

# Multi-level Stochastic Local Search for SAT

---

Camilo Rostoker and Chris Dabrowski  
{rostokec,kdabrows}@cs.ubc.ca  
Department of Computer Science  
University of British Columbia



# Outline

---

1. Introduction to SAT
2. Systematic vs. Stochastic Local Search
3. Hybrid & Multi-level Methods
4. Our Proposed Algorithm
5. Testing & Evaluation
6. Future Work and Improvements
7. Conclusion



# Satisfiability (SAT)

---

- SAT problems are a fundamental problem in combinatorial optimization
  
- Applications in planning, scheduling, circuit verification, economics, etc
  
- Two approaches to solving problems
  - Systematic (Complete)
  - Stochastic Local Search (Incomplete)

# Systemic SAT Methods

---

- Systematic, exhaustive methods guaranteed to find a solution in bounded time (if a solution exists)
- Most state-of-the-art complete methods based on David-Putnam-Logemann-Loveland (DPLL) procedure
- Good for smaller SAT problems, mission-critical applications, or when time/resources is not an issue
- Proven to be extremely effective for structured instances



# Stochastic Local Search Methods

---

- Incomplete, approximate search methods
  - Probabilistically Approximately Complete (PAC)
- Trades guarantee of finding optimal solution for efficiency and scalability
- General approach: Explore search space by iteratively perturbing a candidate solution using a given heuristic
- Proven to be extremely effective for large, random, non-structured instances



# Hybrid Methods

---

- Combine systematic with SLS to get the best of both worlds
- Benefit from systematic handling of structured instances while retaining scalability of SLS methods
- Problem: How to determine how much relative amount of time/effort on systematic and SLS

# Multi-level Methods

---

- Perform search on various “levels”, where a level corresponds to assigning a subset of variables
- When the systematic search reaches a given level, let underlying SLS procedure take control
- Same Problem: Do we spend more time performing the systematic search or more time focusing in on a specific sub-region (SLS)?
- Existing multi-level methods try different combinations of systematic & SLS searches
  - Constrained Local Search (CLS): Randomized backtracking
  - WalkSatz: Satz, then WalkSAT on equivalence graph
  - Multi-level Cooperative Search: agents search different regions

# MLSLS for SAT Overview

---

- A hybrid approach to SAT that iteratively performs systematic search guided by Satz look-ahead heuristic followed by a Novelty+ SLS procedure
- The amount of work to be done in the systematic stage is referred to as the “level”
  - search tree context → we “drill down” to a given depth in the search tree
  - search space context → we “freeze” a subset of the variables
- The purpose of the systematic procedure is to exploit the structure and variable dependencies, as well as reduce search space

# High-level Pseudo-code

---

```
input: problem instance  $\pi \in \Pi$ 
output: solution  $\hat{s} \in S(\pi)$  or  $\emptyset$ 
while termination criteria not satisfied do
   $level \leftarrow$  choose number of variables to assign
   $\pi_r \leftarrow \pi$ 
   $currentLevel \leftarrow 0$ 
  while  $currentLevel < level$  do
     $VS \leftarrow$  calculate and order variable set using  $PROP_z$  heuristic
     $v_{best} \leftarrow$  best variable from  $VS$ 
    assignTruthValue( $v_{best}$ )
     $\pi_r \leftarrow$  unitPropagation( $\pi_r$ )
     $currentLevel \leftarrow currentLevel + 1$ 
  end while
   $s_c \leftarrow$  construct( $\pi_r$ )
   $s_l \leftarrow$  localSearch( $\pi_r, s_c$ )
  if  $f(s_l) < f(\hat{s})$  then
     $\hat{s} \leftarrow s_l$ 
  end if
end while
if  $\hat{s} \in S$  then
  return  $\hat{s}$ 
else
  return  $\emptyset$ 
end if
```

# Key Points

---

- Variable subset selection and ordering is done by a Satz look-ahead procedure, a variation of MOM's heuristic
- The level can be dynamically adjusted every iteration
- After the systematic procedure has set *level* variables, use construction heuristic to get initial candidate solution
- Novelty+ then takes over and performs up to *maxRunSteps* steps
- Novelty+ procedure is essentially the same except “frozen” variables must be excluded

# Variable Set Selection and Ordering

---

- PROPz heuristic selects a set of variables
  - based on idea of MOM's heuristic - Maximum Occurrence of Minimum Size
  - Branch on variables that have greatest chance of finding failed literals as early as possible in the search tree
  
- Satz uses a look-ahead technique to order the variables obtained by PROPz
  - Look-ahead: test if assigning a variable produces empty clauses
  - Choose variables that balance the sub-trees obtained after branching

# Variable Ordering

---

---

**Algorithm 2** Variable Ordering

---

```
for each free variable  $x$  such that  $PROP_z(x)$  is true do
  let  $F'$  and  $F''$  be two copies of  $F$ 
   $F' := \text{UnitPropagation}(F' \cup \{x\})$ 
   $F'' := \text{UnitPropagation}(F'' \cup \{\bar{x}\})$ 
  if both  $F'$  and  $F''$  contain an empty clause then
    do backtracking procedure
  end if
  if  $F'$  contains an empty clause then
     $x := 0$ 
     $F := F''$ 
  end if
  if  $F''$  contains an empty clause then
     $x := 1$ 
     $F := F'$ ;
  end if
  if neither  $F'$  nor  $F''$  contains an empty clause then
     $w(x) := \text{diff}(F', F)$ 
     $w(\bar{x}) := \text{diff}(F'', F)$ 
     $H(x) := w(\bar{x}) * w(x) * 1024 + w(\bar{x}) + w(x)$ 
  end if
end for
Choose variable  $x$  such that  $H(x)$  is greatest
```

---

# Level Selection Mechanism

---

- Total number of variables:  $n$
- Number of variables to assign:  $a$
- Total number of variables assigned:  $t$

$$X = t / n$$

$$Y = t / a$$

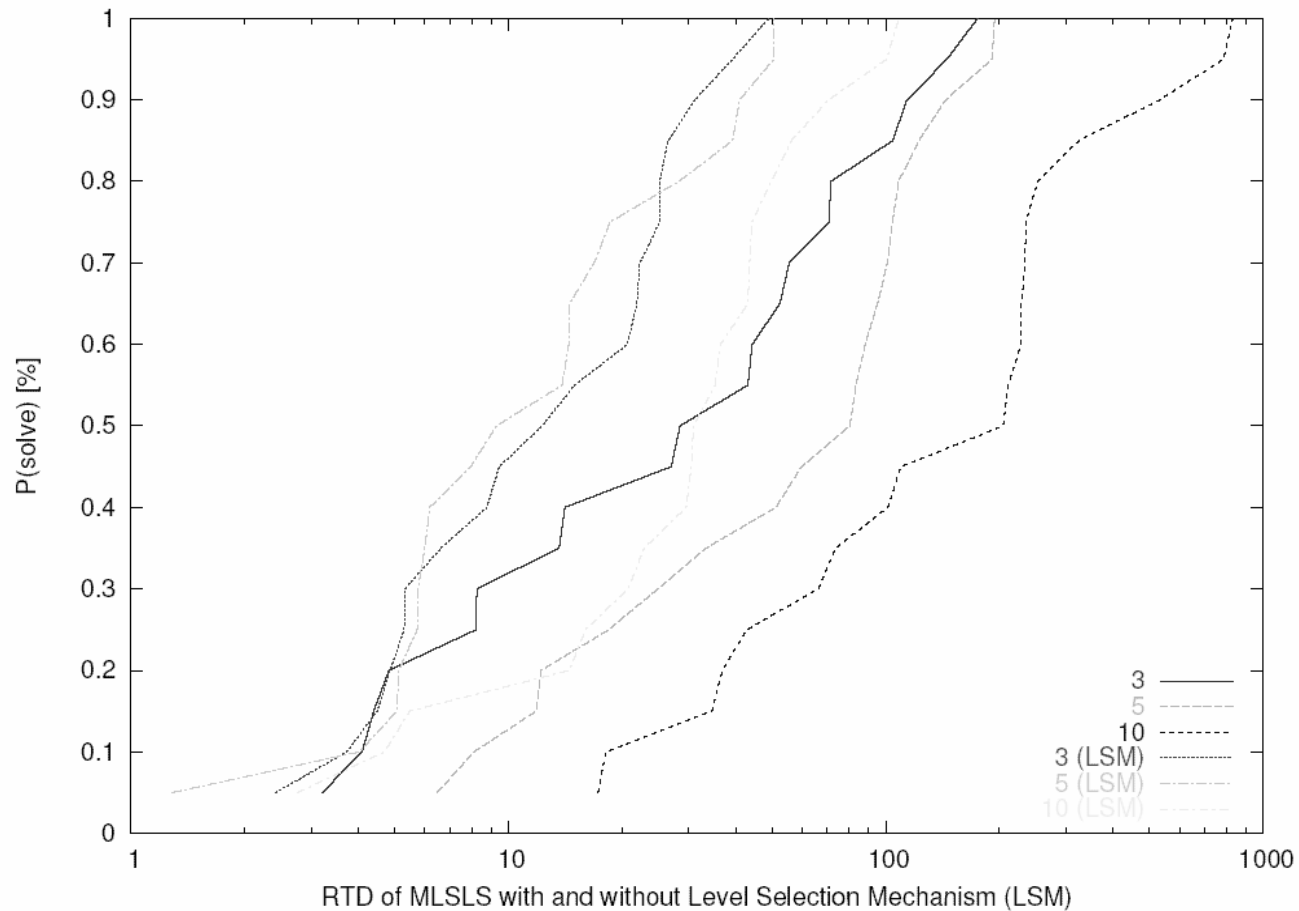
```
if ( X < 0.05 && y > 10 )
```

```
    a = a + 1
```

```
else
```

```
    a = a - 1
```

# Effects of LSM in MLSLS



# Backtracking vs. Backjumping

---

- Backtracking – Flip the most recently set variable.
- Backjumping – Flip the most recently set variable that is causing an empty clause
- MLSLS uses backtracking
  - backjumping is more complex
  - Satz look-ahead claims to choose variables in such a way that backjumping essentially becomes systematic backtracking

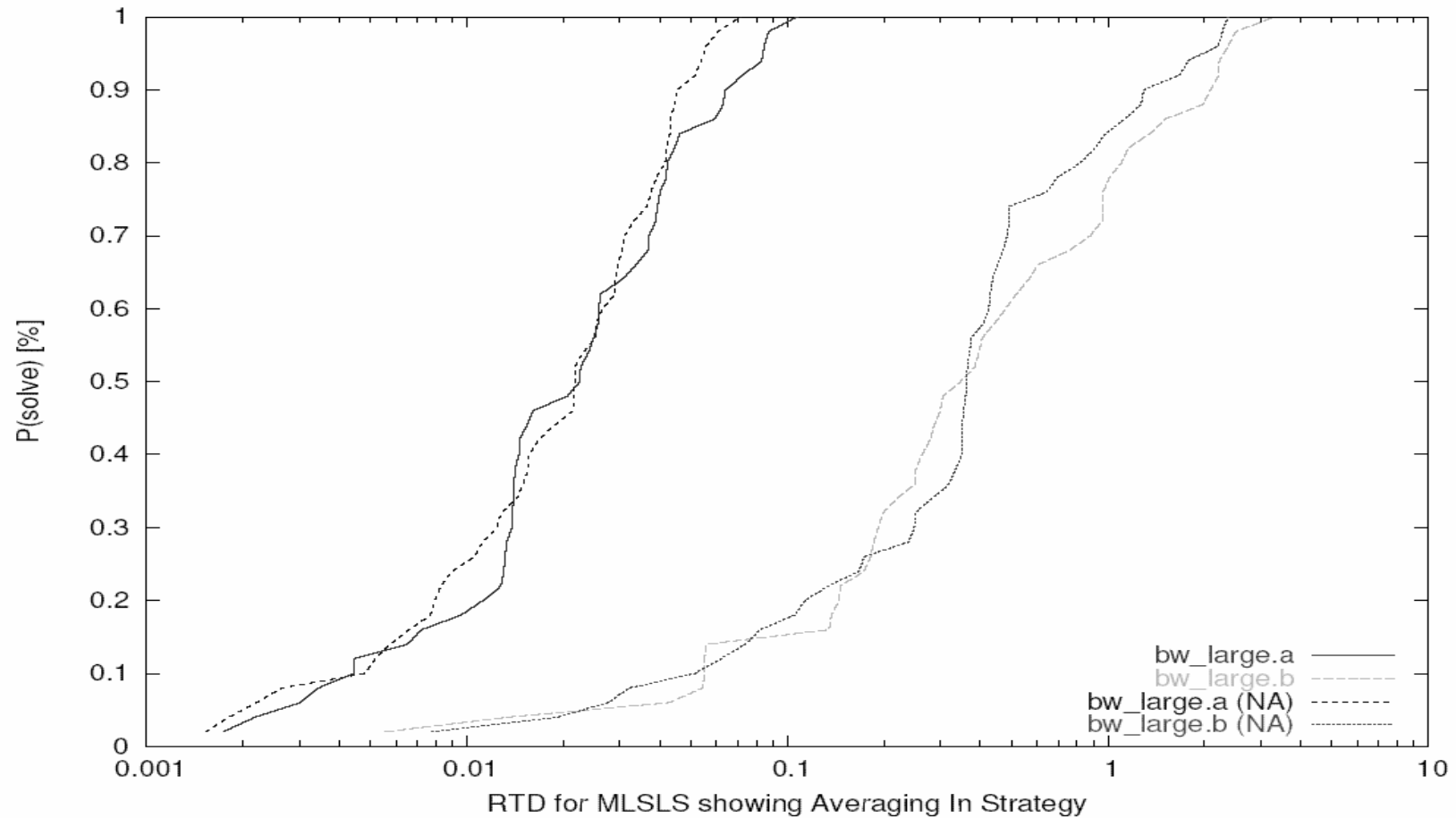
# Construction Heuristics

---

- Random Construction
  - For each unassigned variable, randomly set to 0 or 1
  
- “Averaging In” Heuristic
  - For each unassigned variable, do bitwise average of corresponding variable from past 2 incumbent solutions

$T_1^{init} \leftarrow$  assign all unset variables using random construction  
Do a round of systematic with Satz look-ahead starting with  $T_1^{init}$   
 $T_2^{init} \leftarrow$  bitwise average of  $T_1^{init}$  and  $T_1^{best}$ , excluding frozen variables  
Do a round of systematic with Satz look-ahead starting with  $T_2^{init}$   
**while** (termination criteria not satisfied) **do**  
     $T_i^{init} \leftarrow$  bitwise average of  $T_{i-1}^{best}$  and  $T_{i-2}^{best}$ , excluding frozen variables  
    Do a round of systematic with Satz look-ahead starting with  $T_i^{init}$   
**end while**

# Effects of Averaging In Strategy



# Novelty+ in MLSLS

---

```
if RandomProbability(randwalk) then
  C ← choose a false clause uniformly at random
  flipCandidate ← choose a variable in C uniformly at random
else
  C ← choose a false clause uniformly at random
  bestVar ← variable in C that minimizes the number of unsatisfied clauses if flipped
  secondBestVar ← next best variable in C after bestVar
  youngestVar ← variable in C that was most recently flipped
  if bestVar ≠ youngestVar then
    flipCandidate ← bestVar
  else
    if RandomProbability(noise) then
      flipCandidate ← secondBestVar
    else
      flipCandidate ← bestVar
    end if
  end if
end if
end if
```

# Novelty+ and Frozen Variables

---

| instance(s) | level | # of frozen picks | # of steps  | % of frozen picks |
|-------------|-------|-------------------|-------------|-------------------|
| uf250       | 2     | 678595            | 5828325630  | 0.01164           |
| uf250       | 5     | 695503            | 6049246803  | 0.01149           |
| uf100       | 2     | 1951242           | 10368703436 | 0.01882           |
| uf100       | 5     | 2072399           | 10032938566 | 0.02065           |
| logistics.d | 10    | 5                 | 340342      | 0.00146           |
| logistics.d | 20    | 64                | 1723943     | 0.00371           |
| bw_large.b  | 4     | 68                | 505577      | 0.01344           |

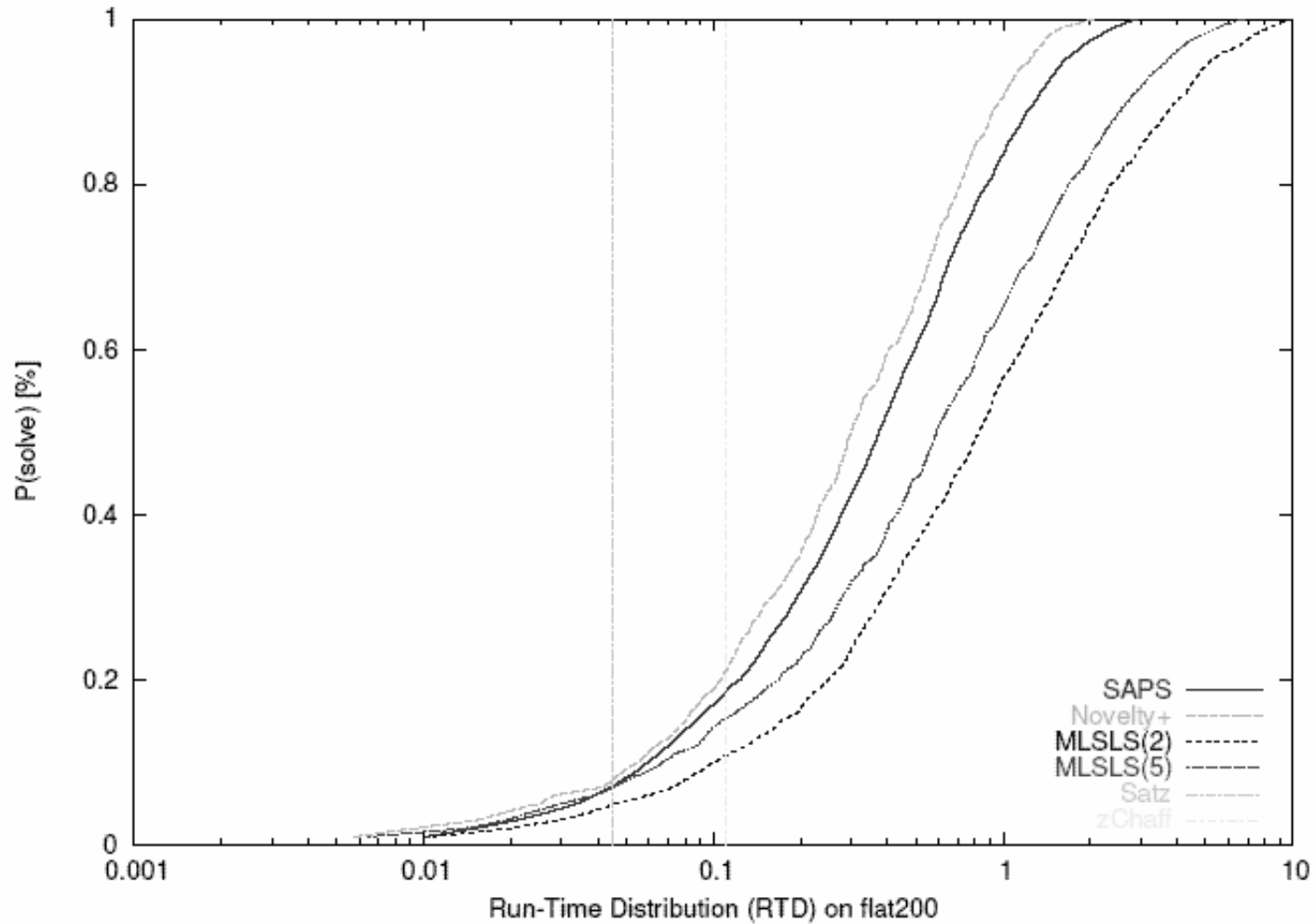
- Frozen variables can only be chosen during random walk
- We used default value of walk probability  $randwalk=0.01$
- This means that a random walk is performed 1% of the time, and approximately 1% of those cases a frozen variable was picked.

# Testing & Evaluation

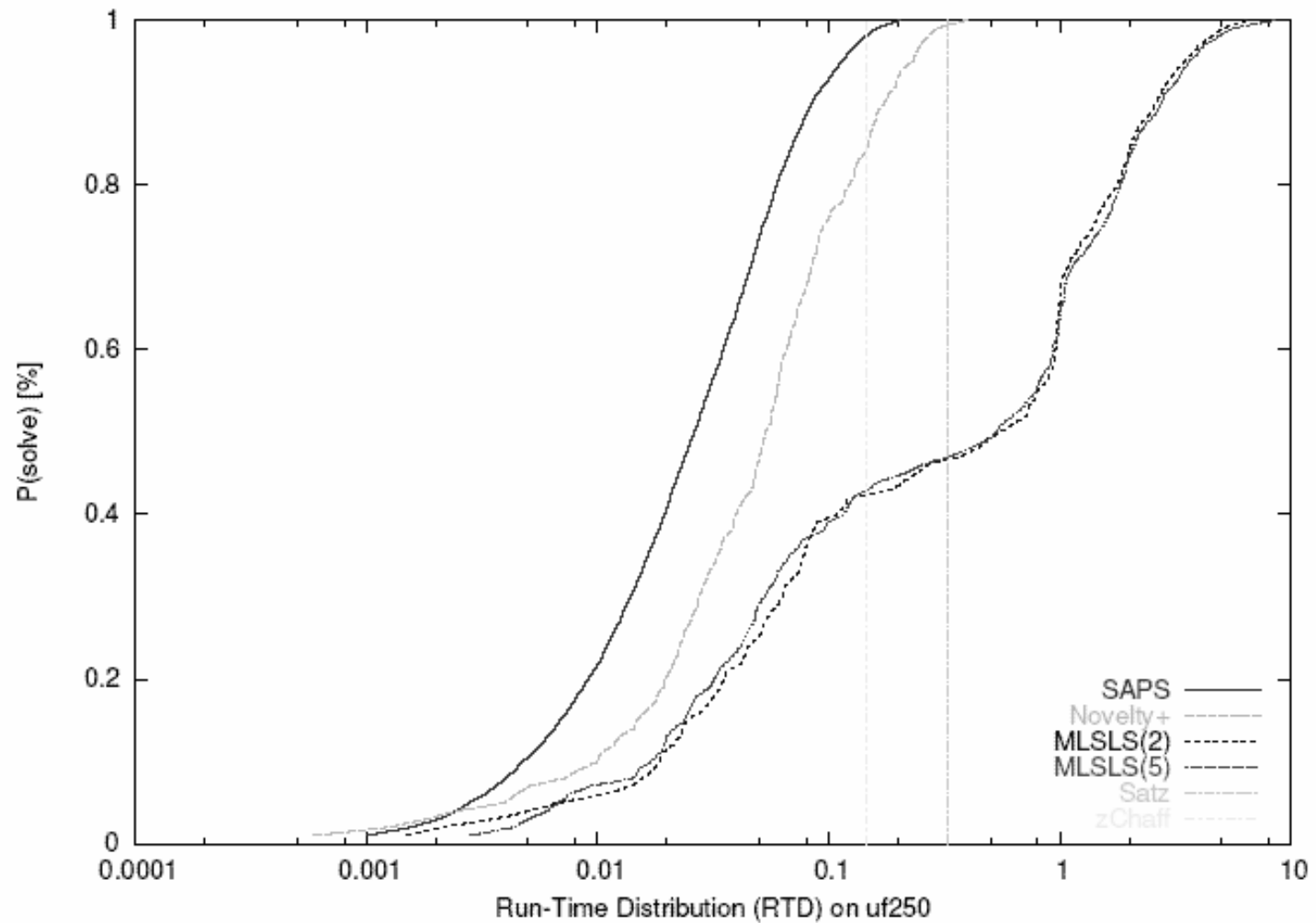
---

- Tested on both random and structured instances.
- Problem domains include blocks world planning, logistics, Velev's microprocessor, uniform random, and flat graph coloring.
- Compared against satz, zChaff, saps, and novelty+.

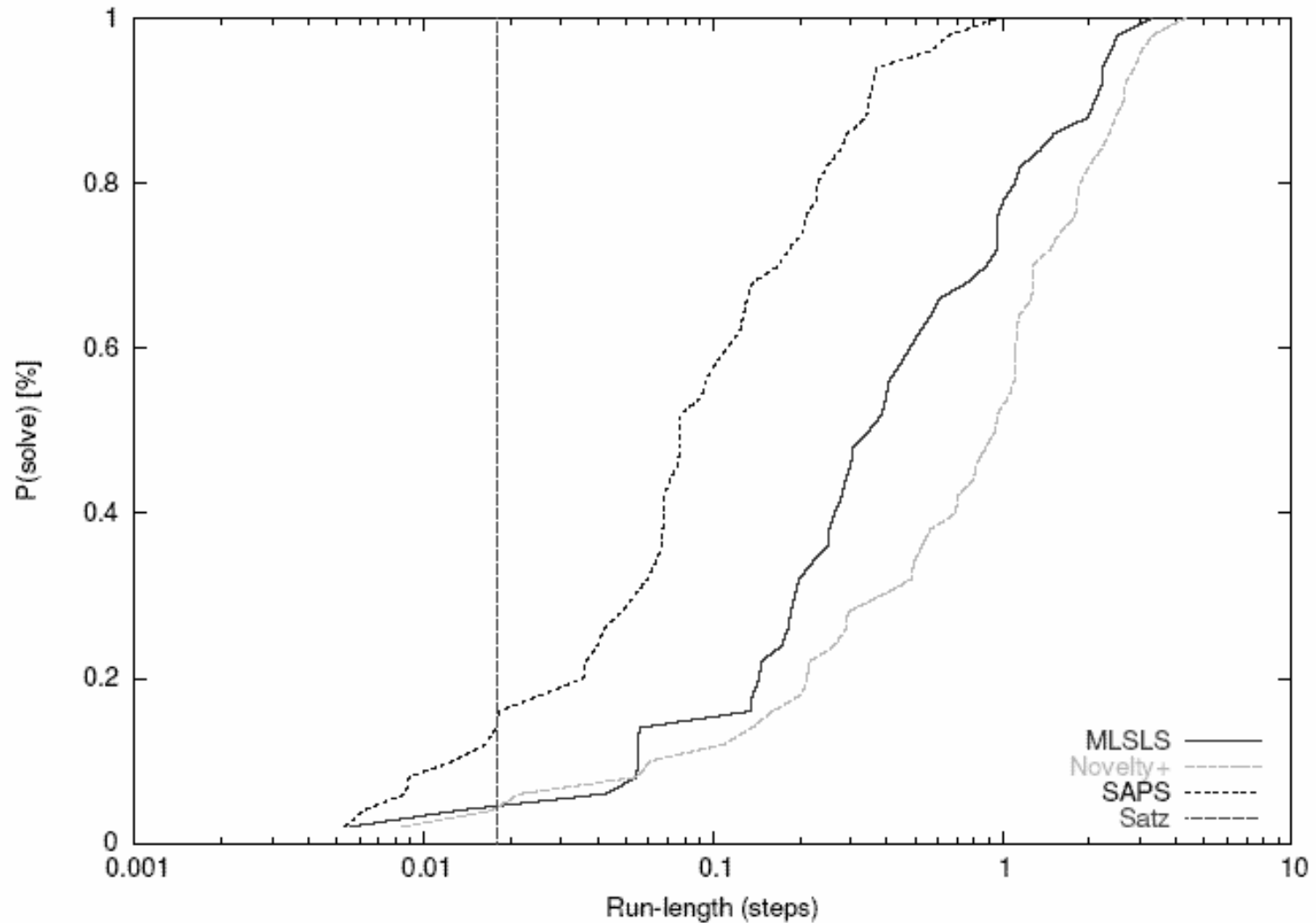
# Results – flat200



# Results – uf250



# Results – bw\_large.b



# Future Work

---

- Improving the SLS component
  - Detailed analysis of frozen variables that the SLS picks
  - Data structure to hold non-frozen variables  
→ like a candidate list
  - Use a different underlying SLS procedure



# Future Work

---

- Improving the systematic component:
  - Efficient Unit Propagation algorithm
  - Detailed empirical analysis of Level Selection Mechanism
  - Parameter estimation techniques for the PROPz heuristic
  - Backtracking vs. backjumping



# Future Work

---

- Improving the overall design
  - Parameter optimization
  - Algorithm profiling
  - Runtime balance between systematic and SLS components

# Conclusion

---

- Original goal: develop an algorithm that would outperform SLS methods on structured instances, while being competitive on random instances (similarly, vice-versa)
- Based on empirical results obtained thus far, we have not achieved this goal
- Future work discussed previously outlines many potential areas of improvement
- With a more efficient systematic procedure, we believe our hypothesis may still be valid

# There is still hope...

---

