Using Sequential Runtime Distributions for the Parallel Speedup Prediction of SAT Local Search

Alejandro Arbelaez - Charlotte Truchet - Philippe Codognet

JFLI
University of Tokyo

LINA, UMR 6241
University of Nantes

JFLI – CNRS / UMPC
University of Tokyo

International Conference on Logic Programming (ICLP)

Aug 24-29, 2013
Motivation

- Not only multi-core machines, but massively parallel computers
- Massively parallel machines easily available
  - Supercomputers, grids and clouds
  - Amazon EC2 or Google Cloud: $ 0.05 p. core-hour
- Cost-benefit: Doubling computational resources to:
  - Double performance 😊
  - Improve 5% performance 😞
K Computer (RIKEN AICS, Kobe)

No.1 @ Top500 Jun+Nov 2011
8 petaflops
88,000 CPUs
each with 8 cores
i.e. total of 704,000 cores

from 04/2012
@ University of Tokyo
1 petaflop
4,800 nodes each with 16 cores
i.e. total of 76,800 cores
SAT

• Boolean Variables: Positive and Negative Literals

• Clauses

\[(\overline{X}_{1} \lor X_{2} \lor \overline{X}_{3}) \land (\overline{X}_{2} \lor X_{3} \lor X_{4}) \land (\overline{X}_{1} \lor \overline{X}_{5} \lor X_{3})\]
Local Search

• Start with a random configuration (values for the variables)
• Iteratively apply local moves in order to find a solution (flip one variable at a time)
Variable Selection in LS

• **GSAT**  [Selman et al. 1992]
  – Select the best variable (score function)

• **WalkSAT**  [Selman et al. 1994]
  – Select an UNSAT clause $C$
  – Select the best variable in $C$ (score function)

• **DLS**  [Hunter et al. 2002]
  – Adding weights to clauses
Local Search for SAT

• Multiple Random decisions
  – Initial starting point (assignment for variables)
  – Noise
  – Random tie-breaking (variable selection)

• The runtime is a random variable, characterized by the runtime distribution (RDT)

\[ F_Y(x) = P[Y \leq x] \]
Runtime Distributions (RTDs)

Probability Density Function (PDF)

Normal Distribution
Expected value = 10

\[ \mathbb{E}[Y] = \int_0^\infty t f_Y(t) dt \]
Runtime Distributions (RTDs)

Probability Density Function (PDF)

Cumulative Distribution Function (CDF)

Normal Distribution
Expected value = 10

\[ \mathbb{E}[Y] = \int_0^\infty t f_Y(t) dt \]

Empirical Runtime Distribution (RTD)

\[ \mathcal{F}_Y(x) = \mathbb{P}[Y \leq x] \]
Parallel Local Search for SAT

• Multiple flips
  – Flipping multiple variables at each iteration

• Parallel portfolio-based algorithm (this talk)
  – Algorithms compete and cooperate on the full problem
  – Use different and complementary strategies
  – No need of load balancing
Parallel Local Search for SAT

• Key questions to build the parallel portfolio:
  – What algorithm(s) should be used?
  – Multiple copies of the best one?
  – What’s the scalability of the algorithm?
  – Is it going to scale up to hundreds of cores?
SAT Challenge (2012)

Main Track: Random SAT

<table>
<thead>
<tr>
<th>Rank</th>
<th>RiG</th>
<th>Solver</th>
<th>#solved</th>
<th>%solved</th>
<th>cum. run-time</th>
<th>median run-time</th>
</tr>
</thead>
<tbody>
<tr>
<td>-</td>
<td>-</td>
<td>Virtual Best Solver (VBS)</td>
<td>558</td>
<td>93.0</td>
<td>72841</td>
<td>39.2</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>CCASat</td>
<td>423</td>
<td>70.5</td>
<td>76206</td>
<td>218.8</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>SATzilla2012 RAND</td>
<td>321</td>
<td>53.5</td>
<td>80796</td>
<td>714.4</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>SATzilla2012 ALL</td>
<td>306</td>
<td>51.0</td>
<td>83273</td>
<td>845.6</td>
</tr>
<tr>
<td>-</td>
<td>-</td>
<td>Sparrow2011 (SAT Competition 2011 Gold) (REFERENCE)</td>
<td>303</td>
<td>50.5</td>
<td>76396</td>
<td>876.1</td>
</tr>
<tr>
<td>-</td>
<td>-</td>
<td>EagleUP (SAT Competition 2011 Bronze) (REFERENCE)</td>
<td>283</td>
<td>47.2</td>
<td>83787</td>
<td>900.0</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>sattime2012</td>
<td>269</td>
<td>44.8</td>
<td>80345</td>
<td>900.0</td>
</tr>
</tbody>
</table>

- CCASat seems to be more robust than Sparrow
- Is CCASat also more robust than Sparrow in a parallel environment?
Parallel Local Search for SAT

• Can we estimate the performance of a given parallel local search algorithm?
  – Yes, analyzing the runtime distribution (RTD) of the sequential algorithm is known

• Order statistics: statistics of sorted random draws
  – First order statistics (or smallest order statistics)
  – Sample values placed in ascending order

\[ X_{(1)} \leq X_{(2)} \leq \ldots \leq X_{(n)} \]
Rationale of our Approach

• Run algorithm sequentially (several times)
• Analyze runtime behavior as a random variable (i.e., runtime distribution)
• Generic, and matches with known statistical distributions
  – Exponential, lognormal, Weibull, etc.
• Predict parallel performance
Notations

• Let $Y$ be the runtime of a Las Vegas algorithm (or the number of iteration of a LS algorithm)

• Cumulative distribution:
  \[ F_Y(x) = \mathbb{P}[Y \leq x] \]
  – Distribution (probability density function):
    \[ f_Y = F'_Y \]
    = derivative of cumulative distribution

• Expectation:
  \[ \mathbb{E}[Y] = \int_0^\infty tf_Y(t)dt \]
Parallel Algorithm

• $n$ copies of original algorithm launched in //
• Runtime of $i^{th}$ instance: $X_i$, follows $f_Y$
• Runtime of parallel algorithm: $Z^{(n)}$
• Cumulative distribution:

$$
\mathcal{F}_{Z^{(n)}} = \mathbb{P}[Z^{(n)} \leq x] \\
= \mathbb{P}\left[\exists i \in \{1...n\}, X_i \leq x\right] \\
= 1 - \mathbb{P}\left[\forall i \in \{1...n\}, X_i > x\right] \\
= 1 - \prod_{i=1}^{n} \mathbb{P}[X_i > x] \\
= 1 - (1 - \mathcal{F}_Y(x))^n
$$
Parallel Algorithm (cont.)

probability

Runtime Distribution parallel algorithm

Sequential Runtime Distribution (Normal Distribution)
Expectation & Speedup

• Expectation of parallel algorithm:

\[ \mathbb{E}[Z^{(n)}] = \int_0^\infty t f_{Z^{(n)}}(t) \, dt \]

\[ = n \int_0^\infty t f_Y(t) (1 - F_Y(t))^{n-1} \, dt \]

• Speedup:

\[ G_n = \frac{\mathbb{E}[Y]}{\mathbb{E}[Z^{(n)}]} \]

• Unfortunately, no general formula
• Depends on the distribution of Y
• NB: related to Order Statistics
Expectation & Speedup

• Expectation of parallel algorithm:

$$\mathbb{E}[Z^{(n)}] = \int_0^\infty t f_{Z^{(n)}}(t) dt$$

$$= n \int_0^\infty t f_Y(t)(1 - F_Y(t))^{n-1} dt$$

• Speedup:

$$G_n = \frac{\mathbb{E}[Y]}{\mathbb{E}[Z^{(n)}]}$$

• Unfortunately, no general formula

• Depends on the distribution of $Y$

• NB: related to

$$f_Y(t) = e^{\frac{(-\mu + \log(t))^2}{2\sigma^2}} \frac{1}{\sqrt{2\pi(t)\sigma}}$$

Lognormal distribution
Parallel Local Search

• Exponential distribution
  – Linear speedup for an unbounded number of cores
Parallel Local Search

• Exponential distribution
  – Linear speedup for an unbounded number of cores

• Shifted exponential distribution

\[ \mathbb{E}[Z^{(N)}] = x_0 + \frac{1}{n\lambda} \]
Parallel Local Search

- Exponential distribution
  - Linear speedup for an unbounded number of cores

- Shifted exponential distribution

- Lognormal distribution
  - May provide a super-linear speedup [Shylo et al. 2011]
Parallel Local Search for SAT

- How to estimate the runtime of a parallel local search algorithm:
  - Identify the RTD of the sequential algorithm
  - Identify the parameters of the closest theoretical distribution (e.g., shifted exponential, lognormal)
  - Apply order statistics to obtain the performance of the parallel portfolio
Parallel Local Search for SAT

Phase transition

Outside phase transition
Parallel Local Search for SAT

- Let’s go back to Sparrow and CCASat (random instances)

**Phase transition**
Lognormal distribution
Pass the kolmogorov-smirnov test

**Time in seconds**

<table>
<thead>
<tr>
<th>Cores</th>
<th>CCASat</th>
<th>Sparrow</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>382.0</td>
<td>781.5</td>
</tr>
<tr>
<td>Predicted</td>
<td>384</td>
<td>132.8</td>
</tr>
<tr>
<td>Actual</td>
<td>158</td>
<td>141.8</td>
</tr>
</tbody>
</table>

CCASat is better using 1 core
But, Sparrow is better using 384 cores
Parallel Local Search for SAT

• Let’s go back to Sparrow and CCASat (random instances)

![Graph showing speedup vs. number of cores]

<table>
<thead>
<tr>
<th>Cores</th>
<th>CCASat</th>
<th>Sparrow</th>
</tr>
</thead>
<tbody>
<tr>
<td>Predicted</td>
<td>384</td>
<td>92</td>
</tr>
<tr>
<td>Actual</td>
<td>67</td>
<td>51</td>
</tr>
</tbody>
</table>

Outside phase transition
Shifting exponential distribution
Pass the kolmogorov-smirnov test
Conclusions

• The parallel speedup of local search algorithms can be predicted using statistical methods
• The speedup varies from instance (+algorithm) to instance (+algorithm)
• Instances (+algorithms) from the same distribution report similar speedups
• The best sequential algorithm is not the best one in a parallel portfolio
• Thanks!
Parallel Local Search for SAT

• Shifted exponential distribution

\[ f_Y(t) = \begin{cases} 
0 & \text{if } t < x_0 \\
\lambda e^{-\lambda(t-x_0)} & \text{if } t > x_0 
\end{cases} \]

\[ \mathbb{E}[Y] = x_0 + 1/\lambda \]

\[ f_{Z(n)}(t) = \begin{cases} 
0 & \text{if } t < x_0 \\
n\lambda e^{-n\lambda(t-x_0)} & \text{if } t > x_0 
\end{cases} \]

• Expected runtime

\[ \mathbb{E}[Z^{(n)}] = n\lambda \int_{x_0}^{\infty} te^{-n\lambda(t-x_0)} dt = x_0 + \frac{1}{n\lambda} \]

\[ G_n = \frac{x_0 + \frac{1}{\lambda}}{x_0 + \frac{1}{n\lambda}} = 1 + \frac{n-1}{n\lambda x_0 + 1} \]
Parallel Local Search for SAT

- Lognormal distribution: the runtime is log-normally distributed

\[ f_Y(t) = e^{-\frac{(-\mu + \log(t))^2}{2\sigma^2}} \frac{1}{\sqrt{2\pi(t)\sigma}} \]

- Computing the expected runtime is difficult and usually a numerical method is required