Contribution ID: **104** Type: **sectional reports**

Customizable system for coarse-grained parallel schemes in branch-and-bound algorithm for discrete optimization

Tuesday, 1 July 2014 17:50 (20 minutes)

One of the main methods for solving discrete optimization problems is branch-and-bound. A branch-and-bound algorithm consists of a systematic enumeration of all candidate solutions, where large subsets of fruitless candidates are discarded en masse, by using upper and lower estimated bounds of the quantity being optimized. One of the ways to speed up an algorithm is to parallelize it. In case of branch-and-bound it is can be achieved on lower level by parallelizing some parts of the algorithm, for example computing lower bounds concurrently. This would be fine-grained parallelization. On the other hand, coarse-grained parallelization can be made by running a set of branch-and-bound algorithms with different parameters concurrently. Or by dividing the problem on a set of subproblems e.g. by fixing some of integer variables, and solving them in parallel.

In this study we examine the coarse-grained approach to parallelization implemented using existing mixed integer programming solver Cbc [1]. Cbc (Coin-or branch and cut) is an open-source mixed integer programming solver written in C++. It can be used as a callable library or using a stand-alone executable. In our approach we have the problem to be solved in the form of an AMPL stub [2]. The initial problem is divided on a set of subproblems by fixing some of integer variables by adding additional constraints to the stub. As a result we have 2^N AMPL stubs where N is the number of fixed variables. Next, every subproblem's stub is passed to CBC. At this step a simple control system written in Erlang [3] orchestrates the solution process by running CBC solver instances on a number of remote machines. Each CBC instance solves a single subproblem. Every time a solver finds a new incumbent, it is passed to the control system which broadcasts it to other instances. After receiving a new incumbent from the outside, solver uses it to discard search tree nodes during branch-and-bound more efficiently. Finally, after all subproblems were solved, control systems returns the best incumbent which is the solution of the whole problem. If all subproblems were proved to be infeasible, then the whole problem is infeasible too.

The system was tested on the traveling salesman problem and a noticeable speedup was shown. Furthermore there is a multi-threaded version of CBC solver which crashes on some of the problems which are solved successfully by our parallel system. However even a single-threaded SCIP solver [4] was performing better than the parallel system presented in the study. Therefore it may be useful to also integrate SCIP in the system. Another possibility for system's customization may be starting solvers with different configuration parameters which may significantly change the solver's behavior.

[1] J. Forrest, and R. Lougee-Heimer, CBC user guide, INFORMS Tutorials in Operations Research, 2005, pp. 257–277.

[2] R. Fourer, D.M. Gay, and B.W. Kernighan, AMPL: A Modeling Language for Mathematical Programming, second edition, Duxbury Press / Brooks/Cole Publishing Company, 2002.

[3] F. Cesarini, and S. Thompson, Erlang Programming, O'Reilly Media, Inc., 2009.

[4] T. Achterberg, SCIP: solving constraint integer programs, Mathematical Programming Computation, volume 1, number 1, 2009, pp. 1–41.

Primary author: Mr SMIRNOV, Sergey (Institute for Information Transmission Problems of the Russian Academy of Sciences)

Presenter: Mr SMIRNOV, Sergey (Institute for Information Transmission Problems of the Russian Academy of Sciences)

Session Classification: Section 8 - Optimization problems and distributed computing

Track Classification: Section 8 - Optimization problems and distributed computing