# AN EFFICIENT GLOBAL-SEARCH STRATEGY IN DISCRETE LAGRANGIAN METHODS FOR SOLVING HARD SATISFIABILITY PROBLEMS

**Benjamin W. Wah and Zhe Wu**

**Department of Electrical and Computer Engineering**
**and the Coordinated Science Laboratory**
**University of Illinois, Urbana-Champaign**
**Urbana, IL 61801, USA**
**E-mail: b-wah@uiuc.edu**
**URL: http://manip.crhc.uiuc.edu**

**August 2000**

# Outline

- Introduction – What is SAT?
- Previous work – Survey of existing methods
- Problem formulation
- Theory of discrete constrained optimization using Lagrange multipliers
- Our previous work in solving SAT
- Applying a new global search strategy in DLM
- Conclusions

# Introduction – What is SAT?

- SAT is a Boolean formula in conjunctive normal form
  - example:
    $(x_1 \vee x_3 \vee x_4) \wedge (x_1 \vee \bar{x_2} \vee \bar{x_3}) \wedge (\bar{x_1} \vee \bar{x_2} \vee x_4) \wedge (\bar{x_1} \vee \bar{x_3} \vee \bar{x_4})$
  - solution: one truth assignment, $(1, 0, 0, 0)$
- Some challenges of SAT
  - local/global search has difficulty in solving some satisfiable DIMACS SAT benchmarks
    - hanoi5,
    - par32-1, par32-2, $\cdots$, par32-5

# Existing Methods

- Complete approaches using resolution and backtracking
  - too expensive and not suitable for large problems
  - some typical methods: Davis Putnam, Grasp
- Incomplete approaches based on local/global searches
  - perturbing a search trajectory until solution is found
  - restarting from a new starting point when the current search is stuck
  - efficient but cannot prove unsatisfiability
  - some typical methods:
    * GSAT/WalkSAT, break-out strategy, guided local search

# Problem Formulation

- Formulate SAT as a discrete, constrained NLP
- Each clause of a SAT problem is considered as a constraint

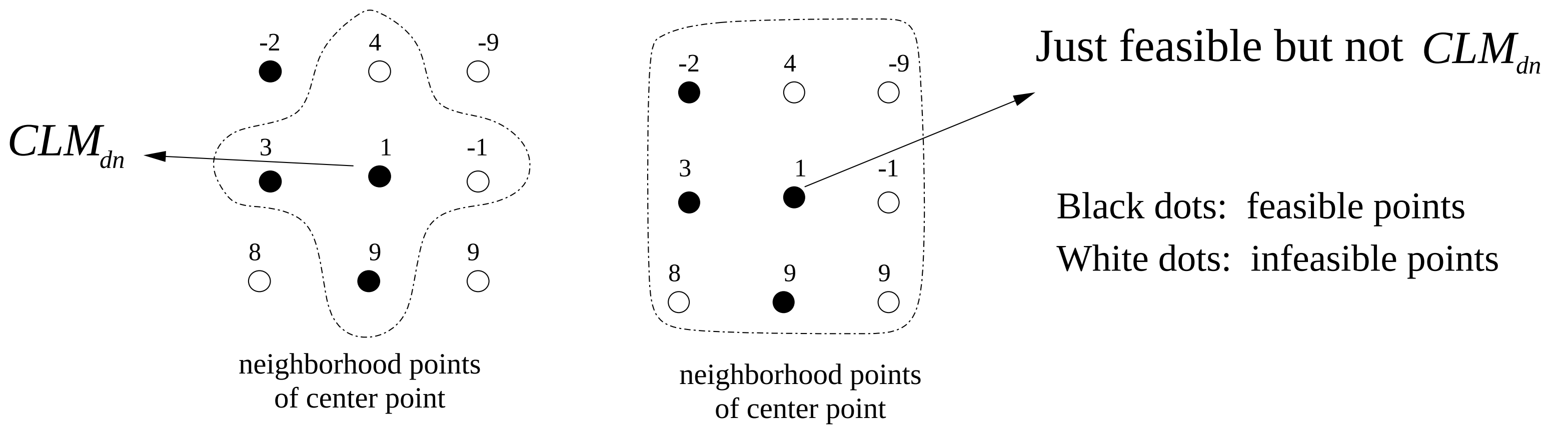
$$\begin{array}{ll} \min_{x \in \{0,1\}^m} & N(x) \\ \text{subject to} & U_i(x) = 0 \quad \forall i \in \{1, 2, \ldots, n\}, \end{array}$$

$$N(x) = \begin{cases} 0 & \\ \text{weighted sum of unsatisfied clauses} & \text{(heuristically} \\ \text{other functions} & \text{chosen objective)} \end{cases}$$

$$U_i(x) = \begin{cases} 0 & \text{when the } i^{th} \text{ clause is satisfied} \\ 1 & \text{otherwise} \end{cases}$$

# Basic Concepts

- $\mathcal{N}_{dn}(x)$: *discrete neighborhood* of $x$
  - a finite set of user-defined points $\{x' \in X\}$ such that $x'$ is reachable from $x$ in one step and that $x' \in \mathcal{N}_{dn}(x) \Longleftrightarrow x \in \mathcal{N}_{dn}(x')$
  - 'reachability' of states must be guaranteed
- $CLM_{dn}$: *constrained local minima in discrete neighborhood*
  - feasible, local minimum with respect to neighboring feasible points



# Discrete Constrained Optimization using Lagrange Multipliers

- Lagrangian formulation of discrete constrained optimization

$$\begin{array}{ll} \text{minimize} & f(x) \\ \text{subject to} & h(x) = 0 \end{array} \quad \Longrightarrow \quad \begin{array}{l} L_d(x,\lambda) = f(x) + \lambda H(h(x)) \\ \text{where } H(y) = 0 \Leftrightarrow \\ y = 0 \text{ and } H(y) \geq 0 \end{array}$$

  – $H$: non-negative transformation function

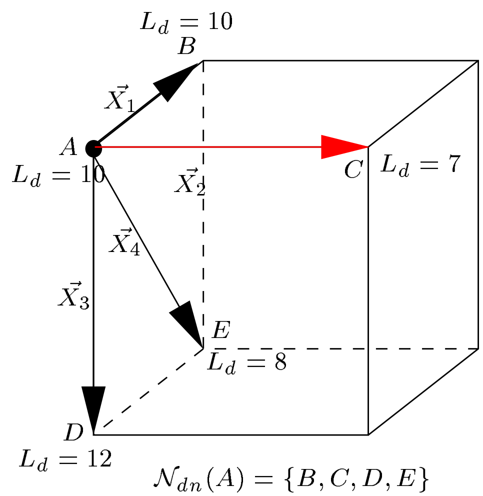
    * examples: absolute function, square function, etc.

- Goal of Solving discrete constrained optimization problems:

  - find $CLM_{dn}$ in discrete space

# Additional Concepts

- $SP_{dn}$: discrete-space saddle point
  - $(x^*, \lambda^*)$ is a $SP_{dn}$ iff $L_d(x^*, \lambda) \le L_d(x^*, \lambda^*) \le L_d(x, \lambda^*)$ for all $\lambda \in \mathcal{R}$ and all $x \in \mathcal{N}_{dn}(x^*)$
- Direction of maximum potential drop (*DMPD*)
  - vector direction in Lagrangian space along which there is maximum drop in Lagrangian value



# Major Concepts in Discrete Optimization

- First-order necessary and sufficient conditions
  - point $x^*$ is a $CLM_{dn}$ if an only if

    there exists $\lambda^*$ such that $(x^*, \lambda^*)$ is a $SP_{dn}$

    or

    $(x^*, \lambda^*)$ satisfies the first-order necessary and sufficient conditions:
    a) $\underset{x \in dn(x^*)}{\Delta} L_d(x, \lambda^*) = 0$ and b) $h(x) = 0$
- General property of $CLM_{dn}$

Solutions satisfying the discrete-space first-order conditions $= CLM_{dn} = SP_{dn}$

# Search Procedures in Discrete Constrained Optimization

- How to locate $CLM_{dn}$?
  - locating $CLM_{dn}$ explicitly is difficult because it is hard to satisfy multiple constraints simultaneously
  - locating $SP_{dn}$ is equivalent to locating $CLM_{dn}$ and is implicit to satisfying multiple constraints simultaneously

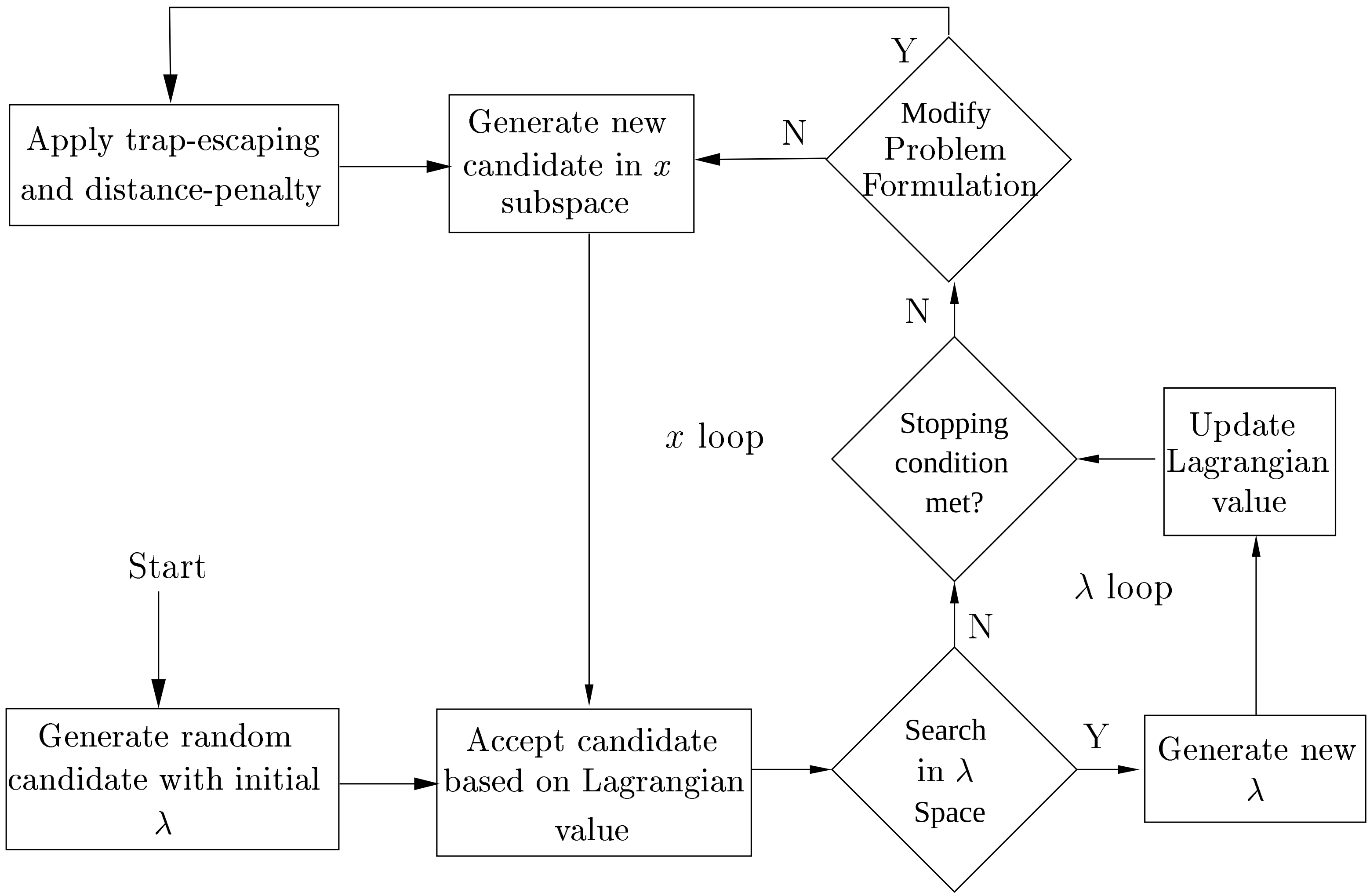
- First-order search method to locate $SP_{dn}$
  - *Descents in $x$ space:*

$$x(k+1) = x(k) \oplus \underset{x \in dn(x(k))}{\Delta} L_d(x(k), \lambda(k))$$

  - *Ascents in $\lambda$ space:*

$$\lambda(k+1) = \lambda(k) + c \times h(x(k))$$

# Discrete-Space First-Order Search Framework



# Our Previous Work: DLM-98-BASIC-SAT

1. Reduce original SAT problem;<br>
2. Generate a random starting point using a fixed seed;<br>
3. Initialize $\lambda_i \longleftarrow 0$;<br>
4. **while** solution not found and time not used up **do**<br>
5.     Pick $x_j \notin$ TabuList that reduces $L_d$ the most; /* $x$ loop */<br>
6.     Maintain TabuList;<br>
7.     Flip $x_j$;<br>
8.     **if** $\#_{UpHillMoves} + \#_{FlatMoves} > \theta_1$ **then** /* $\lambda$ loop */<br>
9.         $\lambda_i \longleftarrow \lambda_i + \delta_o$;<br>
10.         **if** $\#_{Adjust} \% \; \theta_2 = 0$ **then**<br>
11.             $\lambda_i \longleftarrow \lambda_i - \delta_d$ **end_if**<br>
12.     **end_if**<br>
13. **end_while**<br>
**end**

# Performance of DLM-98-BASIC-SAT

| Problem ID | Succ Ratio | Time | Num. Flips |
|---|---|---|---|
| aim-200-1-6-yes1-4.cnf | 10/10 | 0.06 | 29865 |
| aim-200-2-0-yes1-4.cnf | 10/10 | 0.33 | 129955 |
| aim-200-3-4-yes1-4.cnf | 10/10 | 0.53 | 98180 |
| aim-200-6-0-yes1-4.cnf | 10/10 | 0.02 | 632 |
| ii32d3.cnf | 10/10 | 0.33 | 8676 |
| ii32e5.cnf | 10/10 | 0.11 | 5083 |
| jnh220.cnf | 10/10 | 0.08 | 9918 |
| jnh301.cnf | 10/10 | 0.10 | 11039 |
| par8-4.cnf | 10/10 | 0.15 | 48256 |
| par8-5.cnf | 10/10 | 0.40 | 135212 |
| ssa7552-038.cnf | 10/10 | 0.13 | 16250 |
| ssa7552-160.cnf | 10/10 | 0.10 | 13742 |
| sw100-1.cnf | 10/10 | 0.62 | 117577 |
| sw100-2.cnf | 10/10 | 1.43 | 288571 |
| flat100-7.cnf | 10/10 | 2.64 | 859110 |
| flat100-9.cnf | 10/10 | 0.06 | 16428 |
| uf200-07.cnf | 10/10 | 0.13 | 12457 |
| uf200-09.cnf | 10/10 | 0.17 | 15005 |
| ais10.cnf | 10/10 | 0.23 | 18916 |
| ais12.cnf | 10/10 | 2.19 | 140294 |
| logistics-a.cnf | 10/10 | 0.16 | 17427 |
| logistics-b.cnf | 10/10 | 0.16 | 18965 |
| logistics-c.cnf | 10/10 | 0.21 | 16870 |
| logistics-d.cnf | 10/10 | 1.65 | 48603 |

# Global Search by Modifying Problem Formulations

- Problem – DLM-98-BASIC-SAT cannot solve or solve well *par16-**, *f2000, hanoi4, par32-*, hanoi5*
  - search trajectory is easily trapped in hard SAT instances
  - apply global search to overcome local attraction
- Strategy 1: trap escaping to avoid getting stuck in traps [AAAI'99]
  - *trap*: at $(x\ ,\ \lambda)$, any change to a single variable in $x$ will increase $L_d$
  - how to overcome traps?
    a) associate each clause with a counter
    b) implicitly remember traps visited by increasing counters for clauses in a trap
    c) add **extra** penalty on clause with the largest counter
  - implicit effect on Lagrangian formulation: adding extra penalties on constraints

# Global Search by Modifying Problem Formulations (cont'd)

- Strategy 2: trap avoidance to avoid repeating points visited before
  - Intuitively, if a search does not repeat points in its trajectory, then it is performing global search
  - Steps:

    a) keep a finite queue of previously visited points $x_i^s$

    b) add a new "$distance\_penalty$" in the objective:

$$L_d(x, \lambda) = (-distance\_penalty) + \sum_{i=1}^{n} \lambda_i U_i(x)$$

where $distance\_penalty$ is the sum of Hamming distances from $x$ to every point in $x_i^s$

$$distance\_penalty = \sum_i \min(\theta_t, |x - x_i^s|)$$

# DLM-2000-SAT

**procedure** *DLM-2000-SAT*

Reduce the original SAT instance;<br>
Generate a random starting point using a fixed seed, and initialize $\lambda_i \longleftarrow 0$;<br>
**while** solution not found and time not used up **do**<br>
    Pick $x_j \notin$ TabuList that reduces $L_d$ the most; /* $x$ loop */<br>
    Flip $x_j$;<br>
    **If** $\#_{Flips} \% w_s = 0$ **then** Update the queue on historical points **end_if**<br>
    Maintain TabuList;<br>
    **if** $\#_{FlatMoves} > \theta_1$ **then** /* $\lambda$ loop */<br>
        $\lambda_i \longleftarrow \lambda_i + \delta_o$;<br>
        **if** $\#_{Adjust} \% \theta_2 = 0$ **then**<br>
            $\lambda_i \longleftarrow \lambda_i - \delta_d$; **end_if**;<br>
    **end_if**<br>
**end_while**<br>
**end**

# Experimental Results

| Problem ID | DLM-2000-SAT | | | WalkSAT/GSAT | | DLM-99-SAT |
|---|---|---|---|---|---|---|
| | Succ. Ratio | Sec. | # of Flips | SR | Sec. | Sec. |
| par16-1 | 10/10 | 101.7 | $1.3 \cdot 10^7$ | NR | NR | 96.5 |
| par16-2 | 10/10 | 154.0 | $2.1 \cdot 10^7$ | NR | NR | 95.7 |
| par16-3 | 10/10 | 76.3 | $9.8 \cdot 10^6$ | NR | NR | 125.7 |
| par16-4 | 10/10 | 83.7 | $1.1 \cdot 10^7$ | NR | NR | 54.5 |
| par16-5 | 10/10 | 121.9 | $1.5 \cdot 10^7$ | NR | NR | 178.5 |
| par16-1-c | 10/10 | 20.8 | 2786081 | NR | NR | 28.8 |
| par16-2-c | 10/10 | 51.6 | 6824355 | NR | NR | 61.0 |
| par16-3-c | 10/10 | 27.5 | 3674644 | NR | NR | 35.3 |
| par16-4-c | 10/10 | 35.8 | 4825594 | NR | NR | 46.1 |
| par16-5-c | 10/10 | 32.4 | 4264095 | NR | NR | 44.6 |
| f600 | 10/10 | 0.80 | 73753 | NR | 35* | 0.664 |
| f1000 | 10/10 | 3.21 | 285024 | NR | 1095* | 3.7 |
| f2000 | 10/10 | 19.2 | 1102816 | NR | 3255* | 16.2 |
| hanoi4 | 10/10 | 6515 | $6.3 \cdot 10^8$ | NR | NR | 14744 |
| hanoi$4_s$ | 10/10 | 9040 | $1.1 \cdot 10^9$ | NR | NR | 14236 |

*: Results from Selman93 for similar but not the same problems in the DIMACS archive
**: Results from Selman95 on a SGI Challenge computer
Our results are collected from Pentium-III 500MHz with Solaris.

# Experimental Results (cont'd)

| Problem ID | DLM-2000-SAT Succ. Ratio | DLM-2000-SAT Sec. | DLM-2000-SAT # of Flips | WalkSAT/GSAT SR | WalkSAT/GSAT Sec. | DLM-99-SAT Sec. |
|---|---|---|---|---|---|---|
| g125-17 | 10/10 | 41.4 | 434183 | 7/10** | 264** | 144.8 |
| g125-18 | 10/10 | 4.8 | 22018 | 10/10** | 1.9** | 3.98 |
| g250-15 | 10/10 | 17.7 | 2437 | 10/10** | 4.41** | 12.9 |
| g250-29 | 10/10 | 193.1 | 289962 | 9/10** | 1219** | 331.4 |
| anomaly | 10/10 | 0.00 | 259 | NR | NR | NR |
| medium | 10/10 | 0.02 | 1537 | NR | NR | NR |
| huge | 10/10 | 0.19 | 10320 | NR | NR | NR |
| bw-large-a | 10/10 | 0.10 | 6176 | 0.3*** | NR | NR |
| bw-large-b | 10/10 | 1.55 | 67946 | 22*** | NR | 1.9 |
| bw-large-c | 10/10 | 72.36 | 1375437 | 670*** | NR | 292.2 |
| bw-large-d | 10/10 | 146.28 | 1112332 | 937*** | NR | 2390 |

**: Results from Selman95 on a SGI Challenge computer

***: Results from the paper by H. Kautz and B. Selman, AAAI 1996

# More Comparisons

- Comparisons to LSDL by Choi, Lee and Stuckey (ICTAI 1998)

| Problem ID | DLM-2000-SAT | | | *LSDL* | |
|---|---|---|---|---|---|
| | Succ. Ratio | Sec. | # of Flips | GENET | MAX |
| g125-17 | 10/10 | 41.4 | 434183 | 282.0* | 192.0* |
| g125-18 | 10/10 | 4.8 | 22018 | 4.5 | 1.1 |
| g250-15 | 10/10 | 17.7 | 2437 | 0.418 | 0.328 |
| g250-29 | 10/10 | 193.1 | 289962 | 876.0* | 678.0* |

LSDL results were collected from a SUN Sparc classic, model unknown.

- Comparisons to GRASP by Marques-Silva and Sakalla (IEEE TC 1999)

| Problem Class | DLM-2000-SAT | | | GRASP |
|---|---|---|---|---|
| | Succ. Ratio | Sec. | # of Flips | Sec. |
| f- | 10/10 | 7.7* | 487198* | - |
| g- | 10/10 | 64.3* | 187150* | - |
| par8- | 10/10 | 0.25 ** | 81022** | 0.4 |
| par16- | 10/10 | 108 * | $1.4 \cdot 10^{7*}$ | 9844 |
| hanoi- | 10/10 | 7778 * | $8.7 \cdot 10^{8*}$ | 14480 |

GRASP results were collected from a SUN SPARC 5/85 computer.

# Summary and Conclusions

- DLM combined with global search strategies is very efficient
  - sound mathematical foundation - theory of discrete constrained optimization using Lagrange Multipliers
  - nice global-search properties provided by distance_penalty and trap avoidance

- Possible future directions
  - gather and maintain useful yet simple information
    a) location and size of basins;
    b) degrees of difficulty in satisfying different clauses
  - represent efficiently this information for future recall
    a) new objective containing the information
    b) new constraints containing the information
    c) new Lagrangian formulation