European Journal of Operational Research 199 (2009) 89–97

Contents lists available at ScienceDirect

European Journal of Operational Research

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

Discrete Optimization

Improving benders decomposition using a genetic algorithm

C.A. Poojari, J.E. Beasley *

Centre for the Analysis of Risk and Optimisation Modelling Applications (CARISMA), School of Information Systems, Computing and Mathematics, Brunel University, Uxbridge UB8 3PH, UK

ARTICLE INFO

Article history:
Received 14 May 2007
Accepted 31 October 2008
Available online 13 November 2008

Keywords:
Genetic algorithm
Benders decomposition
Mixed-integer linear programs

ABSTRACT

We develop and investigate the performance of a hybrid solution framework for solving mixed-integer linear programming problems. Benders decomposition and a genetic algorithm are combined to develop a framework to compute feasible solutions. We decompose the problem into a master problem and a subproblem. A genetic algorithm along with a heuristic are used to obtain feasible solutions to the master problem, whereas the subproblem is solved to optimality using a linear programming solver. Over successive iterations the master problem is refined by adding cutting planes that are implied by the subproblem. We compare the performance of the approach against a standard Benders decomposition approach as well as against a stand-alone solver (Cplex) on MIPLIB test problems.

1. Introduction

A general mixed-integer linear programming (MILP) problem can be represented as

$$P_{\text{MILP}} \quad \text{Min} \quad cx + fy, \tag{1}$$

$$Ax \geqslant b, \tag{2}$$

$$Bx + Gy \geqslant d, \tag{3}$$

where $x \in \mathbb{Z}_+^{n_1}$, $y \in \mathbb{R}_+^{n_2}$, $n = n_1 + n_2$. The cost vectors are $c \in \mathbb{R}^{n_1}$ and $f \in \mathbb{R}^{n_2}$, the matrices $A \in \mathbb{R}^{m_1 \times n_1}$, $B \in \mathbb{R}^{m_2 \times n_1}$, $G \in \mathbb{R}^{m_2 \times n_2}$, the right-hand-sides $b \in \mathbb{R}^{m_1}$, $d \in \mathbb{R}^{m_2}$.

In this problem we seek to minimise a linear function of the decision vectors (x, y) subject to linear inequality constraints and the requirement that some of the decision variables must be integer valued. If $n_1 = n$ ($n_2 = 0$), then the problem is called a pure integer linear program (PILP); further if $x_i \in \{0,1\}\ \forall i$ then the problem is called a zero-one integer program. On the other hand if $n_2 \geqslant 1$ then the problem is called a mixed-integer linear program (MILP).

MILPs are (except for special cases) known to be NP-hard and therefore finding an optimal (at times even a feasible) solution is difficult. Polyhedral approaches have been increasingly used for MILPs. The main idea is to iteratively strengthen the linear programming relaxation of the MILP by adding inequalities that are violated by fractional solutions, but are satisfied by integer solutions. This results in a better description of the convex hull of the problem. Once a complete description of the convex hull in terms of linear inequalities is obtained (although in practice this may not be achievable) then the resulting problem could be solved as a LP. The optimal solution to the LP would be the optimal solution to the original MILP. Such polyhedral approaches have been combined with branch and bound to produce branch and cut, whereas branch and bound has been combined with column generation to produce branch and price. The advantage of these methods are their mathematical rigor and a guarantee of optimality on termination. However, the large number of cuts or columns which need to be generated means they can be extremely time consuming.

It is clear that one approach that can be followed is a hybrid approach, which combines the positive features of mathematical programming based ideas with metaheuristics. Metaheuristics such as tabu search, simulated annealing and genetic algorithms have often been customised for specific problems so as to generate good quality integer solutions. Metaheuristics for specific problems are often designed such that problem structure is exploited during recursive moves through a search space. However, the suitability of metaheuristics for general MILPs (where there is, perhaps, no distinct problem structure) is less well researched. Representing the original problem so as to customise it for the moves of a metaheuristic is non-trivial for problems having continuous variables. Also, satisfying problem constraints after a move is an issue of ongoing research amongst metaheuristics researchers. Puchinger and Raidl (2005), and Raidl and Puchinger (2008)

* Corresponding author. Tel.: +44 1895 266219; fax: +44 1895 269732.
E-mail addresses: Chandra.Poojari@brunel.ac.uk (C.A. Poojari), John.Beasley@brunel.ac.uk (J.E. Beasley).

0377-2217/$ - see front matter © 2008 Elsevier B.V. All rights reserved.
doi:10.1016/j.ejor.2008.10.033

discuss different state-of-the-art approaches for combining exact algorithms and metaheuristics. For an example of work applying a genetic algorithm to find a feasible solution for MILPs to enable the tree in a branch and bound process to be pruned, see Nieminen et al. (2003).

In this paper, we develop a hybrid approach that integrates a genetic algorithm (GA) with Benders decomposition. Decomposition based techniques are helpful to efficiently process the large scale MILPs that often arise in practical applications. In our approach we decompose the MILP into two smaller problems based on the presence of integer and continuous variables. This decomposition is the same as that given by Benders (1962). One of these decomposed problems contains all the integer variables, whereas the other problem contains only continuous variables. We would note here that recently renewed interest in Benders decomposition has been appearing, e.g. Rei et al. (2006) discuss how local branching can be used to speed Benders decomposition, while Fischetti et al. (submitted for publication) discuss how a better choice of cuts in Benders decomposition can be made.

GAs are typically more suitable for discrete optimisation problems rather than those having continuous variables. We utilise this and use a GA to obtain 'good solutions' for the decomposed problem containing the integer variables. Cuts are generated as in conventional Benders decomposition. We iteratively generate feasibility and optimality cuts that approximate the polyhedron of the decomposed problem to that of the original problem.

We use a GA that can represent and process continuous, general integer and mixed-integer linear mathematical programming problems. Within the GA, we seek feasible solutions. Thus at successive generations feasible solutions are given higher priority over infeasible ones, the intention being to have a set of feasible solutions on termination of the GA.

The reason why we have adopted a population based metaheuristic (a genetic algorithm) rather than any other metaheuristic (such as tabu search or simulated annealing) is that these other metaheuristics are typically single-solution heuristics. That is, they work on a single solution at a time, progressively moving through the search space based on the current (single) solution. By contrast population based approaches utilise a number of solutions (effectively simultaneously, e.g. in crossover). Since in our approach (as will become apparent below) each feasible solution gives rise to a cut for the problem it is clear that a metaheuristic that involves more solutions will (other factors being equal) be preferred. For this reason we have adopted a genetic algorithm (population based metaheuristic).

2. Benders decomposition

The original MILP problem, P_{MILP}, Eqs. (1)–(3), can be decomposed, in a Benders fashion, into two smaller problems. One of these is a MILP problem having $n_1 + 1$ variables, n_1 integer variables and one continuous variable. We refer to this as the *master* problem. The other problem is a LP problem with n_2 variables, as part of which n_1 integer variables have been fixed to their value in the master problem. We refer to this as the *subproblem*.

Benders (1962) showed that the master problem and the subproblem can be solved successively with information being communicated between them. The integer solution is passed from the master to the subproblem, and the subproblem generates a feasible/optimal cut for the corresponding integer solution that can then be added to the master problem.

Consider the MILP problem P_{MILP}, the master problem is written as

$$P_{Master} \quad \text{Min} \quad cx + \vartheta, \tag{4}$$
$$Ax \geqslant b, \tag{5}$$
$$T_s x \geqslant t_s, \quad s = 1 \ldots |S|, \tag{6}$$
$$E_t x + \vartheta \geqslant e_t, \quad t = 1 \ldots |T|, \tag{7}$$
$$x \in \mathbb{Z}_+^{n_1}, \quad \vartheta \in \mathbb{R}, \tag{8}$$

where S and T denote the set of feasibility and optimality cuts respectively. Eq. (6) corresponds to the feasibility cuts and Eq. (7) the optimality cuts. The continuous variable ϑ takes into account the objective function term fy (Eq. (1)). For a given solution $(\hat{x},\hat{\vartheta})$ to the master problem, the subproblem is represented as a LP

$$P_{Sub} \quad \text{Min} \quad fy, \tag{9}$$
$$Gy \geqslant d - B\hat{x}, \tag{10}$$
$$y \in \mathbb{R}_+^{n_2}. \tag{11}$$

Note here that the solution $(\hat{x},\hat{\vartheta})$ to the master problem need only be feasible, there is no requirement to solve the master problem to proven optimality (Cote and Laughton, 1984).

The dual to this subproblem is

$$P_{Sub\text{-}dual} \quad \text{Max} \quad \pi^T(d - B\hat{x}), \tag{12}$$
$$\pi^T G \leqslant f, \tag{13}$$
$$\pi \geqslant 0. \tag{14}$$

There are two possible cases with respect to $P_{Sub\text{-}dual}$:

- it is unbounded, in which case choose any unbounded extreme ray $(\hat{\phi})$ and add to S a Benders feasibility cut

$$\hat{\phi}^T(d - Bx) \leqslant 0 \tag{15}$$

- it is bounded (with optimal value $\hat{P}$), in which case take an optimal solution $(\hat{\pi})$ and add to T a Benders optimality cut

$$\hat{\pi}^T(d - Bx) \leqslant \vartheta. \tag{16}$$

The pseudocode for the Benders decomposition algorithm is described in Algorithm 2.1. In that algorithm we repeatedly: solve the master problem P_{Master}; then solve the associated dual subproblem $P_{Sub\text{-}dual}$; then add either a feasibility or optimality cut as appropriate. The algorithm terminates either when sufficient iterations (solutions of the master problem) have been performed; or when a computational time limit has been reached; or when it has converged. At any Benders iteration a lower bound on the optimal solution to the original problem is given by $c\hat{x} + \hat{\vartheta}$, and an upper bound on the optimal solution to the original problem is given by $c\hat{x} + \hat{P}$. Convergence occurs when the