DM811

Heuristics for Combinatorial Optimization

Lecture 6
SAT

Marco Chiarandini

Department of Mathematics & Computer Science
University of Southern Denmark

Outline

1. Code Speed up

2. SAT

Outline

1. Code Speed up

2. SAT

Where do speedups come from?

Where can maximum speedup be achieved?
How much speedup should you expect?

Code Tuning

- Caution: proceed carefully! Let the optimizing compiler do its work!
  - optimizing flags (C++ `-O3`, java http://java.sun.com/developer/onlineTraining/Programming/JDCBook/perfTech.html)
  - just-in-time-compilation: it converts code at runtime prior to executing it natively, for example bytecode into native machine code. (in java done by deafult – to disable `-Djava.compiler=NONE` – in C++ possible via `llvm-g++` http://vmakarov.fedorapeople.org/spec/2011/llvmgcc64.html)
- Cache aware (`-m32` vs `-m64`)
- Profiling (java: `java -Xrunhprof:cpu=times prog` information on the time spent in each method of the program written to `java.hprof.txt`. C++: `gprof`, instruments)

- Expression Rules: Recode for smaller instruction counts.
- Loop and procedure rules: Recode to avoid loop or procedure call overhead.
- Hidden costs of high-level languages
- String comparisons in C: proportional to length of the string, not constant
- Object construction / de-allocation: very expensive
- Matrix access: row-major order $\neq$ column-major order
- Exploit algebraic identities
- Avoid unnecessary computations inside the loops

Where Speedups Come From?

McGeoch reports conventional wisdom, based on studies in the literature.

- Concurrency is tricky: bad -7x to good 500x
- Classic algorithms: to 1trillion and beyond
- Data-aware: up to 100x
- Memory-aware: up to 20x
- Algorithm tricks: up to 200x
- Code tuning: up to 10x
- Change platforms: up to 10x

Relevant Literature

Bentley, **Writing Efficient Programs; Programming Pearls** (Chapter 8 Code Tuning)

Kernighan and Pike, **The Practice of Programming** (Chapter 7 Performance).
Shirazi, **Java Performance Tuning**, O'Reilly

McCluskey, **Thirty ways to improve the performance of your Java program.** Manuscript and website: www.glenmccl.com/jperf

Randal E. Bryant e David R. O'Hallaron: **Computer Systems: A Programmer's Perspective**, Prentice Hall, 2003, (Chapter 5)

Outline

1. Code Speed up

2. SAT

SAT Problem

Satisfiability problem in propositional logic

$$
\begin{aligned}
&(x_5 \vee x_8 \vee \bar{x}_2) \wedge (x_2 \vee \bar{x}_1 \vee \bar{x}_3) \wedge (\bar{x}_8 \vee \bar{x}_3 \vee \bar{x}_7) \wedge (\bar{x}_5 \vee x_3 \vee x_8) \wedge \\
&(\bar{x}_6 \vee \bar{x}_1 \vee \bar{x}_5) \wedge (x_8 \vee \bar{x}_9 \vee x_3) \wedge (x_2 \vee x_1 \vee x_3) \wedge (\bar{x}_1 \vee x_8 \vee x_4) \wedge \\
&(\bar{x}_9 \vee \bar{x}_6 \vee x_8) \wedge (x_8 \vee x_3 \vee \bar{x}_9) \wedge (x_9 \vee \bar{x}_3 \vee x_8) \wedge (x_6 \vee \bar{x}_9 \vee x_5) \wedge \\
&(x_2 \vee \bar{x}_3 \vee \bar{x}_8) \wedge (x_8 \vee \bar{x}_6 \vee \bar{x}_3) \wedge (x_8 \vee \bar{x}_3 \vee \bar{x}_1) \wedge (\bar{x}_8 \vee x_6 \vee \bar{x}_2) \wedge \\
&(x_7 \vee x_9 \vee \bar{x}_2) \wedge (x_8 \vee \bar{x}_9 \vee x_2) \wedge (\bar{x}_1 \vee \bar{x}_9 \vee x_4) \wedge (x_8 \vee x_1 \vee \bar{x}_2) \wedge \\
&(x_3 \vee \bar{x}_4 \vee \bar{x}_6) \wedge (\bar{x}_1 \vee \bar{x}_7 \vee x_5) \wedge (\bar{x}_7 \vee x_1 \vee x_6) \wedge (\bar{x}_5 \vee x_4 \vee \bar{x}_6) \wedge \\
&(\bar{x}_4 \vee x_9 \vee \bar{x}_8) \wedge (x_2 \vee x_9 \vee x_1) \wedge (x_5 \vee \bar{x}_7 \vee x_1) \wedge (\bar{x}_7 \vee \bar{x}_9 \vee \bar{x}_6) \wedge \\
&(x_2 \vee x_5 \vee x_4) \wedge (x_8 \vee \bar{x}_4 \vee x_5) \wedge (x_5 \vee x_9 \vee x_3) \wedge (\bar{x}_5 \vee \bar{x}_7 \vee x_9) \wedge \\
&(x_2 \vee \bar{x}_8 \vee x_1) \wedge (\bar{x}_7 \vee x_1 \vee x_5) \wedge (x_1 \vee x_4 \vee x_3) \wedge (x_1 \vee \bar{x}_9 \vee \bar{x}_4) \wedge \\
&(x_3 \vee x_5 \vee x_6) \wedge (\bar{x}_6 \vee x_3 \vee \bar{x}_9) \wedge (\bar{x}_7 \vee x_5 \vee x_9) \wedge (x_7 \vee \bar{x}_5 \vee \bar{x}_2) \wedge \\
&(x_4 \vee x_7 \vee x_3) \wedge (x_4 \vee \bar{x}_9 \vee \bar{x}_7) \wedge (x_5 \vee \bar{x}_1 \vee x_7) \wedge (x_5 \vee \bar{x}_1 \vee x_7) \wedge \\
&(x_6 \vee x_7 \vee \bar{x}_3) \wedge (\bar{x}_8 \vee \bar{x}_6 \vee \bar{x}_7) \wedge (x_6 \vee x_2 \vee x_3) \wedge (\bar{x}_8 \vee x_2 \vee x_5)
\end{aligned}
$$

Does there exist a truth assignment satisfying all clauses?

Search for a satisfying assignment (or prove none exists)

SAT Problem

Satisfiability problem in propositional logic

$$
\begin{aligned}
&(x_5 \vee x_8 \vee \bar{x}_2) \wedge (x_2 \vee \bar{x}_1 \vee \bar{x}_3) \wedge (\bar{x}_8 \vee \bar{x}_3 \vee \bar{x}_7) \wedge (\bar{x}_5 \vee x_3 \vee x_8) \wedge \\
&(\bar{x}_6 \vee \bar{x}_1 \vee \bar{x}_5) \wedge (x_8 \vee \bar{x}_9 \vee x_3) \wedge (x_2 \vee x_1 \vee x_3) \wedge (\bar{x}_1 \vee x_8 \vee x_4) \wedge \\
&(\bar{x}_9 \vee \bar{x}_6 \vee x_8) \wedge (x_8 \vee x_3 \vee \bar{x}_9) \wedge (x_9 \vee \bar{x}_3 \vee x_8) \wedge (x_6 \vee \bar{x}_9 \vee x_5) \wedge \\
&(x_2 \vee \bar{x}_3 \vee \bar{x}_8) \wedge (x_8 \vee \bar{x}_6 \vee \bar{x}_3) \wedge (x_8 \vee \bar{x}_3 \vee \bar{x}_1) \wedge (\bar{x}_8 \vee x_6 \vee \bar{x}_2) \wedge \\
&(x_7 \vee x_9 \vee \bar{x}_2) \wedge (x_8 \vee \bar{x}_9 \vee x_2) \wedge (\bar{x}_1 \vee \bar{x}_9 \vee x_4) \wedge (x_8 \vee x_1 \vee \bar{x}_2) \wedge \\
&(x_3 \vee \bar{x}_4 \vee \bar{x}_6) \wedge (\bar{x}_1 \vee \bar{x}_7 \vee x_5) \wedge (\bar{x}_7 \vee x_1 \vee x_6) \wedge (\bar{x}_5 \vee x_4 \vee \bar{x}_6) \wedge \\
&(\bar{x}_4 \vee x_9 \vee \bar{x}_8) \wedge (x_2 \vee x_9 \vee x_1) \wedge (x_5 \vee \bar{x}_7 \vee x_1) \wedge (\bar{x}_7 \vee \bar{x}_9 \vee \bar{x}_6) \wedge \\
&(x_2 \vee x_5 \vee x_4) \wedge (x_8 \vee \bar{x}_4 \vee x_5) \wedge (x_5 \vee x_9 \vee x_3) \wedge (\bar{x}_5 \vee \bar{x}_7 \vee x_9) \wedge \\
&(x_2 \vee \bar{x}_8 \vee x_1) \wedge (\bar{x}_7 \vee x_1 \vee x_5) \wedge (x_1 \vee x_4 \vee x_3) \wedge (x_1 \vee \bar{x}_9 \vee \bar{x}_4) \wedge \\
&(x_3 \vee x_5 \vee x_6) \wedge (\bar{x}_6 \vee x_3 \vee \bar{x}_9) \wedge (\bar{x}_7 \vee x_5 \vee x_9) \wedge (x_7 \vee \bar{x}_5 \vee \bar{x}_2) \wedge \\
&(x_4 \vee x_7 \vee x_3) \wedge (x_4 \vee \bar{x}_9 \vee \bar{x}_7) \wedge (x_5 \vee \bar{x}_1 \vee x_7) \wedge (x_5 \vee \bar{x}_1 \vee x_7) \wedge \\
&(x_6 \vee x_7 \vee \bar{x}_3) \wedge (\bar{x}_8 \vee \bar{x}_6 \vee \bar{x}_7) \wedge (x_6 \vee x_2 \vee x_3) \wedge (\bar{x}_8 \vee x_2 \vee x_5)
\end{aligned}
$$

Does there exist a truth assignment satisfying all clauses?

Search for a satisfying assignment (or prove none exists)

Motivation

- From 100 variables, 200 constraints (early 90s)
  to 1,000,000 vars. and 20,000,000 cls. in 20 years.
- Applications:
  Hardware and Software Verification, Planning, Scheduling, Optimal Control, Protocol Design, Routing, Combinatorial problems, Equivalence Checking, etc.
- SAT used to solve many other problems!

SAT Problem

Satisfiability problem in propositional logic

Definitions:

- Formula in propositional logic: well-formed string that may contain
  - propositional variables $x_1, x_2, \ldots, x_n$;
  - truth values $\top$ ('true'), $\bot$ ('false');
  - operators $\neg$ ('not'), $\wedge$ ('and'), $\vee$ ('or');
  - parentheses (for operator nesting).
- Model (or satisfying assignment) of a formula F: Assignment of truth values to the variables in F under which F becomes true (under the usual interpretation of the logical operators)
- Formula F is satisfiable iff there exists at least one model of F, unsatisfiable otherwise.

SAT Problem (decision problem, search variant):

- **Given:** Formula F in propositional logic
- **Task:** Find an assignment of truth values to variables in F that renders F true, or decide that no such assignment exists.

SAT: A simple example

- **Given:** Formula $F := (x_1 \vee x_2) \wedge (\neg x_1 \vee \neg x_2)$
- **Task:** Find an assignment of truth values to variables x_1, x_2 that renders F true, or decide that no such assignment exists.

Definitions:

- A formula is in conjunctive normal form (CNF) iff it is of the form

$$\bigwedge_{i=1}^{m} \bigvee_{j=1}^{k_i} l_{ij} = (l_{11} \vee \ldots \vee l_{1k_1}) \wedge \ldots \wedge (l_{m1} \vee \ldots \vee l_{mk_m})$$

  where each literal l_{ij} is a propositional variable or its negation. The disjunctions $c_i = (l_{i1} \vee \ldots \vee l_{ik_i})$ are called clauses.

- A formula is in k-CNF iff it is in CNF and all clauses contain exactly k literals (*i.e.*, for all i, $k_i = k$).

- In many cases, the restriction of SAT to CNF formulae is considered.
- For every propositional formula, there is an equivalent formula in 3-CNF.

Example:

$$F := \begin{array}{l} \wedge (\neg x_2 \vee x_1) \\ \wedge (\neg x_1 \vee \neg x_2 \vee \neg x_3) \\ \wedge (x_1 \vee x_2) \\ \wedge (\neg x_4 \vee x_3) \\ \wedge (\neg x_5 \vee x_3) \end{array}$$

- F is in CNF.
- Is F satisfiable?
  Yes, *e.g.*, $x_1 := x_2 := \top$, $x_3 := x_4 := x_5 := \bot$ is a model of F.

MAX-SAT (optimization problem)

Which is the maximal number of clauses satisfiable in a propositional logic formula F?

Exercise

Definition

(Maximum) K-Satisfiability (SAT)
Input: A set U of variables, a collection C of disjunctive clauses of at most k literals, where a literal is a variable or a negated variable in U. k is a constant, $k \geq 2$.
Task: A truth assignment for U or a truth assignment that maximizes the number of clauses satisfied.

1. design one or more construction heuristics for the problem
2. show how the decision version of the graph coloring problem (GCP) can be encoded in a SAT problem
3. show how the constraint satisfaction problem (CSP) can be encoded in a SAT problem
4. are the results of the two previous points proves of the NP-completeness of the CSP and GCP?
5. devise preprocessing rules, ie, polynomial time simplification rules

Pre-processing

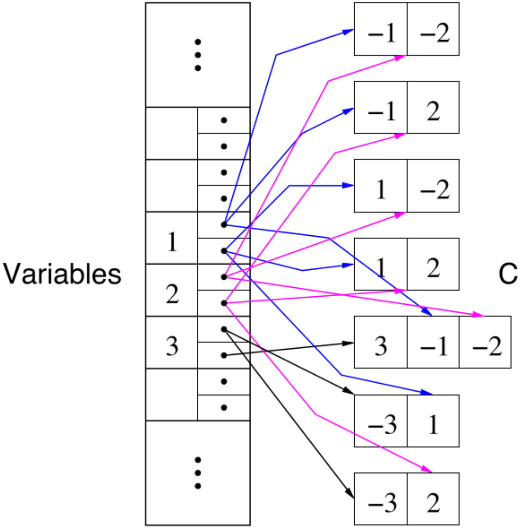
Pre-processing rules: low polynimial time procedures to decrease the size of the problem instance.

Typically applied in cascade until no rule is effective anymore.

Examples in SAT

1. eliminate duplicate literals
2. eliminate tautologies: $x_1 \vee \neg x_1 \ldots$
3. eliminate subsumed clauses
4. eliminate clauses with pure literals
5. eliminate unit clauses
6. unit propagation

Simple data structure for unit propagation



Construction heuristics

- Variable selection heuristics
  aim: minimize the search space
  plus: could compensate a bad value selection

- Value selection heuristics
  aim: guide search towards a solution (or conflict)
  plus: could compensate a bad variable selection

- Restart strategies
  aim: avoid heavy-tail behavior [GomesSelmanCrato'97]
  plus: focus search on recent conflicts when combined with dynamic heuristics

Variable selection heuristics

- Based on the occurrences in the (reduced) formula
  - Maximal Occurrence in clauses of Minimal Size (MOMS, Jeroslow-Wang)
- Variable State Independent Decaying Sum (VSIDS)
  - original idea (zChaff): for each conflict, increase the score of involved variables by 1, half all scores each 256 conflicts [MoskewiczMZZM2001]
  - improvement (MiniSAT): for each conflict, increase the score of involved variables by δ and increase $\delta := 1.05\delta$ [EenSörensson2003]

Value selection heuristics

- Based on the occurrences in the (reduced) formula
  - examples: Jeroslow-Wang, Maximal Occurrence in clauses of Minimal Size (MOMS), look-aheads