Applied Soft Computing 24 (2014) 385–405

Contents lists available at ScienceDirect

Applied Soft Computing

journal homepage: www.elsevier.com/locate/asoc

Solving difficult mixed integer and disjunctive non-linear problems on single and parallel processors

Ralf Östermark*

School of Business and Economics at Åbo Akademi University, Finland

ARTICLE INFO

Article history:
Received 25 March 2013
Received in revised form 15 July 2014
Accepted 16 July 2014
Available online 25 July 2014

Keywords:
Mixed-integer non-linear optimization
General disjunctive programming
Parallel processing

ABSTRACT

Difficult non-linear, mixed integer non-linear and general disjunctive programming (NLP/MINLP/GDP) problems are considered in the present study. We show using random relaxation procedures that sampling over a subset of the integer variables and parallel processing have potential to simplify large scale MINLP/GDP problems and hence to solve them faster and more reliably than conventional approaches on single processors. Gray coding can be utilized to assign problems to the local processors. Some efficient non-linear solvers have been connected to the Genetic Hybrid Algorithm (GHA) through a switch board *minlp_machine*(), monitoring a vector of caller functions. The switch board is tested on single processor computers and massively parallel supercomputers in a sample of optimization problems. A new line search algorithm based on multi-sectioning, i.e. repeated bi-sectioning of parallel threads is applied to a set of irregular NLP-problems. Simulations indicate that step search based on multi-sectioning and re-centering is robust. Time recording indicates that the step search procedure is fast also in large optimization problems. Parallel processing with Gray coding and random sampling over a subset of the integer variables improves the performance of the sequential quadratic programming algorithm with multi-sectioning in large-scale problems. An early mesh interrupt guarantees load balancing in regular problems. General disjunctive programming (GDP) problems can be simplified through parallel processing. Certain problems are solved in larger scale than reported previously. Function pointer logic and intelligent switching procedures provide a fruitful basis for solving high-order GDP problems, where the computational challenge in direct MINLP-formulations would be insurmountable.

Introduction

To date, several efficient solvers for mixed-integer non-linear optimization problems have been proposed and compared in the literature (cf., e.g. [1–8]). The underlying algorithms are based on sequential quadratic programming (SQP), sequential linear programming (SLP) or interior point techniques. Active-set SQP methods and interior point methods are currently considered the most powerful algorithms for large-scale non-linear programming [9]. In non-convex or irregular problems, the algorithms cannot guarantee the global solution. However, the established algorithms mostly yield at least a feasible MINLP-solution also in difficult problems (cf. [10–13]). Certain non-smooth problems can be reformulated as smooth optimization problems, but in general a methodology for non-differentiable functions is required for non-smooth optimization (cf. [14,15]). Semi-definite programming algorithms have been quite successful in solving quadratic assignment (QAP) problems [16]. Integrated systems, where artificial or swarm intelligence is connected to mathematical programming methodology provide a powerful basis for difficult irregular optimization problems (cf. [17–20]). We will subsequently study the performance of some high-performance mathematical programming algorithms connected as support libraries to the Genetic Hybrid Algorithm (GHA). The principal working of the programming platform GHA is summarized in Appendix A.

We consider the general mixed-integer non-linear constrained optimization problem (MINLP):

$$P: \left\{ \begin{array}{l} \min f(\boldsymbol{x},\boldsymbol{y}) | g_i(\boldsymbol{x},\boldsymbol{y}) \geq 0,\ i \in I,\ g_i(\boldsymbol{x},\boldsymbol{y}) = 0,\ i \in E, \\ \{\boldsymbol{x},\boldsymbol{y}\} \in [\boldsymbol{LB},\boldsymbol{UB}],\ \boldsymbol{g}(\boldsymbol{x},\boldsymbol{y}) \in \Re^m,\ \boldsymbol{x} \in \Re^{n_x},\ \boldsymbol{y} \in Y^{n_y} \end{array} \right\} \quad (1.1)$$

E and I denote the sets of equality and inequality constraints and $\{\Re^{n_x}, Y^{n_y}\}$ the real- and discrete-valued sub-spaces respectively (the Lagrange equation for the continuous relaxation of (1.1) is presented in Appendix B). Depending on $\{n_x, n_y\}$ and the functional form of the system, problem P is solved as a continuous or

* Tel.: +358 405205137.
E-mail address: rosterma@abo.fi

http://dx.doi.org/10.1016/j.asoc.2014.07.018
1568-4946/

mixed-integer linear (MI)LP or non-linear (MI)NLP problem. If some of the functions in (1.1) are non-differentiable at a local solution, the derivatives can be approximated as one- or two-way differences in the corresponding interface functions as discussed subsequently.

If $n_y > 0$ and if the problem is regular and not huge, P can be solved by a branch-and-bound strategy: a currently non-integer variable j is selected from Y using a specified criterion and two separate sub-problems are generated with upper bound $floor(y_j)$ and lower bound $ceil(y_j)$ respectively. The sub-problems corresponding to these branches are solved as continuous-valued non-linear optimization problems using efficient NLP-solvers. An advantage of a systematic branch-and-bound search is that sub-trees can be eliminated or *fathomed* at an early stage in certain situations:

- A sub-problem is infeasible. In this case all sub-problems obtained by branching from this node are infeasible as well.
- The sub-problem yields a feasible mixed-integer solution. Further branching is interrupted and the current upper bound for the MINLP-problem updated if necessary.
- The solution of the sub-problem is higher than the current upper bound; hence further branching from the node is interrupted.

In MINLP-problems where the discrete search space is huge, the emerging branch-and-bound tree may become computationally prohibitive. In such problems, random sampling over a subset of the integer variables can be applied as discussed below (cf. the random relaxation procedure 6.1 and its extension 6.2).

In block-separable MINLP-problems, each processor or cluster of processors solves the local sub-problem and delivers the solution directly or through the cluster root to the main root for assembly based on the node id (cf. [21]). Sampling over a subset of the integer variables and parallel processing have potential to simplify large scale MINLPs. If the mesh is divided into clusters, then each cluster can distribute the local block problem to the processors, for example using the Gray code procedure [22] presented below.

If the problem is regular, i.e. the variation in solution time for the different NLP-relaxations derived by Gray coding is relatively low, load balancing is achieved through an early mesh interrupt. In this case, the processor that has detected the global solution will broad cast an early mesh interrupt signal asynchronously. The incoming signal is checked by the receiving nodes in critical places of the MINLP-algorithm. If the complexity of the individual NLP-relaxations is highly variable, the job list can be adjusted according to the solution time accumulating for each problem solved by the processors. In practice, idle processors can be assigned jobs as available from the job lists of busy processors. GHA contains the communication resources for reassignment and fast communication between individual processors, to be applied if load balancing requires adjustments of the job list ([23,24]. cf. [25,26]). In an irregular problem, where the solution time for individual NLP-realizations varies considerably and all jobs are being processed, load imbalance accumulates for each new processor becoming idle. However, the computational complexity of the MINLP-problem is always higher than that of the relaxed NLP-problems determined and assigned through Gray coding. Solving a concurrent set of independent NLPs that guarantee the global MINLP-solution is therefore preferable to tackling the MINLP-problem using branch-and-bound. If the number of NLP- or reduced MINLP-relaxations exceeds the number of parallel processors, the emerging job list can be assigned to the processors and adjusted as explained above.

Some efficient non-linear solvers have been connected to the Genetic Hybrid Algorithm (GHA) through a switch board (vector of caller functions) and thoroughly tested on single processor computers and massively parallel supercomputers: DNCONG [27], FSQP [28], NLPQLP [29] and SNOPT [30]. The source code for these support libraries has been provided by the corresponding authors. A Linux version of the ILOG Cplex optimization studio downloaded from the Academic Initiative web site of IBM was installed on the Sisu supercomputer at the Center of Scientific Computing (CSC) in Helsinki. This is the first Cray XC30 server in use in Europe. Cplex is connected to GHA through an interface function included in the switch board. GHA can be connected to Matlab applications through the MCC compiler of Mathworks.

The choice between alternative algorithms can be automated with GHA using a parallel setting [18,19]. An SQP-algorithm applying a new multi-section step search procedure with re-centering is tested. The local quadratic problems are solved by the QL-solver of Schittkowski [31], applying sparse matrix algebra and an active set strategy.

Some quadratic, cubic [32], heuristic and extended bi-section [33] methods for step search are connected to the switch board. Other efficient solvers e.g. CONOPT [34], IPOPT ([35], or KNITRO-AMPL/MATLAB®) can be connected to GHA through the switch board. For preliminary assessment of the code, a modest number of difficult NLP- and MINLP-problems have been solved. Limited comparisons to results obtained by the algorithms FILTER [36], KNITRO, LOQO [37], MINOS [38], NLPQLP [29] and SNOPT [30] are provided. Initial testing has been conducted both on single processor main frame computers and on massively parallel supercomputers. The results are encouraging for extensive comparisons in future research.

GHA has been developed since the late nineties (cf. e.g. [39–41]). The algorithm has been subjected to rigorous scalability tests and is one of the few algorithms allowed to use all available cores on the massively parallel Cray XT and Cray XC30 supercomputers at the Center of Scientific Computing (CSC) in Helsinki for production runs

(http://www.csc.fi/english/research/Computing_services/computing/servers/louhi_scalability). These computers belong to the group of the top 500 parallel supercomputers in the world (http://www.top500.org/). During the time period 09-10/2011, GHA was tested in a project arranged within the Partnership for Advanced Computing in Europe (PRACE) on the Jugene supercomputer at Jülich in Germany. The calculating power of Jugene is one Petaflop/s or one trillion calculation operations per second, the currently fastest supercomputer in Europe. Scalability was demonstrated with 65536 parallel processors as made available for the test. The algorithm does not restrict the maximum number of processors. This is a topic determined by the computational problem at issue. The intrinsic features of the algorithm allow suppression of processor level output and minimizing the inter-nodal communication when using huge mesh sizes. The support of flexible clustering, node level independence and problem dependent communication allow the design of systems utilizing swarm intelligence [42,43]. An electronic User's Guide for GHA and its support libraries can be downloaded from http://www.abo.fi/fak/esf/gha/research/geno_mathematics/presentation/download_gha_guide.php.

The generic algorithm is presented along with concrete code and examples in the electronic User's Guide. The guide contains numerous cases and examples helping the researcher to conduct comparative studies on the computers where the platform is installed. A great number of hyperlinks in the guide gives access to test problems, their idea and exact C code. GHA and its solvers can be accessed and tested on the CrayXC30 supercomputer at CSC using numerical problems of the researcher. The guide illustrates how special-purpose algorithms can be connected to the system as found necessary.

We have demonstrated scalability of GHA and its MINLP-switchboard on the Cray XC30 server at CSC – the first in production in Europe – in a set of non-convex MINLP-problems up to the maximum number of processors available for testing during 2013. The