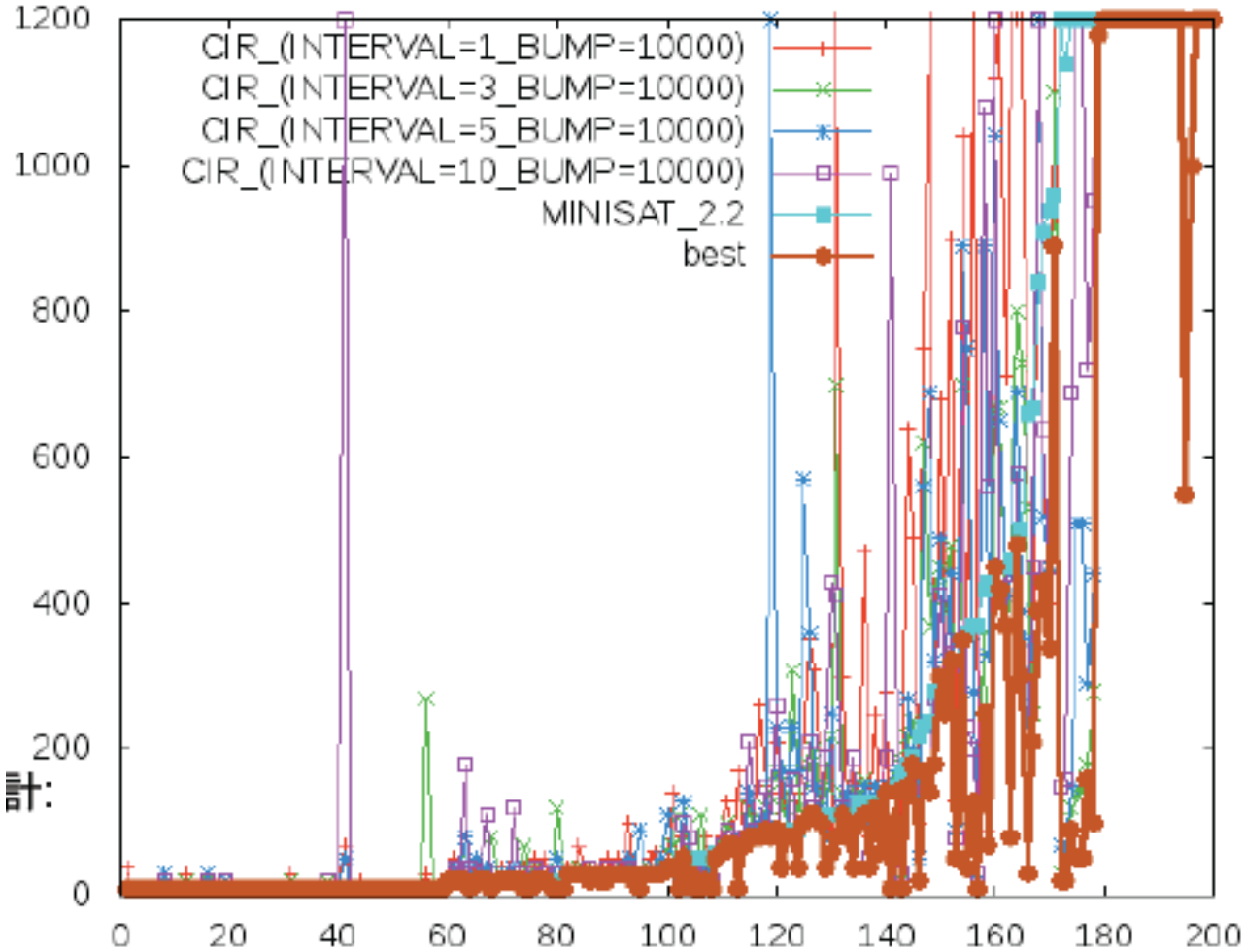
問題

集中と多様

色々走らせてみる


CIR_(INTERVAL=1_BUMP=10000)
CIR_(INTERVAL=3_BUMP=10000)
CIR_(INTERVAL=5_BUMP=10000)
CIR_(INTERVAL=10_BUMP=10000)
MINISAT_2.2
1200
1000
800
600
400
200
0
0
20
40
60
80
100
120
140
160
180
200

Naive portfolio



解探索の特徴

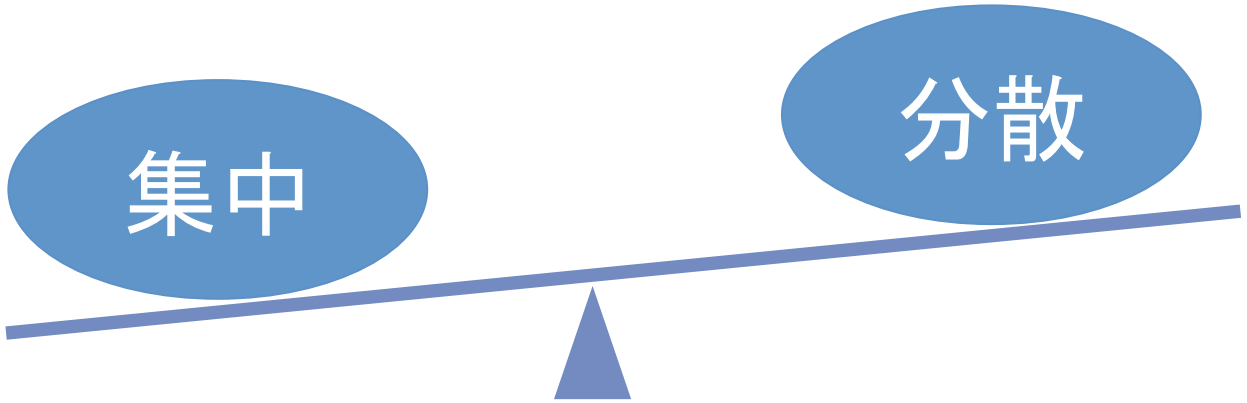
問題の構造を利用

実際の問題から派生した問題には

変数間に強い繋がりが見つかることも

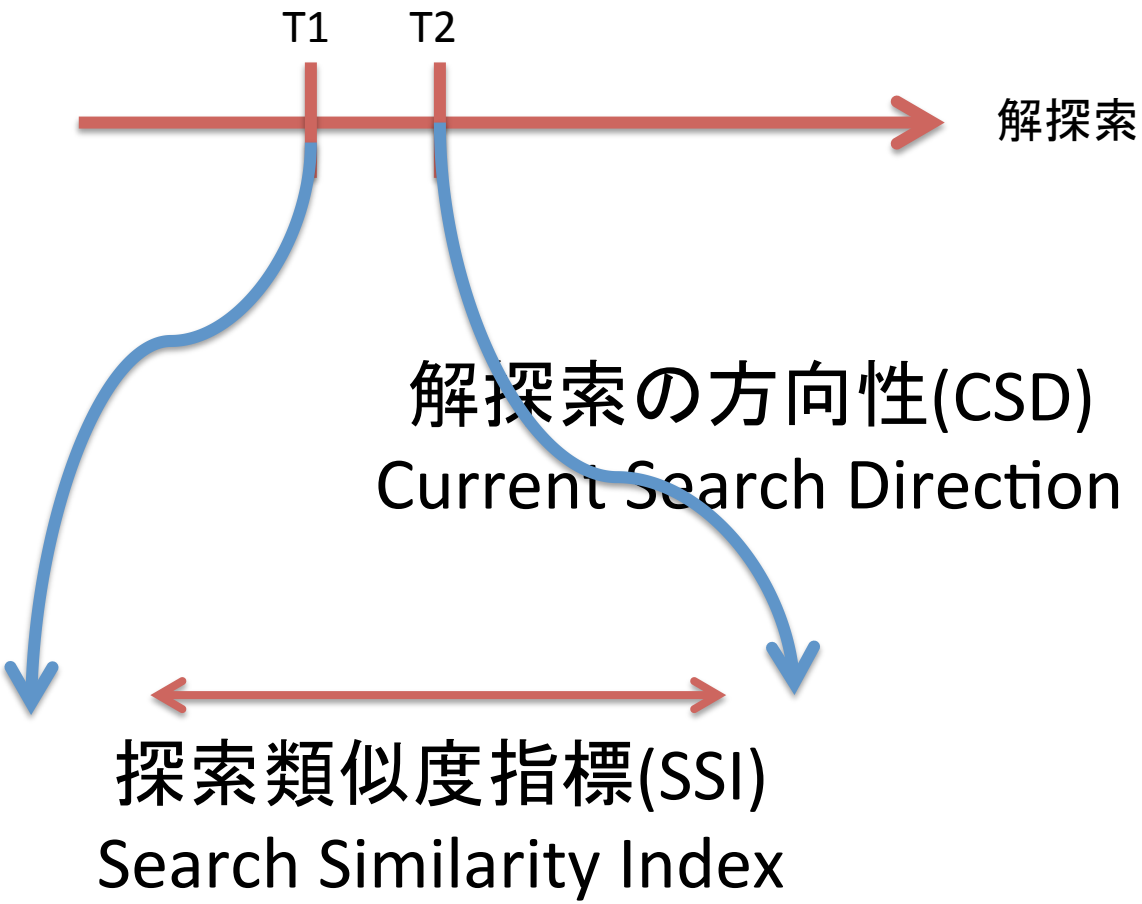
集中・分散のバランス

KEY IDEA

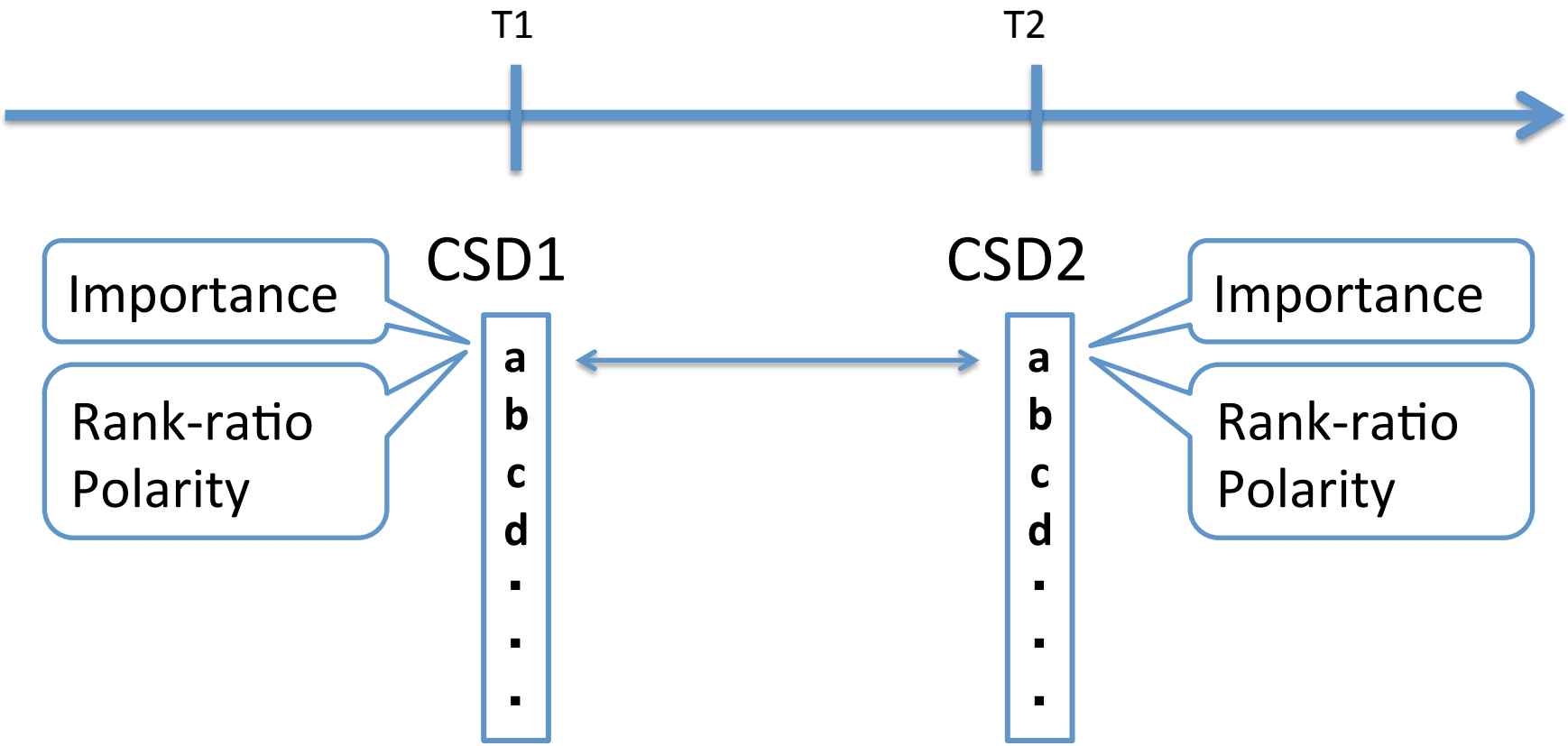


ヒューリスティック → 定量評価

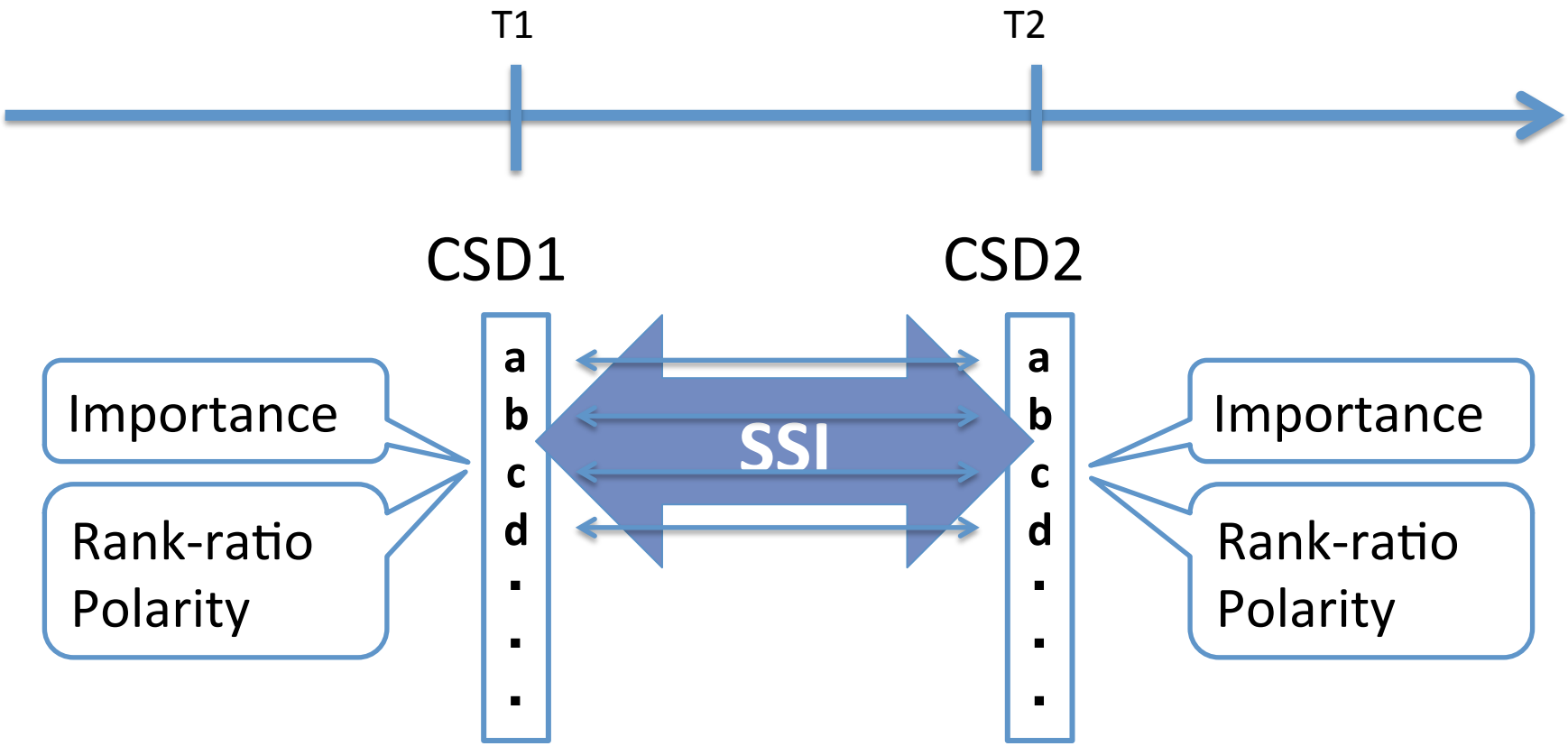
KEY IDEA


T1
T2
解探索
解探索の方向性(CSD)
Current Search Direction
探索類似度指標(SSI)
Search Similarity Index

SSIの基本的な考え方



SSIの基本的な考え方



SSIの定式化

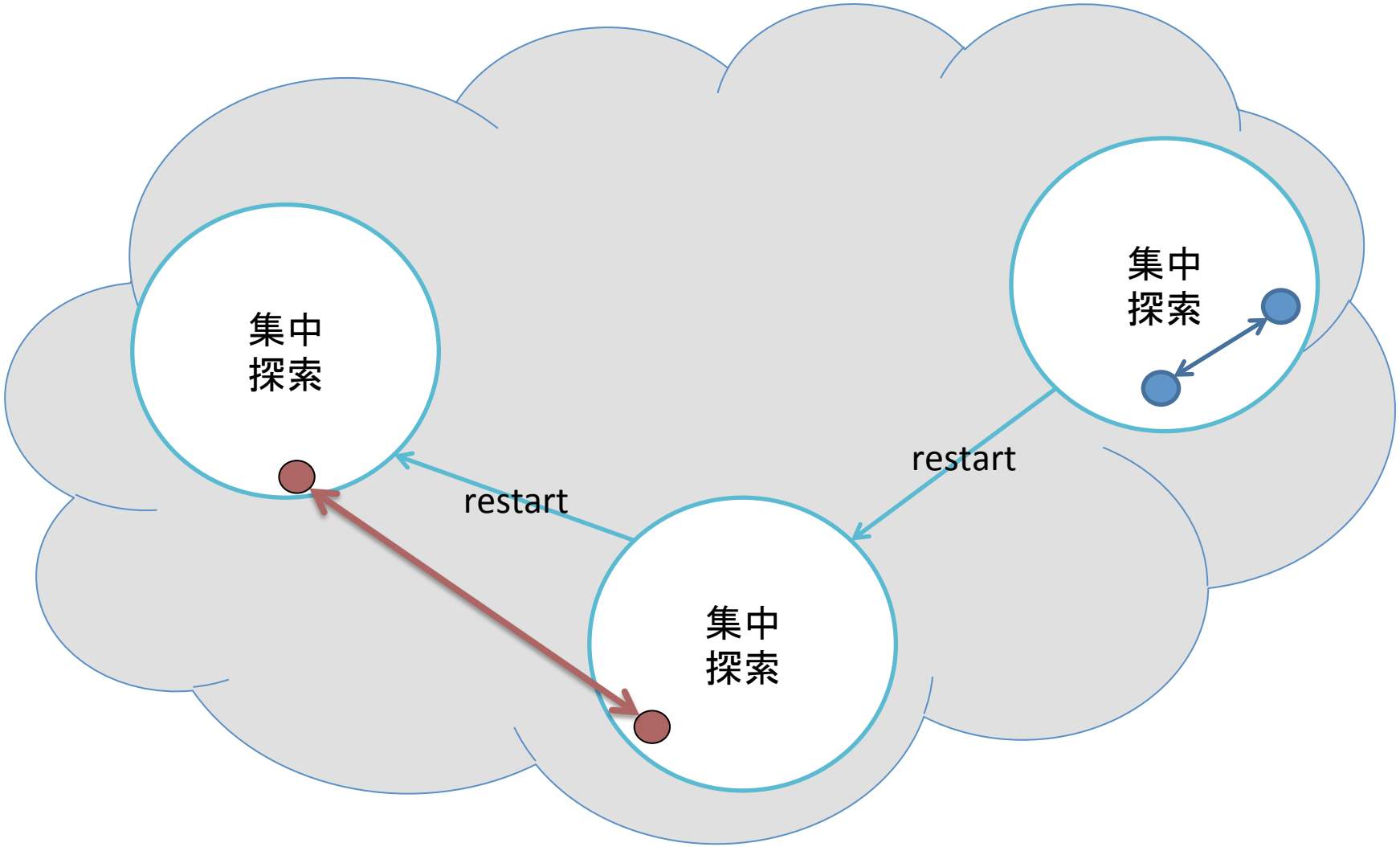
①変数の**優先度**が似ていれば、解探索も似ている
②変数の**真偽値**が似ていれば、解探索も似ている

Similarity = Rank-ratio類似度 × Polarity類似度

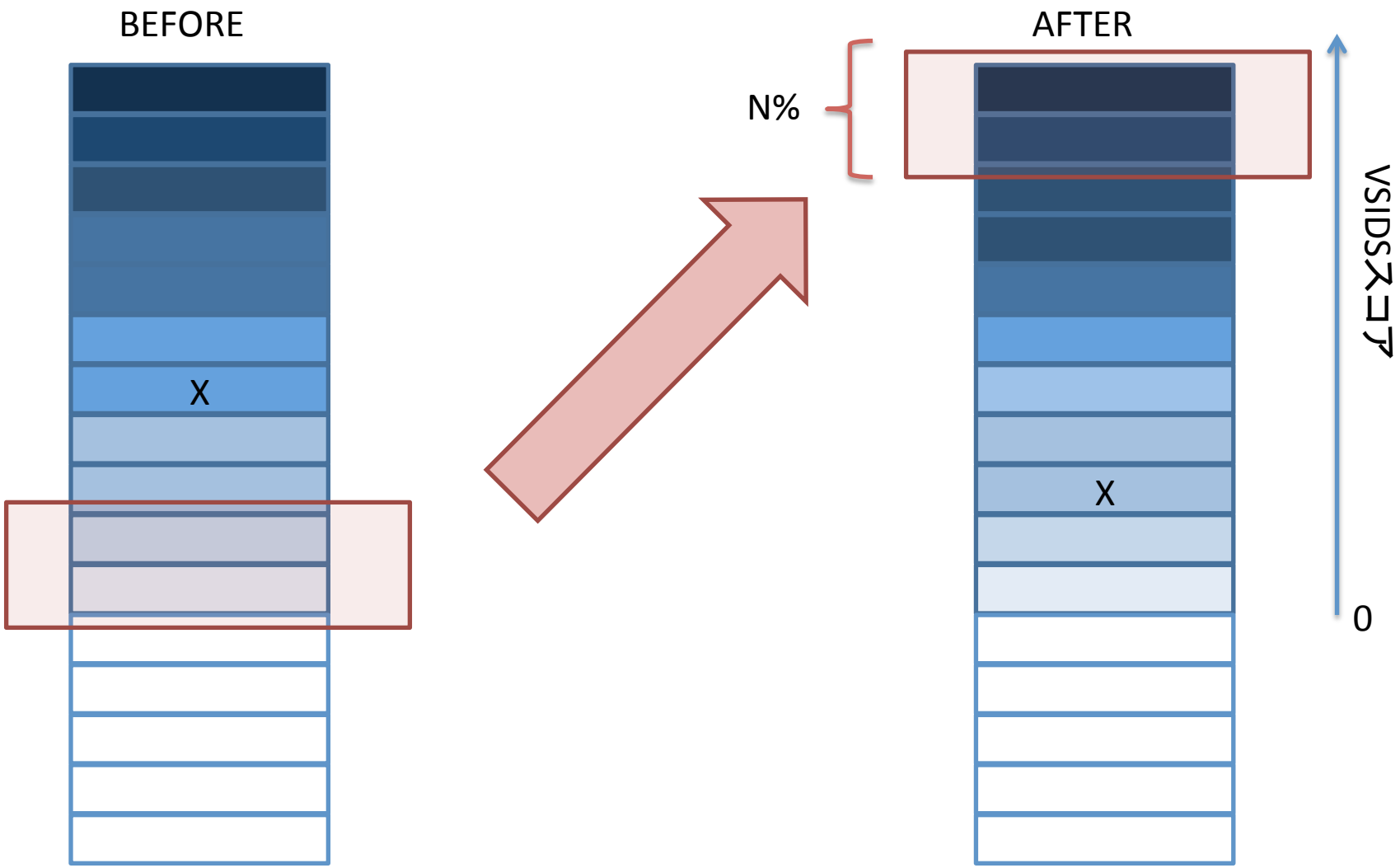
①重要な変数ほど、影響力が大きい
②重要でない変数は、影響力が小さい

SSI = 標準化（Σ Importance × Similarity）

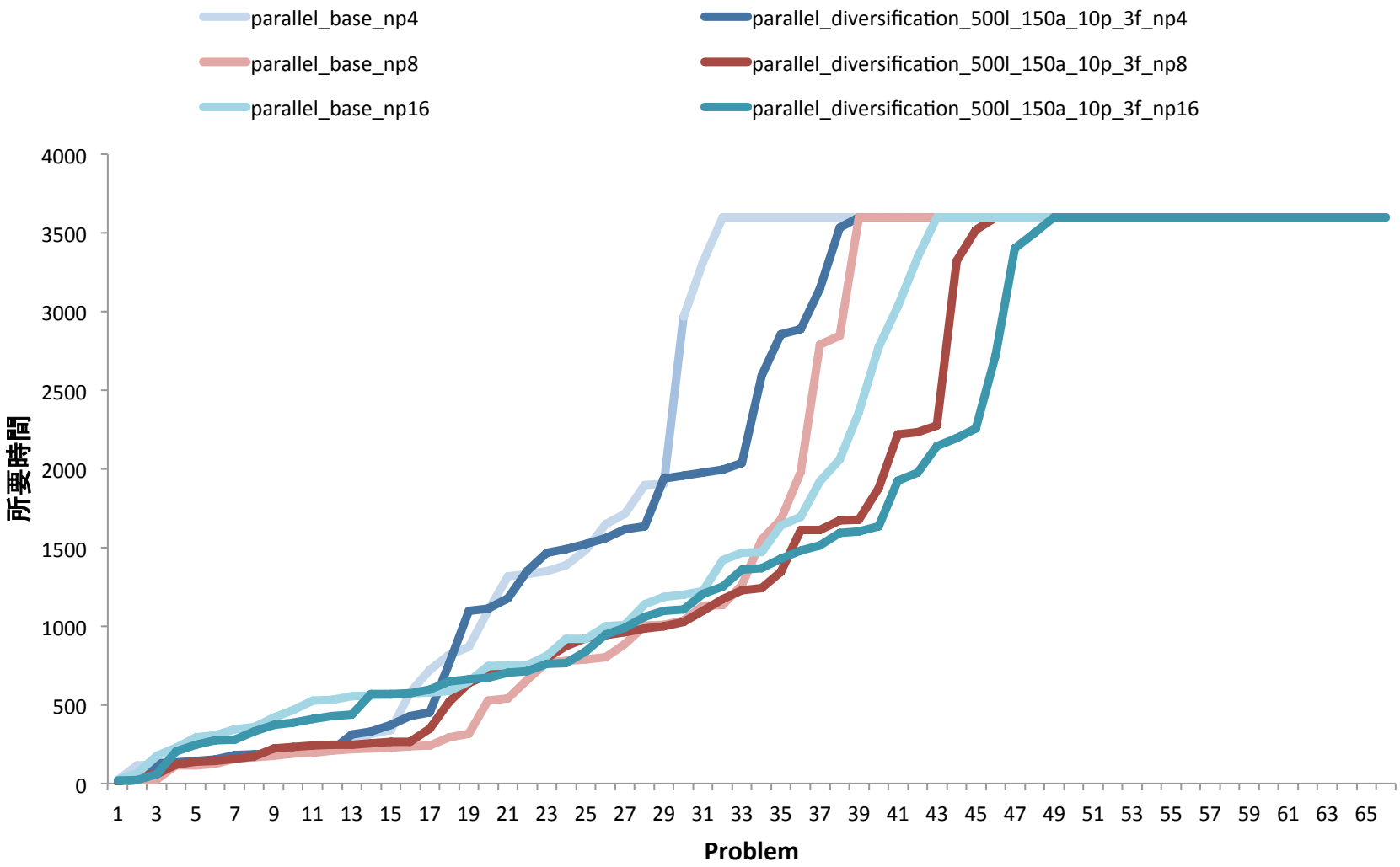
Restartによる解探索の変化



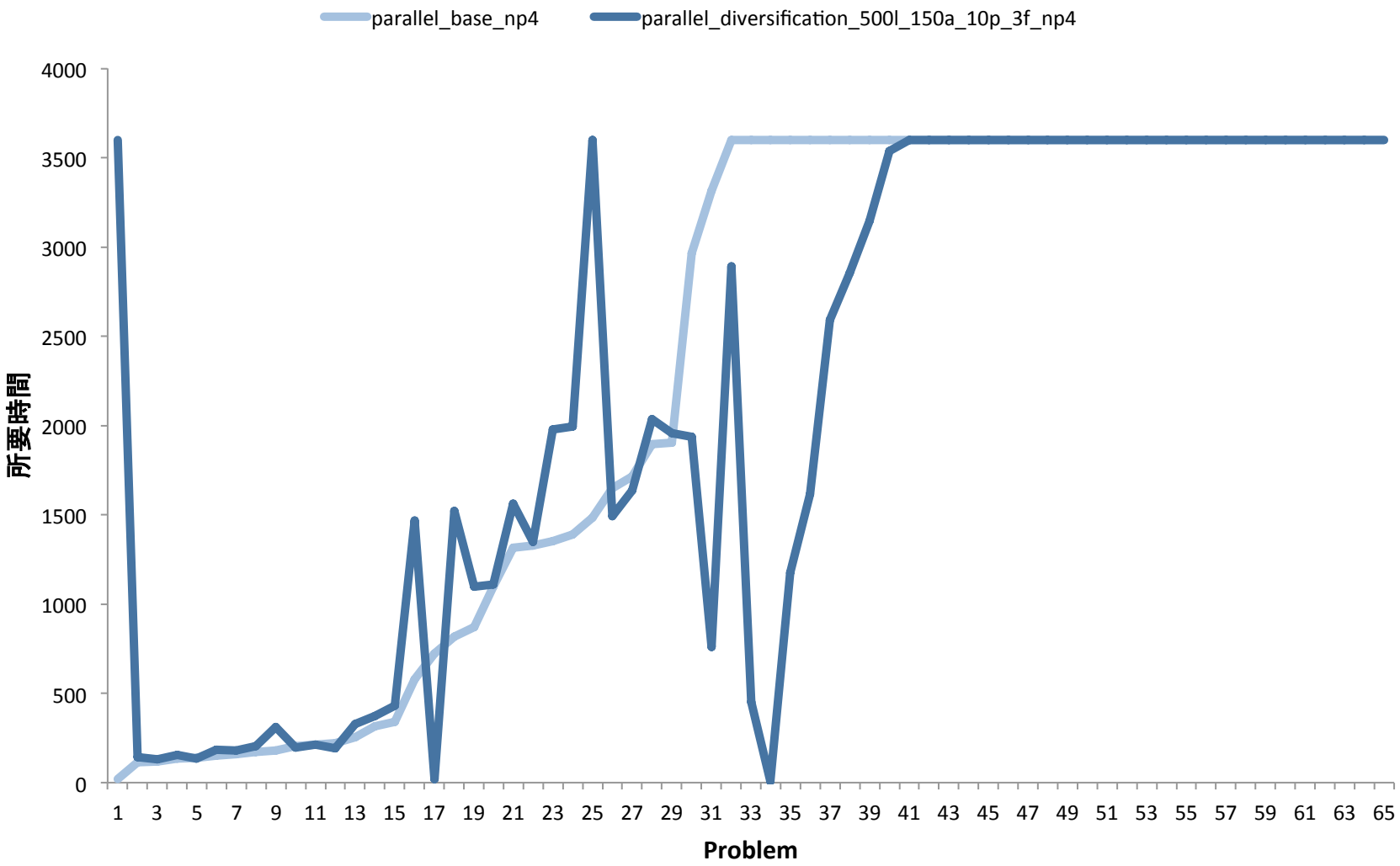
解探索の変更手法



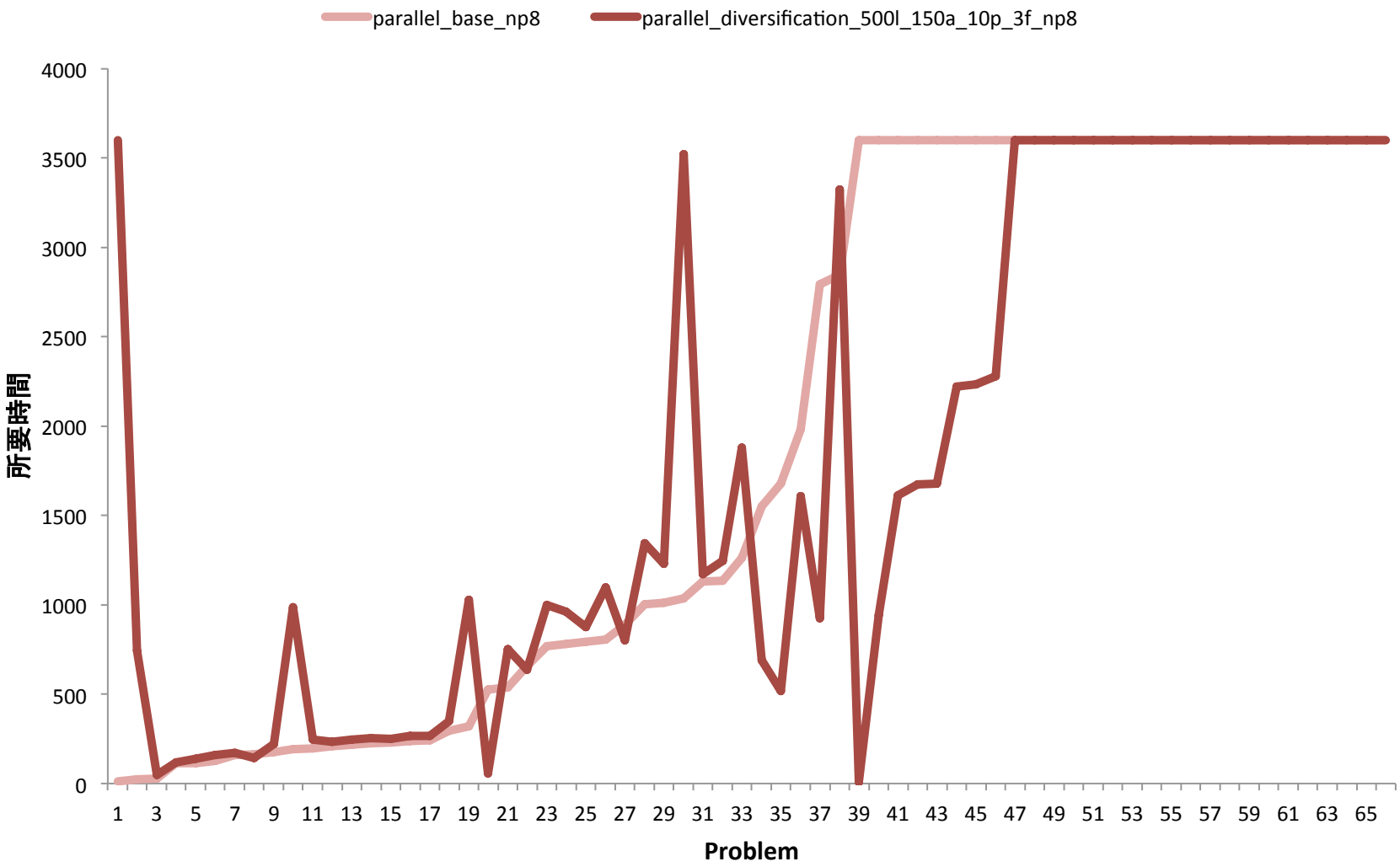
SSI Parallel Diversification



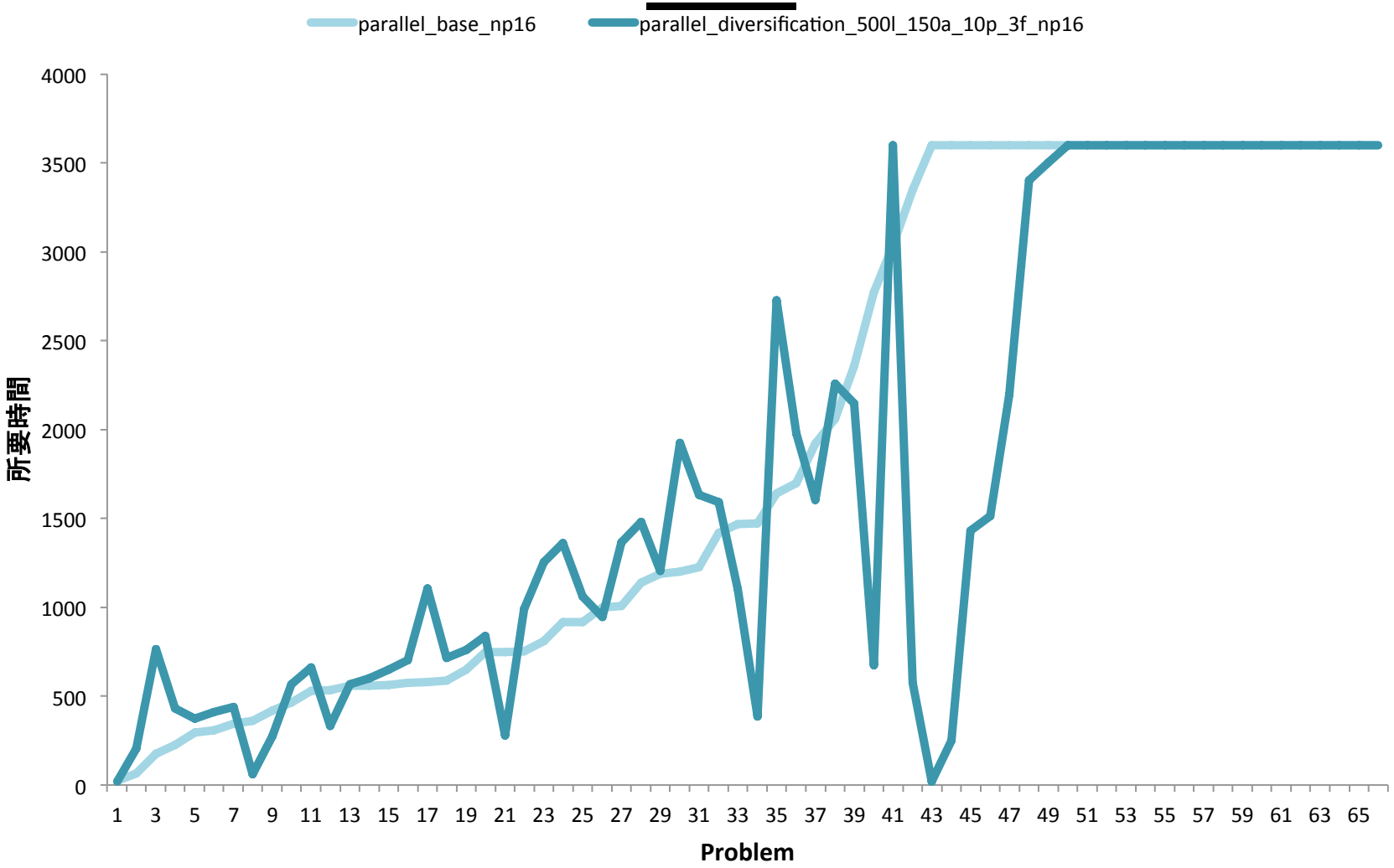
SSI Parallel Diversification（並列数4）



SSI Parallel Diversification（並列数8）



SSI Parallel Diversification（並列数16）



第十一回　課題

レポート・成績について

- ほぼ毎回、プログラミング課題を出題する予定
  - 効率の良い計算機実験のためのツールを使ってみる
  - アルゴリズムの実装
  - ライブラリの利用・・・　など

- ３回以上、レポートのファイルとプログラムのソースを添付し、メールで提出のこと
  - E-mail: algorithm2014@edu.jar.jp
  - サブジェクト 「アルゴリズムとプログラム実践講座・レポート」
  - 学生証番号と名前は、 メールの本文にも書いてください。
  - プログラムやレポートは（見本として）公開することがあります。適宜、作者名や　コピーライトをいれておいてください。公開不可の場合は、プログラムの冒頭にその旨、コメントをいれておいてください。
  - 質問・作問提案も歓迎 （作問については採用の場合は別途加点）
  - サンプルプログラムは「初心者向け」です。 **上級者は無視してください。**

推奨環境など

- Linux, Mac,（Windows+Cygwin）
- 仮想マシン環境（VMware, VirtualBox, Parallels)
  - 余裕があれば、いろいろな組み合わせを試して比較してみると面白いと思います

- 言語
  - 自由。ただし、一般的でない言語については、上記いずれかのOS 上にインストール可能なもの

SAT solver を使ってみる

SAT solver (たとえば MiniSAT) を
インストールし、NP 完全の問題
あるいは、それに類する組み合せ判定
問題を作成し解かせてみよう。