Department of Computer Science, Rochester Institute of Technology
Parallel Computing I, Graduate Team Project

Exhaustive Search Algorithms for 3-SAT
Team Formation Document

Christopher Wood, Eitan Romanoff, Ankur Bajoria

March 9, 2013

1 Problem Description

For our project we propose 3-SAT (or, more formally, 3-CNF-SAT), which is an NP-complete problem [1]. This is a decision problem in which takes as input a 3-CNF Boolean formula and returns YES if the formula is satisfiable, and NO otherwise. A 3-CNF formula, more formally known as a Boolean formula in 3-conjunctive normal form, is expressed as the Boolean AND of arbitrarily many clauses, where each clause is the Boolean OR of three literals, which is a Boolean variable or its negation. Such a Boolean formula is said to be satisfiable if and only if there exists an assignment of truth values to the variables such that substituting them into the literals of the formula will cause it to evaluate to true (or 1). Expressed as a formal language, we have that $3 - SAT = \{\langle\phi\rangle : \phi \text{ is satisfiable }\}$.

2 Exhaustive Search Algorithms for 3-SAT

An exhaustive search algorithm for solving the 3-SAT problem iterates over every combination of variable truth values (of which there exists 2^n total for n variables), substitutes (or assigns) them to the appropriate literal in each clause, and then evaluates the formula to determine if it is satisfiable. If for every possible variable truth assignment the formula does not evaluate to true (or 1), then the exhaustive 3-SAT algorithm returns NO. Otherwise, some satisfiable truth value assignment must exist, and so the 3-SAT algorithm returns YES. Formally, the exhaustive search algorithm is defined as follows:

3 Programs and Performance Metrics

The sequential and parallel programs we will deliver will take as input the 3-CNF formula, encoded using the DIMACS CNF format, and output a single Boolean truth value indicating whether or not the formula is satisfiable. The 3-CNF formula will be entered at the command line or it will be read from a file to facilitate our experiments. Based on the 3-SAT problem, each clause must have exactly three literals, so our program will enable the number of clauses and the number of variables to be parameters defined in the DIMACS CNF format. An example the 3-CNF formula $(x_1 \vee x_2 \vee x_3)$ problem encoded using DIMACS is shown below.

```
p cnf 3 1
1 2 −3 0
```

There are two main parts of the exhaustive search algorithm for 3-SAT: reading in the 3-CNF formula and setting up the appropriate data structures, and traversing and substituting all possible combinations of variable truth assignments into the 3-CNF formula to check for satisfiability. Both the sequential and parallel programs will share the first part so as to set up the globally accessible data structures containing the 3-CNF formula. This is a fixed amount of sequential overhead that must occur before any satisfiability checks can begin.

The second part of the program can be done in parallel. Formally, the second part is an instance of an agenda parallel problem, where each task attempts to satisfy a specific truth value assignment. This is because we are not concerned with all satisfiability results, we are only concerned with the answer to the question for all truth value assignments, "does this truth value assignment satisfy the 3-CNF formula?" Therefore, since we are only seeking the output of one task that answers this question, and there are no sequential dependencies between each task, we can divvy up the execution of these 2^n tasks among 2^n virtual processors. When implemented, we will clump the execution of many tasks together on a single processor because it is unlikely that we will have 2^n processors available.

Since 3-SAT is both an interesting problem in academia and often arises in the industry, we will be measuring the metrics of speedup and speedup, as well as the efficiency and sizeup efficiency. To acquire these metrics and attempt to model our problem with Amdal's and Gustafson's Law, we will measure the execution time of the sequential and parallelizable parts of the sequential and parallel programs. This will enable us to calculate the total execution time and sequential fraction of the program. Using these measurements, along with the problem size N (number of variables) and number of processors K, we can evaluate the aforementioned metrics.

4 Literature Survey and Additional Resources

As part of the graduate requirement, we will analyze [2], [3], and [4]. In addition, we will use the information and resources made available on the International SAT Competition web site [6], as well as the book by Drechsler et al. on advanced SAT solving techniques [5].

References

[1] Gormen, Thomas H., Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. Introduction to Algorithms. MIT Press 44 (1990), 97-138.

[2] Meyer, Quirin, Fabian Schönfeld, Marc Stamminger, and Rolf Wanka. 3-SAT on CUDA: Towards a Massively Parallel SAT Solver. *High Performance Computing and Simulation (HPCS)*, 2010 International Conference on. IEEE, 2010.

[3] Hamadi, Youssef, Said Jabbour, and Lakhdar Sais. ManySAT: A Parallel SAT Solver. *Journal on Satisfiability, Boolean Modeling and Computation* 6.4 (2009) 245-262.

[4] Zhang, Hantao, Maria Paola Bonacina, and Jieh Hsiang. PSATO: A Distributed Propositional Prover and its Application to Quasigroup Problems. *Journal of Symbolic Computation* 21.4 (1996) 543-560.

[5] Drechsler, Rolf and Stephan Eggersglüb. High Quality Test Pattern Generation and Boolean Satisfiability. *Springer*, 2012.

[6] The International SAT Competitions Web Page. `http://www.satcompetition.org/`. Accessed: 3/9/2013.

[7] Kaminsky, Alan. Building Parallel Programs: SMPs, Clusters, and Java. Cengage Course Technology, 2010. ISBN 1-4239-0198-3.