A Parallel Framework for NP Combinatorial Optimization Problems

RIT Department of Computer Science Master’s Project
Author – Michael Adams
Chair – Professor Alan Kaminsky
Reader – Professor Stanisław Radziszowski
Observer – Professor Leon Reznik

May 10th, 2013

Agenda

• Work Synopsis
• Hypotheses
• Related Work
• Description of Solution Strategies
• Description of NP Problems
• Framework Overview
• Experiments
• Results and Analysis
• Future Work
• Lessons Learned
• Conclusion
Agenda

• Work Synopsis
• Hypotheses
• Related Work
• Description of Solution Strategies
• Description of NP Problems
• Framework Overview
• Experiments
• Results and Analysis
• Future Work
• Lessons Learned
• Conclusion

Work Synopsis

• Generic framework designed and developed
• Support for both sequential and parallel execution
• Support for three different solution strategies
• Three NP problems chosen and implemented both with and without framework
• Timing and quality experiments designed and run for each problem using each solution strategy
• Analysis of the results of the experiments
Hypotheses

- Implementation time using the framework will be reduced compared to not using the framework
- Brute force execution time will be reduced as the number of processors increases
- Hill climbing execution time will be reduced (primary) and solution quality will be increased (secondary) as the number of processors increases
- Simulated annealing solution quality will increase as the number of processors increases
Agenda

• Work Synopsis
• Hypotheses
• Related Work
  • Description of Solution Strategies
  • Description of NP Problems
  • Framework Overview
• Experiments
• Results and Analysis
• Future Work
• Lessons Learned
• Conclusion

Related Work

• NP-Opt
  • Also developed in Java
  • Less flexible
  • Does not support parallel execution
  • Provides a GUI

• OptFrame
  • Developed in C++
  • Similar, but different terminology
  • “Simple” versions of solution strategies implemented

• Others exist, but mainly focus on the heuristics and metaheuristics themselves
Agenda

- Work Synopsis
- Hypotheses
- Related Work
- Description of Solution Strategies
- Description of NP Problems
- Framework Overview
- Experiments
- Results and Analysis
- Future Work
- Lessons Learned
- Conclusion

Brute Force

- Sequential – one thread examines all possible solutions, recording the best solution found
- Parallel – solution space is divided amongst available processors. Each thread reduces its local best solution into a global best solution
Hill Climbing

- Sequential – one thread starts at a random solution and generates mutants, transitioning to higher quality mutants
- Parallel – each thread starts with the same random solution, but generates different mutants, transitioning to higher quality mutants

Simulated Annealing

- Sequential – one thread starts at a random solution, generating mutants. Lower quality mutants are transitioned to based on an acceptance probability
- Parallel – each thread starts at its own unique random solution. Each thread then performs the sequential version
Agenda

• Work Synopsis
• Hypotheses
• Related Work
• Description of Solution Strategies
• Description of NP Problems
• Framework Overview
• Experiments
• Results and Analysis
• Future Work
• Lessons Learned
• Conclusion

0-1 Knapsack Problem

• Given list $Q$ of $N$ items, each with weight $W_i$ and value $V_i$, as well as a bag of weight capacity $M$, find values for $X_i$ such that

$$\sum_{i=0}^{N-1} X_i \cdot W_i < M$$

AND

$$\sum_{i=0}^{N-1} X_i \cdot V_i \text{ is maximized}$$

Maximum Satisfiability

- Given a Boolean expression $E$ in conjunctive normal form, find the true/false assignment of literals such that the number of clauses satisfied is maximized

$$E = (x_1 \lor \neg x_2 \lor \neg x_3) \land (x_1 \lor x_2 \lor x_4)$$

Minimum Vertex Cover

- Given a graph $G$, with a set of vertices $V$, find the smallest subset of $V$ such that all edges in $G$ are incident to at least one vertex in the subset
Agenda

- Work Synopsis
- Hypotheses
- Related Work
- Description of Solution Strategies
- Description of NP Problems
- Framework Overview
- Experiments
- Results and Analysis
- Future Work
- Lessons Learned
- Conclusion

Framework Overview

- Some concepts of these solution strategies and problems can be abstracted
- An assignment
- Generating and mutating assignments
- An evaluation of an assignment
- A comparison between evaluations
Framework Diagram

- User must subclass Assignment, Evaluation, NPOptProblem, ExhaustiveGenerator, and MutationGenerator
- User may optionally subclass SolutionStrategy (to add a new strategy) or CoolingSchedule (to add a new cooling schedule)

Agenda
- Work Synopsis
- Hypotheses
- Related Work
- Description of Solution Strategies
- Description of NP Problems
- Framework Overview
- Experiments
- Results and Analysis
- Future Work
- Lessons Learned
- Conclusion
Experiments

- All experiments were run on the tardis hybrid cluster (two 2.6GHz dual core processors per backend node)
- All experiments were tested sequentially and in parallel with 1, 2, 3, and 4 threads. All experiments were run three times to account for timing variance
- Brute force – smallest test ~1 minute, longest test ~1 hour
- Hill climbing
  - Small problem size – same problem sizes as brute force
  - Large problem size – smallest test ~1 minute, longest test ~1 hour
- Simulated annealing
  - Small problem size – same problem sizes as brute force
  - Large problem size – same problem sizes as hill climbing large problem sizes

Agenda

- Work Synopsis
- Hypotheses
- Related Work
- Description of Solution Strategies
- Description of NP Problems
- Framework Overview
- Experiments
- Results and Analysis
- Future Work
- Lessons Learned
- Conclusion
Implementation Time

- Using the framework to implement a problem resulted in halving the implementation time
- Results were rounded to the nearest 15 minutes to account for interruptions

Knapsack Brute Force

- General reduction in running time
- Speedups mostly below or above ideal
MAXSAT Brute Force

- General reduction in running time
- All speedups with two or more processors below ideal

MVC Brute Force

- General reduction in running time, with some variance
- More variance in speedups, but generally below ideal
Brute Force Results

- In most cases, there was a reduction in running time. This resulted in the speedups shown previously.
- Speedups were often significantly above or below the ideal.
- Results skewed by the “one lucky timing”
  - Possibly JDK issues
  - Possibly system administrators working on JDK 1.7 issue
  - Not all experiments happened to have “one lucky timing”
- Overall, enough evidence to support the hypothesis.

Knapsack Hill Climbing (S)

- General reduction in running time
- No improvements in solution quality
MAXSAT Hill Climbing (S)

- General reduction in running time
- Many cases showed slight improvements in solution quality

MVC Hill Climbing (S)

- Running time had a large amount of variance
- Many solutions improved with a second processor
Hill Climbing Results (S)

- Knapsack and MAXSAT showed significant reductions in running time. MVC had greater variance.
- Except for Knapsack, solution quality benefited from additional processors in many cases.
- Results are again skewed by “one lucky timing”
  - Knapsack and MAXSAT seemed to get the lucky timings
  - MVC had many cases where there was no lucky timing
- Overall, enough evidence to support primary hypothesis. MAXSAT and MVC also provided some support for the secondary hypothesis.

Knapsack Hill Climbing (L)

- General reduction in running time
- Very small improvement in some cases with a third processor
MAXSAT Hill Climbing (L)

- General reduction in running time
- No improvements in solution quality

MVC Hill Climbing (L)

- There is again some variance in the running time
- Solution quality actually decreased with a fourth processor in one case
Hill Climbing Results (L)

- The reduction in running time was fairly consistent between the three problems. There was much less variance in MVC with larger problems.
- The improvements to solution quality were fewer and less dramatic.
- Decrease in solution quality in MVC case is due to how the parallel algorithm is designed.
- Overall, the results provided evidence in support of the primary hypothesis. The secondary hypothesis was contradicted in one case. This can likely be the case with small parameter sizes. Secondary hypothesis is problem dependent.

Knapsack Simulated Annealing (S)

- Very small running time with very little variance.
- Many cases showed an increase in solution quality.
MAXSAT Simulated Annealing (S)

- Slightly more variance in running time, but overall short durations
- Very slight improvements to solution quality in a few cases

MVC Simulated Annealing (S)

- Some variance in running time, but overall short duration and small spread
- Solution quality improved in two cases
Simulated Annealing Results (S)

- In general, there was little variance in running times. In addition, running times are very small compared to hill climbing. The variance is due to the nature of the simulated annealing algorithm.
- Every problem had at least one case where solution quality was improved with additional processors.
- Overall, there is enough evidence to support the hypothesis.

Knapsack Simulated Annealing (L)

- Very small running time with very little variance.
- Solution quality improved in a few cases.
MAXSAT Simulated Annealing (L)

- Some variation in running time, but overall small spread
- In all cases, a second processor provided a slight improvement in quality

MVC Simulated Annealing (L)

- Very strange variance, especially in the case of two processors
- All cases showed an improvement in solution quality with two processors
Simulated Annealing Results (L)

- Except for MVC, there was an acceptable amount of variation in running time
- Again, all problems had at least one case where solution quality improved with additional processors
- The large spike in running time with two processors is again due to the “one lucky timing”. The one data point that did not spike happened to have one timing that was ~150 seconds shorter than the other two timings
- Overall, there is strong evidence to support the hypothesis

Agenda

- Work Synopsis
- Hypotheses
- Related Work
  - Description of Solution Strategies
  - Description of NP Problems
- Framework Overview
- Experiments
- Results and Analysis
- Future Work
- Lessons Learned
- Conclusion
Future Work

- Implement cluster and hybrid parallelism
  - Currently only supports shared memory multiprocessor parallelism
  - Distributed memory parallelism would be beneficial

- Implement additional solution strategies
  - Genetic algorithms and Tabu search are of most interest

- Test larger problem sizes or known benchmark examples
  - The large problem size experiments were somewhat small compared to known benchmark problems

Agenda

- Work Synopsis
- Hypotheses
- Related Work
  - Description of Solution Strategies
  - Description of NP Problems
- Framework Overview
- Experiments
- Results and Analysis
- Future Work
- Lessons Learned
- Conclusion
Lessons Learned

- Self motivation
  - No deadlines, no authoritative consequences for procrastination, no periodic grading
  - I had to learn to motivate myself to work on the project. Easy during some phases (implementation), less so during others (report writing)

- Proper planning
  - Failed to perform enough preliminary testing to catch a bug
  - Failed to schedule time for bug fixing
  - Almost caused me to extend schedule

Agenda

- Work Synopsis
- Hypotheses
- Related Work
- Description of Solution Strategies
- Description of NP Problems
- Framework Overview
- Experiments
- Results and Analysis
- Future Work
- Lessons Learned
- Conclusion
Conclusion

- Using the framework to implement a problem provided a significant reduction in implementation time. In addition, it does not require the user to understand the solution strategies or parallel programming.
- Brute force results showed enough support to confirm the reduction in execution time hypothesis.
- Hill climbing results confirmed the primary hypothesis, but the secondary hypothesis can be dependent on the problem.
- Simulated annealing results showed enough support to confirm the increase in solution quality hypothesis.
- I believe the variance in results was due to JDK issues or possibly executing during maintenance by the system administrators.

References


Questions