Hybrid Benders Decomposition Algorithms in Constraint Logic Programming

Andrew Eremin and Mark Wallace

IC-Parc
London, UK
{a.eremin, mgw}@icparc.ic.ac.uk

Abstract. Benders Decomposition is a form of hybridisation that allows linear programming to be combined with other kinds of algorithms. It extracts new constraints for one subproblem from the dual values of the other subproblem. This paper describes an implementation of Benders Decomposition, in the ECLiPSe language, that enables it to be used within a constraint programming framework. The programmer is spared from having to write down the dual form of any subproblem, because it is derived by the system. Examples are used to show how problem constraints can be modelled in an undecomposed form. The programmer need only specify which variables belong to which subproblems, and the Benders Decomposition is extracted automatically. A class of minimal perturbation problems is used to illustrate how different kinds of algorithms can be used for the different subproblems. The implementation is tested on a set of minimal perturbation benchmarks, and the results are analysed.

1 Introduction

1.1 Forms of Hybridisation

In recent years, research on combinatorial problem solving has begun to address real world problems which arise in industry and commerce [1–3]. These problems are often large scale, complex, optimisation (LSCO) problems and are best addressed by decomposing them into multiple subproblems. The optimal solutions of the different subproblems are invariably incompatible with each other, so researchers are now exploring ways of solving the subproblems in a way that ensures the solutions are compatible with each another - i.e. globally consistent. This research topic belongs to the area of "hybrid algorithms" [4, 5], but more specifically it addresses ways of making different solvers cooperate with each other. Following [6] we shall talk about "forms of hybridisation".
An early form of hybridisation is the communication between global constraints in constraint programming, via the finite domains of the shared variables. Different subproblems are handled by different global constraints (for example a scheduling subproblem by a cumulative constraint and a TSP subproblem by a cycle constraint [7]), and they act independently on the different subproblems yielding domain reductions. This is

a clean and sound hybridisation form because a domain reduction which is correct for a subproblem is necessarily correct for any larger problem in which the subproblem is contained.

1.2 Hybridisation Forms for Linear Programming

Master Problems and other Subproblems LSCO problems involve a cost function, and for performance reasons it is important to find solutions quickly that are not only feasible but also of low cost. Usually these cost functions are linear, or can be approximated by a linear or piecewise linear function. Linear programming offers efficient constraint solvers which can quickly return optimal solutions to problems whose cost function and constraints can be expressed using only linear expressions. Consequently most industrial LSCO problems involve one or more linear subproblems which are addressed using linear programming as available in commercial products such as XPRESS [8] and CPLEX [9].
Whilst global constraints classically return information excluding certain assignments from any possible solution, linear solvers classically return just a single optimal solution. In contrast with global constraints, the information returned by a linear solver for a subproblem does not necessarily remain true for any larger problem in which it is embedded. Thus linear solvers cannot easily be hybridised in the same way as global constraints.
Nevertheless several hybridisation forms have been developed for linear solvers, based on the concept of a "master" problem, for which the optimal solution is found, and other subproblems which interact with the master problem. In the simplest case this interaction is as follows. The subproblem examines the last optimal solution produced for the master problem, and determines whether this solution violates any of the constraints of the subproblem. If so the subproblem returns to the master problem one or more alternative linear constraints which could be added to the master problem to prevent this violation occurring again. One of these constraints is added to the master problem and a new optimal solution is found. To prove *global* optimality each of the alternatives are added to the master problem on different branches of a search tree. These alternatives should cover all possible ways of fixing the violation.
A generalisation of this form of hybridisation is "row generation" [10], where a new set of constraints ("rows") are added to the master problem at each node of the search tree. Unimodular probing [11] is an integration of a form of row generation into constraint programming.

Column Generation Another form of hybridisation for linear programming is *column generation* [12]. In this case the master problem is to find the optimal combination of "pieces" where each piece is itself a solution of another subproblem. A typical application of column generation is to crew scheduling: the assignment of crew to a bus or flight schedule over a day or a month. There are complex constraints on the sequence of activities that can be undertaken by a single crew, and these constraints are handled in a subproblem whose solutions are complete

tours which can be covered by a single crew over the time period. The master problem is the optimal combination of such tours. The master problem constraints enforce that each scheduled bus trip or flight must belong to one tour. Each tour is represented in the master problem by a variable, which corresponds to a column in the matrix representing the problem.
In the general case, each call to another subproblem returns a solution which has the potential to improve on the current optimum for the master problem. Each call to a subproblem adds a column to the master problem, and hence the name "column generation".
A number of applications of column generation have been reported in which the subproblem is solved by constraint programming [13, 14]. A column generation library has been implemented in the ECLiPSe constraint logic programming system, which allows both subproblem, communication of solutions and search to be specified and controlled from the constraint program.
While column generation utilises the *dual values* returned from convex solvers to form the optimisation function of a subproblem, a closely related technique exploits them to approximate subproblem constraints within the optimisation function of the master problem. This technique is known as *Lagrangian relaxation* and has been used for hybridising constraint programming and convex optimisation by Sellmann and Fahle [15] and Benoist et. al. [16] in[17].

Other Hybridisation Forms Besides optimal solutions, linear solvers can return several kinds of information about the solution. *Reduced costs* are the changes in the cost which would result from changes in the values of specific variables. These are, in fact, underestimates so if the reduced cost is "-10" the actual increase in cost will be greater than or equal to 10. In case the variable has finite domain, these reduced costs can be used to prune values from the domain in the usual style of a global constraint. (A value is pruned from the domain if the associated reduced cost is so bad it would produce a solution worse than the current optimum). In this way linear programming can be hybridised with other solvers in the usual manner of constraint programming. Indeed the technique has been used very successfully [18].

1.3 Benders Decomposition

Benders Decomposition is a hybridisation form based on the master problem/subproblem relationship. It makes use of an important and elegant aspect of mathematical programming, the dual problem [19]. Benders Decomposition is applicable when some of the constraints and part of the optimisation function exhibit duality. The master problem need not use mathematical programming at all. The subproblems return information which can be extracted by solving the dual. The new constraints that are added to the master problem are extracted from the dual values of the subproblems.

We have implemented Benders Decomposition in ECLiPSe and used it to tackle several commercial applications in transportation and telecommunications. The technique has proved very successful and has outperformed all other hybridisation forms in these applications.
For the purposes of this paper we have also used Benders Decomposition to tackle a set of benchmarks originally designed to test another hybridisation form, Unimodular Probing [11]. Whilst our results on these benchmarks have not been so striking as the applications mentioned above, they nicely illustrate the use of Benders Decomposition and the combination of linear programming with a simple propagation algorithm for the master problem. From these benchmarks we also make some observations about the kinds of problems and decompositions that are most suited to the hybrid form of Benders Decomposition.

1.4 Contents

In the following section we introduce Benders Decomposition, explain and justify it, and present the generic Benders Decomposition algorithm. In section 3 we show how it is embedded in constraint programming. We describe the user interface, and how one models a problem to use Benders Decomposition in ECLiPSe. We also describe how it is implemented in ECLiPSe. In section 4 we present the application of Benders Decomposition to a "minimal perturbation" problem, its definition, explanation and results on a set of benchmarks. Section 5 concludes and discusses the next application, further work on modeling and integration, and open issues.

2 Benders Decomposition

Benders decomposition is a cut or row generation technique for the solution of specially structured mixed integer linear programs that was introduced in the OR literature in [20]. Given a problem $\mathbf{P}$ over a set of variables V, if a subset X of the variables can be identified for which fixing their values results in one or more disconnected *SubProblems* ($\mathbf{SP_i}$) over the variable sets $Y_i : \bigcup_i Y_i = V - X$ which are easily soluble — normally due to some structural property of the resulting constraints — it may be beneficial to solve the problem by a two stage iterative procedure.
At each iteration k a *Relaxed Master Problem* ($\mathbf{RMP^k}$) in the complicating or connecting variables X is first solved and the solution assignment $X = X^k$ used to construct the subproblems $\mathbf{SP_i^k}$; these subproblems are then solved and the solutions used to tighten the relaxation of the master problem by introducing *Benders Cuts*, $\beta_i^k(X)$.
The subproblems optimise over reduced dimensionality subspaces $\mathcal{D}_{Y_i}^k$ of the original problem solution space obtained by fixing the variables $X = X^k$, while the master problem optimises over the optimal solutions of these subspaces augmented by X^k guided by the cuts generated.
In classical Benders Decomposition both the master and subproblems

are linear and are solved by MILP algorithms, while the cuts are derived from Duality theory. In general however, we are free to use any appropriate solution methods for master and subproblems — all that is required is an assignment of the master problem variables $X = X^k$ to construct convex subproblems, and a procedure for generating valid cuts from subproblem solutions. The most naive such scheme would merely result in the master problem enumerating all assignments of X, while more informative cuts can result in substantial pruning of the master problem search space.

2.1 Classical Benders Decomposition

Consider the linear program **P** given by:

$$\begin{aligned} \mathbf{P}: \ \min \ & \mathbf{f^T x} + \sum_{i=1}^{I} \mathbf{c_i^T y_i} \\ \text{subject to} \quad & \mathbf{G_i x} + \mathbf{A_i y_i} \geq \mathbf{b_i} \qquad \forall i \\ & \mathbf{x} \in \mathcal{D}_X \\ & \mathbf{y_i} \geq \mathbf{0} \qquad \forall i \end{aligned} \tag{1}$$

When $\mathbf{x}$ is fixed to some value $\mathbf{x^k}$ we have linear programs in $\mathbf{y_i}$ which may be specially structured or easy to solve, prompting us to partition the problem as follows:

$$\begin{aligned} \mathbf{P}: \quad & \min_{\mathbf{x} \in \mathcal{D}_X} \left\{ \mathbf{f^T x} + \sum_{i=1}^{I} \left(\min \left\{ \mathbf{c_i^T y_i} : \mathbf{A_i y_i} \geq \mathbf{b_i} - \mathbf{G_i x}, \mathbf{y_i} \geq \mathbf{0} \right\} \right) \right\} \\ = & \min_{\mathbf{x} \in \mathcal{D}_X} \left\{ \mathbf{f^T x} + \sum_{i=1}^{I} \left(\max \left\{ \mathbf{u_i}(\mathbf{b_i} - \mathbf{G_i x}) : \mathbf{u_i A_i} \leq \mathbf{c_i}, \mathbf{u_i} \geq \mathbf{0} \right\} \right) \right\} \end{aligned} \tag{2}$$

where the inner optimizations have been dualised. Given that $U_i = \{\mathbf{u_i} : \mathbf{u_i A_i} \leq \mathbf{c_i}, \mathbf{u_i} \geq \mathbf{0}\}$ is non-empty for each i either there is an extreme point optimal solution to each inner optimization or it is unbounded along an extreme ray; letting $\mathbf{u_i^1}, \ldots, \mathbf{u_i^{t_i}}$ and $\mathbf{d_i^1}, \ldots, \mathbf{d_i^{s_i}}$ be respectively the extreme points and directions of U_i we can rewrite (2) as the mixed integer *Master Problem* **MP**:

$$\begin{aligned} \mathbf{MP}: \ \min \ & z = \mathbf{f^T x} + \sum_{i=1}^{I} \beta_i \\ \text{subject to} \quad & \beta_i \geq \mathbf{u_i^k}(\mathbf{b_i} - \mathbf{G_i x}) \quad \forall i \quad \forall k \\ & 0 \geq \mathbf{d_i^l}(\mathbf{b_i} - \mathbf{G_i x}) \quad \forall i \quad \forall l \\ & \mathbf{x} \in \mathcal{D}_X \end{aligned} \tag{3}$$

Since there will typically be very many extreme points and directions of each U_i and thus constraints in (3) we solve relaxed master problems containing a subset of the constraints. If for some relaxed master problem $\mathbf{RMP^k}$ the optimal relaxed solution $(z^k, \mathbf{x^k})$ satisfies all the constraints of (3), then $(z^k, \mathbf{x^k}, \mathbf{y_1^k}, \ldots, \mathbf{y_I^k})$ is an optimal solution of (1); otherwise

there exists some constraint or *Benders Cut* in (3) which is violated for $\mathbf{x} = \mathbf{x^k}$ which we add to $\mathbf{RMP^k}$ to form $\mathbf{RMP^{k+1}}$ and iterate.
To determine such a cut or prove optimality we obtain the optimal solution $(\beta_i^k, \mathbf{u_i^k})$ of the *Subproblems* $\mathbf{SP_i^k}$ formed by fixing $\mathbf{x} = \mathbf{x^k}$ in (2):

$$\begin{array}{rl} \mathbf{SP_i^k}: \max & \beta_i^k = \mathbf{u_i}(\mathbf{b_i} - \mathbf{G_i x^k}) \\ \text{subject to} & \mathbf{u_i A_i} \leq \mathbf{c_i} \\ & \mathbf{u_i} \geq \mathbf{0} \end{array} \tag{4}$$

If any subproblem $\mathbf{SP_i^k}$ has an unbounded optimal solution for some $\mathbf{x^k}$ then the primal of the subproblem is infeasible for $\mathbf{x^k}$; if any subproblem $\mathbf{SP_i^k}$ is infeasible for some $\mathbf{x^k}$ then it is infeasible (and the primal of the subproblem is infeasible or unbounded) for any $\mathbf{x}$ since the (empty) feasible region U_i is independent of $\mathbf{x}$. In either case we proceed by considering the *Homogeneous Dual* of the primal of the subproblem:

$$\begin{array}{rl} \max & \mathbf{u_i}(\mathbf{b_i} - \mathbf{G_i x^k}) \\ \text{subject to} & \mathbf{u_i A_i} \leq \mathbf{0} \\ & \mathbf{u_i} \geq \mathbf{0} \end{array} \tag{5}$$

This problem is always feasible ($\mathbf{u_i} = \mathbf{0}$ is a solution), having an unbounded optimum precisely when the primal is infeasible and a finite optimal solution when the primal is feasible. In the unbounded case we can obtain a cut

$$\mathbf{u_i^k}(\mathbf{b_i} - \mathbf{G_i x}) \leq 0$$

corresponding to an extreme direction of $U_i' = \{\mathbf{u_i} : \mathbf{u_i A_i} \leq \mathbf{0}, \mathbf{u_i} \geq \mathbf{0}\}$. The complete Benders decomposition algorithm proceeds as follows:

Algorithm 1 *The Benders Decomposition Algorithm*

1. Initialisation step: From the original linear program $\mathbf{P}$ (1) construct the *relaxed master problem* $\mathbf{RMP^0}$ (3) with the initial constraint set $\mathbf{x} \in \mathcal{D}_X$ and set $k = 0$.
2. Iterative step: From the current relaxed master problem $\mathbf{RMP^k}$ with optimal solution $(z^k, \mathbf{x^k})$ construct $\mathbf{RMP^{k+1}}$ with optimal solution $(z^{k+1}, \mathbf{x^{k+1}})$: fix $\mathbf{x} = \mathbf{x^k}$ in $\mathbf{P}$, and solve the resulting *subproblems* $\mathbf{SP_i^k}$ (4); there are three cases to consider:
   (a) $\mathbf{SP_i^k}$ is primal unbounded for some i — halt with the original problem having unbounded solution.
   (b) $\mathbf{y_i^k}, \mathbf{u_i^k}$ are respectively primal and dual optimal solutions of subproblem $\mathbf{SP_i^k}$ with objective values β_i^k for each i — there are two cases to consider:
      i. $\sum_{i=1}^{I} \beta_i^k = z^k$ halt with $(z^k, \mathbf{x^k}, \mathbf{y_1^k}, \ldots, \mathbf{y_I^k})$ as the optimal solution to the original problem.
      ii. $\sum_{i=1}^{I} \beta_i^k > z^k$ add the *Benders Cuts* $\beta_i \geq \mathbf{u_i^k}(\mathbf{b_i} - \mathbf{G_i x})$ to $\mathbf{RMP^k}$ to form the new relaxed master problem $\mathbf{RMP^{k+1}}$ set $k = k + 1$ and return to (2).
   (c) $\mathbf{SP_i^k}$ is dual unbounded or both primal and dual infeasible for some i — find an extreme direction $\mathbf{d_i^k}$ of the homogeneous dual leading to unboundedness; add the cut $\mathbf{d_i^k}(\mathbf{b_i} - \mathbf{G_i x}) \leq 0$ to $\mathbf{RMP^k}$ to form the new relaxed master problem $\mathbf{RMP^{k+1}}$ set $k = k + 1$ and return to (2).

2.2 Hybrid Benders Decomposition

The classical linear Benders Decomposition can be generalised to cover problems in which the constraints and objective function are nonlinear, using any appropriate solution method for $\mathbf{RMP^k}$ and $\mathbf{SP_i^k}$ — we require only a procedure for generating valid lower bounds $\beta_i^k(x)$ from the solutions of $\mathbf{SP_i^k}$. In its most general form we have the original problem:

$$\begin{array}{rl} \mathbf{P}: \ \min & f(f_1(\mathbf{x},\mathbf{y_1}),\ldots,f_I(\mathbf{x},\mathbf{y_I})) \\ \text{subject to} & g_i(\mathbf{x},\mathbf{y_i}) \geq \mathbf{b_i} \quad \forall i \\ & \mathbf{x} \in \mathcal{D}_X \\ & \mathbf{y_i} \in \mathcal{D}_Y \quad \forall i \end{array} \tag{6}$$

which we decompose into the master problem:

$$\begin{array}{rl} \mathbf{MP}: \ \min & z = f(\mathbf{x},\beta_1\ldots,\beta_I) \\ \text{subject to} & \beta_i \geq \beta_i^k(\mathbf{x}) \quad \forall i \ \forall k \\ & 0 \geq \beta_i^l(\mathbf{x}) \quad \forall i \ \forall l \\ & \mathbf{x} \in \mathcal{D}_X \end{array} \tag{7}$$

and subproblems:

$$\begin{array}{rl} \mathbf{SP_i^k}: \ \min & f_i(\mathbf{x^k},\mathbf{y_i}) \\ \text{subject to} & g_i(\mathbf{x^k},\mathbf{y_i}) \geq \mathbf{b_i} \\ & \mathbf{y_i} \in \mathcal{D}_Y \end{array} \tag{8}$$

In particular when we can identify one or more distinct sets of variables in which the problem constraints and objective function are linear and a complicating set of variables, it will be useful to decompose the problem into a nonlinear relaxed master problem and linear subproblems.

3 Embedding Benders Decomposition in Constraint Programming

In this section we discuss the implementation of Benders Decomposition in ECLiPSe . In designing the structure of the implementation two important considerations were to maintain the flexibility of the approach and to ensure ease of use for non-mathematicians.

The flexibility of hybrid Benders Decomposition algorithms is due in large part to the possibility of using arbitrary solution methods for master and subproblems; in order to allow appropriate solvers to be simply slotted in to the framework it is essential to cleanly separate the method of solution of master and subproblems from the communication of solutions between them.

As many users of the solver may be unfamiliar with the intricacies of linear programming and duality theory, it is important to provide a user interface that allows for problems to be modeled in a natural and straightforward formulation. All constraints are therefore input in their original formulation — i.e. without having been decomposed and dualised and containing both master and subproblem variables. The sets of variables occurring solely in the subproblems are specified when the optimisation is performed, and the original problem constraints automatically decomposed into master and subproblem constraints and the subproblems dualised.

3.1 ECLiPSe Implementation

The implementation of Benders Decomposition in ECLiPSe uses the same features of the language that are used to implement finite domain and other constraints. These are demons, variable attributes, waking conditions, and priorities.
A demon is a procedure which, on completing its processing, suspends itself. It can be woken repeatedly, each time re-suspending on completion, until killed by an explicit command. Demons are typically used to implement constraint propagation. For Benders Decomposition a demon is used to implement the solver for the master problem, with separate demons for each subproblem.
A variable attribute is used to hold information about a variable, such as its finite domain. Programmers can add further attributes, and for Benders decomposition an attribute is used to hold a *tentative* value for each of the variables in the master problem. Each time the master problem is solved, the tentative values of all the variables are updated to record the new solution.
When the waking conditions for a demon are satisfied, it wakes. For a finite domain constraint this is typically a reduction in the domain of any of the variables in the constraint. For the subproblems in Benders Decomposition the waking condition is a change in the tentative values of any variable linking the subproblem to the master problem. Thus each time the master problem is solved any subproblem whose linking variables now have a new value is woken, and solved again. The master problem is woken whenever a new constraint (in the form of a *Benders cut*) is passed to the solver. Thus processing stops at some iteration either if after solving the master problem no subproblems are woken, or if after solving all the subproblems no new cuts are produced.
Priorities are used in ECLiPSe to ensure that when several demons are woken they are executed in order of priority. For finite domain propagation this is used to ensure that simple constraints, such as inequalities, are handled before expensive global constraints. By setting the subproblems at a higher priority than the master problem, it is ensured that all the subproblems are solved and the resulting Benders cuts are all added to the master problem, before the master problem itself is solved again. While it is possible to wake the master problem early with only some cuts added by setting lower priorities for subproblems, this proved ineffective in practice.

4 Benders Decomposition for Scheduling Problems

4.1 Minimal Perturbation in Dynamic Scheduling with Time Windows

The minimal perturbation dynamic scheduling problem with time windows and side constraints is a variant of the classic scheduling problem with time windows: given a current schedule for a set of n possibly variable duration tasks with time windows on their start and end time points,

a set $\mathcal{C}$ of unary and binary side constraints over these time points and a reduced number of resources r we are required to produce a new schedule feasible to the existing time windows and constraints and the new resource constraint that is minimally different from the current schedule. The user enters these problems in a simple form that is automatically translated into a set of constraints that can be passed to the `bd` library. For the purposes of this paper, in the next section we give the full model generated by the translator. The subsequent section reports how this model is split into a master/subproblem form for Benders Decomposition

4.2 The Constraints Modeling Minimal Perturbation

For each task T_i in the current schedule with current start and end times t_{s_i}, t_{e_i} respectively there are:

Time point variables for the start and end of the task s_i, e_i and **task duration constraints**

$$(s_i, e_i) \in \mathcal{L}_i \tag{9}$$

where $\mathcal{D}_i = \{(s,e) : e-s \geq l_i, e-s \leq u_i, l_{s_i} \leq s \leq u_{s_i},\ l_{e_i} \leq e \leq u_{e_i}\}$ and $l_{s_i}, u_{s_i}, l_{e_i}, u_{e_i}, l_i, u_i$ are derived from the time windows of the task start and end points and any constraints on these time points in $\mathcal{C}$.

Perturbation cost variables c_{s_i}, c_{e_i} and **perturbation cost constraints**

$$(c_{s_i}, s_i, c_{e_i}, e_i) \in \mathcal{P}_i \tag{10}$$

where $\mathcal{P}_i = \{(c_s, s, c_e, e) : c_s \geq s - t_{s_i}\ ,\ c_s \geq t_{s_i} - s,\ c_e \geq e - t_{e_i},\ c_e \geq t_{e_i} - e\}$ so that $c_{s_i} \geq |s_i - t_{s_i}|, c_{e_i} \geq |e_i - t_{e_i}|$

For each pair of tasks T_i, T_j there are:

Binary non-overlap variables $Pre_{ij}, Post_{ij}$ for each task $T_j \neq T_i$ which take the value 1 iff task i starts before the start of task j and after the end of task j respectively, so that we have

$$Pre_{ij} = \begin{cases} 1 \text{ if } s_i < s_j \\ 0 \text{ if } s_i \geq s_j \end{cases} \qquad Post_{ij} = \begin{cases} 1 \text{ if } s_i \geq e_j \\ 0 \text{ if } s_i < e_j \end{cases}$$

and the distances between the time points s_i and s_j, e_j are bounded by

$$\begin{array}{l} s_i - s_j \geq \left(l_{s_i} - u_{s_j}\right) Pre_{ij} \\ s_i - s_j \leq \left(l_{s_j} - u_{s_i} - 1\right) Pre_{ij} + \left(u_{s_i} - l_{s_j}\right) \\ s_i - e_j \geq \left(u_{e_j} - l_{s_i}\right) Post_{ij} + \left(l_{s_i} - u_{e_j}\right) \\ s_i - e_j \leq \left(u_{s_i} - l_{e_j} + 1\right) Post_{ij} - 1 \end{array} \tag{11}$$

The resource feasibility constraint that the start time point s_i overlaps with at most r other tasks

$$\sum_{j \neq i} \left(Pre_{ij} + Post_{ij}\right) \geq n - r - 1 \tag{12}$$

Time point distance constraints between s_i, e_i and all other time points. Since for each task $T_j \neq T_i$ we have the distance bounds (11) between s_i and T_j and between s_j and T_i of which at most half can be binding, we combine them with the binary constraints

$$\begin{array}{ll} s_i \geq s_j + b_{ij} & e_j \geq s_i + b_{u_{ij}} \\ s_i \geq e_j + b_{l_{ij}} & e_i \geq e_j + b_{e_{ij}} \end{array}$$

appearing in the constraint set $\mathcal{C}$ to give the distance constraints

$$\begin{array}{c} (s_i, e_i, s_j, e_j, B_{ij}, L_{ij}, U_{ij}) \in \mathcal{D}_{ij} \\ (s_i, e_i, s_j, e_j, B_{ij}, L_{ij}, U_{ij}, Pre_{ij}, Pre_{ji}, Post_{ij}) \in \mathcal{O}_{ij} \end{array} \tag{13}$$

where

$$\begin{aligned} \mathcal{D}_{ij} = \{&(s_i, e_i, s_j, e_j, B, L, U) : \\ &s_i - s_j \geq B, s_i - e_j \geq L, -s_i + e_j \geq U, e_i - e_j \geq b_{e_{ij}}\} \\ \mathcal{O}_{ij} = \{&(s_i, e_i, s_j, e_j, B, L, U, Pre_{ij}, Pre_{ji}, Post_{ij}) : \\ &B \geq b_{ij}, L \geq b_{l_{ij}}, U \geq b_{u_{ij}}, \\ &B \geq \left(l_{s_i} - u_{s_j}\right) Pre_{ij}, B \geq \left(u_{s_j} - l_{s_i} + 1\right) Pre_{ji} + \left(l_{s_i} - u_{s_j}\right), \\ &L \geq \left(u_{e_j} - l_{s_i}\right) Post_{ij} + \left(l_{s_i} - u_{e_j}\right), \\ &U \geq \left(l_{e_j} - u_{s_i} - 1\right) Post_{ij} + 1\} \end{aligned}$$

Valid ordering constraints for each task $T_j \neq T_i$ there are many additional constraints that we may choose to introduce restricting the binary variables to represent a valid ordering. These constraints are not necessary for the correctness of the algorithm as invalid orderings will be infeasible to the subproblem, but may improve its efficiency as fewer iterations will be needed.

The complete MILP problem formulation is then

$$\mathbf{P}: \min \sum_{i=1}^{n} (c_{s_i} + c_{e_i})$$

subject to

$$\left.\begin{array}{rcl} (c_{s_i}, s_i, c_{e_i}, e_i) & \in & \mathcal{P}_i \\ (s_i, e_i) & \in & \mathcal{L}_i \\ \left.\begin{array}{rcl} (s_i, e_i, s_j, e_j, B_{ij}, L_{ij}, U_{ij}) & \in & \mathcal{D}_{ij} \\ (s_i, e_i, s_j, e_j, B_{ij}, L_{ij}, U_{ij}, Pre_{ij}, Pre_{ji}, Post_{ij}) & \in & \mathcal{O}_{ij} \end{array}\right\} \forall j \neq i \\ \sum_{j \neq i} (Pre_{ij} + Post_{ij}) & \geq & n - r - 1 \end{array}\right\} \forall i \tag{14}$$

4.3 Benders Decomposition Model for Minimal Perturbation

Master Problem

$$\mathbf{MP}: \min z$$

subject to

$$\begin{array}{rcll} \beta^k(\mathbf{B}, \mathbf{L}, \mathbf{U}) & \leq & z & \forall k \\ \beta^l(\mathbf{B}, \mathbf{L}, \mathbf{U}) & \leq & 0 & \forall l \\ \left.\begin{array}{rcl} (s_i, e_i, s_j, e_j, B_{ij}, L_{ij}, U_{ij}, Pre_{ij}, Pre_{ji}, Post_{ij}) & \in & \mathcal{O}_{ij} \ \forall j \neq i \\ \sum_{j \neq i} (Pre_{ij} + Post_{ij}) & \geq & n - r - 1 \end{array}\right\} & & \forall i \end{array} \tag{15}$$

Subproblem There is a single subproblem with primal formulation

$$\begin{array}{l} \mathbf{LP^k}: \min \sum_{i=1}^{n} (c_{s_i} + c_{e_i}) \\ \text{subject to} \\ \left.\begin{array}{r} (c_{s_i}, s_i, c_{e_i}, e_i) \in \mathcal{P}_i \\ (s_i, e_i) \in \mathcal{L}_i \\ (s_i, e_i, s_j, e_j, B_{ij}, L_{ij}, U_{ij}) \in \mathcal{D}_{ij} \ \forall j \neq i \end{array}\right\} \forall i \end{array} \tag{16}$$

The Benders Decomposition library in ECLiPSe automatically extracts a dual formulation of the subproblem. For the current subproblem $\mathbf{LP^k}$, the dual has the form:

$$\begin{array}{l} \mathbf{SP^k}: \max \sum_{i=1}^{n} \left(\alpha_i + \sum_{j \neq i} \left(B_{ij} w_{B_{ij}} + L_{ij} w_{L_{ij}} + U_{ij} w_{U_{ij}} \right) \right) \\ \text{subject to} \\ \left.\begin{array}{r} \sum_{j \neq i} \left(w_{B_{ij}} + w_{L_{ij}} - w_{U_{ij}} - w_{B_{ji}} \right) \\ + w_{t_{s_i}} - w_{l_i} + w_{u_i} + w_{l_{s_i}} - w_{u_{s_i}} \leq 0 \\ \\ \sum_{j \neq i} \left(w_{b_{e_{ij}}} - w_{L_{ji}} - w_{U_{ji}} - w_{b_{e_{ji}}} \right) \\ + w_{t_{e_i}} + w_{l_i} - w_{u_i} + w_{l_{e_i}} - w_{u_{e_i}} \leq 0 \\ w_{t_{s_i}}, w_{t_{e_i}} \geq -1 \\ w_{t_{s_i}}, w_{t_{e_i}} \leq 1 \\ w_{l_i}, w_{u_i}, w_{l_{s_i}}, w_{u_{s_i}}, w_{l_{e_i}}, w_{u_{e_i}} \geq 0 \\ w_{B_{ij}}, w_{L_{ij}}, w_{U_{ij}}, w_{b_{e_{ij}}} \geq 0 \ \forall j \neq i \end{array}\right\} \forall i \end{array} \tag{17}$$

where

$$\begin{array}{rl} \alpha_i = & t_{s_i} w_{t_{s_i}} + t_{e_i} w_{t_{e_i}} + l_i w_{l_i} + u_i w_{u_i} + \sum_{j \neq i} b_{e_{ij}} w_{b_{e_{ij}}} + \\ & l_{s_i} w_{l_{s_i}} - u_{s_i} w_{u_{s_i}} + l_{e_i} w_{l_{e_i}} - u_{e_i} w_{u_{e_i}} \end{array}$$

Solutions to $\mathbf{SP^k}$ produce cuts of the form $z \geq \beta^k(\mathbf{B}, \mathbf{L}, \mathbf{U})$ which exclude orderings with worse cost from further relaxed master problems when the subproblem is feasible, or $\beta^k(\mathbf{B}, \mathbf{L}, \mathbf{U}) \leq 0$ which exclude orderings infeasible to the start windows and durations of the tasks when the subproblem is infeasible, where

$$\beta^k(\mathbf{B}, \mathbf{L}, \mathbf{U}) = \sum_{i=1}^{I} \left(\alpha_i^k + \sum_{j \neq i} \left(w_{B_{ij}}^k B_{ij} + w_{L_{ij}}^k L_{ij} + w_{U_{ij}}^k U_{ij} \right) \right)$$

All coefficients $\mathbf{w^k}$ and constants α_i^k in the cuts are integral since the subproblems are totally unimodular.

4.4 Results and Discussion

Summary We ran this model on 100 minimal perturbation problem instances. The number of variables in the problem model was around 900, and there were some 1400 constraints in the master problem and around 20 in the subproblem. Most problems were solved within 10 iterations

between master and subproblem, though a few notched up hundreds of iterations.
The time and number of iterations for each problem are given in Table 1. The bulk of the time was spent in the finite domain search used to solve the master problem. Typically, for the feasible instances, the optimal solution was found early in the search, and much time was wasted in generating further solutions to the master problem which were not better in the context of the full problem.
Correct and optimal solutions to all the problems were returned, but the performance was an order of magnitude slower than the specially designed algorithm presented in [11].

Analysis Minimal perturbation can be decomposed into a master and subproblem for the Benders Decomposition approach, but the size of the problems is very disparate. The behaviour of the algorithm on the benchmark problem reflect the number of constraints - the subproblems are trivial and almost all the time is spent in the master problem. The imbalance is probably an indication that this algorithm is better suited to problems with larger or more complex subproblems.
Nevertheless it is not always the *number* of constraints that make a problem hard, but the difficulty of handling these constraints. It may be that the master problem constraints, while numerous, are easy to handle if the right algorithm is used.
Currently the algorithm used to solve the master problem is a two-phase finite domain labelling routine. In the first phase a single step lookahead is used to instantiate binary variables that cannot take one of their values. In the second step all the binary variables are labelled, choosing first the variables at the bottleneck of the minimal perturbation scheduling problem. This is not only a relatively naive search method, but it also lacks any active handling of the optimisation function. Linear programming does offer an active handling of the optimisation function. Thus, using a hybrid algorithm to tackle the master problem within a larger Benders Decomposition hybridisation form, could be very effective on these minimal perturbation problems.
Benders Decomposition has proven to be a very efficient and scalable approach in case the problem breaks down into a master problem and multiple subproblems. The minimal perturbation problems benchmarked in this paper involve a single kind of resource. These problems do not have an apparent decomposition with multiple subproblems. This is a second reason why our benchmark results do not compete with the best current approach, on this class of problems. Minimal perturbation problems involving different kinds of resources might, by contrast, prove to be very amenable to the Benders Decomposition form of hybridisation.

5 Conclusion

This paper has investigated hybridisation forms for problems that admit a decomposition. A variety of hybridisation forms can be used in case

Problem	Iterations	Time	Problem	Iterations	Time	Problem	Iterations	Time
1	11	4.92	35	4	1.09	69	26	39.48
2	12	3.16	36	20	7.06	70	13	4.86
3	10	2.40	37	22	20.91	71	-	>200
4	15	11.30	38	36	67.48	72	-	>200
5	16	7.93	39	59	184.57	73	-	>200
6	58	109.22	40	13	5.66	74	26	18.72
7	25	19.82	41	28	27.05	75	91	154.00
8	10	3.27	42	9	5.86	76	12	3.49
9	32	16.25	43	39	21.02	77	54	111.17
10	107	151.01	44	25	9.43	78	35	37.52
11	-	>200	45	11	5.20	79	44	38.00
12	-	>200	46	-	>200	80	10	3.56
13	44	96.77	47	5	1.37	81	28	12.69
14	29	18.30	48	51	51.75	82	8	2.01
15	70	83.87	49	9	2.06	83	16	14.52
16	20	30.96	50	18	8.80	84	32	22.24
17	23	11.65	51	30	19.44	85	20	4.94
18	18	15.16	52	43	119.66	86	-	>200
19	14	4.94	53	28	26.10	87	18	9.56
20	21	8.17	54	33	17.32	88	12	4.72
21	19	5.01	55	14	6.01	89	7	2.26
22	60	180.47	56	14	9.95	90	43	42.51
23	20	8.46	57	45	100.94	91	8	2.12
24	39	82.93	58	4	0.88	92	54	111.5
25	13	2.74	59	8	2.45	93	-	>200
26	3	0.71	60	-	>200	94	25	8.08
27	10	7.14	61	19	9.41	95	8	2.99
28	22	12.23	62	24	11.48	96	22	10.97
29	27	13.24	63	-	>200	97	5	1.59
30	-	>200	64	46	95.07	98	6	2.37
31	42	36.69	65	30	18.62	99	15	4.82
32	15	4.48	66	14	5.57	100	19	47.61
33	15	8.77	67	10	3.10			
34	20	23.70	68	62	132.87			

Table 1. Number of iterations and total solution time for Benders Decomposition on RFP benchmark data

one or more subproblems are handled by linear programming. We aim to make them all available in the ECLiPSe language in a way that allows users to experiment easily with the different alternatives so as to quickly find the best hybrid algorithm for the problem at hand.
Benders Decomposition is a technique that has not, to date, been applied to many real problems within the CP community. Publications on this technique have described a few pedagogical examples and "academic" problem classes such as satisfiability [20, 21]. This paper presents the first application of Benders Decomposition to a set of minimal perturbation problems which have immediate application in the real world. Indeed the benchmarks were based on an industrial application to airline scheduling.
The significance of Benders Decomposition in comparison with other master/subproblem forms of hybridisation (such as row and column generation) is that it takes advantage of linear duality theory. The Benders Decomposition library in ECLiPSe harnesses the power of the dual problem for constraint programmers who may not find the formulation and application of the linear dual either easy or natural.
Moreover the implementation of Benders Decomposition in ECLiPSe has been proven both efficient and scalable. Indeed its results on the minimal perturbation benchmark problems compare reasonably well even against an algorithm specially developed for problems of this class. However the Benders Decomposition for minimal perturbation problems comprises a master problem and a single trivial subproblem. Our experience with this technique has shown that this hybridisation form is more suitable to applications where the decomposition introduces many or complex subproblems.
This paper was initially motivated by a network application where Benders Decomposition has proven to be the best hybridisation form after considerable experimentation with other algorithms. We plan to report on the application of this technique to a problem brought to us by an industrial partner in a forthcoming paper.
There remains further work to support fine control over the iteration between the master and subproblems in Benders Decomposition. The importance of such fine control has been clearly evidenced from our ECLiPSe implementation of another hybridisation form - column generation - applied to mixed integer problems. In particular we will seek to implement early stopping, and more control over the number of Benders cuts returned at an iteration.

References

1. Chic-2 - creating hybrid algorithms for industry and commerce. ESPRIT PROJECT 22165: http://www.icparc.ic.ac.uk/chic2/, 1999.
2. Parrot - parallel crew rostering. ESPRIT PROJECT 24 960: http://www.uni-paderborn.de/ parrot/, 2000.
3. Liscos - large scale integrated supply chain optimisation software. http://www.dash.co.uk/liscosweb/, 2001.
4. *CP98 Workshop on Large Scale Combinatorial Optimisation and Constraints*, volume 1, Pisa, Italy, 1999. http://www.elsevier.nl/gejng/31/29/24/25/23/show/Products/notes/index.htt.

5. *CP99 Workshop on Large Scale Combinatorial Optimisation and Constraints*, volume 4, Alexandra, Virginia, USA, 2000. http://www.elsevier.nl/gejng/31/29/24/29/23/show/Products/notes/index.htt.
6. H. H. El Sakkout. *Improving Backtrack Search: Three Case Studies of Localized Dynamic Hybridization*. PhD thesis, Imperial College, London University, 1999.
7. N. Beldiceanu and E. Contjean. Introducing global constraints in CHIP. *Mathematical and Computer Modelling*, 12:97–123, 1994.
8. XPRESS-MP. http://www.dash.co.uk/, 2000.
9. CPLEX. http://www.ilog.com/products/cplex/, 2000.
10. R. E. Gomory. An algorithm for integer solutions to linear programs. In R. L. Graves and P. Wolfe, editors, *Recent Advances in Mathematical Programming*, pages 269–302. McGraw-Hill, 1963.
11. H. H. El Sakkout and M. G. Wallace. Probe backtrack search for minimal perturbation in dynamic scheduling. *Constraints*, 5(4):359–388, 2000.
12. L. H. Appelgren. A column generation algorithm for a ship scheduling problem. *Transportation Science*, 3:53–68, 1969.
13. U. Junker, S. E. Karisch, N. Kohl, B. Vaaben, T. Fahle, and M. Sellmann. A framework for constraint programming based column generation. In *Proceedings of the 5th International Conference on Principles and Practice of Constraint Programming - LNCS 1713*, pages 261–274. Springer-Verlag, 1999.
14. T. H. Yunes, A. V. Moura, and C. C. de Souza. A hybrid approach for solving large scale crew scheduling problems. In *Proceedings of the Second International Workshop on Practical Aspects of Declarative Languages (PADL'00)*, pages 293–307, Boston, MA, USA, 2000.
15. M. Sellmann and T. Fahle. Cp-based lagrangian relaxation for a multimedia application. In *[17]*, 2001.
16. T. Benoist, F. Laburthe, and B. Rottembourg. Lagrange relaxation and constraint programming collaborative schemes for travelling tournament problems. In *[17]*, 2001.
17. *CP-AI-OR01 Workshop on Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems*, Wye, Kent, UK, 2001. http://www.icparc.ic.ac.uk/cpAIOR01/.
18. F. Focacci, A. Lodi, and M. Milano. Embedding relaxations in global constraints for solving TSP and its time constrained variant. *Annals of Mathematics and Artificial Intelligence*, Special issue on Large Scale Combinatorial Optimization, 2001.
19. G. B. Dantzig. *Linear Programming and Extensions*. Princeton University Press, 1963.
20. J. F. Benders. Partitioning procedures for solving mixed variables programming problems. *Numerische Mathematik*, 4:238–252, 1962.
21. J. N. Hooker and G. Ottosson. Logic-based benders decomposition. http://ba.gsia.cmu.edu/jnh/papers.html, 1999.