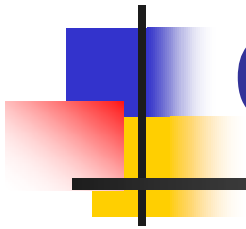
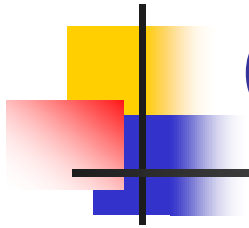


Propositional Satisfiability and Constraint Satisfaction

Stochastic Local Search Application

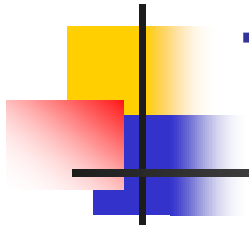
Author: Holger H. Hoos, Thomas Stützle

Presented by: Suling Yang



Outline:

1. The Satisfiability Problem (SAT)
2. The GSAT Architecture
3. The WalkSAT Architecture
4. Dynamic Local Search Algorithms for SAT
5. Constraint Satisfaction Problems (CSP)
6. SLS Algorithms for CSPs



The SAT Problem

- To decide or search for a given propositional formula *F*, whether there exists an truth assignment (a model) to the variables in *F* under which *F* evaluates to true.
- Conjunctive Normal Form (CNF) Representations and Transformations:
  - $\bigwedge_{i=1..m} \bigvee_{j=1..n} l_{ij}$

Alternative Formulations of SAT (1)

- True: 1 ; False: 0.
- Variables' domain: {0,1}
- $I(x) := x$; $I(\neg x) := 1 - x$.
- For $c_i = l_1 \vee l_2 \vee \ldots \vee l_{k(i)}$,
  $I(c_i) = I(l_1) + I(l_2) + \ldots + I(l_{k(i)})$.
- For $c_i = l_1 \wedge l_2 \wedge \ldots \wedge l_{k(i)}$,
  $I(c_i) = I(l_1) * I(l_2) * \ldots * I(l_{k(i)})$.
- A truth assignment satisfies c_i if $I(c_i) \geqslant 1$

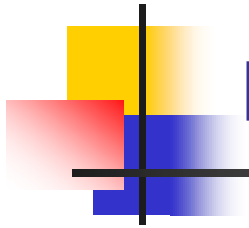
Alternative Formulations of SAT (2)

- u_i (F,a) := 1 if clause c_i of F is unsatisfied under assignment a, *and* u_i (F,a) := 0 otherwise.
- U (F,a) := $\Sigma_{i=1..m} u_i$ (F,a).
- A model of F corresponds to a solution of

$$a^* \in \underset{a \in \{0,1\}^n}{\arg\min} U(F,a)$$

Subject to: $\forall i \in \{1,2,...,m\} : u_i(F,a) = 0$

Which can be consider as a discrete constrained optimization problem.

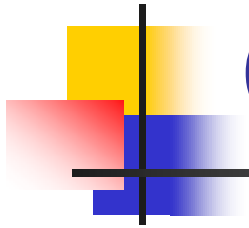


Polynomial Simplification of CNF Formulae

- Elimination of duplicate literals and clauses:
  - E.g. $(a \vee b \vee a) \wedge (a \vee b) = (a \vee b) \wedge (a \vee b) = (a \vee b)$
- Elimination of tautological clauses:
  - E.g. $(a \vee \neg a) = T$
- Elimination of subsumed clauses:
  - E.g. $(a \vee b) \wedge (a \vee b \vee c) = (a \vee b)$
- Elimination of clauses containing pure literals:
  - E.g.

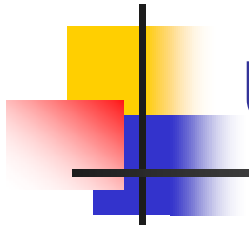
$$(a \vee b) \wedge (a \vee \neg b \vee \neg c) \wedge (c \vee \neg b)$$

$$= (c \vee \neg b) = T \quad \textit{if } b \textit{ is } \textit{false}.$$



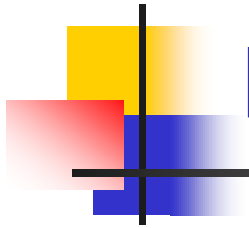
Complete Unit Propagation

- Unit Clause: a clause consisting of only a single literal.
  - E.g. $(a) \vee (\neg a \vee b)$
- Unit Resolution:
  - E.g. $(a) \vee (\neg a \vee b) = (b)$
- Complete Unit Propagation: repeat application of unit resolution until:
  - No more unit clause, or
  - Empty clause, or
  - No more clauses.



Unary and Binary Failed Literal Reduction

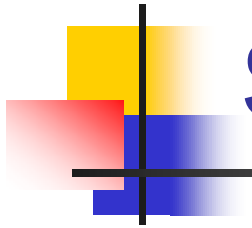
- Unary failed literal reduction: If setting a variable x occurring in the given formula F to true makes F unsatisfiable, i.e. adding a unit clause $c:=x$ to F resulting an empty clause, then add a unit clause $c := \neg x$ to F yields a logically equivalent formula F'. Then, we can apply complete unit propagation.
- Binary failed literal reduction works in the same way.



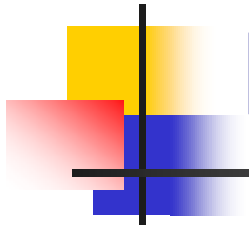
Randomly Generated SAT Instance

- Random clause length model (also called fixed density model):
  - *n* variables and *m* clauses; each of 2*n* literals are chosen with fixed probability *p.*
- Fixed clause length model (also known as Uniform Random k-SAT):
  - *n* variables, *m* clauses, and clause length *k*; *k* literals are chosen uniformly at random from 2*n* literals, and check if it contains multiple copies of the same literal, or it is tautological.

Random *k*-SAT Hardness and Solubility Phase Transition

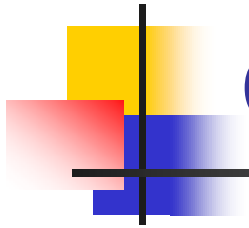


- For a fixed number of variables *n,* when *m* is small, all formulae are underconstrained and therefore satisfiable; however, when increasing the number *m* of clauses crossing some critical value *m**, the probability of generating a satisfiable instance drops sharply to almost zero.
- This rapid change in solubility is called a phase transition.
- It's empirically shown that problem instances from the phase transition region of Uniform Random 3-SAT tend to be hard.



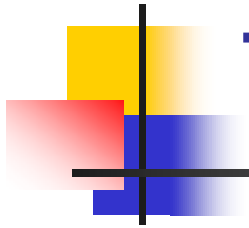
Practical Applications of SAT

- Bounded Model Checking (BMC)
- Binary Decision Diagrams (BDDs)
- Asynchronous circuit design
  - Signal Transition Graphs (STGs)
  - Complete State Coding (CSC) Problem
- Real-world spots scheduling problems



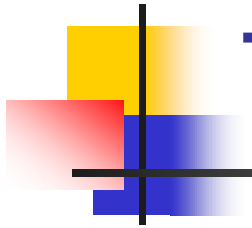
Generalisations and Related Problems

- CSP
  - Multi-Valued SAT (MVSAT)
  - Pseudo-Boolean CSPs
- (unweighted) MAX-SAT
- weighted MAX-SAT
- Dynamic SAT (DynSAT)
- Propositional Validity Problem (VAL)
- Satisfiability Problem for Quantified Boolean Formulae (QSAT)
- #SAT



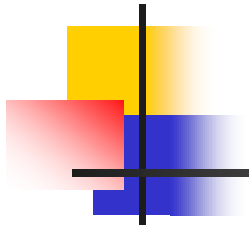
The GSAT Architecture

- SLS algorithms for SAT
- 1-exchange neighbourhood
- Evaluation function *g(F,a)* maps each variable assignment *a* to the number of clauses of the given formula *F* unsatisfied under *a*
- Model *m* (solution) of *F* : *g(F,m)=0*
- Iterative improvement methods
- Differ primarily in underlying variable selection method



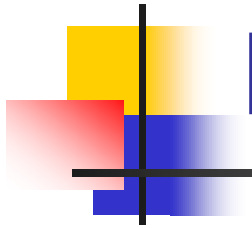
The Basic GSAT Algorithm

procedure *GSAT(F, maxTries, maxSteps)*
    input: CNF formula *F*, positive integers *maxTries* and *maxSteps*
    output: model of *F* **or** 'no solution found'
    for *try* := 1 **to** *maxTries* **do**
        a := randomly chosen assignment of the variables in formula *F*;
        for *step* := 1 **to** *maxSteps* **do**
            if *a* satisfies *F* **then return** *a* **end**
            x := randomly selected variable flipping that minimizes the number of unsatisfied clauses;
            a := *a* with *x* flipped;
        end
    end
    return 'no solution found'
end *GSAT*



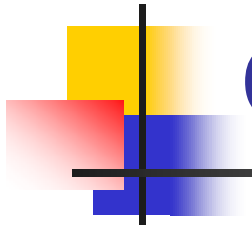
Basic GSAT (1)

- Underlying search procedure: simple best-improvement procedure
- Variable selection method: a variable that results in maximal decrease in the number of unsatisfied clauses
- Tie breaking method: randomly selected according to a uniform distribution
- Escaping local minima method: static restart mechanism
- Termination method: a model is found, or maxTries sequences of maxSteps variable flips have been performed without finding a model
- Evaluation method: change in #unsatisfied clauses



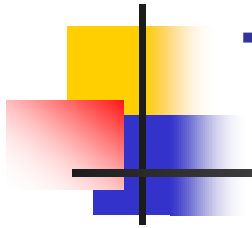
Basic GSAT (2)

- For any fixed number of restarts, GSAT is essentially incomplete, and severe stagnation behaviour is observed on most SAT instances
- Outperformed the best systematic search algorithms for SAT



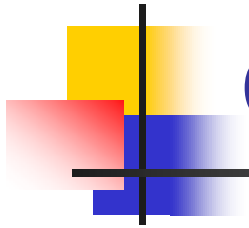
GSAT with Random Walk (GWSAT)

- Underlying search strategy: randomised best-improvement method – incorporate *conflict-directed random walk steps* with probability *wp*
- Escaping from a local minima method (1): with probability *wp>0*, this algorithm allows arbitrarily long sequences of random walk steps; this implies that from arbitrary assignment, a model can be reached with a positive, bounded probability
- Escaping from a local minima method (2): static restart mechanism



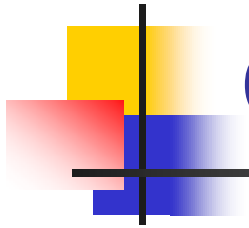
The GWSAT Algorithm

procedure *GSAT(F, maxTries, maxSteps)*
    input: CNF formula *F*, positive integers *maxTries* and *maxSteps*
    output: model of *F* **or** 'no solution found'
    for *try* := 1 **to** *maxTries* **do**
        a := randomly chosen assignment of the variables in formula *F*;
        for *step* := 1 **to** *maxSteps* **do**
            if *a* satisfies *F* **then return** *a* **end**
            x := randomly selected variable flipping that minimizes the number of unsatisfied clauses with probability *1-wp*; otherwise, choose a variable from an unsatisfied clause.
            a := *a* with *x* flipped;
        end
    end
    return 'no solution found'
end *GSAT*



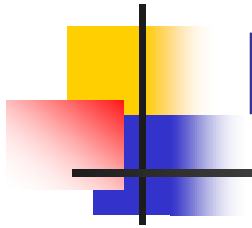
GSAT with Random Walk (GWSAT) (2)

- Outperforms basic GSAT
- Probabilistically approximately complete (PAC)
- Does not suffer from stagnation behaviour with sufficiently high noise setting, and shows exponential RTDs
- For low noise settings, stagnation behaviour is frequently observed



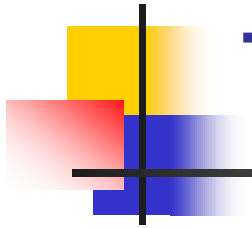
GSAT with Tabu Search (GSAT/Tabu)

- Underlying search strategy: tabu search
  - After a variable *x* has been flipped, it cannot be flipped back within the next *tt* steps
  - Efficient implementation
- For sufficient high *tt* settings, GSAT/Tabu does not suffer from stagnation behaviour, and for hard problem instances, it shows exponential RTDS.
- It's not clear whether GSAT/Tabu with fixed cutoff parameter *maxSteps* has the PAC property.
- Using instance-specific optimised tabu tenure settings for GSAT/Tabu, it typically performs significantly better than GWSAT.



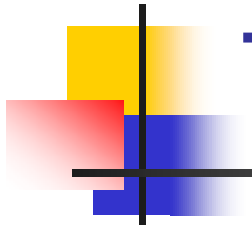
HSAT and HWSAT

- When in a search step there are several variables with identical score, HSAT always selects the least recently flipped variable.
- Although HSAT outperforms basic GSAT, it's more likely to get stuck in local minima.
- HWSAT: HSAT extended with random walk mechanism.
- HWSAT has PAC property.



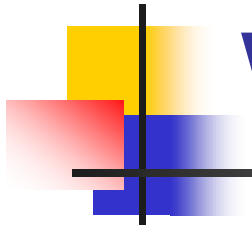
The WalkSAT Architecture

- Based on 2-stage variable selection process focused on the variables occurring in currently unsatisfied clauses:
  - 1st stage: A clause *c* that is unsatisfied under the current assignment is selected uniformly at random.
  - 2nd stage: one of the variables appearing in *c* is then flipped to obtain the new assignment.
- Dynamically determined subset of the GSAT neighbourhood relation – substantially reduced effective neighbourhood size
- Same random search initialisation and static random restart as GSAT



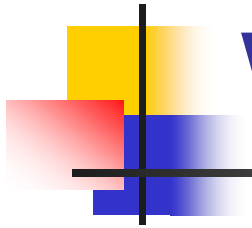
The WalkSAT Architecture

procedure *WalkSAT(F, maxTries, maxSteps, slc)*
input: CNF formula *F*, positive integers *maxTries* and *maxSteps,* heuristic function *slc*
output: model of *F* **or** 'no solution found'
for *try* := 1 **to** *maxTries* **do**
a := randomly chosen assignment of the variables in formula *F*;
for *step* := 1 **to** *maxSteps* **do**
if *a* satisfies *F* **then return** *a* **end**
c := randomly selected clause unsatisfied under *a;*
x := variable selected from *c* according to heuristic function *slc;*
a := *a* with *x* flipped;
end
end
return 'no solution found'
end *WalkSAT*



WalkSAT/SKC

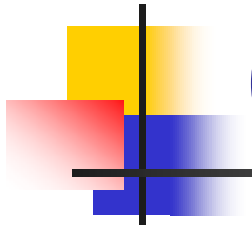
- 1ST WalkSAT algorithm
- The scoring function $score_b(x)$: counts the number of currently satisfied clauses that will be broken – become unsatisfied – by flipping a given variable *x*.
- Variable selection scheme:
  - *Zero damage step*: if there is a variable with $score_b(x)=0$ in the clause selected in stage 1, this variable is flipped.
  - *Greedy step*: if no such variable exists, with a certain probability *1-p*, the variable with minimal score value is selected.
  - *Random walk step*: with probability *p* (*noise setting*), one of the variables from *c* is selected uniformly at random.



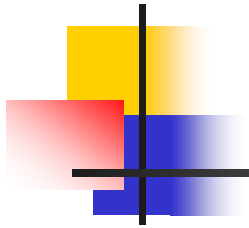
WalkSAT/SKC (2)

- PAC property when applied to 2-SAT; unknown in general case.
- In practice, WalkSAT/SKC with sufficiently high noise setting does not appear to suffer from any stagnation behaviour, and its runtime behaviour is characterized by exponential RTDs.
- Stagnation behaviour is observed for low noise settings.
- With optimised noise setting, WalkSAT/SKC probabilistically dominates GWSAT in terms of the number of variable flips, but not HWSATor GSAT/Tabu; in terms of CPU times, it's always outperforms all GSAT variants.

WalkSAT with Tabu Search (WalkSAT/Tabu)

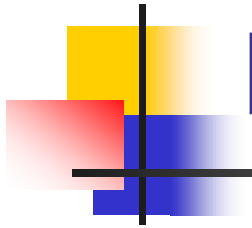


- Similar as WalkSAT/SKC; additionally enforces a tabu tenure of *tt* steps for each flipped variable.
- If the selected clause *c* does not allow a zero damage step, WalkSAT/Tabu picks the one with the highest score of all the variables occurring in c that are not tabu.
- *null-flip*: all variables appearing in *c* are tabu, in which case no variable is flipped.
- WalkSAT/Tabu with fixed *maxTries* parameter has been shown to be essentially incomplete.
- With sufficient high tabu tenure settings, WalkSAT/Tabu's run-time behaviour is characterised by exponential RTDs; but there are cases in which extreme stagnation behaviour is observed.
- Typically, WalkSAT/Tabu performs significantly better than WalkSAT/SKC.



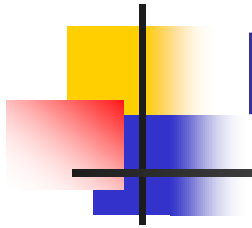
Novelty

- Uses a history-based variable selection mechanism; based on variable's age: the number of local search steps that have been performed since a variable was last flipped.
- Uses the same scoring function as GSAT.
- Variable selection scheme:
  - If the variable with the highest score



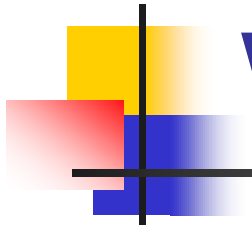
Novelty$^+$

- By extending Novelty with conflict-directed random walk analogously to GWSAT, the essential incompleteness as well as the empirically observed stagnation behaviour can be overcome.
- With probability *1-wp*, Novelty$^+$ selects the variable to be flipped according to the standard Novelty mechanism; otherwise, performs a random walk step.
- Novelty$^+$ is provably PAC for *wp>0* and shows exponential RTDs for sufficiently high setting of the primary noise parameter *p.*



R-Novelty and R-Novelty$^+$

- R-Novelty's variable selection strategy is even more deterministic than Novelty's.
- Diversification mechanism: Every 100 steps, a variable is randomly chosen from the selected clause and flipped – still not sufficient -> R-Novelty is essentially incomplete for fixed *maxTries*.
- Analogous to Novelty$^+$, R-Novelty with a random walk mechanism leads to R-Novelty$^+$.
- R-Novelty$^+$ is provably PAC for *wp>0* and shows exponential RTDs for sufficiently high noise setting.
- Do not reach the performance of R-Novelty$^+$.



WalkSAT with Adaptive Noise

- Low noise settings leads to stagnation behaviour, while high noise setting *maxSteps* has typically little or no impact on the behaviour of algorithm, since the corresponding RTDs are closely approximated by exponential distributions.
- Finding the optimal noise setting is typically rather difficult; it appear to depend on the given problem instance.
- Adaptive WalkSAT use high noise values only when they are needed to escape from stagnation situations.

Dynamic Local Search Algorithms for SAT

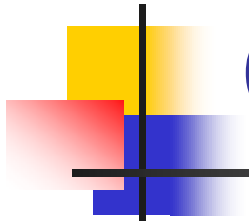
- Most DLS algorithms are based on variants of GSAT as their underlying local search procedure.
- The penalty associated with clause *c*, *clp(c)*, is updated in each iteration.
- Evaluation function:

$$g'(F,a) := g(F,a) + \sum_{c \in CU(F,a)} clp(c)$$

- Another representation:

$$clw(c) := clp(c) + 1$$

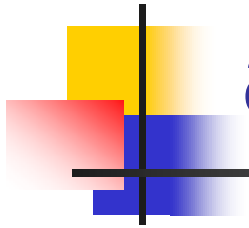
- $$g'(F,a) := \sum_{c \in CU(F,a)} clw(c)$$



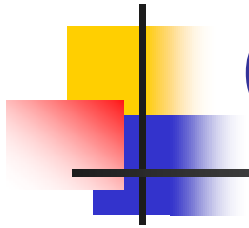
GSAT with Clause Weights

- Weights associated with clauses are initially set to one; before each restart, the weights of all currently unsatisfied clauses are increased by one.
- Underlying local search procedure: a variant of basic GSAT that use the modified evaluation function.
- Different from other DLS methods: begins each local search phase from a randomly selected variable assignment.
- Performs substantially better than basic GSAT on some instances; with GWSAT as underlying local search procedure, further performance improvements can be achieved.
- Breakout method: performs weight updates whenever a local minimum of the modified evaluation function is encountered. With tabu: the most recently visited variable assignments

Methods using Rapid Weight adjustments



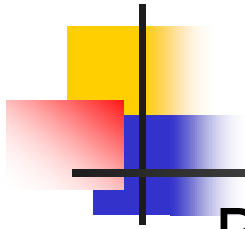
- Benefit from discovering which clauses are most difficult to satisfy relative to recent assignments.
- WGSAT: uses the same weight initialisation and update procedure as GSAT with Clause Weights, but performs only a single GSAT step before updating the clause weights.
- UGSAT: uses a best-improvement local search strategy, but restricts the neighbourhood to the set of variables appearing in currently unsatisfied clauses.
- WGSAT with Decay: uniformly decaying all clause weights in each weight update phase before the weights of the currently unsatisfied clauses are increased.



Guided Local Search (GLS)

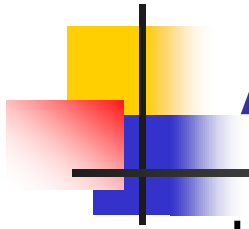
- GLS for SAT (GLSSAT): from the set of variables that lead to a strict decrease in the total penalty of unsatisfied clauses (if no such variable exists, then from those that do not cause an increase in the evaluation function), the one whose last flip has occurred least recently is flipped.
- Performs a complete pass of unit propagation before search begins.
- The penalties of all clauses with maximal utilities are incremented by one after each local search phase.
- GLSSAT2: all clause penalties are multiplied by a factor of 0.8 after every 200 penalty updates.

Discrete Lagrangian Method (DLM)



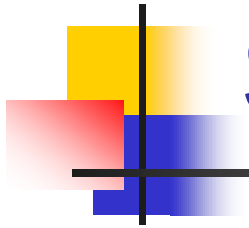
- Basic DLM: underlying local search procedure is based on GSAT/Tabu with clause weights.
- Terminates when the number of neighbouring assignments with larger or equal evaluation function value exceeds a give threshold.
- Bound the range of clause penalties.
- Perform a complete pass of unit propagation.
- DLM-99-SAT: uses temporary clause penalties.
- DLM-2000-SAT: long-term memory mechanism

Exponentiated Subgradient Algorithm (ESG)



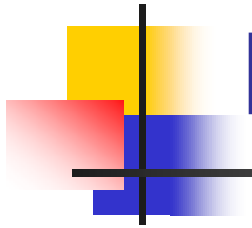
- Underlying local search procedure: best improvement search method (simple variant of GSAT).
- Variable selection: appear in currently unsatisfied clauses and whose flip leads to a maximal reduction in the total weight of unsatisfied clauses.
- Scaling stage: weights of all clauses are multiplied by a factor depending on their satisfaction status.
- Smoothing stage: all clause weights are smoothed using the formula $clw(c) := clw(c) \cdot r + (1 - r) \cdot \overline{w}$
- The weight update steps are computationally much more expensive than the weighted search steps.

Scaling and Probabilistic Smoothing (SAPS)



- Scaling stage is restricted to the weights of currently unsatisfied clauses, and smoothing is only performed with a certain probability.
- By applying the expensive smoothing operation only occasionally, the time complexity of the weight update procedure can be substantially reduced.
- Compared to ESG, SAPS typically requires a similar number of variable flips for finding a model of a given formula, but in terms of time performance it is significantly superior to ESG, DLM-2000-SAT, and best known WalkSAT variants; but SAPS does not reach the performance of Novelty$^+$ for some cases.

Constraint Satisfaction Problems (CSP)



- An instance of the CSP is defined by a set of variables, a set of possible values (or domain) for each variable, and a set of constraining conditions (constraints) involving one or more of the variables.
- The CSP is to decide for a given CSP instance whether all variables can be assigned values from their respective domains such that all constraints are simultaneously satisfied.
- P is a finite discrete CSP instance if and only if all variables in P have discrete and finite domains.
- CSP instances for which at least one solution exists are also called consistent, while instances that do not have any solutions are called inconsistent.

Encoding CSP instances into SAT

- Sparse encoding (unary transform or direct encoding):

  $c_{i,v}$ *represents* $x_i := v$.

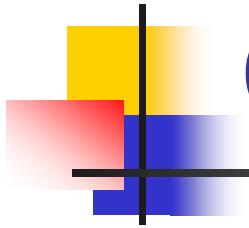
(1) $\neg c_{i,v1} \vee \neg c_{i,v2} \quad (1 \le i \le n; v1, v2 \in D(x_i); v1 \,\mathbf{p}\, v2)$

(2) $c_{i,v0} \vee c_{i,v1} \vee \ldots \vee c_{i,vk-1} \quad (1 \le i \le n)$

(3) $\neg c_{i1,v0} \vee \neg c_{i,v1} \vee \ldots \vee \neg c_{i,vk-1}$ *where* $(x_{i1} := v1; x_{i2} := v2; \ldots; x_{is} := vs)$

violates some constraint $C_j \in C$ *with* $s(C_j) := s$.

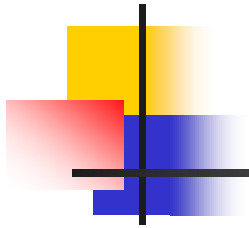
- Compact encoding (binary transform or log encoding).



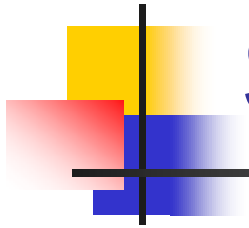
CSP Simplification and Local Consistency Techniques

- Local consistency techniques can reduce the effective domains of CSP variables by eliminating values that cannot occur in any solution.
- Arc consistency
- Path consistency

Prominent Benchmark Instances for the CSP

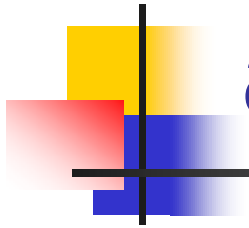


- Uniform Random Binary CSP
- Graph colouring Problem
- Quasigroup Completion Problem
- n-Queen Problem



SLS Algorithms for CSPs

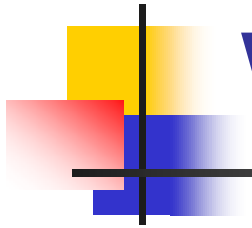
- Categorized into three types:
  - SLS algorithms for SAT applied to SAT-encoded CSP instances
  - Generalisations of SLS algorithms for SAT
  - Native SLS algorithms for CSPs



“Encode and Solve as SAT” approach

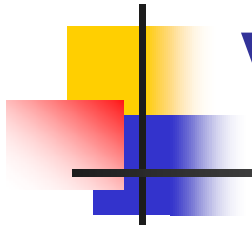
- It allows the use of highly optimised and efficiently implemented “of-the-shelf” SAT solvers.
- Inability of standard SAT algorithms to exploit the structure present in given CSP instances.

Pseudo-Boolean CSP and WSAT(PB)



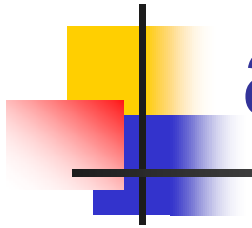
- Pseudo-Boolean CSP, or (Linear) Pseudo-Boolean Programming, have Boolean values for each variables.
- WSAT(PB) is based on direct generalisation of WalkSAT architecture to Pseudo-Boolean CSP.
- The evaluation function is based on the notion of the net integer distance of a constraint from being satisfied.
- Randomly selects a constraint, and flips the variable in the constraint that leads to largest decrease in the evaluation function; if no such variable, choose the least recent flipped one with probability *wp*; otherwise, choose the one with minimal increase in the evaluation function.
- Use simple tabu mechanism.

WalkSAT Algorithms for Many-Valued SAT

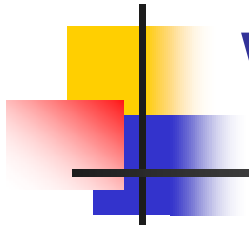


- Non-Boolean SAT: non-Boolean literal is of the form *z/v* or ~*z/v*, where *z* is a variable and *v* a value from the domain of *z.*
- A variable flip corresponds to assigning a different value to a non-Boolean variable such that the literal selected in the corresponding search step, and hence the clause in which it appears becomes satisfied.
- MV-WalkSAT solves a variant of many-valued SAT that is slightly richer than the non-Boolean CNF formulae underlying NB-SAT.

Min Conflicts Heuristic (MCH) and Variants

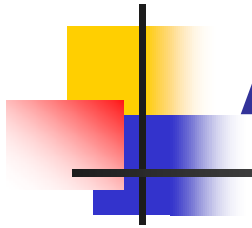


- MCH iteratively modifies the assignment of a single variable in order to minimise the number of violated constraints.
  - Variable initialisation: uniformly at random
  - Variable selection: uniformly at random from the conflict set, which is the set of all variables that appear in a constraint that is unsatisfied under the current assignment.
  - Value selection: the number of unsatisfied constraints (conflicts) is minimised.
- MCH is essentially incomplete.



WMCH and TMCH

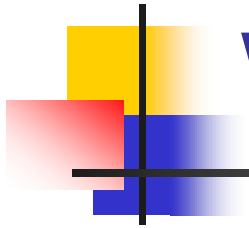
- WMCH is a variant of MCH that uses a random walk mechanism analogous to GWSAT.
- WMCH is PAC for noise setting >0.
- TMCH is MCH extended with a tabu search mechanism.



A Tabu Search for CSPs

- TS-GH by Galinier and Hao:
  - Amongst all pairs (x,v′) such that variable x appears in a currently violated constraint and v′ is any value from the domain of x, TS-GH chooses the one that leads to a maximal decrease in the number of violated constraints.
  - Augmented with the same tabu mechanism used in TMCH.
- When computing evaluation function values, TS-GH uses incremental updating technique analogous to the one is used by GSAT.
- Empirical studies suggest that when applied to the conventional CSP, TS-GH generally achieves better performance than any of the MCH variants.

Further Readings and Related Work



- Covers some additional SLS algorithms for SAT and CSP problems.
- Page 306 of the text.