Solving NLP Models in a Branch & Bound Context

by
Arne Stolbjerg Drud
ARKI Consulting & Development A/S
Bagsvaerdvej 246 A, DK-2880 Bagsvaerd
Denmark
adrud@arki.dk

Overview:

- The Branch and Bound Environment
  - SBB
- The Components of an NLP solver
  - What is critical for B&B?
- A new SQP Component in CONOPT3
  - Combining SQP and GRG Algorithms
- Some Results
  - CONOPT2 vs. CONOPT3 for some general NLP models
  - CONOPT2 vs. CONOPT3 for some MINLPs

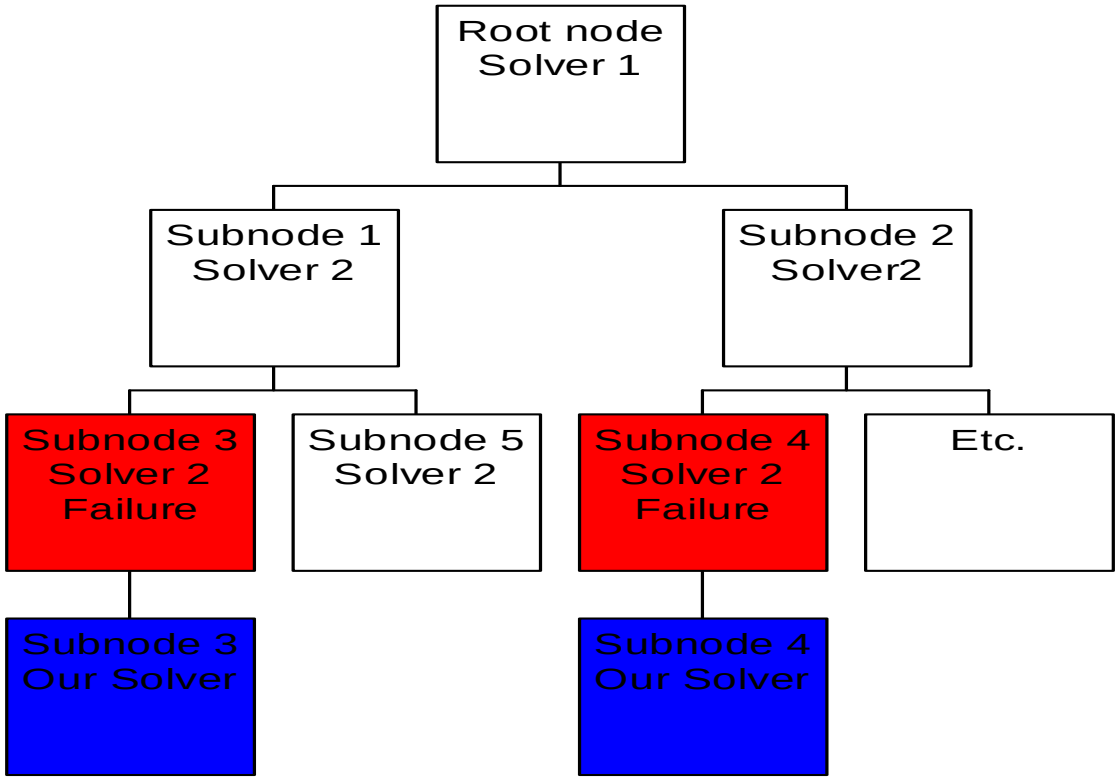
The Branch and Bound Environment: The SBB Perspective:

- **NLP Solvers are called as independent programs to solve the Root-node and Sub-nodes**
- **SBB manages the search tree**
- **Solvers appear to be identical: needs the same input and returns the same output**
- **Different solvers can be used for different nodes**
- **Failure of a solver can be repaired by calling one or more alternative solver**

The Branch and Bound Environment: The Solver Perspective:

- **The Root node is like a standard NLP**
- **A sub-node is a model with a solution from the parent node. This solution is, apart from a few violated bounds, both primal and dual feasible**
- **Communication of initial values and bounds is always via SBB**
- **The parent node may have been solved by another solver**
- **The first call of a solver can appear deep in the tree**
- **The node previously solved by the same solver may have little relation to the current node**

A Branch and Bound Tree



The Components of an NLP solver (Algorithmic)

- **Interface to the Modeling System: Read Model and Write Solution**
- **Model Preparation and Setup**
- **Preprocessing**
- **Various Optimization Components. For CONOPT2 these include:**
  - **Crash procedure - initial basis selection**
  - **Phase 0: cheap Newton-based method used to get close to a feasible solution**
  - **Phase 1: reliable method to find a feasible solution**
  - **Phase 2: optimization through sequence of feasible points**

The Components of an NLP solver (Model Modes)

The optimization components for Phase 1 and 2 exist in different versions selected from statistics about properties of the model around the current point:

- **Linear Mode: The objective appear to be almost linear over large distances (compared to distances to next bound):**
  - **Use Sequential Linear Programming (SLP) methodology to identify many active constraints if large steps are possible**
  - **Use steepest descend (SD) if feasibility cannot be maintained for large steps**

- **Nonlinear Mode: Model has significant curvature in the reduced objective.**
  - **Use Quasi-Newton methods (QN) to estimate second order information**

The Components of an NLP solver (Resources)

The distribution of resource usage in different components depends on the model and its behavior around the optimum:

- **LINEAR: The optimum is mainly defined by bounds / constraints:**
  - **There are few superbasic variables**
  - **The problems has combinatorial aspects**

- **CURVED: The optimum is mainly defined by the curvature of the objective and constraints.**
  - **There are many superbasic variables**
  - **The curvature of the objective and constraints is important**

- **MIXED: The optimum is defined both by curvature of the objective and by bounds / constraints.**
  - **There are many superbasic variables**
  - **The curvature of the objective and constraints is important**
  - **The problems has combinatorial aspects**

The Components of an NLP solver (Resources)

The components and resources needed to solve a standard NLP (or B&B Root node) for different model types:

	LINEAR	CURVED	MIXED
Setup	Fixed work	Fixed work	Fixed work
Phase 0/1	Mainly SLP	Mainly SLP SD/QN to move through difficult areas	Mainly SLP SD/QN to move through difficult areas
Phase 2	Mainly SLP	Mainly QN with few 'subspaces'	Initial SLP Final QN with many 'subspaces'
Resources (CONOPT2)	All parts are fast	Phase 1 is fast Phase 2 is medium per subspace	Phase 1 is fast Phase 2 can be very slow if many subspaces

The Components of an NLP solver (Resources)

The components and resources needed to solve a sub-node for different model types:

	LINEAR	CURVED	MIXED
Setup	Fixed work	Fixed work	Fixed work
Phase 0/1	Newton or SLP with few iterations	Newton or SLP with few iterations	Newton or SLP with few iterations
Phase 2	Mainly SLP with few iterations	QN with very few 'subspaces'	QN with several 'subspaces'
Resources (CONOPT2)	All parts are very fast	Phase 1 is very fast Phase 2 is medium	Phase 1 is very fast Phase 2 can be slow

The expected work is in all cases less than for a Root node solve.
Setup costs are fixed and become a larger percentage of overall cost.

Improving an NLP solver for Branch and Bound

Potential Improvements:

- **Improve Setup: Reuse some preparation work. Usually not worth while because creation is faster than communication.**
- **Reuse Preprocessing: Difficult because it depends heavily on the changing bounds.**
- **Improve Initial Basis. Current routine is conservative to protect against unknown model changes. Should add a less conservative option.**
- **Utilize almost feasible initial point by adding a Combined Phase 1+2 component based on a dual approach. Initial experiments have been disappointing.**
- **Improve General Optimization Components. Best bet is QN -> SQP**

The new SQP Component in CONOPT3

The "active subspace" is defined via basic, superbasic, and nonbasic variables:

Basis variables: Adjust to satisfy constraints. Basis defines dual variables	**Superbasic variables: Adjust to improve objective**	**Nonbasic variables: Keep fixed - they appear to be optimal**

The new SQP Component in CONOPT3 (cont.)

Key task for a given subspace: Change the superbasic variables to their "optimal values". With Quasi-Newton this involves:

- **QN uses gradient combined with a gradually updated second order approximation to the "Reduced Objective" to define search directions.**
- **During line searches, basic variables are adjusted to maintain feasibility.**
- **Several line searches are often needed to get good second order information.**
- **If a bound on a superbasic variable becomes active, the variable is made nonbasic (is fixed) => new subspace.**
- **If a bound on a basic variable becomes active, a basis change is made => new subspace. The second order information is projected (with some loss of quality).**

Costly components: The large number of line searches per subspace and the line searches themselves (including new Jacobian and basis matrices).

The new SQP Component in CONOPT3 (cont.)

The advantages of an SQP approach:

- **Can use exact second order information**
- **Search directions defined inside the SQP will either end at a bound => new subspace is defined, or at local optimum**
- **Only line search with good directions and expected step = 1**

The challenges of an SQP approach:

- **Second order matrix must be projected (reduced) on the superbasic space and for some algorithms on the nonbasic space.**
- **Reduced second order matrix must be adjusted after basis changes.**
- **Reduced second order matrix can be very large. Alternative: Conjugate Gradient methods with or without preconditioning.**

The new SQP Component in CONOPT3 (cont.)

The practical implementation:

- **Exact second order information is delivered by new GAMS routines.**
- **The QP is only defined on the Basic / Superbasic space.**
- **If a bound becomes active, the size of the QP search space is reduced. The space is only increased between SQPs / Major iterations.**
- **The QP is solved with a reduced gradient method (standard CONOPT components).**
- **The Reduced second order matrix:**
  - **Is only stored if the dimension of the superbasis is limited.**
  - **Is not computed via an expensive projection / reduction but updated as needed using QN methods. Matrix from last QP is reused.**
- **Models with many superbasics (and second order matrix) use Conjugate Gradients (CG) or Scaled CG depending on termination criteria.**
- **Early termination based on sufficient progress and statistics for previous iterations (Idea: do not waist time finding an optimal solution if we are far away and the QP approximation is poor).**

CONOPT2 vs. CONOPT3 for some NLP models

Example 1: A version of Alan Manne's RUNS model

The model has 4145 variables and 3428 constraints
The Jacobian has 14905 Jacobian elements, 263 of which are nonlinear.
The Hessian of the Lagrangian has 263 elements on the diagonal, 135 elements below the diagonal, and there are 263 nonlinear variables.

MIXED type model	CONOPT2	CONOPT3
Easy case: First Solve	7590 iterations 284 seconds	514 iterations 63 seconds
Easy case Second Solve	930 iterations 41 seconds	54 iterations 6 seconds
Hard case: First solve	41532 iterations 3109 seconds	997 iterations 414 seconds
Hard case: Second solve	1813 seconds 176 seconds	79 iterations 44 seconds

CONOPT2 vs. CONOPT3 for some NLP models

Example 2: An entropy maximization model of Sherman Robinson (SAM Balancing)

The model has 5048 variables and 2947 constraints
The Jacobian has 19275 elements, 6631 of which are nonlinear.
The Hessian of the Lagrangian has 2657 elements on the diagonal, 1987 elements below the diagonal, and there are 2790 nonlinear variables.

The model has around 2100 superbasic variables and CONOPT2 could not solve it properly (terminated on slow convergence).
CONOPT3 used CG, scaled with the diagonal of the reduced Hessian.

MIXED/CURVED type model	CONOPT2	CONOPT3
Base case	19527 iterations 13758 seconds	183 iterations 56 seconds

CONOPT2 vs. CONOPT3 for some MINLP models

The basis is the MINLPLib collection at http://www.gamsworld.org/minlp/minlplib.htm

A comparison is difficult as seen from some examples:

Model 4STUFEN (150 variables, 99 equations, and 48 discrete variables):

- **CONOPT2 and CONOPT3 have the same objective in the root node.**
- **CONOPT2 has 9 integer infeasibilities in the root and CONOPT3 has 25.**
- **SBB/CONOPT2 solves the model in 943 nodes and 48.84 seconds for the NLPs.**
- **SBB/CONOPT3 stops on time limit after 10097 nodes and 600 seconds for the NLPs without satisfying the optimality conditions.**
- **Time per node is 51.8 msec for CONOPT2 and 59.4 msec for CONOPT3.**
- **SBB/CONOPT3 with pseudo-costs solves the model in 370 nodes and 23.26 seconds including 4.68 seconds to initialize pseudo-costs. Time per node is 50.2 msec.**

CONOPT2 vs. CONOPT3 for some MINLP models

Other examples:

Model WASTE (1992 equations, 2485 variables, and 400 discrete variables):

- **CONOPT2 determines that the root is infeasible**
- **CONOPT3 solves the root, finds an integer solution, but does not satisfy the optimality criteria within 600 seconds.**

Model SPACE960 (6498 equations, 5538 variables, and 960 discrete variables):

- **CONOPT2 cannot solve the root within the selected limit of 600 seconds.**
- **CONOPT3 solves the root + 29 sub-nodes in 600 seconds.**

CONOPT2 vs. CONOPT3 for some MINLP models

The summary statistic for performance is based on:

- **137 models with 107432 equations (770 average), 130423 variables (951 average), and 57329 discrete variables (418 average).**
- **T = sum of Total time spend to solve each model, max 600 seconds.**
- **N = sum of Total number of nodes used to solve each model, max 10000 nodes per solve. Root nodes are counted as 1.**
- **NE = sum of Total number of nodes used to solve each model (max 10000) multiplied by the number of equations is used to give large nodes higher weight.**

CONOPT2 created N = 140627 nodes in T = 20256 seconds (5.6 hours). This gives 0.144 seconds per node. NE = 25.9 mill and T / NE = 7.8 10^{-4} seconds per node*equation.

CONOPT3 created N = 208355 nodes in T = 20794 seconds. This gives 0.100 seconds per node. NE = 39.6 mill and T / NE = 5.2 10^{-4} seconds per node*equation.

Conclusions

The objective was to improve the CONOPT solver in a Branch and Bound environment.

The result with CONOPT3 has so far been:

- **A solver that is slightly faster for B&B nodes (about 30%).**
- **A solver that behaves differently. Some models are solved with SBB/CONOPT3 that were not solved with SBB/CONOPT2 and some are solve with SBB/CONOPT2 but not with SBB/CONOPT3. Combining the two solvers give new possibilities**
- **A solver that is significantly faster on many large NLP models with many superbasic variables. Many models that could not be solved reliably before can now be solved.**

GAMS/CONOPT3 will go in Beta test this month.