Exhaustive Search Algorithms for 3-SAT

Team Proposal Document

Christopher Wood, Eitan Romanoff, Ankur Bajoria
Team Satisfaction
Website: `http://ear7631.github.com/RIT-Parallel3Sat`

March 11, 2013

1 Problem Description

For our project we choose 3-SAT (or, more formally, 3-CNF-SAT), which is an NP-complete decision problem [1]. 3-SAT takes as input a 3-CNF Boolean formula and returns YES if the formula is satisfiable, and NO otherwise. A 3-CNF formula ϕ_n on n variables is expressed as the conjunction (Boolean AND) of arbitrarily many clauses, where each clause is the disjunction (Boolean OR) of exactly three literals and a literal is a Boolean variable or its negation. A formula ϕ_n is said to be satisfiable if and only if there exists an assignment of truth values to the n variables such that substituting them into the literals of phi will cause it to evaluate to true. Expressed as a formal language, we have that $3 - SAT = \{\langle\phi\rangle : \phi \text{ is a satisfiable 3-}CNF \text{ formula}\}$ (i.e. the set of all 3-CNF formula strings that are accepted by the 3-SAT language if they are satisfiable).

2 Exhaustive Search Algorithm for 3-SAT

An exhaustive search algorithm for solving the 3-SAT problem with input ϕ_n iterates over all 2^n configurations of variable truth values, and for each configuration, assigns the truth value of each variable to the appropriate literal in ϕ_n, and then evaluates ϕ_n to determine if it is true. If for every possible variable configuration ϕ_n does not evaluate to true, then the exhaustive 3-SAT algorithm returns NO. Otherwise, some satisfiable truth value configuration must exist, and so the 3-SAT algorithm returns YES.

3 Programs and Performance Metrics

The sequential and parallel programs we will deliver will take as input a 3-CNF formula ϕ_n, encoded using the DIMACS CNF format [2], and output a 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, the only restriction is that each clause must have exactly three literals. Therefore, our programs will enable the number of clauses and the number of variables to be parameters defined in the DIMACS CNF format. An example of the 3-CNF formula $(x_1 \vee x_2 \vee \neg x_3)$ problem encoded using DIMACS is shown below.

```
p cnf 3 1
1 2 −3 0
```

There are two main parts of the programs that implement the exhaustive search algorithm outlined in Section 2. First, the program must read in the 3-CNF formula ϕ_n and set up the appropriate data structures. Second, the program must traverse and assign all possible configurations of variable truth values into ϕ_n 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 an unavoidable sequential task.

The second part of the program can be done sequentially or in parallel. In a parallel program, this part is an instance of an agenda parallel problem, where each task evaluates a specific truth value assignment and returns the Boolean result. Collectively, we are only concerned with the result from a single task (i.e. the one that returns true). Therefore, since we are only seeking the result of an individual task, and there are no sequential dependencies that exist 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 for computation.

Finally, since 3-SAT is both an interesting problem in academia and often arises in the industry, we will be measuring the metrics of speedup ($Speedup(N,K)$) and sizeup ($Sizeup(T,K)$), as well as efficiency ($Eff(N,K)$) and sizeup efficiency ($SizeupEff(T,K)$). To acquire these metrics we will first attempt to model our problem with Amdal's Law. In doing so, we will measure the execution time of the sequential ($T_{seq}(N,K)$) and parallelizable ($T_{par}(N,K)$) parts of the sequential and parallel programs. We will also measure the sequential fraction F of the program. Similarly, we will model our problem with Gustafson's Law by measuring the problem size for the sequential ($N_{seq}(T,K)$) and parallel ($N_{seq}(T,K)$) programs. However, since the First Problem Size Law, as defined by Gustafson, is merely an approximation made under the assumption that the sequential program time $T_{seq}(N,K)$ is independent of N, which is not necessarily true, we will be measuring $Sizeup(T,K)$ under the Second Problem Size Law using the techniques outlined in [8].

4 Literature Survey and Additional Resources

As part of the graduate requirement, we will analyze [3], [4], and [5] in our literature survey. In addition, we will use the information and resources made available on the International SAT Competition web site [7], as well as the book by Drechsler et al. on advanced SAT solving techniques [6].

References

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

[2] Satisfiability: Suggested Format, 1993. Available online at `http://people.sc.fsu.edu/j̃burkardt/pdf/dimacs_cnf.pdf`. Accessed: 3/9/13.

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

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

[5] 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.

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

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

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