Generating Benders Cuts for a General Class of Integer Programming Problems

Yingyi Chu and Quanshi Xia

IC-Parc, Imperial College London,
London SW7 2AZ, UK
{yyc, q.xia}@imperial.ac.uk

Abstract. This paper proposes a method of generating valid integer Benders cuts for a general class of *integer* programming problems. A *generic* valid Benders cut in disjunctive form is presented first, as a basis for the subsequent derivations of simple valid cuts. Under a qualification condition, a *simple valid* Benders cut in linear form can be identified. A cut generation problem is formulated to elicit it. The simple valid Benders cut is further generalized to a *minimally relaxed* Benders cut, based on which a complete Benders decomposition algorithm is given, and its finite convergency to optimality is proved. The proposed algorithm provides a way of applying the Benders decomposition strategy to solve *integer programs*. The computational results show that using the Benders algorithm for integer programs to exploit the problem structures can reduce the solving time more and more as the problem size increases.

1 Introduction

Benders decomposition is a strategy for solving large-scale optimization problems [1]. The variables of the problem are partitioned into two sets: *master problem variables* and *subproblem variables*. The Benders algorithm iteratively solves a *master problem*, which assigns tentative values for the master problem variables, and a *subproblem*, obtained by fixing the master problem variables to the tentative values. In every iteration, the subproblem solution provides certain information on the assignment of master problem variables. Such information is expressed as a *Benders cut*, cutting off some assignments that are not acceptable. The Benders cut is then added to the master problem, narrowing down the search space of master problem variables and eventually leading to optimality. On one hand, Benders method is employed to exploit the problem structure: the problem is decomposed into a series of independent smaller subproblems, reducing the complexity of solving it [2]. On the other hand, Benders method opens a dimension for 'hybrid algorithms' [3, 4] where the master problem and the subproblems can be solved with different methods.

The generation of Benders cuts is the core of Benders decomposition algorithm. Indeed, *valid* Benders cuts guarantee the convergence of the iterations to the optimal solution of the original problem, and also the cuts determine how fast the algorithm converges.

The classic Benders decomposition algorithm [5] was proposed for linear programming problems, the cut generation of which is based on the strong duality property of linear programming [2]. Geoffrion has extended it to a larger class of mathematical programming problems [6].

For integer programming, however, it is difficult to generate valid integer Benders cut, due to the duality gap of integer programming in subproblems. One possible way is to use the *no-good* cut to exclude only the current tentative assignment of master problem variables that is unacceptable. Such no-good Benders cuts will result in an enumerative search and thus a slow convergence. For some specific problems, better Benders cuts can be obtained [7]. For example, in the machine scheduling application, the cut that limits the incompatible jobs in the same machine is generally stronger. For more general integer programming, logic-based Benders decomposition [8] was proposed to generate valid integer Benders cuts, but these cuts contain a large number of disjunctions [9], the linearization of which leads to huge cuts with many auxiliary variables, complicating the master problem significantly.

This paper proposes a new method of generating valid Benders cuts for a class of integer programs, in which the objective function only contains the master problem variables.

As a foundation of our derivation, a *generic* valid integer Benders cut is firstly presented. However it is difficult to use due to its exponential size and nonlinearity. Instead, we can pick only one linear inequality from the disjunction of the generic cut, while preserving the validity. A qualification condition is then given, with which such a simple valid cut can be identified. A cut generation problem is formulated to determine the simple linear valid cut.

However, such an integer Benders cut is not always available. The *minimally relaxed* cut is then proposed as a generalization of it, obtainable in all cases. The simple integer Benders cut is just a special case of the minimally relaxed cut with 'zero' relaxation. Based on this, a complete Benders decomposition algorithm is presented, and its finite convergency to optimality is proved.

The paper is organized as follows. Section 2 introduces the integer programs under consideration and the principle of the Benders decomposition algorithm. Section 3 derives the integer Benders cut. Section 4 generalizes to the minimally relaxed integer Benders cut. Section 5 presents the complete Benders decomposition algorithm that could be used in practice. Section 6 gives computational results. Section 7 concludes the paper. The appendix gives the proofs of all the lemmas.

2 Preliminaries

2.1 Integer Programs

The programs we consider in the paper are written as the following form (Such programs arise from our study on a path generation application in network traffic engineering [10]):

$$P: \quad \max_{\boldsymbol{y},\boldsymbol{x}} \ \boldsymbol{c}^T\boldsymbol{y}$$
$$s.t. \quad \begin{cases} \boldsymbol{D}\boldsymbol{y} \leq \boldsymbol{d}, \\ \boldsymbol{A}\boldsymbol{y} + \boldsymbol{B}\boldsymbol{x} \leq \boldsymbol{b}, \\ \boldsymbol{y} \in \{0,1\}^m, \boldsymbol{x} \in \{0,1\}^n \ . \end{cases}$$

The problem variables are partitioned into two vectors $\boldsymbol{y}$ (master problem variables) and $\boldsymbol{x}$ (subproblem variables), and the objective function only contains the master problem variables.

By the use of Benders decomposition, the problem P is decomposed into the master problem (MP) that solves only the $\boldsymbol{y}$ variables and the subproblem ($SP'(\bar{\boldsymbol{y}})$) that solves only the $\boldsymbol{x}$ variables, by fixing the $\boldsymbol{y}$ variables to the master problem solution, denoted by $\bar{\boldsymbol{y}}$.

$$MP: \quad \max_{\boldsymbol{y}} \ \boldsymbol{c}^T\boldsymbol{y} \qquad\qquad SP'(\bar{\boldsymbol{y}}): \quad \max_{\boldsymbol{x}} \ 0$$
$$s.t. \quad \begin{cases} \boldsymbol{D}\boldsymbol{y} \leq \boldsymbol{d}, \\ \text{Benders cuts}, \\ \boldsymbol{y} \in \{0,1\}^m, \end{cases} \qquad s.t. \quad \begin{cases} \boldsymbol{B}\boldsymbol{x} \leq \boldsymbol{b} - \boldsymbol{A}\bar{\boldsymbol{y}}, \\ \boldsymbol{x} \in \{0,1\}^n, \end{cases}$$

where the Benders cuts in MP are gradually added during the iterations. Note that the subproblem is a feasibility problem with a dummy objective.

2.2 Principle of Benders Decomposition Algorithm

The Benders decomposition algorithm iteratively solves the master problem and the subproblem. In each iteration k, the master problem ($MP^{(k)}$) sets a tentative value for the master problem variables ($\bar{\boldsymbol{y}}^{(k)}$), with which the subproblem ($SP'(\bar{\boldsymbol{y}}^{(k)})$) is formed and solved. Using the subproblem solution, a valid Benders cut over the $\boldsymbol{y}$ variables is constructed and added to the master problem in the next iteration ($MP^{(k+1)}$). The Benders cut added in every iteration cuts off some infeasible assignments, thus the search space for the $\boldsymbol{y}$ variables is gradually narrowed down as the algorithm proceeds, leading to optimality.

Algorithm 1. *Benders Decomposition Algorithm*

1. *Initialization. Construct the initial master problem $MP^{(0)}$ without any Benders cut. Set $k = 0$.*
2. *Iteration.*
   (a) *Solve $MP^{(k)}$. If it is feasible, obtain the optimal solution $\bar{\boldsymbol{y}}^{(k)}$ and the optimal objective $\phi_{MP}^{(k)}$. Otherwise, the original problem is infeasible; set $\phi_{MP}^{(k)} = -\infty$ and go to exit.*
   (b) *Construct the subproblem $SP'(\bar{\boldsymbol{y}}^{(k)})$. If the subproblem is feasible, then optimality is found; obtain the optimal solution $\bar{\boldsymbol{x}}^{(k)}$, and go to exit.*
   (c) *Cut Generation Procedure. Generate a valid integer Benders cut and add it to the master problem to construct $MP^{(k+1)}$. Set $k = k + 1$ and go back to step 2.*

3. *Exit. Return the current* $\phi_{MP}^{(k)}$ *as the optimal objective. If* $\phi_{MP}^{(k)} = -\infty$ *then the problem* P *is infeasible. If not, return the current* $(\bar{\boldsymbol{y}}^{(k)}, \bar{\boldsymbol{x}}^{(k)})$ *as the optimal solution of problem* P*.*

The Cut Generation Procedure *2(c)* is not specified. This is the key step for the algorithm, which must be selected carefully so that the algorithm eventually converges to the optimality of the original problem in finite iterations.

2.3 Always Feasible Subproblem

The subproblem $SP'(\bar{\boldsymbol{y}})$ can be reformulated to an equivalent one that is always feasible:

$$\begin{aligned} SP(\bar{\boldsymbol{y}}): \quad & \min_{\boldsymbol{x},\boldsymbol{r}} \ \mathbf{1}^T \boldsymbol{r} \\ s.t. \quad & \begin{cases} \boldsymbol{B}\boldsymbol{x} - \boldsymbol{r} \le \boldsymbol{b} - \boldsymbol{A}\bar{\boldsymbol{y}}, \\ \boldsymbol{x} \in \{0,1\}^n, \boldsymbol{r} \ge \mathbf{0}, \end{cases} \end{aligned}$$

which simply introduces slack variables $\boldsymbol{r}$ to the constraints of $SP'(\bar{\boldsymbol{y}})$. Obviously, $SP'(\bar{\boldsymbol{y}})$ is feasible *iff* $SP(\bar{\boldsymbol{y}})$ has 0 optimal value. In the Algorithm 1, once the objective of $SP(\bar{\boldsymbol{y}})$ equals 0 during iteration, the algorithm terminates (at step *2(b)*), and the optimal solution is found.

Dual values are very useful in the cut generation for linear programming. For integer programming, however, we need to introduce the *fixed subproblems* and their duals. A fixed subproblem is constructed from *any* feasible assignment $\tilde{\boldsymbol{x}}$ of subproblem $SP(\bar{\boldsymbol{y}})$. It just constrains the $\boldsymbol{x}$ variables to be equal to a *given* feasible $\tilde{\boldsymbol{x}}$.

$$\begin{aligned} SP_f(\bar{\boldsymbol{y}}, \tilde{\boldsymbol{x}}): \quad & \min_{\boldsymbol{x},\boldsymbol{r}} \ \mathbf{1}^T \boldsymbol{r} \\ s.t. \quad & \begin{cases} \boldsymbol{B}\boldsymbol{x} - \boldsymbol{r} \le \boldsymbol{b} - \boldsymbol{A}\bar{\boldsymbol{y}}, \\ \boldsymbol{x} = \tilde{\boldsymbol{x}}, \\ \boldsymbol{x}\text{:free}, \boldsymbol{r} \ge \mathbf{0} \ . \end{cases} \end{aligned} \tag{1}$$

The dual of fixed subproblem $SP_f(\bar{\boldsymbol{y}}, \tilde{\boldsymbol{x}})$ is:

$$\begin{aligned} DSP_f(\bar{\boldsymbol{y}}, \tilde{\boldsymbol{x}}): \quad & \max_{\boldsymbol{u},\boldsymbol{v}} \ (\boldsymbol{A}\bar{\boldsymbol{y}} - \boldsymbol{b})^T \boldsymbol{u} + \tilde{\boldsymbol{x}}^T \boldsymbol{v} \\ s.t. \quad & \begin{cases} -\boldsymbol{B}^T \boldsymbol{u} + \boldsymbol{v} = \mathbf{0}, \\ \boldsymbol{u} \le \mathbf{1}, \\ \boldsymbol{v}\text{:free}, \boldsymbol{u} \ge \mathbf{0} \ . \end{cases} \end{aligned} \tag{2}$$

The optimal solution of $DSP_f(\bar{\boldsymbol{y}}, \tilde{\boldsymbol{x}})$ depends on the value of $\tilde{\boldsymbol{x}}$ (while the feasible region of it does not). Let $(\tilde{\boldsymbol{u}}, \tilde{\boldsymbol{v}})$ denote the corresponding optimal solution of $DSP_f(\bar{\boldsymbol{y}}, \tilde{\boldsymbol{x}})$. Since any value $\tilde{\boldsymbol{x}} \in \{0,1\}^n$ is feasible for $SP(\bar{\boldsymbol{y}})$ (2^n possible combinations), there are $N = 2^n$ possible fixed subproblems, each with its dual.

Given $\tilde{\boldsymbol{x}}$, the fixed subproblem $SP_f(\bar{\boldsymbol{y}}, \tilde{\boldsymbol{x}})$ is itself a linear program. Therefore, strong duality holds for $SP_f(\bar{\boldsymbol{y}}, \tilde{\boldsymbol{x}})$ and $DSP_f(\bar{\boldsymbol{y}}, \tilde{\boldsymbol{x}})$. Furthermore, if $\tilde{\boldsymbol{x}}^*$ is an optimal solution of $SP(\bar{\boldsymbol{y}})$, then $SP(\bar{\boldsymbol{y}})$, $SP_f(\bar{\boldsymbol{y}}, \tilde{\boldsymbol{x}}^*)$ and $DSP_f(\bar{\boldsymbol{y}}, \tilde{\boldsymbol{x}}^*)$ have the same optimal value.

Relations between the optimal primal and dual solutions of $SP_f(\bar{\boldsymbol{y}}, \tilde{\boldsymbol{x}})$ and $DSP_f(\bar{\boldsymbol{y}}, \tilde{\boldsymbol{x}})$ can be established via the *complementary condition*, that is, the Karush-Kuhn-Tucker (KKT) [2] constraints:

$$\begin{cases} \boldsymbol{u}_i(-\boldsymbol{B}\boldsymbol{x} + \boldsymbol{r} - \boldsymbol{A}\bar{\boldsymbol{y}} + \boldsymbol{b})_i = 0 & \forall i, \\ \boldsymbol{r}_i(\boldsymbol{u} - \boldsymbol{1})_i = 0 & \forall i, \end{cases} \tag{3}$$

together with the primal and dual constraints, (1) and (2).

Consider the complementary condition constraints. First, with the equation $\boldsymbol{x} = \tilde{\boldsymbol{x}}$ of (1) and the equation $-\boldsymbol{B}^T\boldsymbol{u} + \boldsymbol{v} = 0$ of (2), variables $\boldsymbol{x}$ and $\boldsymbol{v}$ can be replaced by $\tilde{\boldsymbol{x}}$ and $\boldsymbol{B}^T\boldsymbol{u}$. Secondly, the equation $\boldsymbol{r}_i(\boldsymbol{u}-\boldsymbol{1})_i = 0$ of (3) means that $\boldsymbol{r}_i = \boldsymbol{r}_i\boldsymbol{u}_i$. Putting this into the equation $\boldsymbol{u}_i(-\boldsymbol{B}\boldsymbol{x} + \boldsymbol{r} - \boldsymbol{A}\bar{\boldsymbol{y}} + \boldsymbol{b})_i = 0$ of (3), we get $\boldsymbol{u}_i(-\boldsymbol{B}\tilde{\boldsymbol{x}})_i + \boldsymbol{r}_i = \boldsymbol{u}_i(\boldsymbol{A}\bar{\boldsymbol{y}} - \boldsymbol{b})_i$. The complementary condition constraints can then be simplified to:

$$\begin{cases} \boldsymbol{B}\tilde{\boldsymbol{x}} - \boldsymbol{r} \leq \boldsymbol{b} - \boldsymbol{A}\bar{\boldsymbol{y}}, \\ \boldsymbol{u}_i(-\boldsymbol{B}\tilde{\boldsymbol{x}})_i + \boldsymbol{r}_i = \boldsymbol{u}_i(\boldsymbol{A}\bar{\boldsymbol{y}} - \boldsymbol{b})_i & \forall i, \\ \boldsymbol{r}_i(\boldsymbol{u} - \boldsymbol{1})_i = 0 & \forall i, \\ \boldsymbol{r} \geq \boldsymbol{0}, \boldsymbol{0} \leq \boldsymbol{u} \leq \boldsymbol{1} \ . \end{cases}$$

Finally, we can show the *redundancy* of $\boldsymbol{r}_i(\boldsymbol{u} - \boldsymbol{1})_i = 0$ in the following lemma (Note that the proofs of all the lemmas are given in the appendix):

Lemma 1. *The constraint $\boldsymbol{r}_i(\boldsymbol{u} - \boldsymbol{1})_i = 0$ is redundant in the presence of the constraints:*

$$\begin{cases} \boldsymbol{B}\tilde{\boldsymbol{x}} - \boldsymbol{r} \leq \boldsymbol{b} - \boldsymbol{A}\bar{\boldsymbol{y}}, \\ \boldsymbol{u}_i(-\boldsymbol{B}\tilde{\boldsymbol{x}})_i + \boldsymbol{r}_i = \boldsymbol{u}_i(\boldsymbol{A}\bar{\boldsymbol{y}} - \boldsymbol{b})_i & \forall i, \\ \boldsymbol{r} \geq \boldsymbol{0}, \boldsymbol{0} \leq \boldsymbol{u} \leq \boldsymbol{1} \ . \end{cases} \tag{4}$$

Thus, the complementary condition constraints are finally simplified to (4).

3 Integer Benders Cut Generation

3.1 Generic Valid Integer Benders Cut

In general, the Benders cut is a logic expression over the $\boldsymbol{y}$ variables, generated using the information from the subproblem solution. The valid Benders cut must guarantee that Algorithm 1 finitely converges to the optimal solution. We define the *valid* Benders cut for integer programming.

Definition 1. *In a certain iteration of the Benders algorithm, a valid Benders cut is a logic expression over the master problem variables $\boldsymbol{y}$ that satisfies:*

Condition 1. *if the current master problem solution $\bar{\boldsymbol{y}}$ is infeasible, then the cut must exclude at least $\bar{\boldsymbol{y}}$;*

Condition 2. *any feasible assignment of $\boldsymbol{y}$ variables must satisfy the cut.*

Condition 1 guarantees finite convergence since $\boldsymbol{y}$ has a finite domain. *Condition 2* guarantees optimality since the cut never cuts off feasible solutions.

Lemma 2. *If a valid Benders cut is generated in every iteration (at step 2(c)), then Algorithm 1 finitely converges to the optimality of the original program P.*

Using the solutions of all N fixed subproblems (denoted by $SP_f(\bar{\boldsymbol{y}}, \tilde{\boldsymbol{x}}^{\boldsymbol{i}})$, $\forall i = 1, \cdots, N$), a *generic* integer Benders cut can be obtained as the disjunction of N linear inequalities:

$$\begin{array}{ll} & (\boldsymbol{A}\boldsymbol{y} - \boldsymbol{b})^T \tilde{\boldsymbol{u}}^{\boldsymbol{1}} + (\tilde{\boldsymbol{x}}^{\boldsymbol{1}})^T \tilde{\boldsymbol{v}}^{\boldsymbol{1}} \leq 0 \\ \vee & (\boldsymbol{A}\boldsymbol{y} - \boldsymbol{b})^T \tilde{\boldsymbol{u}}^{\boldsymbol{2}} + (\tilde{\boldsymbol{x}}^{\boldsymbol{2}})^T \tilde{\boldsymbol{v}}^{\boldsymbol{2}} \leq 0 \\ \cdots & \cdots \\ \vee & (\boldsymbol{A}\boldsymbol{y} - \boldsymbol{b})^T \tilde{\boldsymbol{u}}^{\boldsymbol{N}} + (\tilde{\boldsymbol{x}}^{\boldsymbol{N}})^T \tilde{\boldsymbol{v}}^{\boldsymbol{N}} \leq 0, \end{array} \tag{5}$$

where $\{\tilde{\boldsymbol{x}}^{\boldsymbol{1}}, \cdots, \tilde{\boldsymbol{x}}^{\boldsymbol{N}}\}$ are the list of all the possible $\tilde{\boldsymbol{x}}$ values (i.e. an enumeration of $\{0,1\}^n$). For each $\tilde{\boldsymbol{x}}^{\boldsymbol{i}}$, $(\tilde{\boldsymbol{u}}^{\boldsymbol{i}}, \tilde{\boldsymbol{v}}^{\boldsymbol{i}})$ are the corresponding optimal dual solutions from $DSP_f(\bar{\boldsymbol{y}}, \tilde{\boldsymbol{x}}^{\boldsymbol{i}})$.

Similar to the Benders cut for linear programming, each linear inequality in the disjunction follows the expression of the objective function of $DSP_f(\bar{\boldsymbol{y}}, \tilde{\boldsymbol{x}})$. However, for integer programming, where a duality gap exists, we use a large number of dual fixed subproblem solutions, instead of a single dual subproblem solution for linear programming where no duality gap exists.

Lemma 3. *The generic integer Benders cut (5) is a valid cut.*

The generic cut (5) is valid, but it has a nonlinear (disjunctive) form and intrinsically contains all possible $\tilde{\boldsymbol{x}}$ combinations. Although it has theoretical value, it is difficult to use it directly in practical algorithms.

3.2 Integer Benders Cut

Under certain conditions, one of the linear inequalities from the disjunction (5) can still be a valid cut. In such cases the valid integer Benders cut becomes a simple linear inequality and the nonlinear disjunction disappears. We give the following sufficient condition under which such a simple valid cut can be identified:

Theorem 1. *If there exists a solution $\tilde{\boldsymbol{x}} \in \{\tilde{\boldsymbol{x}}^{\boldsymbol{1}}, \cdots, \tilde{\boldsymbol{x}}^{\boldsymbol{N}}\}$ such that $\tilde{\boldsymbol{x}}$ and the corresponding dual $\tilde{\boldsymbol{v}}$ satisfy*

$$\tilde{\boldsymbol{x}}^T \tilde{\boldsymbol{v}} \leq \boldsymbol{x}^T \tilde{\boldsymbol{v}} \quad \forall\ \boldsymbol{x} \in \{\tilde{\boldsymbol{x}}^{\boldsymbol{1}}, \cdots, \tilde{\boldsymbol{x}}^{\boldsymbol{N}}\}, \tag{6}$$

then the linear inequality

$$(\boldsymbol{A}\boldsymbol{y} - \boldsymbol{b})^T \tilde{\boldsymbol{u}} + \tilde{\boldsymbol{x}}^T \tilde{\boldsymbol{v}} \leq 0 \tag{7}$$

is a valid integer Benders cut.

Proof. (*for the valid cut condition 1*) If $\bar{\boldsymbol{y}}$ is infeasible for the subproblem, then $SP(\bar{\boldsymbol{y}})$ has positive objective value. Thus, any possible $SP_f(\bar{\boldsymbol{y}}, \boldsymbol{x})$ ($\boldsymbol{x} \in \{\tilde{\boldsymbol{x}}^{\boldsymbol{1}}, \cdots, \tilde{\boldsymbol{x}}^{\boldsymbol{N}}\}$) has a positive objective value, and so do all $DSP_f(\bar{\boldsymbol{y}}, \boldsymbol{x})$. Therefore, all linear inequalities in (5) are violated by $\bar{\boldsymbol{y}}$. In particular, $(\boldsymbol{A}\boldsymbol{y} - \boldsymbol{b})^T \tilde{\boldsymbol{u}} + \tilde{\boldsymbol{x}}^T \tilde{\boldsymbol{v}} \leq 0$ is violated by $\bar{\boldsymbol{y}}$, that is, the cut (7) excludes the infeasible $\bar{\boldsymbol{y}}$.
(*for the valid cut condition 2*) Let $\hat{\boldsymbol{y}}$ be any feasible solution. There must exist a corresponding $(\hat{\boldsymbol{x}}, \hat{\boldsymbol{u}}, \hat{\boldsymbol{v}})$ such that $SP_f(\hat{\boldsymbol{y}}, \hat{\boldsymbol{x}})$ and $DSP_f(\hat{\boldsymbol{y}}, \hat{\boldsymbol{x}})$ has 0 objective value as $(\boldsymbol{A}\hat{\boldsymbol{y}} - \boldsymbol{b})^T \hat{\boldsymbol{u}} + \hat{\boldsymbol{x}}^T \hat{\boldsymbol{v}} = 0$. Since the feasible region of all $DSP_f(\boldsymbol{y}, \boldsymbol{x})$ are identical and independent of the values of $\boldsymbol{y}$ and $\boldsymbol{x}$, the values of $(\tilde{\boldsymbol{u}}, \tilde{\boldsymbol{v}})$ which are the *optimal* solution for $DSP_f(\bar{\boldsymbol{y}}, \tilde{\boldsymbol{x}})$ are also a *feasible* solution for $DSP_f(\hat{\boldsymbol{y}}, \hat{\boldsymbol{x}})$. Therefore,

$$(\boldsymbol{A}\hat{\boldsymbol{y}} - \boldsymbol{b})^T \tilde{\boldsymbol{u}} + \hat{\boldsymbol{x}}^T \tilde{\boldsymbol{v}} \leq (\boldsymbol{A}\hat{\boldsymbol{y}} - \boldsymbol{b})^T \hat{\boldsymbol{u}} + \hat{\boldsymbol{x}}^T \hat{\boldsymbol{v}} = 0 \quad .$$

From the condition (6), we have $\tilde{\boldsymbol{x}}^T \tilde{\boldsymbol{v}} \leq \hat{\boldsymbol{x}}^T \tilde{\boldsymbol{v}}$. Therefore,

$$(\boldsymbol{A}\hat{\boldsymbol{y}} - \boldsymbol{b})^T \tilde{\boldsymbol{u}} + \tilde{\boldsymbol{x}}^T \tilde{\boldsymbol{v}} \leq (\boldsymbol{A}\hat{\boldsymbol{y}} - \boldsymbol{b})^T \tilde{\boldsymbol{u}} + \hat{\boldsymbol{x}}^T \tilde{\boldsymbol{v}} \leq 0,$$

which means that the feasible $\hat{\boldsymbol{y}}$ is not cut off by (7). □

If one can find an $\tilde{\boldsymbol{x}}$ such that the condition (6) holds, then the single linear inequality from the disjunction (5) that corresponds to $\tilde{\boldsymbol{x}}$ is a valid integer Benders cut by itself. However, the condition (6) involves not only the selected $\tilde{\boldsymbol{x}}$, but also all other possible assignments of $\boldsymbol{x}$, making it difficult to express (6) as a simple constraint. But the sufficient condition (6) can be converted to an equivalent *sign condition*.

Lemma 4. *Inequalities (6) are satisfied iff the following holds:*

$$\begin{array}{l} \tilde{\boldsymbol{x}}_i = 1 \Longrightarrow \tilde{\boldsymbol{v}}_i \leq 0, \\ \tilde{\boldsymbol{x}}_i = 0 \Longrightarrow \tilde{\boldsymbol{v}}_i \geq 0, \end{array} \qquad \forall i = 1, \cdots, n \quad . \tag{8}$$

The above sign condition can be enforced as the constraints:

$$\begin{cases} \tilde{\boldsymbol{x}}_i \tilde{\boldsymbol{v}}_i \leq 0 & \forall i, \\ (\boldsymbol{1} - \tilde{\boldsymbol{x}})_i \tilde{\boldsymbol{v}}_i \geq 0 & \forall i \quad . \end{cases} \tag{9}$$

Unlike the condition (6), the sign condition (9) only involves the selected $\tilde{\boldsymbol{x}}$ itself and the corresponding $\tilde{\boldsymbol{v}}$.

3.3 Integer Benders Cut Generation

The integer Benders cut generation problem is to find a $\tilde{\boldsymbol{x}}$ such that the sign condition (9) is satisfied. We formulate a Cut Generation Program (CGP) to elicit it.

The sign condition relates $\tilde{\boldsymbol{x}}$, an assignment that determines $SP_f(\bar{\boldsymbol{y}}, \tilde{\boldsymbol{x}})$, and $\tilde{\boldsymbol{v}}$, the optimal dual solution from $DSP_f(\bar{\boldsymbol{y}}, \tilde{\boldsymbol{x}})$. The constraints between them

are established by (4). Therefore, the program CGP is composed of constraints (4), the sign condition (9) and a dummy objective function.

$$
\begin{aligned}
CGP(\bar{\boldsymbol{y}}): \quad & \min_{\tilde{\boldsymbol{x}},\boldsymbol{r},\boldsymbol{u}} \ 0 \\
s.t. \quad & \begin{cases}
\boldsymbol{B}\tilde{\boldsymbol{x}} - \boldsymbol{r} \leq \boldsymbol{b} - \boldsymbol{A}\bar{\boldsymbol{y}}, & \\
\boldsymbol{u}_i(-\boldsymbol{B}\tilde{\boldsymbol{x}})_i + \boldsymbol{r}_i = \boldsymbol{u}_i(\boldsymbol{A}\bar{\boldsymbol{y}} - \boldsymbol{b})_i & \forall i, \\
\tilde{\boldsymbol{x}}_i(\boldsymbol{B}^T\boldsymbol{u})_i \leq 0 & \forall i, \\
(\boldsymbol{1} - \tilde{\boldsymbol{x}})_i(\boldsymbol{B}^T\boldsymbol{u})_i \geq 0 & \forall i, \\
\tilde{\boldsymbol{x}} \in \{0,1\}^n, \boldsymbol{r} \geq \boldsymbol{0}, \boldsymbol{0} \leq \boldsymbol{u} \leq \boldsymbol{1} \ . &
\end{cases}
\end{aligned}
\tag{10}
$$

Note that in CGP $\tilde{\boldsymbol{x}}$ (together with $\boldsymbol{r}$, $\boldsymbol{u}$) is a *variable*. The CGP solves for a value for $\tilde{\boldsymbol{x}}$, which, together with the dual values, satisfies the sign condition. If such a solution is found, a corresponding Benders cut is immediately obtained as (7).

Because $\tilde{\boldsymbol{x}}_i \in \{0,1\}$, all the bilinear terms $\tilde{\boldsymbol{x}}_i\boldsymbol{u}_j$ in the CGP can be linearized by introducing the variables $\boldsymbol{w}_{ij} = \tilde{\boldsymbol{x}}_i\boldsymbol{u}_j$ as:

$$
\begin{cases}
\boldsymbol{w}_{ij} \leq \tilde{\boldsymbol{x}}_i, \ \ \boldsymbol{w}_{ij} \leq \boldsymbol{u}_j, & \\
\boldsymbol{w}_{ij} \geq \tilde{\boldsymbol{x}}_i + \boldsymbol{u}_j - 1, \ \ \boldsymbol{w}_{ij} \geq 0, & \forall\, i,j \ .
\end{cases}
$$

Thus, CGP can be in practice solved with MIP solvers such as XPRESS [12].

Note that the CGP is not necessarily feasible due to the enforcement of the additional sign condition constraints (9), and hence the integer Benders cut (7) is not always available in each iteration. Therefore, we need to generalize the cut in order to give a complete Benders decomposition Algorithm 1.

4 Relaxed Integer Benders Cut

4.1 Relaxation

When the sign condition (8) does not hold, one cannot directly use an inequality from (5) as the valid cut. However, we can still select one inequality but relax it to some extent so that the sign condition is satisfied. This provides a generalized way of constructing a valid Benders cut.

In fact, *any* inequality from the disjunction (5):

$$
(\boldsymbol{A}\boldsymbol{y} - \boldsymbol{b})^T\tilde{\boldsymbol{u}} + \tilde{\boldsymbol{x}}^T\tilde{\boldsymbol{v}} \leq 0 \tag{11}
$$

can be relaxed by *inverting* the $\tilde{\boldsymbol{x}}$ values for those elements that violate the sign condition (8) as follows:

$$
\tilde{\boldsymbol{x}}'_i = \begin{cases}
\tilde{\boldsymbol{x}}_i & \text{if (6) is satisfied for the } i\text{th element,} \\
1 - \tilde{\boldsymbol{x}}_i & \text{otherwise.}
\end{cases}
$$

In such way $\tilde{\boldsymbol{x}}'$ and $\tilde{\boldsymbol{v}}$ satisfy the sign condition, and the *relaxed cut* is given by:

$$
(\boldsymbol{A}\boldsymbol{y} - \boldsymbol{b})^T\tilde{\boldsymbol{u}} + (\tilde{\boldsymbol{x}}')^T\tilde{\boldsymbol{v}} \leq 0 \ . \tag{12}
$$

Lemma 5. *The relaxed cut (12) satisfies the valid cut condition 2, and the relaxation gap from (11) to (12) is* $\sum_{i=1}^{n}(\tilde{\boldsymbol{x}} - \tilde{\boldsymbol{x}}')_i \tilde{\boldsymbol{v}}_i \geq 0$.

Note that (12) does not necessarily satisfy the valid cut condition 1, that is, it may not cut off the infeasible $\bar{\boldsymbol{y}}$ in the current iteration due to the relaxation. In this case, however, it can be easily remedied by adding a no-good cut that excludes only one point (the infeasible $\bar{\boldsymbol{y}}$):

$$\sum_{j=1}^{m} \bar{\boldsymbol{y}}_j (1 - \boldsymbol{y}_j) + \sum_{j=1}^{m} (1 - \bar{\boldsymbol{y}}_j) \boldsymbol{y}_j \geq 1 \ . \tag{13}$$

4.2 Relaxed Cut Generation

Since *any* inequality from the disjunction (5) can produce a relaxed cut, one can even avoid solving the CGP during iterations. Only the subproblem $SP(\bar{\boldsymbol{y}})$ is solved to find a solution $\tilde{\boldsymbol{x}}$, and $DSP_f(\bar{\boldsymbol{y}}, \tilde{\boldsymbol{x}})$ is solved to find the duals $\tilde{\boldsymbol{u}}$ and $\tilde{\boldsymbol{v}}$. Then a relaxed cut (12), derived from this $\tilde{\boldsymbol{x}}$, can be generated. To ensure that the valid cut condition 1 is met, the value of $\bar{\boldsymbol{y}}$ is checked against the relaxed cut (12). If it does violate (12), then (12) itself is a valid Benders cut that satisfies valid cut condition 1 and 2. If not, the conjunction of (12) and (13) constitutes a valid Benders cut.

The advantage of such a way of cut generation is its simplicity, since no CGP is involved. The disadvantage is that the selection of the inequality to be relaxed is rather arbitrary, and the generated cut can be loose. In particular, cut (7), which is a tight cut that needs no relaxation, may exist but not be found.

Therefore it is desirable to find a *minimally relaxed cut*, that is, its corresponding relaxation gap (as is given in Lemma 5) is made as small as possible, and thus the cut is as tight as possible. This is indeed a generalization of the valid Benders cut (7), which is just the special case when the minimum relaxation needed is zero.

The minimally relaxed cut can be generated by solving a Relaxed Cut Generation Program CGP_r, constructed by introducing slack variables $(\boldsymbol{p}, \boldsymbol{q})$ to the sign condition constraints of CGP.

$$\begin{aligned} CGP_r(\bar{\boldsymbol{y}}): \quad & \min_{\tilde{\boldsymbol{x}}, \boldsymbol{r}, \boldsymbol{u}, \boldsymbol{p}, \boldsymbol{q}} \ \mathbf{1}^T \boldsymbol{p} + \mathbf{1}^T \boldsymbol{q} \\ s.t. \quad & \begin{cases} \boldsymbol{B}\tilde{\boldsymbol{x}} - \boldsymbol{r} \leq \boldsymbol{b} - \boldsymbol{A}\bar{\boldsymbol{y}}, & \\ \boldsymbol{u}_i(-\boldsymbol{B}\tilde{\boldsymbol{x}})_i + \boldsymbol{r}_i = \boldsymbol{u}_i(\boldsymbol{A}\bar{\boldsymbol{y}} - \boldsymbol{b})_i & \forall i, \\ \tilde{\boldsymbol{x}}_i(\boldsymbol{B}^T \boldsymbol{u})_i - \boldsymbol{p}_i \leq 0 & \forall i, \\ (\mathbf{1} - \tilde{\boldsymbol{x}})_i(\boldsymbol{B}^T \boldsymbol{u})_i + \boldsymbol{q}_i \geq 0 & \forall i, \\ \tilde{\boldsymbol{x}} \in \{0,1\}^n, \boldsymbol{r} \geq \mathbf{0}, \mathbf{0} \leq \boldsymbol{u} \leq \mathbf{1}, & \\ \boldsymbol{p}, \boldsymbol{q} \geq 0 \ . & \end{cases} \end{aligned} \tag{14}$$

As the program CGP, after simple linearization this program is solvable in practice with MIP solvers.

Lemma 6. *If the optimal solution of CGP_r is $(\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{u}}, \tilde{\boldsymbol{v}})$, and the optimal objective value is ϕ_{CGP_r}, then:*

$$\phi_{CGP_r} = \sum_{i=1}^{n} (\tilde{\boldsymbol{x}} - \tilde{\boldsymbol{x}}')_i \tilde{\boldsymbol{v}}_i \quad .$$

Since the right hand side of the above equation is just the relaxation gap and it is minimized, the derived cut (12) is a minimally relaxed cut. In particular, if the optimal objective value of CGP_r is 0, then all the sign condition constraints are satisfied, and no relaxation is necessary. In this case the minimally relaxed cut is reduced to the basic valid Benders cut (7).

In practice, the $CGP_r(\bar{\boldsymbol{y}})$ is solved in every iteration (provided the algorithm does not terminate from step *2(a)* or *2(b)* before the cut generation). Its optimal solution gives a minimally relaxed cut as (12). According to Lemma 5, cut (12) satisfies the valid cut condition 2. If the optimal value is greater than 0, then the current (infeasible) assignment of master problem variables $\bar{\boldsymbol{y}}$ is checked against the cut. If the cut is violated, then (12) by itself satisfies both the valid cut conditions. If not, the conjunction of (12) and the no-good cut (13) constitutes a valid Benders cut.

5 Complete Algorithm

Based on the proposed integer Benders cut, the unspecified cut generation step *2(c)* in Algorithm 1 can now be given as:

Procedure 1. *Cut Generation Procedure (step 2(c) of Algorithm 1)*
Construct the cut generation program $CGP_r(\bar{\boldsymbol{y}}^{(k)})$. Solve it to obtain the optimal solution $(\tilde{\boldsymbol{x}}^{(k)}, \tilde{\boldsymbol{u}}^{(k)}, \tilde{\boldsymbol{v}}^{(k)})$ and its optimal objective value $\phi^{(k)}_{CGP_r}$. Generate the minimally relaxed cut:

$$(\boldsymbol{A}\boldsymbol{y} - \boldsymbol{b})^T \tilde{\boldsymbol{u}}^{(k)} + (\tilde{\boldsymbol{x}}')^{(k)T} \tilde{\boldsymbol{v}}^{(k)} \leq 0 \quad .$$

There are three cases:

A. *if $\phi^{(k)}_{CGP_r} = 0$, then $\tilde{\boldsymbol{x}}'^{(k)} = \tilde{\boldsymbol{x}}^{(k)}$, the above cut is reduced to (7), which is the valid Benders cut.*
B. *if $\phi^{(k)}_{CGP_r} > 0$ and the current $\bar{\boldsymbol{y}}^{(k)}$ violates the above cut, then this cut is the valid Benders cut by itself.*
C. *if $\phi^{(k)}_{CGP_r} > 0$ and the current $\bar{\boldsymbol{y}}^{(k)}$ satisfies the above cut, then this cut, in conjunction with the no-good cut (13), is the valid Benders cut.*

Add the generated Benders cut to the master problem to construct $MP^{(k+1)}$. Set $k = k + 1$ and go back to step 2 of Algorithm 1.

Replacing step *2(c)* of Algorithm 1 with the above procedure, we have a complete Benders decomposition algorithm.

Theorem 2. *The Benders Decomposition Algorithm 1, with its step 2(c) instantiated by the Cut Generation Procedure 1, terminates in finite steps and returns the optimal solution of the original program P.*

The proof is trivial according to Lemma 2, since in all the three cases the cut being generated satisfies the valid cut condition 1 and 2.

6 Computational Experiments

This section presents computational results of using Benders decomposition with the proposed integer cuts in integer programming problems. The algorithm is implemented using the ECLiPSe [11] platform. The test problems have bordered block structure in their coefficient matrices, so that the Benders algorithm can decompose the subproblem. The coefficients are generated randomly, and 20 cases are computed for each problem size configuration. The minimally relaxed cut derived from the CGP_r (14) of Sect. 4.2 is used in the tests.

Table 1. Computational Results using Minimally Relaxed Cut

SPV	MPV	#Iter		#NG	Sol.Time		MP%	SP%	CGP%	MIP.Time	#WIN
		avr	max	(avr)	avr	max	(avr)	(avr)	(avr)	(avr)	
300	100	10.40	14	0	84.50	132.06	5.1	3.6	91.3	785.07	7/20
300	150	11.55	16	0	104.04	187.39	16.2	3.4	80.4	109.84	6/20
300	200	12.40	20	0	136.11	260.56	27.1	2.9	70.0	176.20	13/20
300	250	13.30	25	0	195.67	562.64	46.9	2.1	51.0	318.35	12/20
400	100	12.10	18	0	152.03	245.67	3.4	3.9	92.7	1343.05	13/20
400	150	15.20	28	0.05	229.08	566.93	12.2	3.7	84.1	1697.62	17/20
400	200	14.50	21	0	215.85	434.22	19.4	3.5	77.1	889.04	19/20
400	250	18.05	30	0	371.47	851.96	35.1	2.8	62.1	3655.31	20/20
500	100	15.05	23	0.05	302.98	546.46	2.6	4.0	93.4	6482.65	20/20
500	150	18.20	36	0.10	409.43	873.56	7.9	3.8	88.3	8673.01	20/20
500	200	19.15	39	0.05	483.66	1441.65	15.3	3.4	81.3	8595.30	20/20
500	250	21.40	43	0.10	643.67	1929.80	30.7	3.0	66.3	10059.28	20/20

Table 1 summarizes the computational results for different problem sizes. The number of constraints is fixed to 300 and the number of blocks in the subproblem matrix is fixed to 10. Thus, the subproblem is decomposed into 10 smaller independent problems, each of which can generate a Benders cut in every iteration. We vary the number of master problem variables (MPV) and that of subproblem variables (SPV). For each problem size configuration, the average and maximum number of iterations (#Iter: avr, max) of the 20 test instances, and the average number of no-good cuts that have to be added (#NG) are recorded. Also the average and maximum solving time (Sol.Time: avr, max) of the 20 test instances, and the average percentages of solving time spent in the solution of master problem, subproblem and relaxed cut generation program

(MP%, SP%, CGP%), are recorded. All the solving times are in seconds. For comparison purpose, every problem is also directly solved with MIP solver. The last two columns summarize the average MIP solving time (MIP.Time), and in how many cases (out of the total 20 cases) the Benders algorithm outperforms the directly solving (#WIN). The external solver used in both the decomposition algorithm and the direct solving is XPRESS 14.21 [12].

Table 1 shows that as the problem size increases, the number of iterations and the solving time both increase. But throughout the test instances, no-good cuts being added are rare, which means that the generated cuts are usually tight enough to exclude the infeasible assignment in each iteration. It is also notable that a significant portion of the total solving time is spent in solving the relaxed cut generation program. However, in spite of the time spent in the cut generation, the Benders algorithm still wins over directly solving the problem in more cases when the problem size becomes larger. This shows the benefits of using Benders decomposition for integer programs to exploit the problem structures, that is, a problem is decomposed into a master problem and a series of smaller independent subproblems, reducing the complexity of solving it.

We observed that the decomposition algorithm is especially better for the hard instances. For those problems that take long time by direct solving, the Benders decomposition with integer cuts usually achieves high speedup in terms of solving time. Table 2 shows the comparison. Five hardest instances (in terms of direct MIP solving time) for each fixed subproblem size are recorded.

Table 2. Solving Time Comparison for Hard Instances

SPV	MPV	#Iter	Sol.Time	MIP.Time
300	100	12	94.62	14181.90
300	250	17	270.74	1403.68
300	250	25	562.64	901.83
300	250	14	201.25	833.31
300	200	15	171.69	624.48
400	250	28	697.45	>20000.00
400	150	17	299.12	14577.89
400	100	17	235.38	11086.83
400	250	12	195.52	10335.30
400	250	30	851.96	7061.50
500	250	24	803.22	>20000.00
500	100	18	372.73	>20000.00
500	150	14	297.14	>20000.00
500	200	30	834.20	>20000.00
500	250	28	924.56	>20000.00

We also observed that, for all the test instances that take more than 200 seconds by directly solving, the decomposition algorithm invariably consumes less solving time than the direct solving.

7 Conclusions

This paper studied the generation of valid Benders cuts for a general class of integer programming problems. The valid Benders cuts in the form of linear inequalities were derived, based on which a complete Benders algorithm was presented. The (relaxed) cut generation program was proposed to determine the valid cuts in practice. In theoretical aspect, the paper extended the application scope of Benders decomposition method to integer programming problems. In computational experiments, the results showed the benefits of using Benders algorithm with the proposed cut for integer programs.

The master problem discussed in the paper need not be restricted to linear integer programs. In fact, it can be any formulation and can be solved with any proper algorithm (such as Constraint Programming). More specifically, the linear objective function in problem P (i.e. $\boldsymbol{c}^T\boldsymbol{y}$) can be replaced with a general function $f(\boldsymbol{y})$. The first constraint in P (i.e. $\boldsymbol{D}\boldsymbol{y} \leq \boldsymbol{d}$) can be replaced with a general constraint $\mathcal{C}(\boldsymbol{y})$ (even need not be arithmetic). Since the generalized objective and constraint are only handled in the master problem, they do not affect the theory and method proposed in the paper. Furthermore, the second constraint of P can be generalized to $\boldsymbol{h}(\boldsymbol{y}) + \boldsymbol{B}\boldsymbol{x} \leq \boldsymbol{b}$, that is, the part that relates to the master problem variables can be any function on $\boldsymbol{y}$ (i.e. $\boldsymbol{h}(\boldsymbol{y})$), in place of the linear one, $\boldsymbol{A}\boldsymbol{y}$. Accordingly, all the occurrences of $\boldsymbol{A}\boldsymbol{y}$ in the derivations are changed to $\boldsymbol{h}(\boldsymbol{y})$, and the derivations remain valid. As the master problem is generalized as above, different modelling and solution methods could be combined via the method of Benders decomposition to cooperatively solve a given optimization problem.

References

1. Flippo, O.E., Rinnoy Can, A.H.G.: Decomposition in General Mathematical Programming. Math. Programming. **60** (1993) 361–382
2. Lasdon, L.S.: Optimization Theory for Large Systems. MacMillan, New York. (1970)
3. Schimpf, J., Wallace, M.: Finding the Right Hybrid Algorithm: A Combinatorial Meta-Problem. Annals of Mathematics and Artificial Intelligence. **34** (2002) 259–269
4. Eremin, A., Wallace, M.: Hybrid Benders Decomposition Algorithms in Constraint Logic Programming. In T. Walsh, editor, Principles and Practice of Constraint Programming - CP 2001, Springer. (2001) 1–15
5. Benders, J.F.: Partitioning Procedures for Solving Mixed-Variables Programming Problems. Numerische Mathematik. **4** (1962) 238–252
6. Geoffrion, A.M.: Generalised Benders Decomposition. Journal of Optimization Theory and Application. **10** (1972) 237–260
7. Jain, V., Grossmann, I.E.: Algorithms for Hybrid MILP/CP Models for a Class of Optimisation Problems. INFORMS Journal on Computing. **13** (2001) 258–276
8. Hooker, J.N., Ottosson, G.: Logic-Based Benders Decomposition. Math. Programming. **96** (2003) 33–60

9. Eremin, A.: Using Dual Values to Integrate Row and Column Generation into Constraint Logic Programming. PhD Thesis. Imperial College London. (2003)
10. Xia, Q., Simonis, H., Chu, Y.: Generating Primary/Secondary Path by Benders Decomposition Technique. IC-Parc Internal report, Imperial College London. (2003)
11. Imperial College London: ECLiPSe 5.6 User's Manual. (2003)
12. Dash Inc: Dash XPRESS 14.21 User's Manual. (2003)

Appendix: Proofs of Lemmas

Lemma 1:

Proof. It suffices to show that any feasible solution of (4) automatically satisfies $\boldsymbol{r}_i(\boldsymbol{u}-\boldsymbol{1})_i = 0$. Suppose $\hat{\boldsymbol{x}}, \hat{\boldsymbol{r}}, \hat{\boldsymbol{u}}$ constitute a feasible solution of (4).

The first constraint of (4) implies

$$\hat{\boldsymbol{r}}_i \geq (\boldsymbol{A}\bar{\boldsymbol{y}} + \boldsymbol{B}\hat{\boldsymbol{x}} - \boldsymbol{b})_i \quad . \tag{15}$$

The second constraint of (4) implies

$$\hat{\boldsymbol{r}}_i = \hat{\boldsymbol{u}}_i(\boldsymbol{A}\bar{\boldsymbol{y}} + \boldsymbol{B}\hat{\boldsymbol{x}} - \boldsymbol{b})_i \quad . \tag{16}$$

Consider two cases on the non-negative value of $\hat{\boldsymbol{r}}_i$.
Case 1: $\hat{\boldsymbol{r}}_i = 0$. Then the constraint $\boldsymbol{r}_i(\boldsymbol{u}-\boldsymbol{1})_i = 0$ is trivially satisfied.
Case 2: $\hat{\boldsymbol{r}}_i > 0$. Then we have $\hat{\boldsymbol{u}}_i = 1$ (otherwise, $0 \leq \hat{\boldsymbol{u}}_i < 1$. Then from (16) $0 < \hat{\boldsymbol{r}}_i = \hat{\boldsymbol{u}}_i(\boldsymbol{A}\bar{\boldsymbol{y}} + \boldsymbol{B}\hat{\boldsymbol{x}} - \boldsymbol{b})_i < (\boldsymbol{A}\bar{\boldsymbol{y}} + \boldsymbol{B}\hat{\boldsymbol{x}} - \boldsymbol{b})_i$, which contradicts (15)). Since $\hat{\boldsymbol{u}}_i = 1$, the constraint $\boldsymbol{r}_i(\boldsymbol{u}-\boldsymbol{1})_i = 0$ is again satisfied. □

Lemma 2:

Proof. In every iteration, if $\bar{\boldsymbol{y}}^{(k)}$ is feasible then the algorithm terminates from step *2(b)*. Otherwise a valid cut is added. Due to the valid cut condition 1, the feasible space of $MP^{(k+1)}$ must be smaller than that of $MP^{(k)}$, at least reduced by one point. Since the feasible space of master problem is finite domain, the algorithm terminates finitely. Due to the valid cut condition 2, the feasible space of $MP^{(k)}$ is always a relaxation of that of the original program P. If the algorithm terminates from *2(a)*, then $MP^{(k)}$ is infeasible, and so is the original program P. If the algorithm terminates from step *2(b)*, the current optimal solution of $MP^{(k)}$ is proved to be *feasible* for the subproblem and thus feasible for P. Since $MP^{(k)}$ is a relaxation of P, this solution is optimal for P. □

Lemma 3:

Proof. For the valid cut condition 1, if $\bar{\boldsymbol{y}}$ is infeasible for the subproblem, then $SP(\bar{\boldsymbol{y}})$ has positive objective value. Thus any possible $SP_f(\bar{\boldsymbol{y}}, \boldsymbol{x})$ has positive objective value, and so do all $DSP_f(\bar{\boldsymbol{y}}, \boldsymbol{x})$. Therefore all inequalities in cut (5) are violated, that is, cut (5) excludes $\bar{\boldsymbol{y}}$. For the valid cut condition 2, consider any feasible $\hat{\boldsymbol{y}}$. There must exist one value $\hat{\boldsymbol{x}}$ such that $SP_f(\hat{\boldsymbol{y}}, \hat{\boldsymbol{x}})$ has 0 objective value, and so do its dual $DSP_f(\hat{\boldsymbol{y}}, \hat{\boldsymbol{x}})$, that is, $(\boldsymbol{A}\hat{\boldsymbol{y}} - \boldsymbol{b})^T\hat{\boldsymbol{u}} + \hat{\boldsymbol{x}}^T\hat{\boldsymbol{v}} = 0$ is satisfied, which means the disjunctive cut (5) does not cut off the feasible $\hat{\boldsymbol{y}}$. □

Lemma 4:

Proof. For necessity, we show that if (8) is violated, then (6) must be violated. Suppose the ith element $\tilde{\boldsymbol{x}}_i = 1$ and $\tilde{\boldsymbol{v}}_i > 0$ (The case where the second condition of (8) is violated can be proved similarly). Then $\tilde{\boldsymbol{x}}_i\tilde{\boldsymbol{v}}_i > 0$. Consider another assignment $\boldsymbol{x}$, with the ith element $\boldsymbol{x}_i = 0$ and with all other elements the same as $\tilde{\boldsymbol{x}}$. It is easy to see that $\tilde{\boldsymbol{x}}^T\tilde{\boldsymbol{v}} > \boldsymbol{x}\tilde{\boldsymbol{v}}$, which means that (6) is violated.

For sufficiency, we suppose (8) is satisfied for every element. Then the inequality $\tilde{\boldsymbol{x}}_i\tilde{\boldsymbol{v}}_i \leq b\tilde{\boldsymbol{v}}_i$ holds no matter b is 0 or 1. Therefore, for every element of any $\boldsymbol{x} \in \{0,1\}^n$, we have $\tilde{\boldsymbol{x}}_i\tilde{\boldsymbol{v}}_i \leq \boldsymbol{x}_i\tilde{\boldsymbol{v}}_i$, which implies that (6) holds. □

Lemma 5:

Proof. We first prove that the relaxed cut satisfies the valid cut condition 2. Following the same reasoning as the proof of Theorem 1, we have:

$$(\boldsymbol{A}\hat{\boldsymbol{y}} - \boldsymbol{b})^T\tilde{\boldsymbol{u}} + \hat{\boldsymbol{x}}^T\tilde{\boldsymbol{v}} \leq (\boldsymbol{A}\hat{\boldsymbol{y}} - \boldsymbol{b})^T\hat{\boldsymbol{u}} + \hat{\boldsymbol{x}}^T\hat{\boldsymbol{v}} = 0 \ .$$

According to the construction of relaxed cut, we have $\tilde{\boldsymbol{x}'}_i\tilde{\boldsymbol{v}}_i \leq b\tilde{\boldsymbol{v}}_i$ for any binary value $b \in \{0,1\}$. In particular, $\tilde{\boldsymbol{x}'}_i\tilde{\boldsymbol{v}}_i \leq \hat{\boldsymbol{x}}_i\tilde{\boldsymbol{v}}_i, \forall i$. Therefore,

$$(\boldsymbol{A}\hat{\boldsymbol{y}} - \boldsymbol{b})^T\tilde{\boldsymbol{u}} + \tilde{\boldsymbol{x}'}^T\tilde{\boldsymbol{v}} \leq (\boldsymbol{A}\hat{\boldsymbol{y}} - \boldsymbol{b})^T\tilde{\boldsymbol{u}} + \hat{\boldsymbol{x}}^T\tilde{\boldsymbol{v}} \leq 0,$$

which means that the feasible $\hat{\boldsymbol{y}}$ is not cut off by (12).

The relaxation gap is directly obtained by subtracting the left hand side of (12) from that of (11). Because $\tilde{\boldsymbol{x}'}_i\tilde{\boldsymbol{v}}_i \leq \hat{\boldsymbol{x}}_i\tilde{\boldsymbol{v}}_i$, the relaxation gap $\sum_{i=1}^{n}(\tilde{\boldsymbol{x}} - \tilde{\boldsymbol{x}'})_i\tilde{\boldsymbol{v}}_i \geq 0$. □

Lemma 6:

Proof. Consider the program CGP_r. Since $\boldsymbol{v} = \boldsymbol{B}^T\boldsymbol{u}$, the constraint $\tilde{\boldsymbol{x}}_i(\boldsymbol{B}^T\boldsymbol{u})_i - \boldsymbol{p}_i \leq 0$ in (14) becomes $\tilde{\boldsymbol{x}}_i\boldsymbol{v}_i \leq \boldsymbol{p}_i$, and $(1-\tilde{\boldsymbol{x}})_i(\boldsymbol{B}^T\boldsymbol{u})_i + \boldsymbol{q}_i \geq 0$ becomes $-(1-\tilde{\boldsymbol{x}})_i\boldsymbol{v}_i \leq \boldsymbol{q}_i$.

For each element $\tilde{\boldsymbol{x}}_i$ and its corresponding $\tilde{\boldsymbol{v}}_i$, there are three cases.
Case 1: $\tilde{\boldsymbol{x}}_i$ and $\tilde{\boldsymbol{v}}_i$ satisfy the sign condition. Then the optimal slack variable $\boldsymbol{p}_i^*$ and $\boldsymbol{q}_i^*$ are 0, and $\tilde{\boldsymbol{x}}_i = \tilde{\boldsymbol{x}'}_i$. Therefore, $\boldsymbol{p}_i^* + \boldsymbol{q}_i^* = (\tilde{\boldsymbol{x}} - \tilde{\boldsymbol{x}'})_i\tilde{\boldsymbol{v}}_i = 0$.
Case 2: $\tilde{\boldsymbol{x}}_i$ and $\tilde{\boldsymbol{v}}_i$ violates the sign condition as $\tilde{\boldsymbol{x}}_i\tilde{\boldsymbol{v}}_i > 0$, which implies that $\tilde{\boldsymbol{x}}_i = 1$ and $\tilde{\boldsymbol{v}}_i > 0$. The optimal slack variable $\boldsymbol{p}_i^* = \tilde{\boldsymbol{v}}_i > 0$, and $\boldsymbol{q}_i^*$ equals 0. Since the sign condition is violated, the value of $\tilde{\boldsymbol{x}}_i$ will be changed from 1 to 0 ($\tilde{\boldsymbol{x}'}_i = 0$) to construct the relaxed cut, and thus $(\tilde{\boldsymbol{x}} - \tilde{\boldsymbol{x}'})_i\tilde{\boldsymbol{v}}_i$ equals $\tilde{\boldsymbol{v}}_i$. Therefore, $\boldsymbol{p}_i^* + \boldsymbol{q}_i^* = (\tilde{\boldsymbol{x}} - \tilde{\boldsymbol{x}'})_i\tilde{\boldsymbol{v}}_i > 0$.
Case 3: $\tilde{\boldsymbol{x}}_i$ and $\tilde{\boldsymbol{v}}_i$ violates the sign condition as $(1-\tilde{\boldsymbol{x}}_i)\tilde{\boldsymbol{v}}_i < 0$, which implies that $\tilde{\boldsymbol{x}}_i = 0$ and $\tilde{\boldsymbol{v}}_i < 0$. The optimal slack variable $\boldsymbol{q}_i^* = -\tilde{\boldsymbol{v}}_i > 0$, and $\boldsymbol{p}_i^*$ equals 0. Since the sign condition is violated, the value of $\tilde{\boldsymbol{x}}_i$ will be changed from 0 to 1 ($\tilde{\boldsymbol{x}'}_i = 1$) to construct the relaxed cut, and thus $(\tilde{\boldsymbol{x}} - \tilde{\boldsymbol{x}'})_i\tilde{\boldsymbol{v}}_i$ equals $-\tilde{\boldsymbol{v}}_i$. Therefore, $\boldsymbol{p}_i^* + \boldsymbol{q}_i^* = (\tilde{\boldsymbol{x}} - \tilde{\boldsymbol{x}'})_i\tilde{\boldsymbol{v}}_i > 0$.

In all cases the equation $\boldsymbol{p}_i^* + \boldsymbol{q}_i^* = (\tilde{\boldsymbol{x}} - \tilde{\boldsymbol{x}'})_i\tilde{\boldsymbol{v}}_i \geq 0$ holds. Therefore,

$$\phi_{CGP_r} = \sum_{i=1}^{n}(\boldsymbol{p}_i^* + \boldsymbol{q}_i^*) = \sum_{i=1}^{n}(\tilde{\boldsymbol{x}} - \tilde{\boldsymbol{x}'})_i\tilde{\boldsymbol{v}}_i \geq 0 \ .$$

□