European Journal of Operational Research 244 (2015) 77–85

Contents lists available at ScienceDirect

European Journal of Operational Research

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

Discrete Optimization

A parallelised distributed implementation of a Branch and Fix Coordination algorithm

Adela Pagès-Bernaus[a,*], Gerardo Pérez-Valdés[a,b], Asgeir Tomasgard[a]

[a] Institute for Industrial Economics and Technology Management, Norwegian University of Science and Technology, Norway
[b] Applied Economics Department, SINTEF Technology and Society, Norway

ARTICLE INFO

Article history:
Received 7 January 2014
Accepted 3 January 2015
Available online 19 January 2015

Keywords:
Stochastic mixed-integer problems
Branch and fix coordination algorithm
Parallel programming

ABSTRACT

Branch and Fix Coordination is an algorithm intended to solve large scale multi-stage stochastic mixed integer problems, based on the particular structure of such problems, so that they can be broken down into smaller subproblems. With this in mind, it is possible to use distributed computation techniques to solve the several subproblems in a parallel way, almost independently. To guarantee non-anticipativity in the global solution, the values of the integer variables in the subproblems are coordinated by a master thread. Scenario 'clusters' lend themselves particularly well to parallelisation, allowing us to solve some problems noticeably faster. Thanks to the decomposition into smaller subproblems, we can also attempt to solve otherwise intractable instances. In this work, we present details on the computational implementation of the Branch and Fix Coordination algorithm.

1. Introduction

Many infrastructure problems nowadays are solved using optimisation and equilibrium models (Li, Gabriel, Shim, & Azarm, 2011; Rømo et al., 2009). These problems often imply handling investments, which can be represented by yes/no decisions ("Should a pipeline be built or not?") and modelled with binary variables. Mixed Integer Programming problems (MIPs), and specially those containing binary variables, are common in problems of transportation (Christiansen, Fagerholt, & Ronen, 2004), energy (Wallace & Fleten, 2003), real-state, etc. In these cases, strategic investment decisions influence a project's development over long timespans. For example, expensive building projects induce costs and deliver returns for many years after the decision of building was made. Once built, under-utilisation impacts the economic prospects of the project.

Dealing with these long-lasting investments in the real world, almost invariably, involves uncertainty in the parameters of the model we are trying to create. Prices, supplies and consumptions might differ significantly from estimates, unexpected events could make these estimates imprecise, or new legislation can turn once attractive investment into expensive ones (Alonso-Ayuso, Escudero, & Ortuño, 2003; Kall & Wallace, 1994).

Uncertainty, therefore, further complicates already hard-to-solve MIPs with the introduction of additional variables/parameters: MIPs are complex because of the large number of combinatorial choices they imply; stochastic problems are complicated because of the amount of scenarios they involve if a lot of variability happens over long time spans. As a result, stochastic MIPs are likely to result in complex problems, even while working exclusively with linear constraints (SMILPs).

In view of this, the Branch and Fix Coordination (BFC) algorithm was developed to tackle a certain class of SMILPs, namely, those in which both integer and continuous variables appear in (generally) every stage of the problem, all integer variables are binary, and no multi-scenario constraints appear (Alonso-Ayuso et al., 2003; Escudero, Garín, Merino, & Pérez, 2010b). BFC is based on the well-known Branch and Bound (B&B) algorithm, with the main difference that the search tree evaluates many subproblems at each step, and the decisions to branch, prune or bound are done taking all subproblems into consideration. The first version of the BFC was developed by the groups of Alonso-Ayuso et al. (2003) to take advantage of scenario-wise decomposition schemes for solving special cases of two-stage SMILPs (for more applications see also Escudero, Garín, Merino, & Pérez, 2007, 2009b, 2010a). The algorithm was generalised to multi-stage SMILPs with binary and continuous variables in any stage (Escudero, Garín, Merino, & Pérez, 2009a, 2009c; Escudero et al.,

* Corresponding author. Tel.: +47 73591267.
E-mail addresses: adela.pages@iot.ntnu.no (A. Pagès-Bernaus), gerardo.valdes@iot.ntnu.no (G. Pérez-Valdés), asgeir.tomasgard@iot.ntnu.no (A. Tomasgard).

http://dx.doi.org/10.1016/j.ejor.2015.01.004
0377-2217/© 2015 The Authors. Published by Elsevier B.V. This is an open access article under the CC BY-NC-ND license (http://creativecommons.org/licenses/by-nc-nd/4.0/).

2010b). Further refinements such as the parallelisation of the algorithm or contemplating explicitly non-symmetric trees have proven to be successful in reducing running times (Aldasoro, Escudero, Merino, & Pérez, 2013; Escudero, Garín, Merino, & Pérez, 2012).

In this paper, we present a particular implementation of the BFC routine, which uses parallel processing to solve many subproblems at the same time in a way that makes the process arguably faster and more efficient. The resulting application is able to solve problems with dimensions which are orders of magnitude larger than those reported in past papers, and also coordinating significantly more clusters and variables without apparent loss of efficiency. Moreover, this implementation improves over the existing ones with added flexibility (allowing for more branching options and data storage), as well as doing away with some bounding strategies in favour of looser but faster searches.

The framework currently in place allows for a mostly seamless transition into solving fully-decomposed problems, which can then be stored in several files without the need to load the full problem anywhere in the solution process.

The paper is organised as follows: first, we present some formulations useful for the description of the algorithm in Section 2. After that, in Section 3, we state the problems needed to be solved at several steps of this particular BFC implementation, which is itself described in Section 4, along with some details on the parallelisation we use. Later in Section 5, we show several problem instances, whose analysis suggests that BFC is competitive when compared to commercial MIP solvers.

2. Problem formulation

Consider the Deterministic Equivalent Model (DEM) of a stochastic mixed integer linear programming problem with only binary and continuous variables X, Y, and its matrix form, $[AX + BY]$. For most SMILPs, we can easily identify blocks in these matrices that represent constraints specific to a scenario tree node, or constraints that link one node in the tree to its parent nodes, children nodes, and so on. We can use this idea of blocks, and the constraints and variables they involve, to represent different variations of the problem by indexing sections of the constraint matrices which have special interest to us with a set of nodes which in turn corresponds to a row of blocks in the matrix.

$$\begin{bmatrix} A_{G_1} \\ A_{G_2} \\ \vdots \end{bmatrix} X + \begin{bmatrix} B_{G_1} \\ B_{G_2} \\ \vdots \end{bmatrix} Y$$

Suppose $\mathbb{G}$ is the set of all nodes g of a (not necessarily symmetric) scenario tree, and $\mathbb{G}^2$ the set of all subsets of $\mathbb{G}$ (i.e. its power set). Further, let $\mathbb{G}^1$ be the subset of one-element sets in $\mathbb{G}^2$, i.e., $\mathbb{G}^1 := \{m \in \mathbb{G}^2 : |m| = 1\}$; clearly, there is a one-to-one correspondence between the nodes in $\mathbb{G}$ and the sets in $\mathbb{G}^1$.

At each node g, we have binary variables $x_{g,i}$ and continuous variables $y_{g,j}$. We use the paired sets of indices $\mathbb{I}_g, \mathbb{J}_g$ so that $X_g = \{x_{g,i} : i \in \mathbb{I}_g\}$ and $Y_g = \{y_{g,j} : j \in \mathbb{J}_g\}$, and in turn $X = \{X_g : g \in \mathbb{G}\}$, and $Y = \{Y_g : g \in \mathbb{G}\}$.

First, let us present a simplified formulation of a general stochastic problem in *compact formulation*. Here, we only differentiate between groups of constraints that (a) affect variables belonging to one node, or (b) affect in general all variables (for example, recourse constraints, among others). We then index each set of rows using either the set $G \in \mathbb{G}^1$ corresponding to the node the set of rows involves, or the entire node set $\mathbb{G}$. This gives A_G and B_G, for each $G \in \mathbb{G}^1 \cup \{\mathbb{G}\}$, in which each G is a member of a set of sets. The DEM of this problem is

$$\min: f(X, Y) = \sum_{g \in \mathbb{G}} w_g(a_g X_g + b_g Y_g) \tag{1a}$$

$$\text{s.t. } A_G X + B_G Y \le C_G, \quad G \in \mathbb{G}^1 \cup \{\mathbb{G}\}; \tag{1b}$$

$$x_{g,i} \in \{0, 1\}, \quad i \in \mathbb{I}_g, g \in \mathbb{G}; \tag{1c}$$

$$y_{g,j} \in \mathbb{R}, \quad j \in \mathbb{J}_g, g \in \mathbb{G}; \tag{1d}$$

with properly defined weights w_g for each node.

In problem (1), all constraints which affect the variables of more than one node of the scenario tree appear as $A_{\mathbb{G}} X + B_{\mathbb{G}} Y \le C_{\mathbb{G}}$ in expression (1b). However, they do not necessarily need to be aggregated like this. By using this indexing idea, we can just as easily group nodes into relevant sets which correspond to scenarios, or to groups of scenarios, or to stages in the scenario tree, and use these sets to index parts of the constraint matrices. This helps us to write different formulations and variations of a SMILP DEM in a condensed notation.

Let $\mathbb{S}_i$ be the set of nodes g which belong to stage i (i.e. $\mathbb{S}_1$ contains only the root node(s), $\mathbb{S}_2$ contains the immediate children to the root nodes, and so on). If the tree has S stages, make $\mathbb{S} = \{\mathbb{S}_1, \dots, \mathbb{S}_S\}$ the family of all stage sets. Also, for the set of leaf nodes $g \in \mathbb{S}_S$, create a partition $\mathbb{L} = \{\mathbb{L}_c\}$ so that all nodes in a given $\mathbb{L}_c$ have a common root. This makes it possible to define a *cluster* (Escudero et al., 2010a) as we use it in this work:

Definition 1 (Cluster). For each element $\mathbb{L}_c \in \mathbb{L}$, a cluster $\mathbb{C}_c$ is a set of nodes which contains all elements of $\mathbb{L}_c$ and their parent nodes up to the root. Then $\mathbb{C} = \{\mathbb{C}_c\}$ is the set of all clusters in a scenario tree.

With the families $\mathbb{S}$ and $\mathbb{C}$ thus defined, we can write constraints for all the nodes in one stage, or all the nodes of one scenario, or all the nodes in a given set of scenarios, and index them with sets regardless of the formulation chosen for the model.

For example, if we consider the stochastic scenario tree in Fig. 1a, the set of nodes will be $\mathbb{G} = \{g_1, \dots, g_6\}$, and consequently $\mathbb{G}^1 = \{\{g_1\}, \dots, \{g_6\}\}$.

The only possible cluster for this tree is $\mathbb{C} = \{\{g_1, g_2, \dots, g_6\}\}$, with $\mathbb{S} = \{\{g_1\}, \{g_2, g_3\}, \{g_4, g_5, g_6\}\}$. In this manner, the formulation would be equivalent to that in problem (1).

On the other hand, if we define a different set of nodes $\mathbb{G} = \{g_1, \dots, g_9\}$, such as the ones shown in Fig. 1b, with $\mathbb{S} = \{\{g_1, g_2, g_3\}, \{g_4, g_5, g_6\}, \{g_7, g_8, g_9\}\}$ and $\mathbb{C} = \{\{g_1, g_4, g_7\}, \{g_2, g_5, g_8\}, \{g_3, g_6, g_9\}\}$ being the only possible set of clusters, we can formulate the *split-variable formulation* of (1), **MP1**, once we define $\mathbb{A}$ as $\mathbb{A} = \{\{g_1, g_2, g_3\}, \{g_4, g_5\}\}$ as the set of all (non-trivial) non-anticipativity classes of equivalence. Notice that the weights w_g should be modified accordingly for the problems to be equivalent.

MP1 :

$$\min: f(X, Y) = \sum_{g \in \mathbb{G}} w_g(a_g X_g + b_g Y_g) \tag{2a}$$

$$\text{s.t. } A_G X + B_G Y \le C_G, \quad G \in \mathbb{C} \cup \mathbb{G}^1 \cup \mathbb{S}; \tag{2b}$$

$$X_g = X_h, Y_g = Y_h, \quad \forall g, h \in \mathbb{A}_n, \mathbb{A}_n \in \mathbb{A}; \tag{2c}$$

$$x_{g,i} \in \{0, 1\}, \quad i \in \mathbb{I}_g, g \in \mathbb{G}; \tag{2d}$$

$$y_{g,j} \in \mathbb{R}, \quad j \in \mathbb{J}_g, g \in \mathbb{G}. \tag{2e}$$

Formulation **MP1** is flexible enough to describe both the usual stochastic formulations, but also to describe formulations in between those, such as that shown in Fig. 1c. For large scenario trees, there are as many such possible *mixed formulations* (analogous to the splitting-compact representation used in Escudero et al., 2010a), as stages in the tree.

The definitions of families $\mathbb{L}$, $\mathbb{A}$ and, more important to us, $\mathbb{C}$, are closely linked to the selection of a *break stage* (effectively equivalent to that in Escudero et al., 2012 for most scenario trees).