Local Search Techniques

Md Solimul Chowdhury
mdsolimc@andrew.cmu.edu

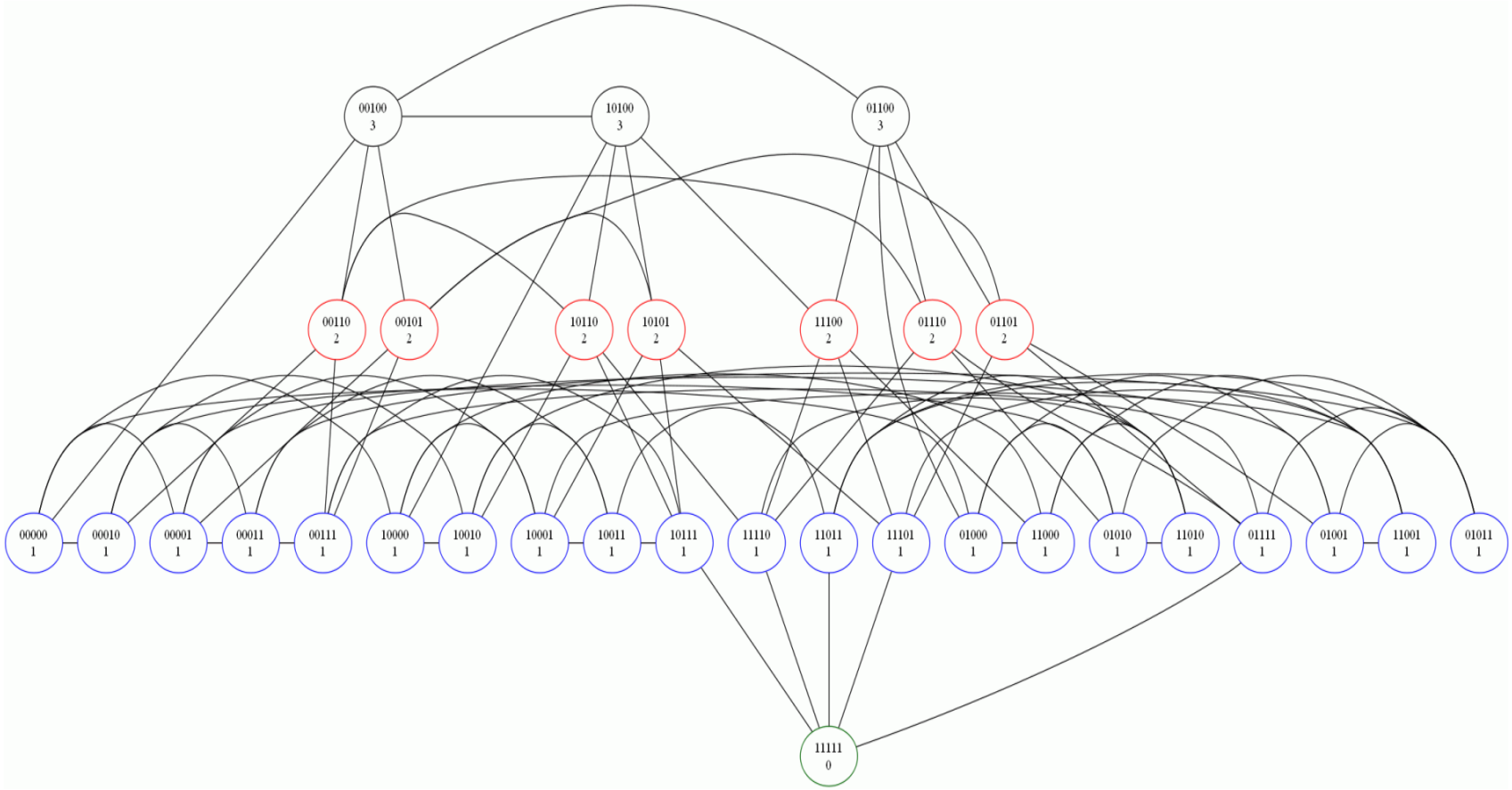
Carnegie Mellon University

http://www.cs.cmu.edu/~mheule/15816-f23/

Automated Reasoning and Satisfiability
September 25, 2023

Introduction[1]

- Locals Search: for optimization and decision problems
- Explores the solution space by visiting neighbors



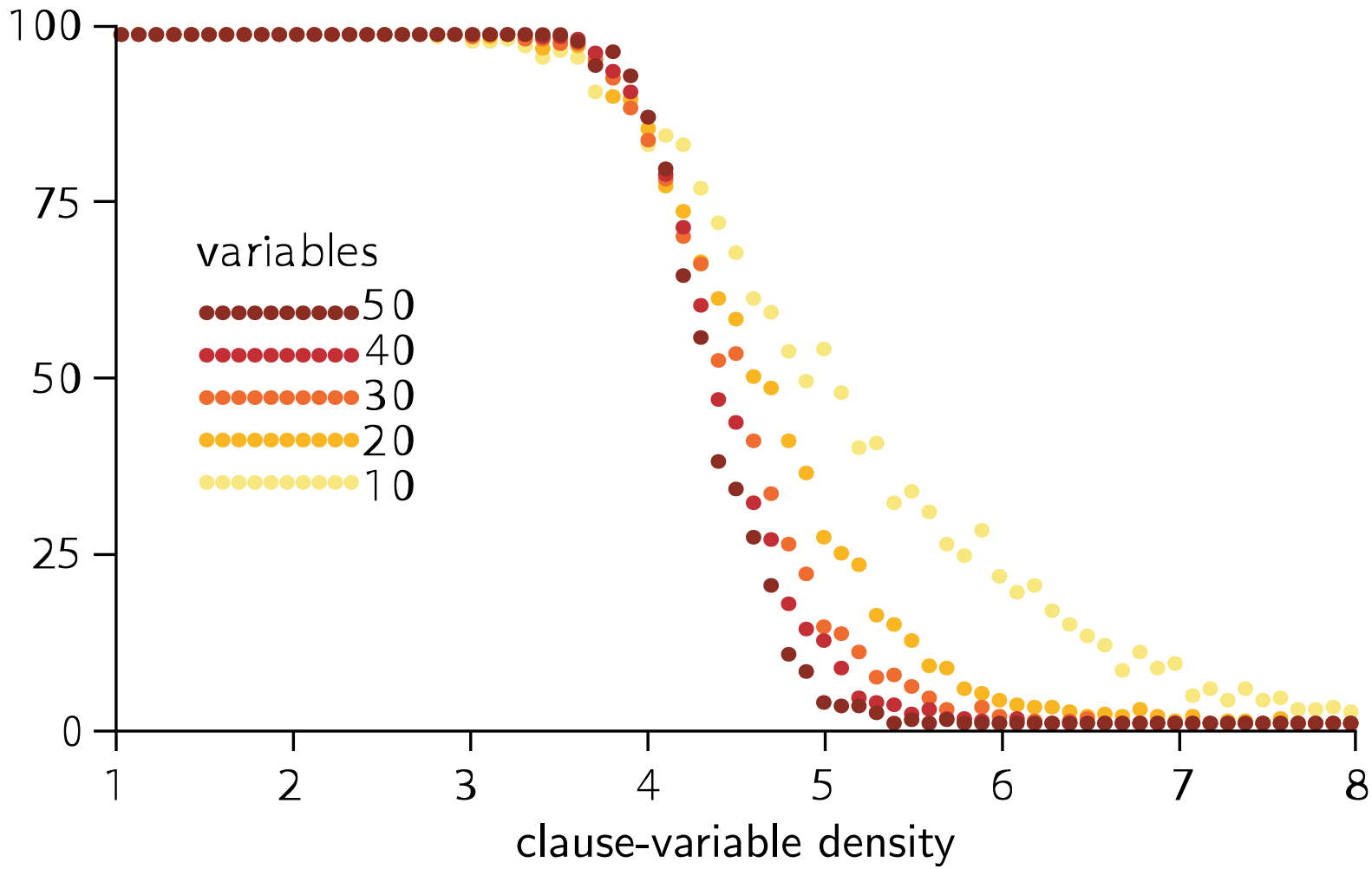
[1] Picture Source: Tompkins, D.: Dynamic Local Search for SAT: Design, Insights and Analysis. Ph.D. thesis (2010)

Random k-SAT

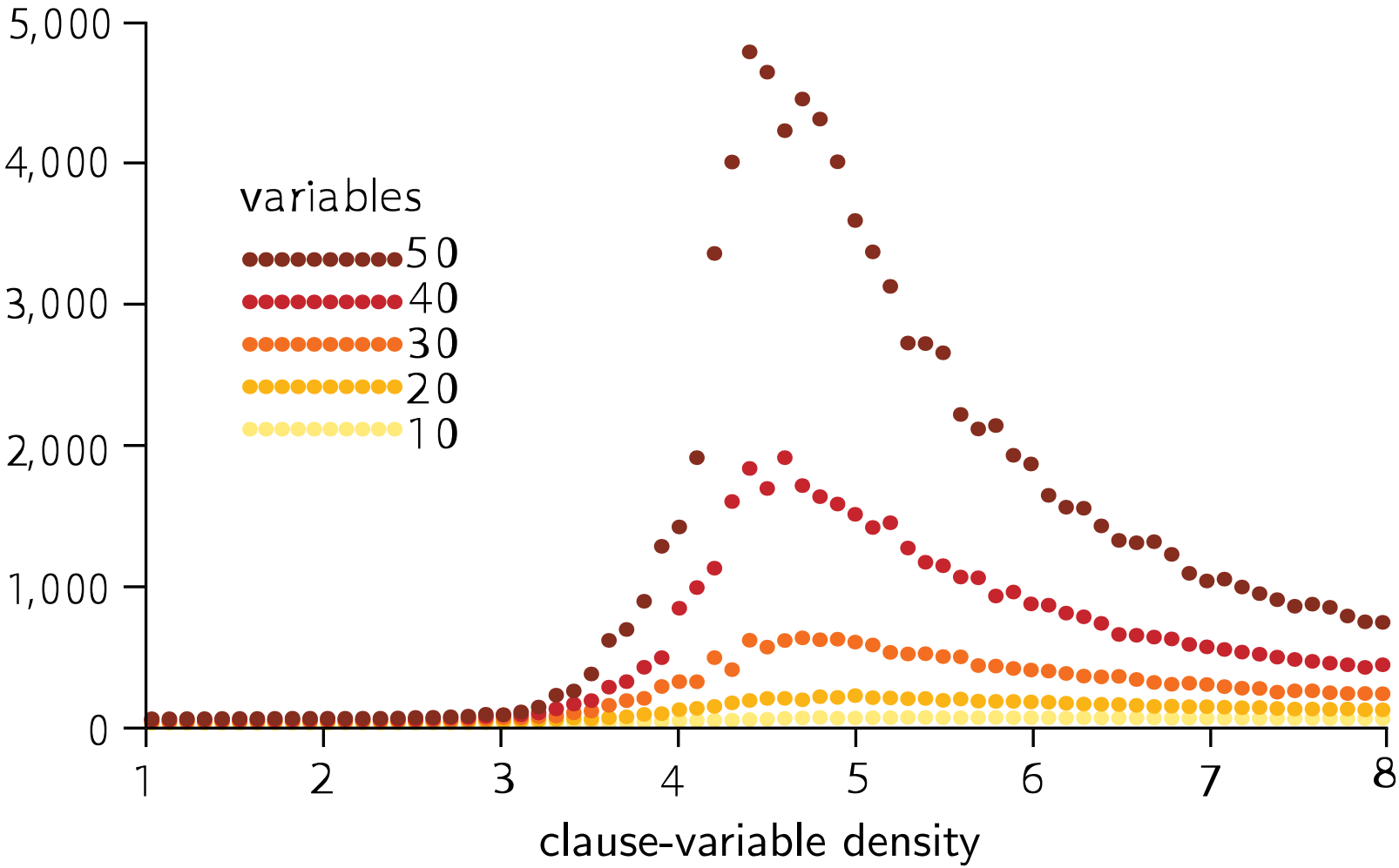
Local search solvers are particularly effective (and optimized) on hard uniform random (satisfiable) k-SAT problems

- All clauses have length k
- Variables have the same probability to occur
- Each literal is negated with probability of 50%
- Density is ratio Clauses to Variables

Random 3-SAT: % satisfiable, the phase transition



Random 3-SAT: exponential runtime, the threshold



Local Search: Generic structure

Generic structure of local search SAT solvers

```
1:  for i in 1 to MAX_TRIES do
2:      α := random initial assignment
3:      for j in 1 to MAX_STEPS do
4:          if α satisfies F then
5:              return satisfiable
6:          end if
7:          α := Flip (α)
8:      end for
9:  end for
10: return unknown
```

Local Search: Global vs Local flips

Global flips

- Pro: Big improvements
- Neg: Probabilistic incomplete

Local flips

- Neg: Small improvements
- Pos: Probabilistic complete

Local Search: Types of Flips

Select a random unsatisfied clause C

- Free flip
- Random flip
- Heuristic flip

Local Search: WalkSAT Code [Selman, Kautz, and Cohen '93]

FLIP_WALKSAT (α)

1: $C :=$ random falsified clause by $\alpha \circ \mathcal{F}$
2: **if** a variable in C can be flipped for free **then**
3: flip in α that variable
4: **else**
5: flip in α with p a random $x_i \in C$
6: flip in α with $1 - p$ the "optimal" $x_i \in C$
7: **end if**
8: **return** α

Local Search: ProbSAT [Balint and Schöning '12]

ProbSAT generalizes the WalkSAT code.

Let $\mathrm{break}(x, \alpha)$ denote the number of clauses that are only satisfied by x or $\overline{x}$ under the assignment α

- $C :=$ random falsified clause by $\alpha \circ \mathcal{F}$
- randomly pick a variable x in C using weights $c^{-\mathrm{break}(x,\alpha)}$
- an effective constant for random 3-SAT: $c = 2.5$
- update α by flipping x

Introduction

Random k-SAT

Stochastic Local Search

WalkSAT and ProbSAT

Weight Transfer in Local Search

Local Search: Weight Transfer

- All clauses have a weight
- Only do global flips
- Pick the variable that reduces the falsified weight the most
- If there is no weight-reducing variable, modify the weights

Local Search: Weight Transfer Pseudo-code

Generic structure of local search SAT solvers

```
1:  for i in 1 to MAX_TRIES do
2:      α := random initial assignment
3:      for j in 1 to MAX_STEPS do
4:          if α satisfies F then
5:              return satisfiable
6:          else if there exists a weight-reducing variable then
7:              flip the most weight-reducing variable in α
8:          else
9:              increase the weight of clauses falsified by α
10:         end if
11:     end for
12: end for
13: return unknown
```

Weight Transferring Algorithms [Ishtaiwi '04; Hutter '02]

1. Pure Additive Weighting Scheme (PAWS)
2. Scaling and Probabilistic Smoothing (SAPS)

```
6:      else if there exists a weight-reducing variable then
7:          flip the most weight-reducing variable in α
8:      else
9:          increase the weight of clauses falsified by α
```

PAWS: Additive Increase | SAPS: Multiplicative Increase

Numerical overflow $\longrightarrow$ periodic/probabilistic scaling

Weight Transfer Algorithms [Ishtaiwi, Sattar, and Pham '05]

3. Divide and Distribute Fixed Weights (DDFW)
   - Transfers a fixed integer amount from $C_{\texttt{sat}}$ to $C_{\texttt{false}}$ if,

$$|C_{\text{sat}} \cap C_{\text{false}}| \geq 1 \rightarrow \text{Neighbors}(C_{\texttt{sat}}, C_{\texttt{false}})$$

   - Cumulative weights of the clauses remains constant
   - No need to re-scale!

The DDFW Algorithm

Finds solution for $\mathcal{F}$ by `minimizing` falsified clause weight

1. Each clause is initialized with a fixed weight

$$\mathrm{Weight}[\mathrm{C}] \leftarrow \mathtt{w_{init}} (= 8) \text{ for } \mathrm{C} \in \mathcal{F}$$

2. Iteratively, flips variable

   `flip` (v), if a flip with v reduces falsified weight the most

The DDFW Algorithm

3. If no weight-reducing variable is available,

   (a) If Neighbors, moves fixed weight to C_{false} from C_{sat}

   With $\alpha_1 > \alpha_2$,
   (i) $\text{Weight}[C_{\text{sat}}] > \texttt{w}_{\texttt{init}} \rightarrow$ move α_1 from C_{sat} to C_{false}
   (ii) $\text{Weight}[C_{\text{sat}}] = \texttt{w}_{\texttt{init}} \rightarrow$ move α_2 from C_{sat} to C_{false}

   (b) With 1% probability, selects C_{sat} randomly

Linear Weight Transfer [Chowdhury, Codel, and Huele '23]

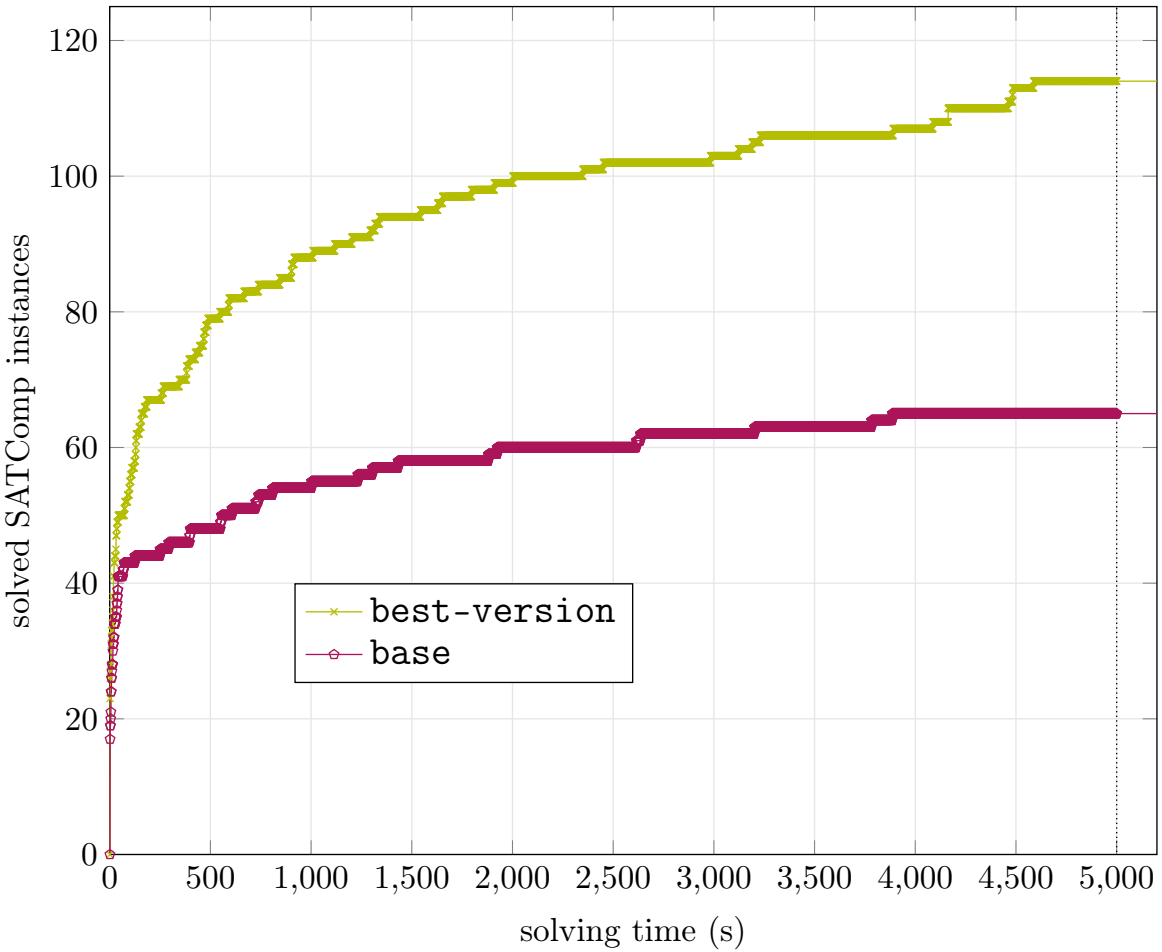
- Non-fixed amount of weight transfer

$$\alpha \leftarrow \mathrm{Weight}[\mathrm{C}_{\mathtt{sat}}] * \mathsf{a} + \mathsf{c} \qquad (0 \leq \mathsf{a} \leq 1,\ \mathsf{c} > 0)$$

  - Transfers α to $\mathrm{C}_{\mathtt{false}}$ from $\mathrm{C}_{\mathtt{sat}}$

- Two randomized flip-heuristics
- Adjustment of a performance critical parameter

Linear Weight Transfer for DDFW[2]



- Next?
  - Non-linear Weight Transfer?
  - Explore alternative definitions of neighbors?
  - Predicting parameters with ML?

[2] https://github.com/solimul/yal-lin