Approximate Uni-directional Benders Decomposition

Christina N. Burt and **Nir Lipovetzky** and **Adrian R. Pearce** and **Peter J. Stuckey**
Department of Computing and Information Systems
The University of Melbourne, Parkville, Australia

Abstract

We examine a decomposition approach to find good quality feasible solutions. In particular, we study a method to reduce the search-space by decomposing a problem into two partitions, where the second partition (i.e., the subproblem) contains the fixed solution of the first (i.e., the master). This type of approach is usually motivated by the presence of two sub-problems that are each more easily solved by different methods. Our work is motivated by methods for which it is nontrivial to return a strong 'no-good', 'Benders feasibility', or 'optimality' cut. Instead, we focus our attention on a uni-directional decomposition approach. Instead of providing a relaxation of the sub-problem for the master problem, as in Benders decomposition, we provide an approximation of the sub-problem. Thus, we aim at finding good quality feasible solutions in the first iteration. While the quality of the approximation itself affects the impact of this approach, we illustrate that even using a simple approximation can have strong positive impact on two examples: the Travelling Purchaser Problem and a Mine Planning Problem.

Consider a class of problems which are not efficiently solved by any one method. A simple, quick-and-dirty method for finding feasible solutions to such problems is to partition them into two subproblems, each of which is more easily solved by different algorithms (see, e.g., (Fernández and Borrajo 2009) and (Flórez et al. 2011)). Such hybrid systems are useful for quickly finding feasible solutions for problems which are otherwise unsolvable. In the naive decomposition, we can split the problem into two *partitions* by separating the variables into two sets. In the first partition (called the master, in Benders literature), the variables from the second partition (called the sub-problem) are ignored completely. We then solve the master without any knowledge or regard for the components in the subproblem. The subproblem can then be solved by fixing the variables from the master to the solution obtained from the naive solve. If the partitioning is devised such that a solution to the master problem guarantees a feasible solution in the subproblem and thereby a feasible solution to the overall problem, then we have a method for finding feasible solutions more quickly (perhaps) than solving the full problem. In this paper, we restrict our focus to such cases—i.e., decompositions which guarantee feasibility. We describe a method to improve the solution space produced by the master, by approximating the subproblem within it. If we are using specialised solvers that don't provide feedback, as in Benders, and solving the full problem to optimality is also not possible, then the alternative is to give up on completeness in order to get a strong approximation. In our case studies, we illustrate this scenario by solving the master using mixed-integer programming and the subproblem by an Automated Planner. However, the approach is not limited to these methods.

State-space search is the predominant approach to solving Automated Planning problems (see Section *Planning and State-space Search*). Typically, such search methods focus on finding feasible solutions—partly because this is all that is required and partly because the state-space can be overwhelmingly large and non-convex. Since planning methods search for paths connecting an initial state to a goal state within an implicitly defined graph, they can be used to solve problems with state-dependent components, which is generally challenging for mixed-integer programming solvers. Considering the full state-space, however, leads to an obvious explosion in the search space. Therefore, decomposition can be necessary to reduce the complexity of these problems and necessary to deal with more expressive planning problems that have continuous, numeric, and temporal variables.

Benders Decomposition is a prominent decomposition approach which guarantees optimal solutions for mixed-integer linear systems. In the Benders method, we partition the full problem into a master and subproblem, each of which is easier to solve than the overall problem. This method is based on the concept of *projection*: a procedure that eliminates dimension from a problem in such a way that the feasible region for the reduced system remains intact. Furthermore, if no solution exists for the projected system, then no solution exists for the original system. Obtaining the correct projection is analogous to performing pivots in Gaussian Elimination—there is usually a residual in the coefficient and/or in the right-hand-side once the projection is obtained. This is the key result on which Benders decomposition is based. The master is originally an overapproximation of the correct projection. Then, in an iterative process, the solution space for the master is pruned back until it is provably the correct projection. Once this is obtained, it is trivial to find the optimal solution.

Our approximation is based on the insight that the residual in the correct projection represents the part of the problem that was projected away, or, the subproblem. Instead of eliminating variables as in the quick-and-dirty approach, consider replacing them with an approximation. If the approximation is perfect, then we simply have the original problem. However, in this paper we formulate the approximation in such a way that it both resembles the subproblem and has little computational burden. We use this approximation to drive the master to be closer to the correct projection. We show empirically that the residual can be effectively approximated by even a simple approximation of the subproblem.

Benders decomposition and Logic-based Benders decomposition are not automated methods—the partitions must be designed such that strong optimality and feasibility cuts can be returned from the subproblem. This itself is a non-trivial task. Contrarily, in our method we bypass this requirement. The only challenge in our method is to devise an approximation of the subproblem within the master. While the provision of a relaxation of the subproblem within the master has been an important step in Benders and Logic-based Benders since the beginning, the key difference with our method is that the approximation leads to a feasible solution (whereas the relaxation doesn't guarantee feasibility). Therefore, our method can also be used as a starting solution for Benders decomposition or Logic-based Benders decomposition if the first partition is provably an overestimation—i.e., it does not eliminate all the optimal solutions.

In the Section *Uni-Directional Decomposition*, we outline how this approximation can be achieved in a simple way. We describe our illustrative problems in sections *Example 1: The Travelling Purchaser Problem (TPP)* and *Example 2: Mining Operations Planning (MOP)*. Both problems are such that solving the entire problem with a planner leads to overwhelmingly large search spaces, and the state-of-the-art planners subsequently do not admit solutions. Our experiments (Section *Experiments*) show that even by creating simple approximations, we obtain strong improvement over naive partitionings on our motivating examples.

Planning and State-space Search

Planning is the problem of finding a sequence of actions that maps a given initial state to a goal state, where in its simplest form, the environment and the actions are deterministic—this is also known as *classical planning*. The computational challenge is to devise effective methods to obtain such action sequences, called plans. The last two decades have brought significant advances in classical planning (see, e.g., (Kautz and Selman 1996; Blum and Furst 1995; Bonet and Geffner 2001; Richter and Westphal 2010)), with *heuristic search*—using search algorithms on the underlying state model guided by heuristic estimators extracted automatically from the problem—being the most successful.

For large problems, explicitly enumerating the state space is not feasible. In this scenario, *factored* representations are used in which states are complete assignments to a set of variables whose domains are finite and discrete. Any graph-search algorithm can be used in order to find a plan on the underlying state model. Even so, blind search algorithms such as Dijkstra (Cormen et al. 2001) do not scale up due to the size of the state space, which can be exponential in the number of variables in the factored problem. On the other hand, heuristic search algorithms have proven to perform effectively, provided they use a heuristic function sufficiently informed to guide the search. State-of-the-art planners derive heuristic functions automatically from the description of the problem, and use search enhancements to prune the state-space.

One of the keys to scaling up is to have accurate heuristic functions in order to explore a reduced amount of the search space. Most of the heuristics are far from optimal, and very small errors can cause an exponential amount of exploration (Helmert and Röger 2008). An alternative solution is to decompose the original problem into a series of more easily solvable subproblems. Moreover, decomposition techniques are widely used to incorporate more expressive problems formalised in different theories. Specialised solvers are then used to deal with such theories, e.g., linear program solvers deal with linear equations and Presburger Arithmetic to reason over continuous effects and metric resources (Coles et al. 2008a; Ivankovic et al. 2014), AC power flow solvers deal with balancing problems in an electricity network (Piacentini et al. 2013), simple temporal network (STN) solvers deal with difference logic to reason over durative actions (Coles et al. 2008b), etc. The following approach reduces the complexity of the planning problem and improves the solution quality outcomes for hybrid systems.

Uni-directional Decomposition

Let $x \in D_x$ and $y \in D_y$ be vectors of variables, where the domains may be continuous or integer. Then, let $f(x, y)$ be a function of these variables such that f may contain some part that is not trivially linearisable, such as state-dependent components. As a simple example of state-dependence, let the 'state' of the system be given by $\dot{\mathrm{s}}(t)$, at a given time t. Then, a possible function that we are considering in this class is one where the cost function for a particular variable is state-dependent: $f(x, y, t) = cx + d\left(\dot{\mathrm{s}}(t)\right) y$. Such a function can be cumbersome to linearise, because we would need to encode exponentially many ancillary variables to indicate which combinations of events are a part of the current state. Let $C(x, y, t)$ be a set of constraints on x, y and t, where the constraints may contain some part that is not trivially linearisable. Then, we wish to solve problems of the following class:

$$\begin{aligned} P: \qquad \min \quad & f(x,y,t) \\ \text{s.t.} \quad & C(x,y,t)\ , \\ & x \in D_x\ , \\ & y \in D_y\ , \\ & t \in D_t\ . \end{aligned} \tag{1}$$

We partition problem P into master $P1$ and subproblem $P2$, such that all constraints in P are represented in $P1 \cup P2$. The partitions themselves are motivated by subproblem structure that lends each subproblem to be solved more easily by specialised algorithms. Let $P1$ be the naive projection of P onto x, where we simply relax all y and constraints containing y. Furthermore, let $P1$ be linear—i.e., the non-linearisable component is relaxed—such that $f(x,y,t) = cx + f'(y,t)$ where $f'(y,t) = f(x,y,t) - cx$, and, similarly, $C(x,y,t) = A_1x \leq b_1 \wedge C'(y,t)$. Importantly, we choose $P1$ such that any solution for $P1$ is feasible for P.

$$\begin{aligned} P1: \quad & \min \quad cx \\ & \text{s.t.} \quad A_1x \leq b_1 \ , \\ & \qquad\ \ x \in D_x \ . \end{aligned} \tag{2}$$

Then, let $P2$ be the remaining, restricted problem. A feasible solution, $\bar{x}$, to $P1$ is then fixed within $P2$, as follows:

$$\begin{aligned} P2: \quad & \min \quad c\bar{x} + f'(y,t) \\ & \text{s.t.} \quad C(\bar{x},y,t) \ , \\ & \qquad\ \ y \in D_y \ . \end{aligned} \tag{3}$$

Clearly, if $P2$ finds a feasible solution, then this solution is valid for the whole problem, P, since it has satisfied all the constraints of the problem. Of course, such a solution is unlikely to be optimal, and since the master is only a naive projection, there are no guarantees of finding a solution to $P2$ even if one exists.

Logic-based Benders Decomposition

We now deviate from our approximate approach to describe Logic-based Benders decomposition (e.g., see Hooker and Ottosson (2003)). From the solution obtained from $P2$ (if any), we try to *learn* a new constraint that cuts off either infeasible or sub-optimal solutions in $P1$. In Classical Benders decomposition, subproblem $P2$ is a linear program, and we use *dual* information to infer the tightest possible bound to pass back to $P1$, before resolving. Hooker and Ottosson (2003) proposed that, as long as we can prove the bound is valid—by which, they mean, that it is guaranteed to be a lower bound—any method can be used to generate this bound. They called this the *inference dual* of $P2$, i.e., $Q2$. More formally, we wish to find the closest possible value of u to $f(\bar{x},y,t)$ from $C(\bar{x},y,t)$.

$$\begin{aligned} Q2: \quad & \max \quad u \\ & \text{s.t.} \quad C(\bar{x},y,t) \overset{p}{\Rightarrow} f(\bar{x},y,t) \geq u \ , \\ & \qquad\ \ u \in \mathcal{R}^+, \end{aligned} \tag{4}$$

where p is a proof that the bound is valid (Hooker 2007). Then u and the relevant cuts are added to $P1$, and we can repeat the whole process, until we can prove convergence on the optimal solution. That is, the *iterative* version of problem $P1$ becomes

$$\begin{aligned} P1i: \min \quad & cx + \eta \\ \text{s.t.} \quad & A_1x \leq b_1 \ , && (5) \\ & \eta \geq \left\{ \begin{array}{ll} & \max \ u \\ \text{s.t.} C(\bar{x},y,t) & \overset{p}{\Rightarrow} f(\bar{x},y,t) \geq u \ , \end{array} \right. && (6) \\ & x \in D_x \ . \end{aligned}$$

In the class of mathematical programs of interest here—i.e., problems that are difficult to linearise—we have no luxury of easy-to-obtain proofs, p.

Approximating the projection

With this in mind, the alternate heuristic we propose is both simple and cheap. Since η and the corresponding Benders cuts are characterised by the solution to $P2$, and we wish to drive $P1$ into a good, and preferably near-optimal, space in order to arrive at better solutions to $P2$, we approximate η and the corresponding cuts as follows. We know, from Classical Benders decomposition (Benders 1962), that for linear systems, if we substitute the optimal values of x (wrt P) into $P2$, and then solve $P2$, we obtain the optimal solution to P. Let the optimal $P1$ values be $\bar{x}^*$, and the corresponding optimal subproblem, $P2^*$. Furthermore, let z be a vector of *ancillary* variables to x. Then, the heuristic is to define η and its corresponding constraints using the ancillary variables, such that they approximate the problem $P2^*$ and this approximation problem is easier to solve. That is,

$$\begin{array}{llcll} \min & dz + c\bar{x}^* & & \min & f(\bar{x}^*,y) \\ \text{s.t.} & A_2z \leq b_2 \ , & \approx & \text{s.t.} & C'(\bar{x}^*,y) \ , \\ & B(\bar{x}^*,z) \ , & & & y \in D_y \ , \\ & z \in D_z \ , & & & \end{array}$$

for some function dz, constraints $A_2z \leq b_2$, linking constraints $B(\bar{x}^*,z)$ and domain D_z. It should be a goal that the approximation does not create infeasible solutions for $P2$, although clearly this is difficult to guarantee cheaply.

We denote our approximate master and subproblem by $H1$ and $H2$ respectively. Now, the master becomes:

$$\begin{aligned} H1: \quad \min \quad & cx + dz \\ \text{s.t.} \quad & A_1x \leq b_1 \ , && (7) \\ & A_2z \leq b_2 \ , && (8) \\ & B(x,z) \ , && (9) \\ & x \in D_x \ , \\ & z \in D_z \ . \end{aligned}$$

The ancillary variables will now drive the master solution toward the optimal solution to problem P. We then fix the solution to $H1$ within $H2$ as follows:

$$\begin{aligned} H2: \quad & \min \quad f(\bar{x},y) \\ & \text{s.t.} \quad C'(\bar{x},y) \ , \\ & \qquad\ \ y \in D_y \ . \end{aligned} \tag{10}$$

In the following sections, we provide complete examples of this approach. The first we choose for the simplicity of the decomposition and illustration—we do not suggest that our method is the best approach for this particular problem. However, the second example is a real-world problem which cannot be easily solved by any one method. For this problem, our approach is very effective.

Example 1: Travelling Purchaser Problem (TPP)

Consider a transportation problem defined on a graph $\mathcal{G} = (\mathcal{N}, \mathcal{E})$, where the nodes represent a set of markets M, and a depot, s, (i.e., $M \cup \{s\} = \mathcal{N}$), and the edges represent direct links between them. Suppose we wish to route a purchaser on this graph—beginning at the depot—with the intention of visiting as many markets as is necessary to meet demand for a set of products, K. The markets may have limited quantities of each product (capacity C). This is known as the Travelling Purchaser Problem (Gerevini et al. 2009). The optimisation version of this problem is as follows.

Definition 1 (TPP). *Given a graph $\mathcal{G} = (\mathcal{N}, \mathcal{E})$; a set of products, $k \in K$; demand for each product in the set, D_k; a set of markets, $m \in M \subseteq \mathcal{N}$; capacities on products available at each market, $C_{m,k}$; constant prices for each product at each market, $c_{m,k}$; travel cost, d_e, along each edge, $e \in \mathcal{E}$; and the depot s: select a subset of the markets such that the demand for each product is satisfied, the tour begins and ends at the starting depot, and the overall cost of purchasing and travelling is minimised.*

To formalise the problem: let x_e be 1 if edge e is traversed during the route (and 0 otherwise), $z_{m,k}$ be the units of product k that are picked up from market m, and y_m be 1 if market m is visited (and 0 otherwise). We let n be the number of markets selected in an incumbent solution, and S be a subtour of markets detected in the incumbent solution. Recall that $\mathcal{N} = \{s \cup M\}$. Now we can express the symmetric TPP (labelled TPP) as follows:

$$TPP: \quad \min \quad \sum_e d_e x_e + \sum_{m,k} c_{m,k} z_{m,k}$$

$$\text{s.t.} \quad z_{m,k} \le C_{m,k} y_m \qquad \forall\, m \in N \backslash s, k\,, \tag{11}$$

$$\sum_m z_{m,k} \ge D_k \qquad \forall\, k\,, \tag{12}$$

$$\sum_{e \in \{(m,j) \forall\, j \in N \backslash m\}} x_e = 2 y_m \qquad \forall\, m \in N\,, \tag{13}$$

$$\sum_{e \in S} x_e \le |S| - \frac{1}{n - |S|} \sum_{m \notin S} y_m \qquad \forall\, S \subset N\,, \tag{14}$$

$$y_s = 1\,, \tag{15}$$

$$y_m, x_e \in \{0, 1\}\,,$$

$$z_{m,k} \in \mathcal{Z}^+\,.$$

In the objective function, we represent the cost of the tour of the selected markets and the cost of purchasing goods at each market. In constraint (11), we prevent purchases from markets that are not visited and place an upper bound on the quantity of product that may be purchased at each market. In constraint (12), we ensure that our demand for each product is met by the purchasing variables. In constraint (13), we ensure that each market has an incoming and outgoing edge. Constraint (15) ensures the source node is visited. This formulation matches that presented in (Laporte, Riera-Ledesma, and Salazar-González 2003) with the exception of the subtour elimination constraint. The subtour elimination constraint, (14), ensures that we make a complete tour through the selected markets. Note that, for computational effectiveness, we implement this constraint in callback within the branch-and-bound tree. This allows us to access information from the incumbent solution, and generate cuts to eliminate illegal solutions. This constraint differs from traditional subtour elimination constraints, because we cannot assume that all of the markets selected in the incumbent solution will be in the optimal solution. In fact, it is possible that a subset of these markets is the optimal market selection. Therefore, the subtractive term

$$\frac{1}{n - |S|} \sum_{m \notin S} y_m$$

ensures that we only extend the size of a subtour to remaining markets if the inclusion of those markets provides a better solution. The multiplicative term

$$\frac{1}{n - |S|}$$

is required to scale the number of additional markets to one. That is, we can only try to force the subtour to take in at least one more market at a time, as the optimal market selection is not known in the relaxation. This constraint can be interpreted as a 'feasibility' cut for a logic-based Benders decomposition where the subproblem is forming a single tour. Therefore, this example has potential to illustrate the effectiveness of our approximation as a warm-start to a Benders decomposition itself—however, we will explore this in future work.

We chose this problem to illustrate our approach because of the presence of two clear sub-problems: capacitated facility location (e.g., see Melkote and Daskin (2001)), and the remaining tour problem. Also, we can obtain the exact solution for this problem, and therefore provide a more robust evaluation of our proposed method. The TPP can easily be extended to incorporate more complex constraints, such as time windows and state-dependent components. In these cases, solving the entire problem using MIP technology is completely out of the question, and planning would be the most suitable search framework to solve the subproblem. Our second example (Section *Mining Operations Planning*) will consider such a state-dependent scenario.

Approximate Benders Decomposition on TPP

The motivation for choosing the partitioning lends to the algorithmic strengths of the two methods, MIP and planning. For MIP, optimally solving the capacitated facility location problem is well studied. However, for planning this subproblem is challenging due to the numerical quantities (e.g., demand and capacities) that must be tracked throughout the search. On the other hand, for MIP the subtour elimination component of the remaining tour problem is computationally challenging. Classical planners can be employed for this component.

Therefore, for the master, we take the capacitated facility location variables and constraints, and relax the rest:

$$\begin{aligned}
P1: \quad & \min \quad \sum_{m,k} c_{m,k} z_{m,k} \\
& \text{s.t.} \quad z_{m,k} \le C_{m,k} y_m \quad \forall\, m,k\,, && (16)\\
& \sum_m z_{m,k} \ge D_k \quad \forall\, k\,, && (17)\\
& y_s = 1\,, && (18)\\
& y_m \in \{0,1\}\,,\\
& z_{m,k} \in \mathcal{Z}^+\,.
\end{aligned}$$

Since the partition already contains ancillary variables, y, we utilise these to approximate η as follows. We overestimate the cost of traversing an edge using the distance from the start node. That is, we let the cost of visiting a market be equal to the cost of travelling to that market from the source, i.e. $d_{s,m}$, where s is the depot node and m is the visited market, and back. The selected markets are guaranteed to form a circuit (and therefore is feasible with respect to the original formulation), because all selected markets are connected to the source. Then, we have:

$$\begin{aligned}
H1: \min \quad & \sum_m 2d_{s,m} y_m + \sum_{m,k} c_{m,k} z_{m,k}\\
\text{s.t.} \quad & z_{m,k} \le C_{m,k} y_m \quad \forall\, m,k\,, && (19)\\
& \sum_m z_{m,k} \ge D_k \quad \forall\, k\,, && (20)\\
& y_s = 1\,, && (21)\\
& y_m \in \{0,1\}\,,\\
& z_{m,k} \in \mathcal{Z}^+\,.
\end{aligned}$$

We can now fix the solution to $P1$ or $H1$ within the planning subproblem. Suppose that we set $\bar{y}_{m,k}$ and $\bar{z}_m$ to the master solution. Then, the planning problem is to solve the following:

$$\begin{aligned}
H2: \min \quad & \sum_e d_e x_e + \sum_{m,k} c_{m,k} \bar{z}_{m,k}\\
& \sum_{e \in \{(m,j) \forall\, j \in N \setminus m\}} x_e = 2\bar{y}_m \quad \forall\, m \in N\,, && (22)\\
& \sum_{e \in S} x_e \le |S| \quad \forall\, S \subset N\,, && (23)\\
& x_e \in \{0,1\}\,.
\end{aligned}$$

Notice that the markets to visit are now fixed, there is no facility-location problem, and we simply seek an appropriate tour to complete the problem. That is, the remaining problem is a TSP.

Example 2: Mining Operations Planning (MOP)

Here, we present an example from Lipovetzky et al. (2014), where we study the problem of sequencing equipment that will extract ore from a mine.

Definition 2 (MOP)**.** *Given a short-term mine plan which dictates available equipment, ore demands, and required ore blends, determine the shortest (time) extraction sequences for the equipment to perform that satisfy the requirements of the short-term plan.*

Key to this problem is that the ore has varying quality, and must be *blended* to produce a saleable product. Also, the time required to travel to a particular task depends on what has already been mined and therefore is *state-dependent*.

The set of traversible edges, $\mathcal{E}\big(\dot{\mathrm{s}}(t)\big)$, represent the state-dependent shortest path between two nodes, j and j'. These edges are state-dependent in two ways: as resources are picked up, new pathways are created (i.e., nodes that were once inaccessible may become accessible), and traversal times improve over time.

Let the binary variable $y_{j,k}$ indicate if mining task j is completed while contributing to stockpile 'job' k. Let $d(j,j')$ be a function that returns the traversal time for the shortest path between two task locations. Then, the 'state' of the system at any moment in time, t, is given by the 'state' vector of resource collection up to time t, $\dot{\mathrm{s}}(t)$, where

$$\dot{\mathrm{s}}(t)\cdot e_j = \begin{cases} 1 & \text{if } \sum_{d' \le k} y_{j,d'} = 1\,,\\ 0 & \text{otherwise}\,,\end{cases}$$

where e_j is the jth column of the identity matrix. Then, $\tau\big(j,j',\dot{\mathrm{s}}(t)\big)$ is the shortest path traversal time from j to j', given the state of the mine. That is,

$$\tau\big(j,j',\dot{\mathrm{s}}(t)\big) = \begin{cases} \infty & \text{if } (j,j') \notin \mathcal{E}\big(\dot{\mathrm{s}}(t)\big)\,,\\ d(j,j') & \text{otherwise}\,.\end{cases}$$

The key variables are as follows:

ω is the makespan;

$w^s_{i,j,k}(w^f_{i,j,k})$ is the start (finishing) time of event tuple (i,j,k) for equipment i; and,

$x_{j,k}$ is the proportion of task j performed for job k.

Additionally, let C_j be the task capacity, $D^{\mathcal{L}}(D^{\mathcal{U}})$ the lower (upper) bound on job demand, G_j the quality of tasks, and $G^{\mathcal{L}}(G^{\mathcal{U}})$ the lower (upper) quality bound.

$$\begin{aligned}
P: \quad & \min\ \omega\\
\text{s.t.} \quad & \sum_k x_{j,k} = 1 \quad \forall\, j\,, && (24)\\
& D^{\mathcal{L}} \le \sum_j C_j x_{j,k} \le D^{\mathcal{U}} \quad \forall\, k\,, && (25)\\
& G^{\mathcal{L}} \sum_j C_j x_{j,k} \le \sum_j G_j C_j x_{j,k} \quad \forall\, k\,, && (26)\\
& \sum_j G_j C_j x_{j,k} \le G^{\mathcal{U}} \sum_j C_j x_{j,k} \quad \forall\, k\,, && (27)\\
& B(x,y)\,, && (28)\\
& C(\omega, w^s_{i,j,k}, w^f_{i,j,k}, x_{j,k})\,, && (29)\\
& x_{j,k} \in [0,1],\ y_{j,k} \in \{0,1\}\,,\\
& \omega, w^s_{i,j,k}, w^f_{i,j,k} \in \mathcal{R}^+\,,
\end{aligned}$$

where constraint (24) ensures that all tasks are fully completed, constraint (25) enforces demand restrictions for each job k, constraints (26)–(27) enforce quality restrictions, the constraints $B(x,y)$ describe precedences on all tasks, and constraints $C(\omega, w^s_{i,j,k}, w^f_{i,j,k}, x_{j,k})$ describe the state-dependent sequencing problem.

We partition P into a linear feasibility problem that solves the allocation of mining tasks to jobs and the blending problem, and a planning problem that solves the sequencing problem with state-dependencies. This is a standard decomposition in the literature, with early contributions from (Hooker 2007). In our version, we choose to use a planner for the sub-problem due to the state-dependent traversal times for loaders. We represent the fact that the master, $P1$, is a feasibility problem—that is, that it has no objective function—by multiplying the variable vector by zero. Then, $P1$ can be expressed as follows:

$$P1: \quad \min \mathbf{0}^\top (x_{j,k}, y_{j,k})$$

$$\text{s.t.} \sum_k x_{j,k} = 1 \qquad \forall j , \qquad (30)$$

$$D^{\mathcal{L}} \leq \sum_j C_j x_{j,k} \leq D^{\mathcal{U}} \qquad \forall k , \qquad (31)$$

$$G^{\mathcal{L}} \sum_j C_j x_{j,k} \leq \sum_j G_j C_j x_{j,k} \qquad \forall k , \qquad (32)$$

$$\sum_j G_j C_j x_{j,k} \leq G^{\mathcal{U}} \sum_j C_j x_{j,k} \qquad \forall k , \qquad (33)$$

$$B(x,y) , \qquad (34)$$

$$y_{j,k} \in \{0,1\}, \; x_{j,k} \in [0,1] .$$

The problem $P2$ is to solve the sequencing problem that arises when the solution to $P1$ is fixed in P.

Approximate Benders Decomposition on MOP

We need to approximate the minimum makespan sequencing problem using $P1$ and ancillary variables. We achieve this by minimising the difference in workload allocated to each piece of equipment. Suppose that each piece of equipment is allocated a series of tasks, j, to complete for each job session, k, and that once these tasks are complete the equipment must wait until the remainder of the equipment is also complete before it may begin the next job session.

$P1$ has no variables relating to equipment, so we need to include some representation of equipment workloads. We first introduce continuous variables, $\delta^+_{i,i',k} (\delta^-_{i,i',k})$, to infer the minimum positive (negative) deviation between the job workload for any equipment pair (i,i'). Next, we estimate possible equipment to task allocations by creating preferential sets, $\mathcal{P}(i)$, heuristically. We allow each piece of equipment equipment to nominate its preferred task (e.g., closest) in turn, until all tasks are selected. These tasks are grounded in variables in the $P1$, so we define the deviation variables as follows:

$$\delta^+_{i,i',k} \geq \sum_{j \in \mathcal{P}(i)} C_b x_{j,k} - \sum_{j' \in \mathcal{P}(i')} C_{b'} x_{j',k} \quad \forall i, i', k , \qquad (35)$$

$$\delta^-_{i,i',k} \geq \sum_{j' \in \mathcal{P}(i')} C_{b'} x_{j',k} - \sum_{j \in \mathcal{P}(i)} C_b x_{j,k} \quad \forall i, i', k . \qquad (36)$$

We then minimise the bottleneck deviation for each job session, as follows:

$$\rho_k \geq \delta^j_{i,i',k} \qquad \forall i, i', k, j \in \{+,-\} . \qquad (37)$$

Thus, we obtain an objective function minimising the bottleneck in each job. That is, we add $\eta = \sum_k \rho_k$ and the constraints (35)–(37) to $P1$ to obtain $H1$.

Experiments

We performed all experiments single threaded on a 2.4GHz Intel Processor with processes time or memory-out after 2 hours or 2GB.

The Travelling Purchaser Problem

We solved the mixed-integer programming problems and sub-problems using Gurobi (v. 5.6). We used an A^*-based optimal planner guided by h_{max} heuristic (Bonet and Geffner 2001) to solve $H2$. We defined our test instances on a graph, where nodes represent markets that can only be visited once. We generate 10 random instances for every combination of markets (between 5 and 30), products (every fifth number between 30 and 50), and a special parameter *distval* (between 1 and 10), so that we obtain 14300 instances overall. We use the distval parameter to increase the relative difference between distances and product prices, where markets positions are random integers $\in (0, distval * 20)$, and product cost is a random number $\in (1, 10)/distval$. We select the demand randomly from $(1, 10)$ for each product, and select the quantity of product per market relative to the demand as follows: quantity$_{m,k}$ =demand$_k \times rand(0,1) \times randint(0,1)$. Data sets that were infeasible for the master were discarded.

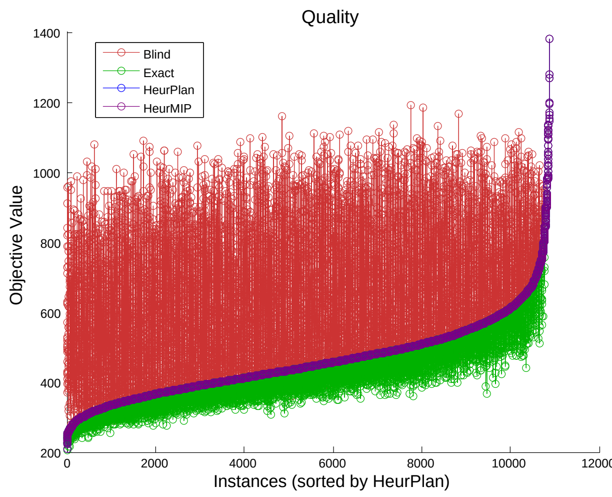

Figure 1: Solution impact on quality sorted by HeurMIP and HeurPlan (which obtained the same solution value). The dark (purple) curve captures the sorted HeurMIP values, which cover the HeurPlan values. Above the curve are the Blind values, and below are the Exact values.

We ran four experiments on this data set. The first solving approach is labelled 'Blind', which we obtain by solving $P1$ and $P2$ with a MIP solver. This approach is equivalent to a naive partitioning of the problem. The second solving approach is labelled 'Exact', which we obtain by solving the complete TPP problem with a MIP solver. Thus, we obtain bounds on the quality of our proposed methods. We ran two versions of our proposed method. The first is labelled 'HeurPlan', which corresponds to solving $H1$ with a MIP solver, and $H2$ with a basic optimal Planner. The second is labelled 'HeurMIP', which corresponds to solving both $H1$ and $H2$ with a MIP solver. We threw away instances for which HeurPlan could not solve $H2$ in 20 seconds (373 instances out of the total)—simply because MIP is the stronger technology for solving this partition, and we only provide the planning solution as an illustration of a decomposition that could benefit planning technology.

In Figure 1, we sort the solution quality by HeurMIP and HeurPlan, which obtained the same solution value. It is clear that the approximation is of good quality in general. Blind never beats HeurMIP or HeurPlan in terms of solution quality. The average quality of HeurMIP and HeurPlan is 1.0598 times the optimal solution; whereas Blind obtains an average of 1.4268 times the optimal solution.

	Average solve time (s) for number of markets										
Methods	20	21	22	23	24	25	26	27	28	29	30
Blind	0.05	0.06	0.08	0.07	0.09	0.10	0.14	0.17	0.18	0.19	0.19
HMIP	0.12	0.13	0.15	0.15	0.16	0.17	0.19	0.20	0.20	0.22	0.22
Hplan	0.43	0.37	0.47	0.43	0.50	0.49	0.53	0.53	0.47	0.43	0.46
Exact	1.72	2.22	3.18	2.25	2.51	4.36	4.65	6.61	6.81	7.68	7.48

Table 1: Experiment times for TPP example.

In Table 1, we present the computation times for each method. Blind is the fastest approach, which is expected since the approximation adds some complexity to the master. HeurMIP outperforms HeurPlan—this is also expected, as MIP technology is well tuned to solve the TSP, which is the subproblem. The Exact method, which we implemented with a lazy constraint callback on the subtour elimination constraints, performs the worst—16 times slower than HeurPlan, and 34 times slower than HeurMIP, on average.

Mining Operations Planning

We solved the MIPs and feasibility problems using CPLEX (v.12.6). We used temporal planners CPT and POPF (Vidal and Geffner 2006; Coles et al. 2010) to solve the mining operations problems, where actions can be executed concurrently with different durations. POPF solutions are suboptimal, while CPT and $A^*(h_{max})$ are optimal planners.

We provide experimental results for 5 real case studies provided by our industry partner and a synthetic benchmark testing the scalability of our approach.

Since $P1$ had no variables from the complete problem objective function, it became a feasibility problem. $H1$ became a MIP, but was still efficiently solved.

Minimising makespan is of extreme importance for our industrial partner, as each excavator, on average, can extract 3000 tonnes in 60 minutes—shorter makespans translate directly to increased overall production. In our experiments, the heuristic MIP improves makespan over $P1$ with either planner. Since CPT is optimal it yields a significantly shorter makespan than POPF, thereby evidencing the impact of computing optimal plans for each subproblem.

The 5 real case scenarios range from 22–75 blocks with 1173–3704 kt, with an average block size of 50kt varying from 5–150kt, and contaminants grade ranging from within the bounds up to 30 times outside the minimum and maximum blend targets. In all instances we permitted the average available equipment of 5 excavators.

The results of Table 2 over the real case scenarios are consistent with the results over the benchmark problems. $P1/POPF$ and $H1/POPF$ solve all instances. $P1/CPT$ and $H1/CPT$ solve 3 and 2 instances, respectively. $H1$ partitions render longer plans than $P1$ with both planners. As a result, the makespan of $H1$ partitions are significantly better than $P1$ with both planners. Remarkably, the approximate $H1$ MIP solutions computed with the suboptimal planner POPF yield a *much* smaller makespan than the solutions computed by the optimal planner CPT with just the feasible MIP P. While computing optimal plans improves the quality of the makespan, this result highlights the importance of finding 'good' partitions from blocks to stockpiles with the MIP solvers, which $H1$ is able to infer through the approximation of $P2$.

Conclusions

In this paper, we have presented a clear methodology for decomposing challenging problems in a mindful way. In particular, we show that we can approximate the correct projection effectively by using ancillary variables in a naive partitioning, and that even simple approximations of this kind can put the subproblem in a good space.

Our approach has the potential to be effective for any type of problem, but is of especial interest for those problems where devising strong 'no-good' cuts is challenging, or where obtaining good feasible solutions quickly is a priority.

Real-world problems can be difficult, and yet require prompt feasible solutions. In these cases, our approach fills the gap between the 'quick-and-dirty' approach, and the optimal Benders approach. Furthermore, our approach can be utilised in a more diverse range of problems, including non-convex and state-dependent problems. Our method can also be used to warm-start a Benders decomposition or Logic-based Benders decomposition if care has been taken to ensure the master is overestimating the correct projection.

Acknowledgments. This research was co-funded by the Australian Research Council linkage grant LP11010015 "Making the Pilbara Blend: Agile Mine Scheduling through Contingent Planning" and industry partner Rio Tinto. The authors thank two anonymous reviewers for providing insightful feedback that lead to an improved version of this paper.

Data				Makespan [minutes]			
Inst.	$\|B\|$	P	$\sum$	P1/POPF	H1/POPF	P1/CPT	H1/CPT
(1)	37	26	1614	12218	**9704**	—	—
(2)	23	8	1173	8131	**7130**	7805	—
(3)	25	0	1449	14906	9475	14659	**7835**
(4)	21	18	1375	12171	7645	10438	**6021**
(5)	66	68	3704	37405	**26761**	—	—

Table 2: The case study data and complete method results. In column (1) we list the instances. $|B|$ refers to the number of tasks. P refers to the number of precedences. $\sum$ is the total ore in kilotonnes. A "—" indicates a timeout

References

Benders, J. F. 1962. Partitioning procedures for solving mixed-variables programming problems. *Numerische Mathematik* 4:238–252.

Blum, A., and Furst, M. 1995. Fast planning through planning graph analysis. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence (IJCAI 1995)*, 1636–1642.

Bonet, B., and Geffner, H. 2001. Planning as heuristic search. *Artificial Intelligence* 129:5–33.

Coles, A.; Fox, M.; Long, D.; and Smith, A. 2008a. A hybrid relaxed planning graph'lp heuristic for numeric planning domains. In *Proceedings of the Eighteenth International Conference on Automated Planning and Scheduling, ICAPS-08*, 52–59.

Coles, A.; Fox, M.; Long, D.; and Smith, A. 2008b. Planning with problems requiring temporal coordination. In *Proceedings of the Eighteenth International Conference on Automated Planning and Scheduling, ICAPS-08*, 892–897.

Coles, A. J.; Coles, A.; Fox, M.; and Long, D. 2010. Forward-chaining partial-order planning. In *Proceedings of ICAPS*, 42–49.

Cormen, T. H.; Leiserson, C. E.; Rivest, R. L.; and Stein, C. 2001. *Introduction to Algorithms*. The MIT Press.

Fernández, S., and Borrajo, D. 2009. Solving clustered oversubscription problems for planning e-courses. In *Proceedings of SPARK Workshop*, volume 9.

Flórez, J. E.; de Reyna, A. T. A.; García, J.; López, C. L.; Olaya, A. G.; and Borrajo, D. 2011. Planning multi-modal transportation problems. In *Proceedings of ICAPS*, 66–73.

Gerevini, A. E.; Haslum, P.; Long, D.; Saetti, A.; and Dimopoulos, Y. 2009. Deterministic planning in the fifth international planning competition: PDDL3 and experimental evaluation of the planners. *Artificial Intelligence* 173(5–6):619–668.

Helmert, M., and Röger, G. 2008. How good is almost perfect? In *Proceedings of the Twenty-Third Conference on Artificial Intelligence,(AAAI 2008)*, 944–949.

Hooker, J. N., and Ottosson, G. 2003. Logic-based Benders decomposition. *Mathematical Programming* 96(1):33–60.

Hooker, J. N. 2007. Planning and scheduling by logic-based Benders decomposition. *Operations Research* 55(3):588–602.

Ivankovic, F.; Haslum, P.; Thiébaux, S.; Shivashankar, V.; and Nau, D. S. 2014. Optimal planning with global numerical state constraints. In *Proceedings of the Twenty-Fourth International Conference on Automated Planning and Scheduling, ICAPS-14*.

Kautz, H., and Selman, B. 1996. Pushing the envelope: Planning, propositional logic, and stochastic search. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence (AAAI 1996)*, 1194–1201. AAAI Press.

Laporte, G.; Riera-Ledesma, J.; and Salazar-González, J.-J. 2003. A branch-and-cut algorithm for the undirected traveling purchaser problem. *Operations Research* 51(6):940–951.

Lipovetzky, N.; Burt, C. N.; Pearce, A. R.; and Stuckey, P. J. 2014. Planning for mining operations with time and resource constraints. In *Proceedings of the Twenty-Fourth International Conference on Automated Planning and Scheduling, ICAPS-14*.

Melkote, S., and Daskin, M. S. 2001. Capacitated facility location/network design problems. *European journal of operational research* 129(3):481–495.

Piacentini, C.; Alimisis, V.; Fox, M.; and Long, D. 2013. Combining a temporal planner with an external solver for the power balancing problem in an electricity network. In *Proceedings of the Twenty-Third International Conference on Automated Planning and Scheduling, ICAPS-13*.

Richter, S., and Westphal, M. 2010. The LAMA planner: Guiding cost-based anytime planning with landmarks. *Journal of Artificial Intelligence Research (JAIR)* 39:127–177.

Vidal, V., and Geffner, H. 2006. Branching and pruning: An optimal temporal pocl planner based on constraint programming. *Artificial Intelligence* 170(3):298–335.