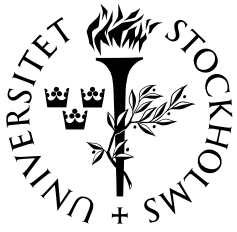

SJÄLVSTÄNDIGA ARBETEN I MATEMATIK

MATEMATISKA INSTITUTIONEN, STOCKHOLMS UNIVERSITET

Solution approaches for the opportunistic replacement problem: Benders decomposition and Chvátal-Gomory cut generation

av

Sara Soltani

2011 - No 2

Solution approaches for the opportunistic replacement problem: Benders decomposition and Chvátal-Gomory cut generation

Sara Soltani

Självständigt arbete i tillämpad matematik 30 högskolepoäng, avancerad nivå

Handledare: Ann-Brith Strömberg

2011

To the love of my life, Hossein

Abstract

The purpose of this project is to study mathematical properties of the opportunistic replacement problem introduced in [18]. The goal is to examine and determine new techniques to compute the solutions for the opportunistic replacement problem faster. In this project the Benders decomposition method and rank-1 Chvátal-Gomory cut generation are applied to the opportunistic replacement problem.

Regarding the Benders Method, for the opportunistic replacement problem with fixed maintenance occasions the dual of the resulting linear programming problem is derived and it is shown that when the maintenance occasions are non-increasing with time, this problem can be solved through a greedy procedure. The feasibility constraints for the master problem in the Benders decomposition method when implemented on the opportunistic replacement problem are explicitly derived. Then, the algorithm of Benders decomposition method applied to the opportunistic replacement problem is presented.

Furthermore, the rank-1 Chvátal-Gomory separation problem for the opportunistic replacement problem is modeled. A branch and bound approach is then used to generate valid inequalities for the replacement polytope, by solving the separation model and finding the most violated cuts for non-integral extreme points of the constraint set defining the opportunistic replacement problem.

Results from computation tests of the two solution procedures and conclusions are finally reported.

Acknowledgments

Here, I would like to specially thank my supervisor Docent Ann-Brith Strömberg for her help and guidance during the work. I would also thank Professor Michael Patriksson for his comments and support during the work on this thesis. I express my gratitude towards LicEng. Adam Wojciechowski for his suggestions and nice discussions we had on the thesis work.

I also would like to thank my examiner Docent Yishao Zhou and Professor Andrzej Szulkin at the Department of Mathematics, Stockholm University, for their great help. In particular, I want to thank all for giving me the opportunity to work on this thesis at the Department of Mathematical Sciences, Chalmers University of Technology.

Contents

List of Tables

List of Figures

List of Algorithms

1 Introduction and background

1.1 Maintenance planning and problems

In aviation industry, power plants, and processing industry, expensive equipments need to be used efficiently with few interruptions to pay back the huge costs of the investments. Due to the huge costs of breakdowns of a system, it is of essential importance to avoid them as much as possible. In a typical setting every machine consists of different modules, where each module contains several components. When a component breaks or reaches its life a replacement of that component is unavoidable. The life of each component can be considered as deterministic or stochastic.

In the literature sometimes it is assumed that the maintenance opportunity is independent of the failure; sometimes the opportunity is equal to the first failure of an individual part. In this case failure of one part is used as an advantage of preventive maintenance for the other parts. The term *opportunistic maintenance* refers to the situation that every maintenance occasion is considered as an opportunity to prevent possible future failures of the system ([6]). This is often the case for aircraft engines.

The maintenance of aircraft engines is crucial in aviation industry. Since, in particular, the major concern is safety. If an essential component of an aircraft breaks, it may crash; therefore the interruption of the system function should be avoided at any cost. In this case, the maintenance planning should be scheduled in such a way that the system works without interruption between the planned maintenance occasions. Since some of the parts are of great safety importance, they are assigned fixed deterministic lives. Other parts of the engine are considered to have stochastic lives.

The opportunistic replacement is motivated mainly by the unavoidable fixed costs associated with each maintenance occasion rather than each component's cost [6]. In simple words, when an engine is taken to the maintenance workshop, a spare engine should replace it. Therefore at every maintenance occasion –accompanying the cost of each part to be replaced– there is often also an independent large fixed cost. In opportunistic maintenance, the extra cost of a maintenance occasion should be balanced with the costs of individual modules which have to be replaced, so there is an optimization problem to be solved.

When a deterministic life of a component is reached, the engine must be taken to the workshop for maintenance. This is a good opportunity to replace some of the non-failed components with stochastic or deterministic lives. There are some information needed to formulate the optimization problem such as the remaining lives of the deterministic parts, costs of new spare parts, and the work cost for the workshop when replacing components, etc. In case of not knowing the lives of the components, it is possible to estimate their stochastic life time using historical data and/or condition measurements.

A relevant optimization model for opportunistic replacement is to minimize the expected costs in order to have functioning engine during a predetermined time period. The optimization should create a maintenance schedule with as low total cost as possible. In the opportunistic replacement problem for an aircraft engine the time horizon is finite. This problem computationally is harder to solve than the infinite time horizon counterpart ([18]).

A conclusion in [2] is that it is extremely hard to find an optimal replacement

schedule when the number of parts is large. Different replacement policies can help to simplify the solution process but they possibly lead to non-optimal solutions. On the other hand, if all the parts have stochastic lives it is difficult to compute a reliable schedule. In such cases it is essential that one uses replacement policies rather than solving an optimization model.

In the aircraft engine studied about 75% of the components are considered to have deterministic lives. If the lives of all the components are deterministic, an optimal maintenance schedule is found by solving the opportunistic replacement problem. This thesis constitutes a study of the mathematical structure and properties of the problem when all the component lives are assumed to be deterministic.

1.2 The opportunistic replacement problem

This thesis is part of a research on analyzing and solving the maintenance decision problems. A system (e.g., a jet engine or a wind power turbine) to maintain during a finite planning time is considered. The idea is to use a mathematical model to decide whether or not to perform maintenance at the time when the system needs a corrective maintenance or a scheduled preventive one. A fairly simple maintenance problem is presented.

The problem statement is as follows. Consider a system that consists of components $\mathcal{N} = \{1, \ldots, N\}$ with known deterministic lives. We assume that every component must be replaced before its failure. Moreover, every maintenance occasion generates the cost d and the replacement of a component $i \in \mathcal{N}$ generates the replacement cost c_i. We wish to minimize the expected maintenance cost over the planning time $[0, T]$. This problem is denoted the opportunistic replacement problem.

The research work in this thesis is based on the opportunistic replacement problem. Although, the intention is to generalize the results obtained to more general maintenance problems in the future.

1.3 Outline

The aim of this thesis is to study some mathematical aspects of the opportunistic replacement problem and then utilize these when solving the problem. The thesis is organized as follows.

First some essential background from integer programming is briefly reviewed. Chapter 2 is divided into five main sections in which the integer linear programming is described and the complexity of such problems is discussed. Then, some of the concepts helping to a better understanding of the geometry behind integer linear programming are defined. Valid inequalities and facets for an integer linear program is discussed and the Chvátal-Gomory procedure to obtain valid inequalities is presented. Section 2.4 discusses the Chvátal-Gomory cuts and the separation problem for the integer linear programming problems. Finally, in Section 2.5, the general Benders partitioning procedure to solve an integer linear programming problem is presented.

In Chapter 3 a linear programming model is introduced for the opportunistic replacement problem and some of its mathematical properties are discussed. Also a study of the facial structure of the opportunistic replacement problem is presented.

Benders decomposition method is applied to the opportunistic replacement problem in Chapter 4. In Section 4.1 it is shown that the dual linear programming problem when the maintenance occasions are non-increasing with time and the maintenance occasions are fixed is solvable by a greedy rule. An implementation of the Benders decomposition method on the opportunistic replacement problem is discussed in Section 4.2. The feasibility constraints for the master problem in the Benders decomposition method are explicitly derived. Results from numerical experiments are reported.

In Chapter 5 the effect of generating rank-1 Chvàtal-Gomory cuts for the opportunistic replacement problem is questioned. The rank-1 Chvàtal-Gomory separation problem is modeled as an mixed integer linear programming problem. Then, the model is solved in a pure cutting plane framework to find the most violated cuts for the convex hull of the polyhedron defining the opportunistic replacement problem. Some computational results are also presented.

Finally, in Chapter 6 conclusions and remarks on future work are stated.

2 Integer linear programming

Optimization in the simplest way means to maximize (or minimize) a real-valued function of real or integer variables by choosing the values of these variables within an allowed set, which is described by a set of constraints on the variables. A technique for optimizing a linear objective function of real valued variables with respect to linear equality and/or inequality constraints is called *linear programming*. Adding extra restrictions to the variables such as belonging to an integer set or taking binary values gives new types of problems, which has a wide range of applications in everyday life.

Integer programming is about how to solve optimization problems with discrete (or integer) variables. A wide variety of practical problems in management and the efficient use of resources can be formulated as integer linear programming problems. Problems such as distribution of goods, production scheduling, transportation network design, facility location, telecommunications or electricity generation planning most often fall into the integer linear programming category.

This thesis is dealing with an integer linear problem, where the function to be minimized and the inequality restrictions are all linear. In this chapter, we briefly address some basic and essential concepts of integer linear programming problems. First the integer linear programming problem is defined, followed by definitions of a polyhedron, a convex hull, the ideal formulation, and bounds for the integer linear programming problem. Then, the complexity of linear integer programming problems is discussed. Valid inequalities and facets are defined. Also Chvátal-Gomory procedure to derive valid inequalities for the integer linear programming problems and the septation problem are presented. The chapter closes with the description of the general Benders decomposition procedure.

2.1 An integer linear programming model

Consider the linear programming problem in the canonical form:

$$\text{(LP)} \quad \max_x \{cx : Ax \leq b, x \in R^n_+\},$$

where R^n_+ is the set of non-negative real n-dimensional vectors, A an $m \times n$ matrix, c an n-row vector, b an m-column vector, and x an n-column vector. x is the vector of decision variables. Letting some of the variables be integer, then *the mixed integer linear programming problem* is defined as

$$\text{(MILP)} \quad \max_{x,y} \{cx + hy : Ax + Gy \leq b, x \in R^n_+, y \in Z^p_+\},$$

where Z^p_+ is the set of non-negative integral p-dimensional vectors, G is an $m \times p$ matrix, h is a p-row vector, y is a p-column vector of integer decision variables. If all variables are integer (i.e., if $n = 0$), we have *(pure) integer linear programming problem*

$$\text{(ILP)} \quad \max_y \{hy : Gy \leq b, y \in Z^p_+\}.$$

Note that ILP is the special case of MILP in which there are no continuous variables and LP is another special case of MILP in which there are no integer variables. At

the end we define a *binary linear program*:

$$\text{(BLP)} \quad \max_x \{cx : Ax \leq b, x \in \{0,1\}^n\}.$$

One should note that in understanding and solving integer linear programming problems, the linear programming theory is fundamental. As in linear programming, translating a problem description into a mathematical formulation should be done systematically. The data of the problem instance and the variables should be distinct in the model. In the next section we formulate a very famous problem called, the set covering problem, as n BLP.

2.1.1 The set covering problem

Two of the very well known integer linear programming problems are the set covering problem and the set packing problem. The set covering problem can be described as follows. Given a certain number of regions, the set covering problem is to decide where to install a set of emergency service centers. For each possible center the cost of installing a service center, and which regions it can service are known. The goal is to chose a minimum cost set of service centers so that each region is covered. Now, we formulate it as a BLP. Let $M = \{1, \ldots, m\}$ be the set of regions, and $N = \{1, \ldots, n\}$ the set of potential centers. Let $S_j \subseteq M$ be the regions that can be serviced by a center at $j \in N$, and c_j its installation cost. To facilitate the description, we first construct a $0-1$ *incidence matrix* A such that $a_{ij} = 1$ if $i \in S_j$, and $a_{ij} = 0$ otherwise. Let the decision variables be $x = (x_1, \ldots, x_n)$ where $x_j = 1$ if center j is selected, and $x_j = 0$ otherwise. The set covering problem is defined as

$$\min_x \sum_{j=1}^{n} c_j x_j, \tag{1a}$$

$$\sum_{j=1}^{n} a_{ij} x_j \geq 1, \quad i = 1, \ldots, m, \tag{1b}$$

$$x_j \in \{0,1\}, \qquad j = 1, \ldots, n. \tag{1c}$$

The inequalities (1b) state that at least one center must service region i. The set packing problem is the integer linear program of the form

$$\max_x \sum_{j=1}^{n} c_j x_j, \tag{2a}$$

$$\sum_{j=1}^{n} a_{ij} x_j \leq 1, \quad i = 1, \ldots, m, \tag{2b}$$

$$x_j \in \{0,1\}, \qquad j = 1, \ldots, n. \tag{2c}$$

There are very strong ties between set covering problem and set packing problems. For more details see [12]. The set covering problem is structurally similar to the opportunistic replacement problem defined in Chapter 3.

2.1.2 Definitions of some central concepts

In integer programming, for the formulation of problems we are given a set of feasible points, often described as the set of integer solutions to a linear inequality system

$S = \{x \in Z_+^n : Ax \leq b\}$. For a better understanding of how to deal with integer linear programming problems, we give some necessary definitions and propositions without proof from linear algebra. The books [12] and [19] are used as references for this section.

Definition 2.1 (**Convex combination and convex hull**). *Given a set $S \in R_n$, a point $x \in R^n$ is a convex combination of points of S if there exists a finite set of points $\{x_i\}_{i=1}^t \in S$ and a $\lambda \in R_+^t$ such that $\sum_{i=1}^t \lambda_i = 1$ and $x = \sum_{i=1}^t \lambda_i x^i$. The convex hull of S, denoted by $conv(S)$, is the set of all points that are convex combinations of points in S.*

Definition 2.2 (**Linear and affine independence**). *A set of points $x^1, \ldots, x^k \in R^n$ is linearly independent if the unique solution to the equations $\sum_{i=1}^k \lambda_i x^i = 0$ is $\lambda_i = 0$, $i = 1, \ldots, k$. A set of points $x^1, \ldots, x^k \in R^n$ is affinely independent if the unique solution to the equations $\sum_{i=1}^k \alpha_i x^i = 0$ and $\sum_{i=1}^k \alpha_i = 0$ is $\alpha_i = 0$ for $i = 1, ..., k$.*

Note that the maximum number of linearly and affinely independent points in R^n are n and $n+1$ respectively. Linear independence implies affine independence, but the converse is not true.

Definition 2.3 (**Polytope and polyhedron**). *A polyhedron $P \subseteq R^n$ is the set of points that satisfy a finite number of linear inequalities; that is, $P = \{x \in R^n : Ax \leq b\}$, where (A, b) is an $m \times (n+1)$ matrix. A polyhedron is bounded if there exists an $\alpha \in R_+^1$ such that $P \subseteq \{x \in R^n : -\alpha \leq x_j \leq \alpha \quad for \quad j = 1, \ldots, n\}$. A bounded polyhedron is called a polytope.*

Definition 2.4 (**Ray**). *Let $P^0 = \{r \in R^n : Ar \leq 0\}$. If $P = \{x \in R^n : Ax \leq b\} \neq \emptyset$, then $r \in P^0 \setminus \{0\}$ is called a ray of P.*

Definition 2.5 (**Extreme point and extreme ray**). *$x \in P$ is an extreme point of P if there do not exist $x^1, x^2 \in P$, $x^1 \neq x^2$, such that $x = \frac{1}{2}x^1 + \frac{1}{2}x^2$. A ray of P is an extreme ray if there do not exist $r^1, r^2 \in P^0$, $r^1 \neq \lambda r^2$ for any $\lambda \in R_+^1$, such that $r = \frac{1}{2}r^1 + \frac{1}{2}r^2$.*

Definition 2.6 (**Cone**). *$C \in R^n$ is a cone if $x \in C$ implies $\lambda x \in C$ for all $\lambda \in R_+^n$.*

A polyhedron has a finite number of extreme points and extreme rays. Let $V = (v_1, v_2, \ldots, v_k)$ and $E = (e_1, e_2, \ldots, e_l)$ be the set of extreme points and extreme rays, respectively, of the polyhedron P. Then $P = conv(V) + cone(E)$ and if P is a polytope then $P(A, b) = conv(V)$.

Definition 2.7 (**Dimension of a polyhedron**). *A polyhedron P is of dimension k, denoted by $dim(P) = k$, if the number of affinely independent points in P is $k+1$. A polyhedron $P \subseteq R^n$ is full-dimensional if $dim(P) = n$.*

Proposition 2.1 (**Pro.2.1 chapter I.4. [12]**). *A polyhedron is a convex set.*

Proposition 2.2 (**Pro.1.2 chapter 1 [19]**). *The extreme points of $conv(S)$ all lie in S.*

Definition 2.8. *A polyhedron $P \subseteq R^{n+p}$ is a formulation for a set $S \subseteq R^n \times Z^p$ if and only if $S = P \cap (R^n \times Z^p)$.*

Note that in a formulation of an ILP problem the integrality requirements are disregarded and only the (in)equalities are considered, i.e., we consider the polyhedron defined by the (in)equality constraints. Most integer linear problems can be mathematically formulated in several ways, but not every choice of formulation is a good one. In an *ideal formulation* each extreme point is integer, so if the ideal formulation is solved –as the integer optimal solution is at an extreme point– this yields an optimal integer solution. Propositions (2.1) and (2.2) enable us to replace The ILP $\max\{cx : x \in S\}$ by the equivalent LP $\max\{cx : x \in conv(S)\}$. The reduction of the formulation of the polyhedron P to the ideal formulation $conv(S)$ also holds for unbounded integer sets, and mixed integer sets. However, this is in general a theoretical construction, because in most ILP problems an enormous number of inequalities are needed to describe $conv(S)$. This makes it very hard and almost impossible to find all such inequalities for realistic–size real world problems. But since $S \subseteq conv(S) \subseteq P$, for all formulations P, we have the following definition:

Definition 2.9 (**Strong formulation**). *Given a set $S \subseteq R^n$ and two formulations P_1 and P_2 for S, P_1 is a stronger formulation than P_2 if $P_1 \subseteq P_2$.*

2.1.3 Bounds on integer linear programming problem

This section explains how upper and lower bounds to an ILP can be found. Consider the integer linear programming problem:

$$\text{(ILP)} \qquad z_{ILP} = \max_x\{cx : x \in S\}, \quad S = \{x \in Z_+^n : Ax \leq b\}, \tag{3}$$

where c is an n-vector with integral coefficients and (A, b) is an $m \times (n+1)$ matrix with integral coefficients. In an algorithm to solve ILP, finding a tight bound on z_{ILP} would provide a stopping criterion, since it can be considered as a fundamental way of proving optimality for a feasible solution to the integer linear programming problem. Practically, this means that any algorithm will find an increasing sequence of lower bounds: $\underline{z_1} < \underline{z_2} < \ldots < \underline{z_s}$ and a decreasing sequence of upper bounds: $\overline{z_1} > \overline{z_2} > \ldots > \overline{z_t}$, and stop when $\overline{z_t} - \underline{z_s} \leq \epsilon$, where $\epsilon \geq 0$ is suitably chosen for each ILP.

Every feasible solution $x \in S$ for (3) provides a lower bound $\underline{z} = cx \leq z_{ILP}$. An upper bound on a maximization integer linear problem can be found by considering the linear programming dual of (3). This is called dual bound in contrast to the primal bound. Let $P = \{x \in R_+^n : Ax \leq b\}$, the polyhedron corresponding to the integer program (3). Consider the linear program

$$z_{LP} = \max_x\{cx : x \in P\}. \tag{4}$$

The linear program (4) is called the *linear programming relaxation* of (3). The dual of (4) is defined as

$$\text{(D)} \qquad w_{LP} = \min_u\{ub : u \in P_D\}, \tag{5}$$

where $P_D = \{u \in R_+^m : uA \geq c\}$. It can be easily proven that the integer program (3) and the linear program (5) form a weak dual pair[1]. A relaxation of (3) must

[1][Prop. 2.2, Chapter I.2 [12]]**(weak duality)** If x^* is primal feasible and u^* is dual feasible, then $cx^* \leq z_{LP} \leq w_{LP} \leq u^*b$.

be solved to optimality to provide an upper bound on z_{ILP}. So we need to define a dual problem such that any dual feasible solution yields an upper bound on z_{ILP}. A weak dual of (3) is any minimization problem

$$\text{(DP)} \qquad w_D = \min_u \{z_D(u) : \ u \in S_D\},$$

that satisfies $z_D(u) \geq cx$ for all $x \in S$ and all $u \in S_D$ where $S_D \subseteq P_D$. We now present a proposition whose proof is given in Chapter 2 of [19].

Proposition 2.3. *Suppose that (ILP) and (DP) form a weak-dual pair.*

(i) *If (DP) has an unbounded objective value, then (ILP) is infeasible.*

(ii) *(***strong dual***) If $x^* \in S$ and $u^* \in S_D$ satisfy $cx^* = z_D(u^*)$, then x^* is optimal for (ILP) and (u^*) is optimal for (DP).*

2.2 Computational complexity and well solved problems

The purpose of the following discussion on computational complexity of integer linear optimization problems is to provide a better insight in how difficult it is to find an optimal solution, and what are the properties of the so called well solved problems. The theory in this section is mainly adopted from the references [12] and [19].

2.2.1 $\mathcal{NP}$-hard problems

One might imagine an algorithm for solving an optimization problem: $\max\{cx : x \in S\}$ where the *decision problem*:

$$\text{Is there an} \quad x \in S \quad \text{with value} \quad cx \geq k \quad \text{for} \quad k \in Z?$$

is replaced by the corresponding optimization problem. Let us call this decision problem P. For a problem instance X, the *length of the input* $L = L(X)$ is the length of the binary representation of a "standard" representation of the instance. An algorithm A is defined to be *polynomial* for a problem P and an instance X with $L(X) = l$ if the running time of algorithm A is proportional to $\mathcal{O}(l^p)$ for some positive integer p.

Definition 2.10 (**The classes $\mathcal{P}$ an $\mathcal{NP}$**). *$\mathcal{NP}$ is the class of decision problems with the property that where there exists a primal solution of value as good as or better than k, there is a polynomial proof. $\mathcal{P}$ is the class of decision problems in $\mathcal{NP}$ for which there exists a polynomial algorithm.*

Definition 2.11 (**The class $\mathcal{NPC}$ and reducibility**). *If the problems $P, Q \in \mathcal{NP}$, and if an instance of P can be converted in polynomial time to an instance of Q, then P is polynomial reducible to Q. $\mathcal{NPC}$, the class of $\mathcal{NP}$-complete problems, is the subset of problems $P \in \mathcal{NP}$ such that for all $P, Q \in \mathcal{NP}$, Q is polynomially reducible to P.*

$\mathcal{P}$ is the class of easy problems i.e., for which a polynomial algorithm exists for solving all instances of such problems. A large number of famous optimization problems, e.g., the 0-1 knapsack problem, the set covering problem, the integer

programming problem, belong to the class $\mathcal{NPC}$. Since no polynomial algorithm for problems in $\mathcal{NPC}$ is known today, this class is said to be the class of "the most difficult" problems. An optimization problem for which the decision problem lies in $\mathcal{NPC}$ is called $\mathcal{NP}$-hard. Set covering problem and set packing problem are $\mathcal{NP}$-hard problems. There still remains a question to answer which is how one can prove that a problem lies in $\mathcal{P}$?

We continue this section with defining the separation problem and introducing special cases where a polynomial algorithm exists for the integer linear programming problem. The problems belonging to the class $\mathcal{P}$ are called well solved problems.

2.2.2 Separation problem

The separation problem associated with $\max\{cx : x \in S\}$ is the problem: Given $\mathring{x} \in R^n$, does $\mathring{x} \in conv(S)$? If not, find an inequality $\pi x \leq \pi_0$, where $\pi \in R^n$ and $\pi_0 \in R$ satisfied by all points $x \in S$, but violated by the point $\mathring{x}$. A class of optimization problems has the *efficient optimization property* if and only if a polynomial algorithm exists for the class. The *efficient separation property* refers to the property that there exists an efficient algorithm for the separation problem associated with the problem class.

If a problem has the efficient separation property then the explicit description of the corresponding convex hull is at hand. Therefore, the efficient optimization and the efficient separation problems are equivalent. The family of optimization problems $\max\{cx : x \in conv(S)\}$ is polynomially solvable if and only if the corresponding family of separation problems is polynomially solvable.

2.2.3 Integer programming with totally unimodular matrices

A starting point in solving ILP is that to answer the question whether there exists some problems with the property that $conv(S) = \{x \in R^n_+ : Ax \leq b\}$, i.e., special cases for which an efficient algorithm can be found, or when the LP relaxation possesses an optimal solution which is integer. The following definitions and results from [12] and [19] are utilized to answer this question.

Definition 2.12 (**Totally unimodular matrix**). *A matrix A is totally unimodular (TU) if every square sub-matrix of A has determinant $+1, -1$ or 0.*

Definition 2.13 (**Interval matrix**). *An $m \times n$ $(0,1)$–matrix A is called an* interval matrix *if in each column the 1's appear consecutively; that is, if $a_{ij} = a_{kj} = 1$ and $k > i+1$, then $a_{lj} = 1$ for all l with $i < l < k$.*

Corollary 2.10 on page 544 in [12] states that interval matrices are (TU).

Proposition 2.4 (**Integral polyhedron**). *If A is TU, then $P(b) = \{x \in R^n_+ : Ax \leq b\}$ is integral for all $b \in Z^m$ for which it is not empty. I.e. $P(b) = conv(P(b) \cap Z^n)$.*

Proof. Consider the linear program with constraint set $Ax + Iy = b$, $x \in R^n_+$, $y \in R^m_+$, where A is TU and b is integral. From linear programming theory, we know that the basic feasible solutions are $x = (x_B, x_N)$ where $x_B = B^{-1}b$ and $x_N = 0$. Also $(A, I) = (B, N)$, where B is an invertible $m \times m$ sub-matrix of (A, I), called a *basis*

matrix for the linear program. Since A is TU, B and B^{-1} are also integral [form Pro 2.1 page 540 [12]]. Thus $B^{-1}b$ is integral, so the correspondence between basic feasible solutions and extreme points yields the result. □

Proposition 2.5. *The linear program* $\max\{cx : Ax \leq b, x \in R^n_+\}$ *has an integral solution for all integer vectors b for which it has a finite optimal value if and only if A is totally unimodular.*

By propositions 2.4 and 2.5 we have shown that for $S = \{x \in Z^n_+ : Ax \leq b\}$, and $P = \{x \in R^n_+ : Ax \leq b\}$, where A is TU and b is integral, It holds that $S = conv(S) = P$. Thus, when A is TU the linear programming relaxation solves IP. The converse also holds.

2.3 Valid inequalities and facets

A primarily and practically important problem in solving integer linear programming problem is to find an equivalent representation of the integer program by a linear program that have the same optimal solution. In Sections 2.1.2 and 2.2.2 it is noted that if an ideal description of $conv(S)$ is at hand then the integer linear programming problem is polynomially solvable. Unfortunately this is not the case for most of the integer linear programming problems. Therefore one needs to find additional valid inequalities for the set S in hope to obtain a stronger formulation for $conv(S)$. In this section, by using integrality and valid inequalities for P, we address constructing suitable valid inequalities for the set S.

Consider the general integer program (3), and let $P = \{x \in R^n_+ : Ax \leq b\}$ so that $S = P \cap Z^n$. Define the convexified integer program.

$$(CIP) \qquad \max\{cx : x \in conv(S)\}. \tag{6}$$

Theorem 2.1. *Given the set* $S = P \cap Z^n \neq \emptyset$, *where* $P = \{x \in R^n_+ : Ax \leq b\}$, *and any* $c \in R^n$, *it follows that:*

1. *The objective value of* (3) *is unbounded from above if and only if the objective value of* (6) *is unbounded from above.*

2. *If* (6) *has a bounded optimal value, then it has an optimal solution (namely, an extreme point of* $conv(S)$*), that is an optimal solution to* (3).

3. *If* $\mathring{x}$ *is an optimal solution to* (3), *then* $\mathring{x}$ *is an optimal solution to* (6).

For a proof of this theorem see Chapter I.4.6 in [12]. This theorem states that reducing an integer linear program to a linear program amounts to find the linear inequalities representative of $conv(S)$. For $\mathcal{NP}$-hard problems, finding a good description of $conv(S)$ in terms of linear inequalities is a very hard problem. Generally, the strategy one could take when trying to solve an $\mathcal{NP}$-problem is to find effective ways to approximate $conv(S)$ for some instances of that problem. That is to deduce the relevant inequalities from the linear inequality representation of the polyhedron P and the integrality constraints on the variables.

2.3.1 Valid inequalities and facets

In the polyhedral description of a integer linear programming problem, it is important to find necessary inequalities and to get rid of redundant inequalities.

Definition 2.14 (Valid inequality). *The inequality $\pi x \leq \pi_0$, denoted as (π, π_0), is a valid inequality for a set $P \subseteq R^n$ if $\pi x \leq \pi_0$, for all $x \in P$.*

Proposition 2.6 (Valid inequality for S). *If $\pi x \leq \pi_0$ is valid for $S = \{x \in Z_+^n : Ax \leq b\}$, it is also valid for $conv(S)$.*

Proof. Consider an $x \in conv(S)$. Then $x = \sum_{j \in J} \lambda_j x^j$, where $x^j \in S$ for $j \in J$, $\sum_{j \in J} \lambda_j = 1$, and $\lambda_j \geq 0$ for $j \in J$. Hence,

$$\pi x = \sum_{j \in J} \lambda_j (\pi x^j) \leq \sum_{j \in J} \lambda_j \pi_0 = \pi_0, \tag{7}$$

which yields the result. □

Definition 2.15 (Face and facet). *If $\pi x \leq \pi_0$ is a valid inequality for P and $F = \{x \in P : \pi x = \pi_0\}$, F is called a* face *of P, and we say that (π, π_0) represents F. A face F of P is said to be* proper *if $F \neq \emptyset$ and $F \neq P$. A face F of P is a* facet *of P if $dim(F) = dim(P) - 1$.*

Proposition 2.7 (Proposition 4.2 Chapter I.4 [12]). *x is an extreme point of P if and only if x is a zero-dimensional face of P.*

Let P be a full-dimensional polyhedron. Then $P = \{x \in R^n : a^i x \leq b_i \ i = 1, \ldots, m\}$, where each inequality is unique within a positive multiple, is the unique description of P. These inequalities are necessary to define P: without any of them, P is not completely defined. If a valid inequality for P is not a positive multiple of these inequalities, it is redundant and can be removed. Note that an inequality $\pi x \leq \pi_0$ representative of the facet F is necessary in the description of P. Moreover, the facets are sufficient for the description of P.

Identifying new classes of facets and including them in the problem description, help to solve the $\mathcal{NP}$–hard ILP problem more efficiently. This is a strong motivation to look for some techniques to generate all valid inequalities for an ILP or an MILP problem. However, finding facets of the polyhedra defined by different integer linear programming problems is not an easy task. The determination of families of strong valid inequalities is more of an art than a formal methodology As a well known integer linear programming problem finding classes of facets for the set covering problem and set packing problems has been always of interest. Looking back to the classical literature on these problems, two classes of facets for the set packing polyhedron has been identified in [14]. In Chapter II.2 of [12] the problem structure is used to determine facets for convex hull of the constraint sets of some $\mathcal{NP}$–hard ILP problems. By some examples, it is shown in [14] and Chapter II.2 [12] that generating facets is considerably effective when solving the $\mathcal{NP}$–hard ILP problems. Studying known facets for well known integer linear programming problems can always be used as a guide for finding classes of facets for the arising new problems in the field of integer programming. The following result from Theorem 3.6 of Chapter I.4 in [12], is widely used to determine facets of $conv(S)$.

Theorem 2.2 (characterization of facets). *Let P be a full-dimensional polyhedron and let $F = \{x \in P \mid \pi^T x = \pi_0\}$ be a proper face of P (i.e., $\emptyset \neq F \subset P$). Then the following two statements are equivalent:*

1. *F is a facet of P.*
2. *If $\lambda^T x = \lambda_0$ for all $x \in F$, then $(\lambda, \lambda_0) = \alpha(\pi, \pi_0)$ for some $\alpha \in R$.*

We obtain valid inequalities for a given set P by taking non-negative linear combinations of rows of $Ax \leq b$. This would give an infinite family of valid inequalities. Moreover under some technical assumptions stated in the theorem below (Chapter II.1, Prop. 1.1 in [12]), all valid inequalities for P can be obtained this way. The linear combinations can be restricted to using, at most, $\min(m, n)$ rows of A.

Theorem 2.3. *Let $\pi x \leq \pi_0$ be any valid inequality for $P = \{x \in R^n_+ : \ Ax \leq b\}$. Then $\pi x \leq \pi_0$ is either equivalent to or dominated by an inequality of the form $uAx \leq ub$, $u \in R^m_+$, if and only if $P \neq \emptyset$, $\{u \in R^m_+ : \ uA \geq \pi\} \neq \emptyset$ and $A = \binom{A'}{I}$, where I is an $n \times n$ identity matrix and A' is a $(m-n) \times n$ sub-matrix of A.*

A simple procedure for combining the rows of the matrix A to obtain valid inequalities for the set P has been developed by Chvátal and Gomory. The Chvátal–Gomory procedure stated in the next section is from references [12] and [19].

2.3.2 The Chvátal–Gomory procedure to construct valid inequalities

Consider the feasible region S of the general ILP problem. How one can obtain valid inequalities for S is based on the simple principle that if a is an integer and $a \leq b$, then $a \leq \lfloor b \rfloor$, where $\lfloor b \rfloor$ is the largest integer less than or equal to b. Let $X = \{y \in Z^1 \ : \ y \leq b\}$; then the inequality $y \leq \lfloor b \rfloor$ is valid for X.

The Chvátal–Gomory procedure: For the set $S = \{x \in Z^n_+ : \ Ax \leq b\}$, where $A = (a_1, a_2, \ldots, a_n)$ and $N = \{1, \ldots, n\}$ it holds that
i. The inequality $\sum_{j \in N} u a_j x_j \leq ub$ is valid for P for all $u \geq 0$ since $\sum_{j \in N} a_j x_j \leq b$,
ii. The inequality $\sum_{j \in N} \lfloor u a_j \rfloor x_j \leq ub$ is valid for P for all $u \geq 0$,
iii. The inequality $\sum_{j \in N} \lfloor u a_j \rfloor x_j \leq \lfloor ub \rfloor$ is valid for S whenever x is integer, and thus $\sum_{j \in N} \lfloor u a_j \rfloor x_j$ is integer.

The valid inequality in ***iii*** can be added to the linear system $Ax \leq b$, and then the Chvátal–Gomory procedure can be repeated to the original set of inequalities or the system with new inequalities. Note that it is sufficient to combine at most n inequalities. This general procedure is called the *Chvátal–Gomory (CG) rounding method*, and the inequalities it produces are called *CG inequalities*. It can be proved that by applying the *Chvátal–Gomory (CG) procedure* a finite number of times all of the valid inequalities for S can be generated (Chapter 8, Theorem 8.4 in [19]). Note that non-dominated CG-cuts only arise for $u \in [0, 1)^m$ provided that (A, b) is integral (Chapter II.2 of [12]).

2.3.3 Software for finding facets by projection

In Section 2.2.2 it is stated that if a complete description of the polyhedron defining the ILP problem is at hand, then a polynomial algorithm which solves the ILP

exists. So, we may ask if is it possible to find all the facets of the polyhedron defining ILP. *Porta* and *Polymake* ([16]) are two computer software designed for analyzing polytopes and polyhedra; both are capable of generating all facets for small instances of ILP. These two software find facets based on a projection method which is described very briefly here.

In integer programming it is in principle possible to find all the feasible solutions if the polyhedron defining the problem is bounded. Define $P(A,b) = \{x \in R^n : \ Ax \leq b\}$ where $A \in R^{m\times n}$ and $b \in R^m$ to be a polytope. If $F = (f_1, f_2, \ldots, f_M)$ is the set of integral feasible solutions inside the polytope $P(A,b)$ then $conv(P(A,b) \cap Z^n) = conv(F)$. The set of all the integer feasible solutions in $P(A,b)$ can be utilized to find the facets defining the convex hull of $P(A,b) \cap Z^n$. Every $x \in P(A,b) \cap Z^n$ can be defined by

$$\sum_{i=1}^{M} \lambda_i f_i \quad = \quad x, \quad \forall x \in P(A,b) \cap Z^n, \tag{8a}$$

$$\text{where} \quad \sum_{i=1}^{M} \lambda_i, \quad = \quad 1, \quad \lambda_i \geq 0, \qquad\qquad i = 1, \ldots, M, \tag{8b}$$

for some $\lambda_i \in R^1_+$, $i = 1, \ldots, M$. The idea is to eliminate the variables λ_i, to yield a system of equations containing only x. At the start we project the set defined by inequalities (8) on the plane defined by $\lambda_1 = 0$ and obtain a system of inequalities which does not contain λ_1. The projection is done by the Fourier-Motzkin elimination procedure ([8]). By iteratively projecting out the variables $\lambda_1, \ldots, \lambda_M$, we obtain a system of inequalities containing the variable x and which are facets to the convex hull of $P(A,b) \cap Z^n$. A big disadvantage with the Fourier–Motzkin method is that its computational complexity is exponential. (For more details, see [16].) Hence, even for small size instances this approach to find facets can be very time consuming. However, studying the facets of small instances of an ILP may help to an understanding of the facial structure for that class of problems.

2.4 Chvátal–Gomory cuts and the separation problem

In Section 2.3.2 we have presented the Chvátal–Gomory procedure to obtain valid inequalities (CG-cuts) for the feasible set of the ILP problem. Also, the separation problem is defined in Section 2.2.2. In this section we discuss some further definitions and notions of the Chvátal–Gomory separation problem. This section is divided into two major parts. In the beginning we give the definition of the first Chvátal closure and then present the rank-1 separation problem. The remainder of this section discusses the projected cuts for the mixed integer linear programming problems. The refrences [7] and [4] are mainly used for this section.

Assume $P = \{x \in R^n_+ : \ Ax \leq b\}$ where A is an $m \times n$ integer matrix, and b an m-dimensional integer vector. Let $P_I = conv(S)$ with $S = P \cap Z^n$ and assume $P_I \neq P$. As defined in Section 2.3.2, a CG cut is a valid inequality for P_I of the form $\lfloor u'A \rfloor x \leq \lfloor u'b \rfloor$, where $u \in R^m_+$. The vector u is called the *CG multiplier vector*. Note that CG-cuts depend on P and not directly on P_I, i.e. different formulations of the same problem can produce different CG-cuts. The *rank-1* closure or *the first Chvátal Closure* of P is defined as:

$$P_1 = \{x \in P : \lfloor u'A \rfloor x \leq \lfloor u'b \rfloor, \text{ for all } u \in [0,1)^m\}. \tag{9}$$

We define a $\{0, 1/2\}$–CG-cut as a CG-cut with multipliers $u' \in \{0, 1/2\}^m$ and define $P_{1/2}$ the polyhedron obtained by intersecting P with the half-spaces induced by all $\{0, 1/2\}$–CG-cuts, i.e:

$$P_{1/2} = \{x \in P : \lfloor u'A \rfloor x \leq \lfloor u'b \rfloor, \text{ for all } u \in \{0, \frac{1}{2}\}^m\}. \tag{10}$$

Notice that $P_{1/2}$ is a function of A and b. Clearly, $P_I \subseteq P_1 \subseteq P_{1/2} \subseteq P$. Although $P_1 = P$ holds if and only if $P = P_I$, one can have $P_{1/2} = P$ even if $P \neq P_I$. This case occurs, e.g., when $\frac{1}{2}b \in Z^m$. Therefore, $P_1 \subset P$ in case $P \neq P_I$, i.e., P_1 gives a better approximation of P_I than P. In some cases, $P_1 = P_{1/2} = P$ as, e.g, when P is the solution set for the matching problem ([5]).

2.4.1 Chvátal-Gomory separation problem

Because of the well-known equivalence between optimization and separation, we will address the *CG separation problem* in which we are given any point $x^* \in R^n$ and search for a hyperplane separating x^* from P_1, if any exists. Without loss of generality assume that this x^* lies in P. Therefore the separation problem is the following.

Definition 2.16 (CG-SEP). *Given any point $x^* \in P$ find a CG cut that is violated by x^*, i.e., find $u \in R^m_+$ such that $\lfloor u'A \rfloor x^* > \lfloor u'b \rfloor$, or prove that no such u exists.*

If, in addition to the assumption in the definition of CG-SEP, $u \in \{0, \frac{1}{2}\}$, then the separation problem is called $\{0, \frac{1}{2}\}$-SEP. It seems necessary to remind that the availability of a polynomial-time algorithm for CG-SEP would allow to optimize in polynomial time, a linear objective function over P_1 or $P_{1/2}$. $\{0, \frac{1}{2}\}$-SEP is equivalent to finding the minimum-weight member of a binary clutter (see [5]), which is an $\mathcal{NP}$-hard problem, implying that $\{0, \frac{1}{2}\}$-SEP is $\mathcal{NP}$-complete [5]. There exist, however, special cases where $\{0, \frac{1}{2}\}$-SEP is polynomially solvable. $\{0, \frac{1}{2}\}$-SEP can be solved in polynomial time if $\bar{A}^T$ is an *edge-path incidence matrix of a tree* (EPT matrix)[2] or if $\bar{A} = \binom{M}{I}$, and M is an EPT matrix ([5]).

In general CG-SEP is $\mathcal{NP}$-hard, so optimizing over P_1 is also $\mathcal{NP}$-hard. Here we are interested in optimizing the objective vector $c'x$ over the polyhedron P_1 in order to get a hopefully tight lower bound on the optimal value of the original integer linear programming problem.

2.4.2 Projected Chvátal-Gomory cuts for mixed integer linear programs

Suppose we are given a mixed-integer problem with the feasible region:

$$T = \{x \in R^p_+,\ y \in Z^n_+ : Ax + Gy \leq b\},$$

where A and G are $m \times n$ and $m \times p$ rational matrices, respectively and, b is an m-vector. One should consider that the CG-procedure does not work when there are continuous variables, in particular, we can not round down the right-hand side of an inequality to its integer part even when all of the coefficients on the left-hand

[2] A $p \times q$ $\{0,1\}$-matrix A is an *EPT-matrix* if there is a tree T with $p+1$ nodes such that each column of A is the characteristic vector of the edges of a path in T.

side are integers. However, we can obtain a procedure, related to the disjunctive procedure, that generalizes the CG-procedure and generates valid inequalities for T ([12]). These cuts are called *Gomory Mixed Integer* (GMI) cuts (also known as MIR cuts and split cuts). Although it is easy to find a GMI cut that separates an integer infeasible basic solution of the linear programming relaxation, separating other points by GMI cuts is $\mathcal{NP}$-hard ([4]).

Consider the Mixed Integer Linear Program MILP defined in the region T as

$$\min\{c'x + h'y : Ax + Gy \leq b,\ x \in R^p_+,\ y \in Z^n_+\},$$

where $c \in R^p$ and $h \in R^n$. Also consider the following two polyhedra in the (x, y)-space

$$P^{XY} = \{(x, y) \in R^n_+ \times R^p_+ : Ax + Gy \leq b\}; \tag{11}$$
$$P^{XY}_I = conv(\{(x, y) \in P(x, y) : y \text{ integral}\}). \tag{12}$$

Our aim here is to project first the linear programming relaxation of the MILP at hand onto the space of the integer variables y, and then to derive CG-cuts for the projected polyhedron. For this purpose, we define the projection of P^{XY} onto the space of the y variables as

$$\begin{aligned} P^Y &= \{y \in R^p_+ :\ \exists\ x \in R^n_+ \text{ s.t. } Ax + Gy \leq b\} \\ &= \{y \in R^p_+ :\ u^k Gy \leq u^k b,\ k = 1, \ldots, K\} \\ &= \{y \in R^p_+ :\ \bar{G}y \leq \bar{b}\}, \end{aligned}$$

where $u^1, \ldots, u^K$ are the extreme rays of the projection cone $\{u \in R^m_+ \ :\ u'A \geq 0'\}$, $\bar{G} = u^k G$ and $\bar{b} = u^k b$.

The *projected Chvtátal-Gomory* (pro-CG) can be obtained from the system $\bar{G}y \leq \bar{b}$, $y \geq 0$, i.e., $\lfloor w'\bar{G} \rfloor y \leq \lfloor w'\bar{b} \rfloor$ for some $w \geq 0$. Note that any row of $\bar{G}y \leq \bar{b}$ is a linear combination of the rows of $Gy \leq b$ with multipliers $\bar{u} \geq 0$ where $\bar{u}A \geq 0$. Therefore a pro-CG cut can equivalently and directly be defined as an inequality of the form

$$\lfloor u'G \rfloor y \leq \lfloor u'b \rfloor \quad \text{for any } u \geq 0 \text{ such that } u'A \geq 0'. \tag{13}$$

Denote the rank-1 Chvátal closure of P^Y by P^Y_1 and the convex hull of $P^Y \cap Z^n$ by P^Y_I.

In Chapter 5 we model and solve the rank-1 separation problem for the opportunistic replacement problem. We also generate pro-CG cuts for the opportunistic replacement problem defined in Chapter 3.

2.5 Benders decomposition procedure for mixed-variable programming problems

The decomposition method refers to an algorithm which partitions the variables of an optimization problem into two subsets. The first step of the Benders algorithm consist of fixing a certain amount of variables in our original ILP problem, hereby making the resulting sub-problem easy to solve. The essence of Benders decomposition lies in determining which variables to fix, such as to simplify the resulting

sub-problem. This decision will often require specific knowledge of the problem at hand as well as known ways to solve similar problems quickly.

This section is divided into two parts, first we present the theoretical developments that leads to the decomposition. It is followed by the Benders algorithm.

2.5.1 An equivalent representation of mixed integer programing problems

Consider the general mixed integer programming problem defined by

$$\begin{aligned} &\text{minimize } c'x + d'y, && \text{(14a)}\\ &\text{s.t.} \quad Ax + Fy \geq b, && \text{(14b)}\\ &\qquad x \geq 0, \qquad y \in S, && \text{(14c)} \end{aligned}$$

where A and F are $m \times n$- and $m \times p$- matrices, respectively, x and c are n-vectors, d and y are p-vectors. S is a nonempty and bounded subset of Z^n. Here, the x variables are continuous and y discrete. Except for the integrality requirements on y, the model (14) has a linear programming format. Benders method decomposes this model in such a way that it can be solved as an alternating sequence of linear programs and pure integer programs.

Assume that the vector y is fixed to some specific value. For this vector to be feasible, it must lie in the set

$$R = \{y \in S \mid \exists x \geq 0 \quad \text{such that} \quad Ax \geq b - Fy\}. \tag{15}$$

We assume that the set R is nonempty, otherwise the original problem (14) is infeasible. We can rewrite the problem (14) as that to

$$\underset{y \in R}{\text{minimize}} \left\{ d'y + \min\{c'x \mid Ax \geq b - Fy,\ x \geq 0\} \right\}. \tag{16}$$

When the value of y is fixed, the minimization subproblem of (16) is a linear programming problem in the variables x and we formulate the linear programming dual of this subproblem. By the fundamental theorem of duality in linear programming (Theorem 6.1, page 267, [3]) it holds that

$$\min_x\{c'x \mid Ax \geq b - Fy,\ x \geq 0\} = \max_u\{(b - Fy)u' \mid A'u \leq c,\ u \geq 0\}, \tag{17}$$

which states that if the primal and dual problems are feasible they possess finite optimal solutions with equal objective values. This lets us formulate a new equivalent problem to (14) as that to

$$\underset{y \in R}{\text{minimize}} \left\{ d'y + \max\{(b - Fy)'u \mid A'u \leq c, \quad u \geq 0\right\}. \tag{18}$$

Followed from the fact that an optimum solution to the maximization subproblem of (18), (i.e., the right problem in (17)) must be at one of the extreme points of its feasible region, a different approach can be taken to solve it. Consider the polyhedron P defined by the constraint set of the dual problem; $P = \{u \in R^m \mid A'u \leq c, u \geq 0\}$. Assume that P is nonempty; otherwise, the dual problem is infeasible which implies that (14) is unbounded. The optimum of the maximization subproblem is at an

extreme point of P or approaches $+\infty$ along an extreme ray. If the dual is unbounded then, by the duality theorem (see [3]), the corresponding primal problem and —as a result— the original problem (14) is infeasible. But we assume that $R \neq \emptyset$, so the polyhedron P is bounded and the number of extreme points is finite, so we only need to check the extreme points to find the maximum (P unbounded $\Longleftrightarrow$ (14) infeasible $\Longleftrightarrow R \neq \emptyset$). Let u_i^p, $i = 1, \ldots, n_p$ be the extreme points of P. Then the problem (18) can be rewritten as

$$\underset{y \in R}{\text{minimize}} \left\{ d'y + \max_{1 \leq i \leq n_p} (b - Fy)' u_i^p \right\}, \tag{19}$$

which is equivalent to

$$\begin{aligned} &\text{minimize} && z && && (20a)\\ &\text{subject to} && z \geq d'y + (b - Fy)' u_i^p, && i = 1, \ldots, n_p, && (20b)\\ & && y \in R. && && (20c) \end{aligned}$$

In the model (20), there is one constraint for each extreme point of P.

Applying the Farkas lemma (Lemma 5.1, Sec.5.3, [3]) to the linear equality system $Ax - s = b - Fy$ with $x \geq 0$, $s \geq 0$, where $y \in R$ is fixed, yields that y is feasible for (14) if and only if

$$(b - Fy)' u \leq 0 \tag{21}$$

for all $u \in P_0 = \{u \in R^m \mid A'u \leq 0,\ u \geq 0\}$. The cone P_0 is a polyhedron; therefore each vector $u \in P_0$ can be written as a convex combination of the generators u_i^r, $i = 1, \ldots, n$ (i.e., extreme rays of P_0). Each $u \in P_0$ can then be expressed as

$$u = \sum_{i=1}^{n_r} \lambda_i u_i^r, \quad \text{where} \quad \lambda_i \geq 0,\ i = 1, \ldots, n_r. \tag{22}$$

By substituting (22) in (21) we have that $\sum_{i=1}^{n_r} \lambda_i (b - Fy)' u_i^r \leq 0$ which holds for all $\lambda_i \geq 0$ if and only if

$$(b - Fy)' u_i^r \leq 0, \quad i = 1, \ldots, n_r. \tag{23}$$

Therefore, the vector $y \in S$ is feasible for (14) if and only if (22) holds. Hence the set R can be written as

$$R = \{y \in S \mid (b - Fy)' u_i^r \leq 0,\ i = 1, \ldots, n_r, \}. \tag{24}$$

Using this explicit definition of R we obtain a new formulation of (14) given by

$$\begin{aligned} &\text{minimize} && z && && (25a)\\ &\text{subject to} && z \geq d'y + (b - Fy)' u_i^p, && i = 1, \ldots, n_p, && (25b)\\ & && 0 \geq (b - Fy)' u_i^r, && i = 1, \ldots, n_r, && (25c)\\ & && y \in S. && && (25d) \end{aligned}$$

The problem (25) is often referred to as the "*complete master problem*" of Benders decomposition algorithm. In the model (25) there are integer variables y and one real variable z. Theorem 1 on page 374 in [11] summarizes the previous statements.

Theorem 2.4 (Equivalence of (25) and (14)).

a. *(25) has a feasible solution* $\Longleftrightarrow$ *(14) has a feasible solution.*

b. *(25) is feasible without having an optimal solution* $\Longleftrightarrow$ *(14) is feasible without having an optimal solution.*

c. *If* $(\mathring{z}, \mathring{y})$ *solves (25) and* $\mathring{x}$ *solves the linear program to*

$$\begin{aligned} &\textit{minimize} && c'x && (26a)\\ &\textit{subject to} && Ax \geq b - F\mathring{y}, \quad x \geq 0. && (26b) \end{aligned}$$

then $(\mathring{x}, \mathring{y})$ *solves (14) and* $\mathring{z} = c'\mathring{x} + d'\mathring{y}$

d. *If* $(\mathring{x}, \mathring{y})$ *solves (14) and* $\mathring{z} = c'\mathring{x} + d'\mathring{y}$*, then* $(\mathring{z}, \mathring{y})$ *solves (25).*

2.5.2 Benders algorithm

Theorem 2.4 states that for obtaining the optimal solution to the original model (14), one needs to solve (25) to find the solution $(\mathring{z}, \mathring{y})$ and then obtain the optimal value $\mathring{x}$ by solving the primal problem in (17) with $y = \mathring{y}$. The new model (25) can not be practically solved because its formulation requires that all extreme points and extreme rays of (25) are identified. The number of extreme points and extreme rays can be considerably large even for small dimension problems. Since only a small number of constraints will be binding at the optimal solution, (25) can be relaxed to a problem with no or few constraints. Define the new modified problem (the restricted master problem)

$$\begin{aligned} &\text{minimize} && z && && (27a)\\ &\text{subject to} && z \geq d'y + (b - Fy)'u_i^p, && i \in I_1, && (27b)\\ &&& 0 \geq (b - Fy)'u_i^r, && i \in I_2, && (27c)\\ &&& y \in S. && && (27d) \end{aligned}$$

where I_1 and I_2 are proper subsets of the sets $\{1, \ldots, n_p\}$ and $\{1, \ldots, n_r\}$, respectively. Let G, G' be the set of all (z, y) satisfied by the constraints (25a)–(25d) and (27b)–(27d) respectively. Then $G \subseteq G'$. Benders algorithm begin with solving the problem (27). If the solution satisfies the remaining constraints in (25a) and (25d), the solution is also optimal to (25), i.e., it lies in G. If not, at least one constraint in (25) is not satisfied. The linear programming problem or its dual in (17) is then solved to find a new extreme point u_i^p or an extreme ray u_i^r. This solution is used to define a new constraint which will be added to (27b) or (27c).

The maximization dual problem is solved in each step of Benders method, so it is of importance to note the conditions where it is unbounded or has no feasible solution. The feasible region P of the dual problem is independent of the variable y. If the dual problem has no feasible solution, the primal problem is either infeasible or unbounded for all $y \in R$ which yields that (14) is unbounded or infeasible. This case is not interesting for practical problems so we assume that the dual problem in (17) possesses feasible solutions. If the dual problem has an unbounded solution for some $y \in S$, the primal is then infeasible for that y and the simplex method locates an extreme ray. This case may frequently happen and should be considered in the

algorithm by adding a new constraint to (27c) with the new obtained extreme ray. Now we state the general Benders method in details.

Let (27) have a finite optimal solution $(\mathring{z}, \mathring{y})$. The solution is optimal to (25) if and only if it holds that

$$(b - F\mathring{y})'u_i^p \leq \mathring{z} - d'\mathring{y}, \quad i = 1, \ldots, n_p, \tag{28a}$$
$$(b - F\mathring{y})'u_i^r \leq 0, \qquad i = 1, \ldots, n_r. \tag{28b}$$

We intend to find the most unsatisfied constraint in (28a) or (28b). The most unsatisfied constraint of (28a) is given by

$$\arg \max_{1 \leq i \leq n_p} (b - F\mathring{y})'u_i^p \leq \mathring{z} - d'\mathring{y}. \tag{29}$$

Since the linear function $(b - F\mathring{y})'u$ attains a finite maximum over P at an extreme point of P, the constraint (29) can be obtained by solving the linear program $\max\left\{(b - F\mathring{y})'u \mid u \in P\right\}$, which is the dual problem in (17) with $y = \mathring{y}$. This problem has either a finite optimal solution or an unbounded solution. In the unboundness case the objective value approaches $+\infty$ along the half line, $u_i^p + \lambda u_i^r$, $\lambda \geq 0$ and $(b - F\mathring{y})'u_i^r > 0$ for some $i \in \{1, \ldots, n_r\}$. This implies that one of the constraints (28b) is violated. Thus both sets of constraints (28a) and (28b) are satisfied if and only if it holds that

$$\max_u \{(b - F\mathring{y})'u \mid A'u \leq c,\ u \geq 0\} \leq \mathring{z} - d'\mathring{y}. \tag{30}$$

If some of the constraints (28a) and (28b) are not satisfied, then it holds that

$$\max_u \{(b - F\mathring{y})'u \mid A'u \leq c,\ u \geq 0\} > \mathring{z} - d'\mathring{y}. \tag{31}$$

Consider the problem (27) with few or no constraints. If (27) is infeasible so are (25) and (14). Let $(\mathring{z}, \mathring{y})$ be finite optimal solution of the problem (27). If (27) has an unbounded solution let $\mathring{z} = -\infty$ and $\mathring{y}$ be any vector in S. If the solution to the maximization linear dual problem in (17) is bounded it is obtained at an extreme point of P, say $\mathring{u}$, then $(b - F\mathring{y})'\mathring{u} > \mathring{z} - d'\mathring{y}$. This constraint is not satisfied by the solution to the current problem (27), therefore we add the constraint $z \geq (b - F\mathring{y})'\mathring{u} + d'y$ to the current problem which yields a new (27). In the case where the solution of the linear dual program (17) leads to unboundness, an extreme ray of P, $\mathring{v}$ is found, where $\mathring{v}$ satisfy $(b - F\mathring{y})'\mathring{v} > 0$, but $\mathring{y}$ does not satisfy the constraint $0 \geq (b - Fy)'\mathring{v}$. Thus, this constraint is added to (27). The problem (27) is solved again with the new constraint, a new solution $\mathring{x}$ is obtained by solving the primal problem in (17) with $y = \mathring{y}$. Now the optimality of the solution to (27) should be checked. An optimality test can be obtained directly from (30), according to the following theorem.

Theorem 2.5 (**Optimality test**(Theorem 2 Sec.7.3 [11])). *$(\mathring{z}, \mathring{y})$ is optimal for* (25) *if and only if*

$$\max\{(b - F\mathring{y})'u \mid A'u \leq c,\ u \geq 0\} = \mathring{z} - d'\mathring{y}. \tag{32}$$

Theorem 2.6 (**Finite convergence**(Theorem 3 Sec.7.3 [11])). *Benders iterative procedure will terminate in a finite number of iterations, either with the information that* (14) *is infeasible or unbounded, or with an optimal solution to* (14).

Proof. The program (25) has a finite number of constraints. If the optimality test is not passed, then one or more new constraints are added to the program (27). Thus, in a finite number of iterations either the optimality test is passed or a full set of constraints will be obtained. The program (14) is infeasible if and only if the program (27) is infeasible. The program (14) is unbounded if and only if the dual linear program is infeasible, which will be detected in the first step. □

In Chapter 4 some special properties of the opportunistic replacement problem will be detected which will be essential in solving it using Benders decomposition method. Then Benders algorithm adopted to our problem will be presented.

In the following chapter we introduce the opportunistic replacement problem.

3 The opportunistic replacement problem

This chapter is dedicated to defining the opportunistic replacement model and the study of some of its mathematical properties. In Section 3.1 a mathematical model for determining an optimal opportunistic replacement schedule when component lives are deterministic is introduced. The opportunistic replacement problem is modeled as an integer linear programming problem. This basic opportunistic replacement problem is $\mathcal{NP}$-hard. In Section 3.2 it is stated that the convex hull of the set of feasible replacement schedules is full–dimensional. When the maintenance occasions are fixed, the remaining problem can be stated as a linear program for the case when the maintenance costs are monotone with time, this linear program can be solved by a greedy procedure. Furthermore, all the inequalities that are necessary in the definition of the problem are facet-inducing (Section 3.3). At the end of this chapter some facets of the polyhedron defining the opportunistic replacement problem which has been developed in [13] is briefly presented. The references used in this chapter are [18], [17] and [13].

3.1 The opportunistic replacement model

In this section we introduce an optimization model for determining an optimal maintenance schedule when the problem data is deterministic. Consider a set $\mathcal{N}$ of components; with $|\mathcal{N}| = N$. Consider also a set $\mathcal{T} = \{1, \ldots, T\}$ of times, with $T \geq 2$. T is considered as the time horizon for the maintenance planning. Each component $i \in \mathcal{N}$ has a fixed life of T_i. Without loss of generality we can assume that for all $i \in \mathcal{N}$, $T_i \geq 2$ holds, otherwise replacement of component i is necessary at each time step. Also $T_i \leq T$, $i \in \mathcal{N}$, i.e., each component needs at least one replacement during the time horizon. The purchase cost at time $t \in \mathcal{T}$ for component i is $c_{it} > 0$. There is a fixed cost of $d_t > 0$ associated with performing maintenance for any component i at time t, independent of the number of parts replaced. For any given component $i \in \mathcal{N}$ in the system, $\{l+1, \ldots, l+T_i\}$ corresponds to a window of T_i time steps, starting at time step $l+1$, in which component i must be replaced.

The objective is to minimize the total cost for having a functional system without failure between times 1 and T, i.e., for each component $i \in \mathcal{N}$, no period without replacement longer than the component's life T_i may exist. The model considers the cost of maintenance occasions and minimizes the maintenance costs. We define the decision variables

$$z_t = \begin{cases} 1, & \text{if maintenance shall occur at time } t, \\ 0, & \text{otherwise,} \end{cases} \qquad t \in \mathcal{T},$$

$$x_{it} = \begin{cases} 1, & \text{if component } i \text{ shall be replaced at time } t, \\ 0, & \text{otherwise,} \end{cases} \qquad i \in \mathcal{N}, \quad t \in \mathcal{T},$$

The opportunistic replacement problem is defined as to

$$\underset{(x,z)}{\text{minimize}} \quad \sum_{t \in \mathcal{T}} \left(\sum_{i \in \mathcal{N}} c_{it} x_{it} + d_t z_t \right), \tag{33a}$$

$$\text{subject to} \quad \sum_{\substack{t=l+1 \\ l+T_i}} x_{it} \geq 1, \qquad \ell = 0, \ldots, T - T_i, \quad i \in \mathcal{N}, \tag{33b}$$

$$
\begin{aligned}
x_{it} &\le z_t, && t \in \mathcal{T}, \quad i \in \mathcal{N}, && \text{(33c)}\\
x_{it} &\ge 0, && t \in \mathcal{T}, \quad i \in \mathcal{N}, && \text{(33d)}\\
z_t &\le 1, && t \in \mathcal{T}, && \text{(33e)}\\
x_{it} &\in \{0,1\}, && t \in \mathcal{T}, \quad i \in \mathcal{N}, && \text{(33f)}\\
z_t &\in \{0,1\}, && t \in \mathcal{T}. && \text{(33g)}
\end{aligned}
$$

The constraints (33b) state that for all $i \in \mathcal{N}$ and in any time window of length T_i, the component i must be replaced at least once. The constraints (33c) ensure that if component i is replaced at time t a maintenance occasion occurs, which enforces the payment of the fixed maintenance cost d_t. When this cost is paid it leads to no extra maintenance costs. The constraints (33d)–(33g) define the restrictions on the variables x_{it} and z_t for all $i \in \mathcal{N}$ and $t \in \mathcal{T}$. If the constraints (33f) and (33g) are removed, a so called LP relaxation of the problem is obtained.

3.2 Complexity analysis and special properties

According to integer programming literature the set covering problem is considered to be an $\mathcal{NP}$-hard problem. In [18], Theorem 1, it is proved that the set covering problem is polynomially reducible to the opportunistic replacement problem. This leads to the conclusion that the opportunistic replacement problem is $\mathcal{NP}$-hard. However when fixing the values of some of the variables (e.g., z variables) in the model (33), the resulting subproblem turns out to possesses "nicer" properties. In this section some special properties of the opportunistic replacement problem (33) is presented.

Consider the polyhedron in $R^{N\times T}$ defined by (33b)–(33d) when z_t, $t \in \mathcal{T}$ are fixed to binary values . Let $\tilde{z}_t \in \{0,1\}$, $t \in \mathcal{T}$ and define $\widetilde{\mathcal{T}} = \{t \in \mathcal{T} \mid \tilde{z}_t = 1\}$. The following proposition is stablished in [18].

Proposition 3.1 (integrally polyhedron). *The polyhedron defined by (33b), (33d), and*

$$
\begin{aligned}
x_{it} &\le 1, && t \in \widetilde{\mathcal{T}}, && \text{(34a)}\\
x_{it} &\le 0, && t \in \mathcal{T} \setminus \widetilde{\mathcal{T}}, && \text{(34b)}
\end{aligned}
$$

for $i \in \mathcal{N}$, is integral.

Proof. The constraint matrix A corresponding to the system of inequalities given by (33b) and (34) is an interval matrix (see Definition 2.13). Hence A^T is an interval matrix and hence TU (see Section 2.2.3). Proposition 2.1. on page 540 in [12] states that the transpose matrix of a TU matrix is TU. Thus the constraint matrix A is TU. Since the right-hand sides of (33b) and (34) are all integral it follows from Proposition 2.4 that the corresponding polyhedral is integral. □

A direct result of Proposition 3.1 is that the binary requirements (33f) on the variables x_{it} can be relaxed, provided that the opportunistic replacement model is to be solved using an algorithm that detects extreme optimal solutions to the linear programming subproblems.

A special instance of the model (33) occurs when the costs are monotonous with time, i.e., costs are non-increasing ($c_{i,t+1} \le c_{it}$ and $d_{t+1} \le d_t$ for all i and t) or

non-decreasing ($c_{i,t+1} \geq c_{it}$ and $d_{t+1} \geq d_t$ for all i and t) with time. For these cases interesting special properties of the optimal solutions can be proven.

Letting the variables z_t, $t \in \mathcal{T}$, be assigned binary values, $\tilde{z}_t \in \{0,1\}$, the remaining optimization model separates over the components $i \in \mathcal{N}$ and the corresponding constraint matrix is TU. Thus for every component $i \in \mathcal{N}$ the linear programming subproblem is given by

$$\underset{x_i}{\text{minimize}} \quad \sum_{t \in \mathcal{T}} c_{it} x_{it}, \tag{35a}$$

$$\sum_{t=l+1}^{l+T_i} x_{it} \geq 1, \quad l = 0, \ldots, T - T_i, \tag{35b}$$

$$0 \leq x_{it} \leq \tilde{z}_t, \quad t \in \mathcal{T}. \tag{35c}$$

Assume without loss of generality that for each $i \in \mathcal{N}$, the costs c_{it} and d_t for all $t \in \mathcal{T}$ are non-increasing with time. We claim that Algorithm 1, based on a greedy rule, yields an optimal solution to the linear program (35). In Algorithm 1, from [17], component i is replaced as late as possible within its life and among the times $t \in \widetilde{\mathcal{T}}$. Algorithm 1 is followed by a proposition, stated in [17], which proves that the non-increasing greedy rule yields the optimum to the subproblem (35).

Algorithm 1 (Non-increasing cost greedy rule for component $i \in \mathcal{N}$)

$\widetilde{\mathcal{T}} \leftarrow \{t \in \mathcal{T} \mid \tilde{z}_t = 1\} \cup \{T+1\}$;
$\tilde{x}_{it} \leftarrow 0 \ \forall t \in \mathcal{T}$; $\tilde{t} \leftarrow \min\{t \mid t \in \widetilde{\mathcal{T}}\}$; $s \leftarrow 0$; $\widetilde{\mathcal{T}} \leftarrow \widetilde{\mathcal{T}} \setminus \{\tilde{t}\}$;
while $\widetilde{\mathcal{T}} \neq \emptyset$ **do**
  $\hat{t} \leftarrow \min\{t \mid t \in \widetilde{\mathcal{T}}\}$;
  if $T_i < \hat{t} - s$ **then**
    $\tilde{x}_{i\tilde{t}} \leftarrow 1$; $s \leftarrow \tilde{t}$;
  end if
  $\tilde{t} \leftarrow \hat{t}$; $\widetilde{\mathcal{T}} \leftarrow \widetilde{\mathcal{T}} \setminus \{\tilde{t}\}$;
end while
Return $\tilde{x}_{it}$, $t \in \mathcal{T}$.

Proposition 3.2 (**Non-increasing greedy rule yields optimum**). *Assume that $c_{i,t+1} \leq c_{it}$ holds, $i \in \mathcal{N}$, $t \in \mathcal{T} \setminus \{T\}$. Let $\tilde{z}_t \in \{0,1\}$, $t \in \mathcal{T}$, and assume that the set $\widetilde{\mathcal{T}} = \{t \in \mathcal{T} \mid \tilde{z}_t = 1\}$ is such that for each $t \in \widetilde{\mathcal{T}} \cup \{0\}$ there is an $s \in \widetilde{\mathcal{T}} \cup \{T+1\}$ with $1 \leq s - t \leq \min_{i \in \mathcal{N}} T_i$. Then, Algorithm 1 produces an optimal solution to the model* (35).

Proof. By assumption, $\tilde{x}_i$ is feasible in (35). Let $\bar{x}_i \neq \tilde{x}_i$ be feasible in (35). Postpone, where possible, replacements corresponding to $\bar{x}_i$ to the next time point in $\widetilde{\mathcal{T}} \cup \{T+1\}$. This will transform $\bar{x}_i$ to $\tilde{x}_i$ without introducing any additional replacements and at a non-increasing cost. Hence, $\sum_{t \in \mathcal{T}} c_{it}(\tilde{x}_{it} - \bar{x}_{it}) \leq 0$ holds; the result follows. □

For the non-decreasing costs an analogous algorithm and result can be obtained. The algorithm and more details is found in [17] and [18].

3.3 The replacement polytope

In this section properties of the polyhedron defined by the opportunistic replacement is stated. A complete description of the polytope defined the opportunistic replacement problem can be achieved by a finite set of linear inequalities. From Sections 2.1.2 and 2.2.2 it follows that by knowing all the inequalities describing the convex hull of the set defining the problem, the ILP can be solved as a linear programming problem. Unfortunately, for $\mathcal{NP}$-hard problems, there is almost no hope of finding a good description. Still, for given instances of the opportunistic replacement problem our goal here is to find effective ways to approximate the convex hull and to contribute to this description by studying polyhedral properties and searching for classes of facets.

Let the set $S \subset R^{N\times T} \times \{0,1\}^T$ be defined by the values of (x, z) that satisfy the inequalities (33b)–(33e). Define the *replacement polytope* as $conv(S)$. The following proposition is stated and proven in [18].

Proposition 3.3 (**Dimension of the replacement polytope).** *If $T_i \geq 2$ for all $i \in \mathcal{N}$, then the dimension of $conv(S)$ is $(N + 1)T$, that is, $conv(S)$ is full-dimensional.*

Proposition 3.4 (**The inequalities** (33b)–(33e) **define facets of the replacement polytope).** *If $T_i \geq 2$ for all $i \in \mathcal{N}$, then each of the inequalities: $\sum_{t=l+1}^{l+T_i} x_{it} \geq 1$, $l = 0, \ldots, T - T_i$, $i \in \mathcal{N}$; $x_{it} \leq z_t$, $i \in \mathcal{N}$, $t \in \mathcal{T}$; $x_{kt} \geq 0$, $k \in \mathcal{N}$: $T_k \geq 3$, $t \in \mathcal{T}$; and $z_t \leq 1$, $t \in \mathcal{T}$, define a facet of $conv(S)$.*

For proving Proposition 3.4 the Theorem 2.2 can be utilized (see [18] or [17]). The inequalities (33b)–(33e) define facets for $conv(S)$ and they are thus necessary in the description of the polyhedron $conv(S)$ but they are not completely describing the convex hull of S, i.e., they are not sufficient to define $conv(S)$ ([18]).

3.4 Previous work on facet generation for the opportunistic replacement problem

The polyhedron defined by the constraints of the opportunistic replacement problem has non-integer extreme points; therefore, finding the hyperplanes which, in addition to the constraints of the associated linear programming problem, define the convex hull of integer solutions to the problem is necessary. New classes of facets has been found for the opportunistic replacement problem in [13] derived by combinatorial implications and $\{0, 1/2\}$-Chvátal-Gomory cuts. In this section we briefly discuss these classes of facets.

Assume an instance of the opportunistic replacement problem (33) is in hand. Let $p, q \in \mathcal{N}$ be such that $T_q < T_p$. It is shown in [13] for $s \in \{1, \ldots, l + T_p - T_q\}$ and $l \in \{0, \ldots, T - T_p\}$ that the inequality

$$\sum_{t=l+1}^{l+s-1} x_{pt} + \sum_{t\in\{l+s,l+s+T_q\}} z_t + \sum_{t=l+s+1}^{l+s+T_q-1} (x_{pt} + x_{qt}) + \sum_{t=l+s+T_q+1}^{l+T_p} x_{pt} \geq 2, \qquad (36)$$

defines a facet for $conv(S)$. Some extensions can be done on this inequality. Assume that p and l are fixed and let m inequalities of the form (36) be given as

$\sum_{i\in\mathcal{N}}\sum_{t\in\mathcal{T}}\lambda_{it}^{(k)}x_{it}+\sum_{t\in\mathcal{T}}\mu_t^{(k)}z_t\geq\rho^{(k)}$ for $k=1,\ldots,m$, where $\lambda_{it}^{(k)}$ and $\mu_t^{(k)}$ are the coefficients of x_{it} and z_t in each inequality k, and $\rho^{(k)}$ is the right hand side constant in inequality k. Define a new inequality by

$$\sum_{i\in\mathcal{N}}\sum_{t\in\mathcal{T}}\lambda_{it}x_{it}+\sum_{t\in\mathcal{T}}\mu_t z_t\geq\rho, \tag{37}$$

$$\begin{aligned}
\text{where}\quad & \lambda_{it}=\sum_{k=1}^{m}\lambda_{it}^{(k)}, && i\in\mathcal{N}\setminus\{p\},\quad t\in\mathcal{T},\\
& \lambda_{pt}=\min_k\{\lambda_{it}^{(k)}\}, && t\in\mathcal{T},\\
& \mu_t=\max_k\{\mu_t^{(k)}\}, && t\in\mathcal{T},\\
\text{and}\quad & \rho=1+\sum_{k=1}^{m}\big(\rho^{(k)}-1\big).
\end{aligned}$$

We now seek to find some conditions under which the inequality (37) is valid, and some conditions under which it defines a facet. Let $\tau_1^{(k)}=\min\{t\mid\mu_t^{(k)}=1\}$ and $\tau_2^{(k)}=\max\{t\mid\mu_t^{(k)}=1\}$. We will assume that these parameters are ordered such that: $\tau_1^{(1)}\leq\tau_1^{(2)}\leq\ldots\leq\tau_1^{(m)}$ and $\tau_2^{(1)}\leq\tau_2^{(2)}\leq\ldots\leq\tau_2^{(m)}$. The following proposition from [13] states the conditions under which (37) is facet.

Proposition 3.5. *An inequality of the form* (37)*, such that if* $\tau_1^{(k)}=\tau_1^{(k')}$ *for some* $k\neq k'$*, then* $\tau_2^{(k)}\neq\tau_2^{(k'')}$ *for any* $k''=1,\ldots,m$*,* $k''\neq k$*, is valid. Furthermore, if* $\tau_2^{(k)}=\tau_1^{(k+1)}$*,* $k=1,\ldots,m-1$*, then* (37) *defines a facet for* $conv(S)$*.*

Another valid inequality can be obtained from the inequality (37) with conditions in Proposition 3.5. Assume that we are given the inequality (37) of the form (λ,μ,ρ) with the same component p at the window of time $\{l+1,\ldots,l+T_p\}$. Also assume $\mu_{l+1}=0$, pick a new component p' such that $T'_p\geq T_p+1$ and $l'\in\{l+T_p+1-T_{p'},\ldots l\}$. Define a new inequality (λ',μ',ρ') by:

$$\sum_{it}\lambda'_{it}x_{it}+\sum_{t}\mu'_t z_t\geq\rho', \tag{38}$$

where $\lambda'_{it}=\lambda_{it}$, $i\in\mathcal{N}\setminus\{p,p'\}$, $t\in\mathcal{T}$, $\lambda'_{pt}=\lambda_{pt}-\mathcal{X}_{\{l+1\}}$,
$\lambda'_{p't}=\mathcal{X}_{\{l'+1,\ldots,l'+T'_p\}\setminus\{l+1,l+T_p+1\}}$, $\mu'_t=\mu_t+\mathcal{X}_{\{l+1,l+T_p+1\}}$, $\rho'=\rho+1$ and $\mathcal{X}$ is the indicator function. If (λ,μ,ρ) defined in (37) is a facet of $conv(S)$ then the inequality (λ',μ',ρ') in (38) defines a facet for $conv(S)$.

From Section 2.3.2 we know that by applying the *Chvátal-Gomory procedure* a finite number of times all of the valid inequalities for S can be generated. It is known that iteratively generating mod 2-cuts[3] gives the convex hull of bounded integer feasible sets ([9]). In [13] the generation of valid inequalities for the opportunistic replacement problem with Chvátal-Gomory inequalities using only $\{1/2\}$ as multipliers has also been studied. The procedure is to pick an odd number of inequalities

[3] If $P=\{x\in R^n|Ax\leq b\}$ with $A\in Z^{m\times n}$, then a mod-2 is an inequality of the form $\frac{1}{2}u'Ax\leq\lfloor\frac{1}{2}u'b\rfloor$, where $u_i\in\{0,1\}$ for all $i=1,\ldots,m$ and $\frac{1}{2}u'A\in Z^n$; i.e., $u'A\equiv$ mod 2.

of the form (33b) that overlap in time, and mix them together with the inequalities of the form (33c) in the Chvátal-Gomory procedure defined in Section 2.3.2, using $\frac{1}{2}$ as multipliers to obtain CG-valid inequalities for the replacement polyhedron. The author's conclusion is that the characteristics of when $\{0, \frac{1}{2}\}$-cuts become facets or even valid inequalities seems very bad to include in a computer program. For producing facets and valid inequalities for our problem stated above a constraint generation approach has been implemented, in which the separation problem is formulated as a shortest path-problem in a specific graph. Several graphs of reasonable sizes has been constructed. The facet generation seem to behave nicely when the associated graphs are simple. Graphs corresponding to useful inequalities, however, are not simple and hence computationally very hard to generate.

4 Benders decomposition method applied to the opportunistic replacement problem

This chapter is dedicated to the implementation of the Benders decomposition method adopted to the opportunistic replacement problem.

Consider the opportunistic replacement model (33). The variables z_t refer to the maintenance occasions at time $t \in \mathcal{T}$ and x_{it} to the replacement of component $i \in \mathcal{N}$ at time t. From Proposition 3.1 it is known that the binary requirements on the variables x_{it} can be relaxed, and if the maintenance occasions are fixed–when the cost are monotonous with time– the linear subproblems can be solved by a greedy rule. Therefore it seems natural to fix variables z_t, $t \in \mathcal{T}$ and (to attempt) to solve the mixed-integer model using an algorithm that detects optimal solutions to linear programming subproblems and search among these solutions to obtain "the best" one, which is the optimal solution to the opportunistic replacement problem. By this knowledge, we expect Benders decomposition method to be efficient for the opportunistic replacement problem.

In this chapter Benders partitioning method adopted to our problem is discussed and the summary of the algorithm is presented. Some computational tests and results are also presented.

4.1 Special properties of the opportunistic replacement linear dual programming problem

Before we describe the Benders decomposition of the opportunistic replacement problem, we present some of the properties of the dual problem of the linear programming subproblems of the opportunistic replacement problem where the maintenance occasions are fixed.

Consider the opportunistic replacement problem in (33). Let the variables z_t, $t \in \mathcal{T}$, in the problem be assigned binary values, $\tilde{z}_t \in \{0,1\}$. Then the remaining optimization problem separates over each component $i \in \mathcal{N}$. Let $i \in \mathcal{N}$ be fixed. In order to simplify the presentation, we define the sets $\widetilde{\mathcal{T}} = \{t \in \mathcal{T} |\ \tilde{z}_t = 1\}$, $\mathcal{L}_i = \{0, \ldots, T - T_i\}$, $\overline{\mathcal{T}}_{il} = \{l+1, \ldots, l+T_i\}$ for $l \in \mathcal{L}_i$ and for $t \in \mathcal{T}$, $\overline{\mathcal{L}}_{it} = \big\{\max\{0, t-T_i\}, \ldots, \min\{t-1, T-T_i\}\big\}$, i.e., $\overline{\mathcal{L}}_{it} = \{l \in \mathcal{L}_i \mid t \in \overline{\mathcal{T}}_{il}\}$.

The linear programming subproblem for component $i \in \mathcal{N}$ is then to

$$
\begin{aligned}
(P) \qquad & \underset{x_i}{\text{minimize}} && \sum_{t \in \mathcal{T}} c_{it} x_{it}, && && \text{(39a)}\\
& \text{subject to} && \sum_{t \in \overline{\mathcal{T}}_{il}} x_{it} \geq 1, && l \in \mathcal{L}_i, && \text{(39b)}\\
& && 0 \leq x_{it} \leq \tilde{z}_t, && t \in \mathcal{T}. && \text{(39c)}
\end{aligned}
$$

The linear programming dual problem to the problem (P) is to

$$
\begin{aligned}
(D) \qquad & \underset{(v_i, u_i)}{\text{maximize}} && \sum_{l \in \mathcal{L}_i} v_{il} - \sum_{t \in \mathcal{T}} \tilde{z}_t u_{it}, && && \text{(40a)}\\
& \text{subject to} && \sum_{l \in \overline{\mathcal{L}}_{it}} v_{il} - u_{it} \leq c_{it}, && t \in \mathcal{T}, && \text{(40b)}\\
& && v_{il} \geq 0, && \forall i \in \mathcal{N}, \quad l \in \mathcal{L}_i, && \text{(40c)}
\end{aligned}
$$

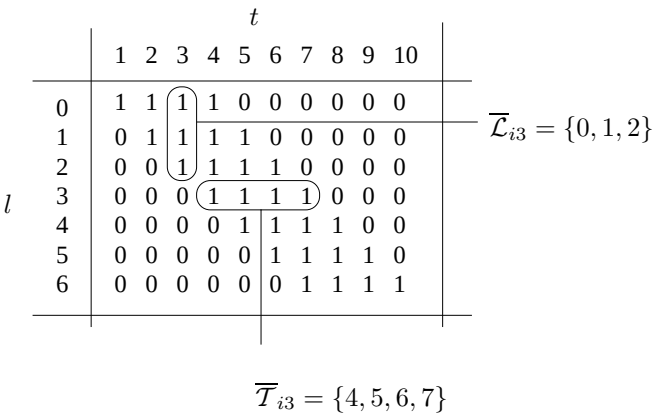

Figure 1: The x_{it} coefficients in the constraints (39b) in problem (P) for component i with life $T_i = 4$ and time horizon $T = 10$.

$$u_{it} \geq 0, \qquad \forall i \in \mathcal{N}, \quad t \in \mathcal{T}. \tag{40d}$$

Using the definitions of $\widetilde{\mathcal{T}}$ and $\overline{\mathcal{T}}_{il}$ we may rewrite the problem (P) as that to

$$\begin{aligned}
(P') \qquad \underset{x_i}{\text{minimize}} \quad & \sum_{t \in \widetilde{\mathcal{T}}} c_{it} x_{it}, && && (41a)\\
\text{subject to} \quad & \sum_{t \in \widetilde{\mathcal{T}} \cap \overline{\mathcal{T}}_{il}} x_{it} \geq 1, && l \in \mathcal{L}_i, && (41b)\\
& x_{it} \geq 0, && t \in \widetilde{\mathcal{T}}. && (41c)
\end{aligned}$$

The problems (P) and (P') are equivalent, since in (P) must hold that $x_{it} = 0$ for $t \in \mathcal{T} \setminus \widetilde{\mathcal{T}}$. The linear programming dual of the problem (P') is given by

$$\begin{aligned}
(D') \qquad \underset{v_i}{\text{maximize}} \quad & \sum_{l \in \mathcal{L}_i} v_{il}, && && (42a)\\
\text{subject to} \quad & \sum_{l \in \overline{\mathcal{L}}_{it}} v_{il} \leq c_{it}, && t \in \widetilde{\mathcal{T}}, && (42b)\\
& v_{il} \geq 0, && l \in \mathcal{L}_i. && (42c)
\end{aligned}$$

Let v_i' be feasible in (D'), Let $v_i = v_i'$ and

$$u_{it} = \max\left\{0, \sum_{l \in \overline{\mathcal{L}}_{it}} v_{il} - c_{it}\right\}, \quad t \in \mathcal{T}. \tag{43}$$

Proposition 4.1. *If $(v_i'^*)$ is optimal in (D') then $(\mathring{v}_i, \mathring{u}_i)$ is optimal in (D) where $\mathring{v}_i = v_i'^*$ and $\mathring{u}_{it}$ are given by (43).*

Proof. Let x_i, x_i', (v_i, u_i) and v_i' be any feasible solution to problems (P), (P'), (D) and (D') respectively. Define:

$$f^P(x_i) \quad = \quad \sum_{t \in \mathcal{T}} c_{it} x_{it},$$

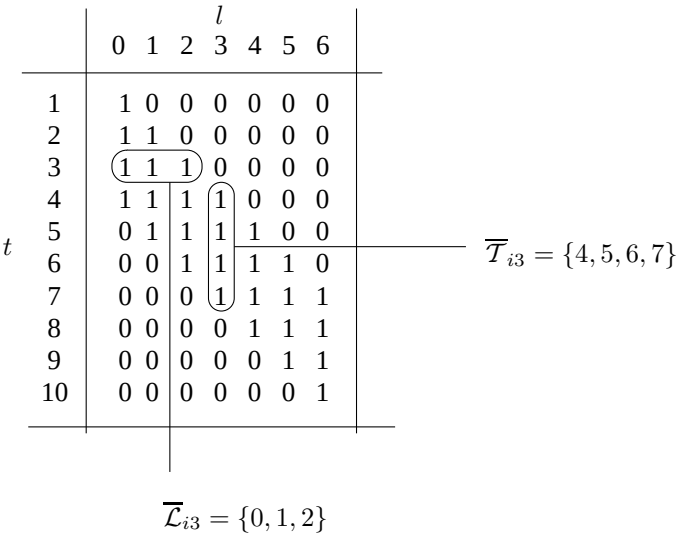

Figure 2: The v_{il} coefficients in the constraints (42b) in problem (D') for component i with life $T_i = 4$ and time horizon $T = 10$.

$$\begin{aligned} f^{P'}(x_i') &= \sum_{t \in \widetilde{\mathcal{T}}} c_{it} x_{it}', \\ f^{D}(v_i, u_i) &= \sum_{l \in \mathcal{L}_i} v_{il} - \sum_{t \in \mathcal{T}} \tilde{z}_t u_{it}, \\ f^{D'}(v_i') &= \sum_{l \in \mathcal{L}_i} v_{il}'. \end{aligned}$$

If v_i' is feasible in (D') then (v_i, u_i) given by (43) is feasible to (D), because (43) implies that:

$$\sum_{l \in \overline{\mathcal{L}}_{it}} v_{il} - c_{it} \leq u_{it}, \quad t \in \mathcal{T}, \tag{45}$$

and $u_{it} \geq 0$. Note that from (43) and (42b)

$$u_{it} = 0, \quad t \in \widetilde{\mathcal{T}}. \tag{46}$$

Assume that x_i^* and $x_i'^*$ are the optimal solutions to the problem (P) and (P'), respectively. The equivalence of (P) and (P') yields that $f^P(x_i^*) = f^{P'}(x_i'^*)$. By the strong duality theorem (Theorem 6.1 Chapter 6 [3]) the following statement's are true:

$$f^P(x_i^*) = f^D(v_i^*, u_i^*), \tag{47a}$$
$$f^{P'}(x_i'^*) = f^{D'}(v_i'^*), \tag{47b}$$

where $(v_i'^*)$ and (v_i^*, u_i^*) are optimal solutions to problems (D') and (D) respectively. The equations (47a) and (47b) then yield that

$$f^D(v_i^*, u_i^*) = f^{D'}(v_i'^*). \tag{48}$$

Let $(v_i'^*)$ be the optimal solution obtained by solving (D') and let $(\mathring{v}_i, \mathring{u}_{it})$ be defined by (43). It then holds that:

$$f^D(\mathring{v}_i, \mathring{u}_i) = \sum_{l \in \mathcal{L}_i} \mathring{v}_{il} - \sum_{t \in \mathcal{T}} \tilde{z}_t \mathring{u}_{it} = \sum_{l \in \mathcal{L}_i} \mathring{v}_{il} - \sum_{t \in \widetilde{\mathcal{T}}} \mathring{u}_{it},$$

and by (46),

$$f^D(\mathring{v}_i, \mathring{u}_i) = \sum_{l=0}^{T-T_i} \mathring{v}_{il} = \sum_{l=0}^{T-T_i} v'^*_{il} = f^{D'}(v'^*_i).$$

Hence $(\mathring{v}_i, \mathring{u}_i)$ is an optimal solution to D. □

This proposition shows that we can solve the problem (D') instead of (D). The problem (D') has fewer variables than (D). Now we claim that for cases for which the costs are non-increasing in time (i.e., $c_{i,t+1} \leq c_{it}$ and $d_{t+1} \leq d_t$ for all $i \in \mathcal{N}$ and $t \in \mathcal{T}$) (D') can be solved by a greedy rule.

Algorithm 2 solves the problem (D'). For each fixed component i, it starts from the last constraint row in (D') and the last indexed dual variable $(v_{i,T-T_i})$. It assigns to each dual variable v_{il}, the most positive (largest) feasible value such that the solution remains feasible. Each constraint in (D') corresponds to a t in $\widetilde{\mathcal{T}}$ and the algorithm terminates when all indices's $t \in \widetilde{\mathcal{T}}$ have been investigated.

Algorithm 2 (Non-increasing cost greedy rule for problem (D'), $\forall i \in \mathcal{N}$)

$A \leftarrow \emptyset$
while
$\widetilde{\mathcal{T}} \neq \emptyset$ **do**
  $\tilde{t} \leftarrow \max\{t \mid t \in \widetilde{\mathcal{T}}\}$
  $B \leftarrow \overline{\mathcal{L}}_{i\tilde{t}}$
  if $B \setminus A \neq \emptyset$ **then**
    $\tilde{l} \leftarrow \max\{l \mid l \in B \setminus A\}$
    $\mathring{v}_{i\tilde{l}} \leftarrow c_{i\tilde{t}} - \sum_{l \in A \cap B} \mathring{v}_{il}$
    $\mathring{v}_{il} \leftarrow 0, \quad l \in B \setminus \{A \cup \{\tilde{l}\}\}$
  end if
  $\widetilde{\mathcal{T}} \leftarrow \widetilde{\mathcal{T}} \setminus \{\tilde{t}\}$
  $A \leftarrow B$
end while

The next proposition shows that for $c_{i,t+1} \leq c_{it}$, $i \in N$, and $z_t \in \{0,1\}$, $t \in \mathcal{T}$, Algorithm 2 yields an optimal solution to (D').

Proposition 4.2 (non-increasing cost greedy algorithm for problem (D') yields optimum). *Assume that $c_{it+1} \leq c_{it}$, for all $i \in \mathcal{N}$ and for all $t \in \mathcal{T} \setminus \{\mathrm{T}\}$. Assume that $\tilde{z}_t \in \{0,1\}$, $t \in \mathcal{T}$, and define the set $\widetilde{\mathcal{T}} = \{t \in \mathcal{T} \mid \tilde{z}_t = 1\}$ such that for each $t \in \widetilde{\mathcal{T}} \cup \{0\}$ and fixed $i \in N$ there is an $s \in \widetilde{\mathcal{T}} \cup \{T+1\}$ with $1 \leq s - t \leq T_i$. Then Algorithm 2 produces an optimal solution to (D').*

Proof. Let i be fixed. Consider the primal problem (P'), the condition on $\widetilde{\mathcal{T}}$ implies that (P) has a feasible solution. According to Section 3.2, the solution of (P') is obtained by replacing component i as late as possible within its life and among the time points $t \in \widetilde{\mathcal{T}}$. Assume that x_i^* is an optimal solution to problem (P') given by Algorithm 1. Let $\widehat{\mathcal{T}}_i$ be the set of time points at which maintenance for component i is performed, that is,

$$\widehat{\mathcal{T}}_i = \{t_k \in \widetilde{\mathcal{T}} \mid x^*_{it_k} = 1\} \quad \text{for each component} \quad i \in \mathcal{N}. \tag{49}$$

Let $\widehat{\mathcal{T}}_i$ be an ordered set, i.e., $t_1 \leq t_2 \leq \ldots \leq t_k$. Define:

$$\underline{l_k} = \min\{l \mid l \in \overline{\mathcal{L}}_{it_k}\},$$

$$\overline{l_k} = \max\{l \mid l \in \overline{\mathcal{L}}_{it_k}\}.$$

Let $\mathring{v}_i$ be the solution to problem D' obtained by Algorithm 2. If we can prove that $\sum_{l\in\mathcal{L}_i} \mathring{v}_{il} = \sum_{t\in\mathcal{T}} c_{it}x^*_{it}$ then weak duality implies that $\mathring{v}_i$ is an optimal solution to (D'). We build our proof by iteration over the set $\widehat{\mathcal{T}}_i$

Initial Step Take $t_1 \in \widehat{\mathcal{T}}_i$. If $0 \notin \overline{\mathcal{L}}_{it_1}$, then $t_1 \notin \overline{\mathcal{T}}_{i0} = \{1, 2, \ldots, T_i\}$, therefore $t_1 > T_i$; which violates the primal feasibility. Hence $0 \in \overline{\mathcal{L}}_{it_1}$. On the other hand if $0 \in \overline{\mathcal{L}}_{it}$ for $t > t_1$ and $t \in \mathcal{T}$, then $t \in \overline{\mathcal{T}}_{i0}$. This contradicts the primal greedy rule; Therefore $0 \notin \overline{\mathcal{L}}_{it}$ for $t > t_1$ and $t \in \mathcal{T}$. t_1 corresponds to the time where the first maintenance occurs as late as possible in $\widetilde{\mathcal{T}}$. Since $0 \in \overline{\mathcal{L}}_{it_1}$ and $0 \notin \overline{\mathcal{L}}_{it}$ for $t > t_1$ dual greedy (Algorithm (2)) yields

$$\sum_{l=0}^{\overline{l_1}} \mathring{v}_{il} = c_{it_1}, \qquad \text{by greedy.}$$

Since $x^*_{it_1} = 1$ and $x^*_{it} = 0$ for $t < t_1$ it holds that

$$c_{it_1} = c_{it_1}x^*_{it_1} = \sum_{t=1}^{t_1} c_{it}x^*_{it}.$$

Thus for t_1 : $\sum_{l=0}^{\overline{l_1}} v^*_{il} = \sum_{t=1}^{t_1} c_{it}x^*_{it}$.

Iterative Step Assume that the following holds for t_{k-1}

$$\sum_{l=0}^{\overline{l_{k-1}}} \mathring{v}_{il} = \sum_{t=1}^{t_{k-1}} c_{it}x^*_{it}.$$

It is obvious that $\overline{l_{k-1}} + 1 \notin \overline{\mathcal{L}}_{it_{k-1}}$. If $\overline{l_{k-1}} + 1 \notin \overline{\mathcal{L}}_{it_k}$, then $t_k \notin \overline{\mathcal{T}}_{i,\overline{l_{k-1}}+1} = \{\overline{l_{k-1}} + 1 + 1, \ldots, \overline{l_{k-1}} + 1 + T_i\}$. Hence $t_k > \overline{l_{k-1}} + 1 + T_i$. It contradicts the feasibility of the primal problem. Also if for $t \in \widetilde{\mathcal{T}}$ and $t > t_k$ then $\overline{l_{k-1}} + 1 \in \overline{\mathcal{L}}_{it}$. This implies $t \in \overline{\mathcal{T}}_{i,\overline{l_{k-1}}+1}$, which contradicts the primal greedy rule. Since $\overline{l_{k-1}} + 1 \in \overline{\mathcal{L}}_{it_k}$ and $\overline{l_{k-1}} + 1 \notin \overline{\mathcal{L}}_{it}$ for $t \in \widetilde{\mathcal{T}}$ and $t > t_k$, dual greedy yields: $\sum_{l=\underline{l_k}}^{\overline{l_k}} \mathring{v}_{il} = c_{it_k}$ and $\mathring{v}_{il} = 0$ for $l \in \overline{\mathcal{L}}_{it_k}$ and $l < \overline{l_{k-1}} + 1$, this implies that:

$$\sum_{\overline{l_{k-1}}+1}^{\overline{l_k}} \mathring{v}_{il} = c_{it_k}.$$

Thus:

$$\sum_{\overline{l_{k-1}}+1}^{\overline{l_k}} \mathring{v}_{il} + \sum_{l=0}^{\overline{l_{k-1}}} \mathring{v}_{il} = \sum_{t=1}^{t_k-1} c_{it}x^*_{it} + c_{it_k}.$$

Since $x^*_{it_k} = 1$ and $x^*_{it} = 0$ for $t_{k-1} < t < t_k$ we have:

$$\sum_{l=0}^{\overline{l_k}} \mathring{v}_{il} = \sum_{t=1}^{t_k} c_{it} x^*_{it}.$$

Note that $\exists k$ such that $\overline{l_k} = T - T_i$, otherwise if for $t_k \in \widehat{\mathcal{T}}_i$, $T - T_i \notin \overline{\mathcal{L}}_{it_k}$ then $t_k \notin \overline{\mathcal{T}}_{i,T-T_i} = \{T - T_i + 1, \ldots, T\}$. Therefore $t_k < T - T_i + 1$, which leads to an infeasible primal problem. As $\widehat{\mathcal{T}}_i$ is a finite set, and $x_{it} = 0$ for $t > t_k$ we will finally obtain: $\sum_{l \in \mathcal{L}_i} \mathring{v}_{il} = \sum_{t \in \mathcal{T}} c_{it} x^*_{it}$ □

Note that this proposition confirms that if component $i \in \mathcal{N}$ is replaced as late as possible within its life and among the times in $\widetilde{\mathcal{T}}$ then the solution is optimal.

One would ask about the fact that by knowing that problem (P) with non-increasing costs can be solved using a greedy rule, whether is it possible to solve the primal and find the optimal dual variables by complementary slackness theorem (Theorem 6.2 [3]). The solution to this question is that complementary slackness theorem will not give us more information than we already have. This become clear if we take a better look at the dual problem (D). Let us assume that x^*_{it} is optimal in (P). Consider the sets $\widetilde{\mathcal{T}}$ and $\widehat{\mathcal{T}}_i$ in (49). Note that $\widehat{\mathcal{T}}_i \subseteq \widetilde{\mathcal{T}}$ for each $i \in \mathcal{N}$. To obtain the corresponding dual solution (v^*_i, u^*_i) to x^*_i, complementary slackness applied to the problems (P) and (D) implies

$$\sum_{l \in \overline{\mathcal{L}}_{it_k}} v^*_{il} = u^*_{it_k} + c_{it_k}, \quad t_k \in \widehat{\mathcal{T}}_i.$$

The remaining constraints of the dual problem (D) are:

$$u_{it} \geq 0, \qquad t \in \mathcal{T}, \tag{50a}$$
$$u_{it} \geq \sum_{l \in \overline{\mathcal{L}}_{it}} v_{il} - c_{it}, \quad t \in \mathcal{T} \setminus \widehat{\mathcal{T}}_i. \tag{50b}$$

Equality of the optimal objective values of problems (D) and (D') known from (48) implies that $u^*_{it} = 0$ for all $t \in \widetilde{\mathcal{T}}$. To maximize the objective value in (D), u^*_{it} should be chosen as small as possible subject to (50a)–(50b), i.e., should be set equal to the maximum of the right-hand-side values in (50a) or (50b) . This yields the optimal dual solution as follows:

$$\sum_{l \in \overline{\mathcal{L}}_{it}} v^*_{il} = c_{it_k}, \qquad t_k \in \widehat{\mathcal{T}}_i, \tag{51a}$$
$$u^*_{it} = 0, \qquad \forall t \in \widetilde{\mathcal{T}},$$
$$u^*_{it} = \max\Big\{0, \sum_{l \in \overline{\mathcal{L}}_{it}} v^*_{il} - c_{it}\Big\}, \quad \forall t \in \mathcal{T} \setminus \widetilde{\mathcal{T}}, \tag{51b}$$

where v^*_{il}'s are yet to be found by solving the problem (D'). In general the proof of Proposition 4.2 can be seen as checking complementary slackness theorem for x^*_i and $\mathring{v}_i$.

4.2 An implementation of Benders decomposition method applied to the opportunistic replacement problem

The opportunistic replacement program in (33) with continuous variables x_{it} is to

$$\begin{aligned}
\underset{(x,z)}{\text{minimize}} \quad & \sum_{t\in\mathcal{T}}\Big(\sum_{i\in\mathcal{N}} c_{it}x_{it} + d_t z_t\Big) && && \text{(52a)}\\
\text{subject to} \quad & \sum_{t\in\overline{\mathcal{T}}_{il}} x_{it} \geq 1\ , && l\in\mathcal{L}_i,\ i\in\mathcal{N}, && \text{(52b)}\\
& 0\leq x_{it}\leq z_t, && i\in\mathcal{N},\ t\in\mathcal{T}, && \text{(52c)}\\
& z_t = \{0,1\}, && t\in\mathcal{T}, && \text{(52d)}
\end{aligned}$$

where $\overline{\mathcal{T}}_{il} = \{l+1,\ldots,l+T_i\}$ for each $i\in\mathcal{N}$ and $l\in\mathcal{L}_i$. We can rewrite the problem (52) as to

$$\begin{aligned}
&\text{minimize } c'x + d'z, && \text{(53a)}\\
&\text{s.t.} \quad Ax + Bz \geq b, && \text{(53b)}\\
&\qquad x\geq 0, \qquad z\in\{0,1\}^{\mathrm{T}}, && \text{(53c)}
\end{aligned}$$

where A and B are $m\times n$ and $m\times p$ matrices with $m = N(2T+1) - \sum_{i=1}^{N} T_i$, $n = NT$ and $p = T$, respectively, x and c are n-vectors, d and z are p-vectors and b is an m-vector. This representation helps us to compare problems (52) and (14).

Letting $z = (z_1, z_2, \ldots, z_T)$ be fixed to $\tilde{z}$, makes (52) a linear programming problem as to

$$\begin{aligned}
\underset{(x,z)}{\text{minimize}} \quad & \sum_{t\in\mathcal{T}}\Big(\sum_{i\in\mathcal{N}} c_{it}x_{it} + d_t \tilde{z}_t\Big) && && \text{(54a)}\\
\text{subject to} \quad & \sum_{t\in\overline{\mathcal{T}}_{il}} x_{it} \geq 1\ , && l\in\mathcal{L}_i,\ i\in\mathcal{N}, && \text{(54b)}\\
& 0\leq x_{it}\leq \tilde{z}_t, && i\in\mathcal{N},\ t\in\mathcal{T}. && \text{(54c)}
\end{aligned}$$

The linear dual programming of the problem (54) is to

$$\begin{aligned}
\underset{(v,u)}{\text{maximize}} \quad & \sum_{i\in\mathcal{N}}\Big(\sum_{l\in\mathcal{L}_i} v_{il} - \sum_{t\in\mathcal{T}} \tilde{z}_t u_{it}\Big) && && \text{(55a)}\\
\text{subject to} \quad & \sum_{l\in\overline{\mathcal{L}}_{it}} v_{il} - u_{it} \leq c_{it}, && t\in\mathcal{T}, \quad i\in\mathcal{N}, && \text{(55b)}\\
& v_{il}\geq 0, && i\in\mathcal{N}, \quad l\in\mathcal{L}_i, && \text{(55c)}\\
& u_{it}\geq 0, && i\in\mathcal{N}, \quad t\in\mathcal{T}, && \text{(55d)}
\end{aligned}$$

where, for all $i\in\mathcal{N}$ and $t\in\mathcal{T}$, $\overline{\mathcal{L}}_{it} = \{l\in\mathcal{L}_i \,|\, t\in\overline{\mathcal{T}}_{il}\}$.

Knowing from Section 3.2, when the variables z_t, $t\in\mathcal{T}$, are assigned binary values, the remaining optimization model separates over the components $i\in\mathcal{N}$. The primal of the subproblem, for $z_t = \tilde{z}_t$, $t\in\mathcal{T}$, is then given by

$$\sum_{t\in\mathcal{T}} d_t\tilde{z}_t + \sum_{i\in\mathcal{N}}\left(\begin{array}{lrcll} \min & \sum_{t\in\mathcal{T}} c_{it}x_{it}, & & & \\ \text{s.t.} & \sum_{t\in\overline{\mathcal{T}}_{il}} x_{it} & \geq & 1, & l\in\mathcal{L}_i,\\ & 0\leq x_{it} & \leq & \tilde{z}_t, & t\in\mathcal{T} \end{array}\right). \qquad \text{(56)}$$

We denote the optimal solution to this program by $\tilde{x}_{it}$, $i \in \mathcal{N}$, $t \in \mathcal{T}$.

If $z_t = 1$, maintenance occurs at time t and the cost d_t is incurred. If $z_t = 0$, then $x_{it} = 0$ for all $i \in \mathcal{N}$ and the cost of maintenance is zero. Hence, since $\tilde{z}_t = 0$, $t \in \mathcal{T} \setminus \widetilde{\mathcal{T}}$, it follows that $\tilde{x}_{it} = 0$, $i \in \mathcal{N}$, $t \in \mathcal{T} \setminus \widetilde{\mathcal{T}}$, which yields the simplified subproblem formulation:

$$\sum_{t \in \widetilde{\mathcal{T}}} d_t + \sum_{i \in \mathcal{N}} \left(\begin{array}{lrcll} \min & \sum_{t \in \widetilde{\mathcal{T}}} c_{it} x_{it}, & & & \\ \text{s.t.} & \sum_{t \in \overline{\mathcal{T}}_{il} \cap \widetilde{\mathcal{T}}} x_{it} & \geq & 1, & l \in \mathcal{L}_i, \\ & 0 \leq x_{it} & \leq & 1, & t \in \widetilde{\mathcal{T}} \end{array} \right). \tag{57}$$

Assuming that $c_{it} \geq 0$, $i \in \mathcal{N}$, $t \in \mathcal{T}$, the constraints "$x_{it} \leq 1$" are unnecessary (redundant), according to the following argument:

If $\tilde{x}_{it} > 1$ for some $i \in \mathcal{N}$ and $t \in \widetilde{\mathcal{T}}$, the optimal value of the subproblem is always reducing (or constant, if $c_{it} = 0$) with the value of x_{it}. Since the constraints "$x_{it} \geq 0$" must hold for $i \in \mathcal{N}$ and $t \in \mathcal{T}$, the constraints "$\sum_{t \in \overline{\mathcal{T}}_{il}} x_{it} \geq 1$" will not be violated (until $x_{it} < 1$).

This leads to the following further simplification of the subproblem formulation as

$$\sum_{t \in \widetilde{\mathcal{T}}} d_t + \sum_{i \in \mathcal{N}} \left(\begin{array}{lrcll} \min & \sum_{t \in \widetilde{\mathcal{T}}} c_{it} x_{it}, & & & \\ \text{s.t.} & \sum_{t \in \overline{\mathcal{T}}_{il} \cap \widetilde{\mathcal{T}}} x_{it} & \geq & 1, & l \in \mathcal{L}_i, \\ & x_{it} & \geq & 0, & t \in \widetilde{\mathcal{T}} \end{array} \right) \tag{58}$$

with the corresponding linear programming dual

$$\sum_{t \in \widetilde{\mathcal{T}}} d_t + \sum_{i \in \mathcal{N}} \left(\begin{array}{lrcll} \max & \sum_{l \in \mathcal{L}_i} v_{il}, & & & \\ \text{s.t.} & \sum_{l \in \overline{\mathcal{L}}_{it}} v_{il} & \leq & c_{it}, & t \in \widetilde{\mathcal{T}}, \\ & v_{il} & \geq & 0, & l \in \mathcal{L}_i \end{array} \right). \tag{59}$$

Note that the subproblems in (58) and (59) are the problems (P') and (D'), from Section 4.1, respectively. The optimal dual solution $(\tilde{v}_{il}, \tilde{u}_{it})$ for $\tilde{z}_t = \{0, 1\}$ can be obtained by solving problem (55) directly, or by solving the subproblems in (59) with Algorithm 2 where $\tilde{u}_{it}$ is given by (43).

The polyhedron P is the set of all (v, u) satisfying (55b)–(55d). The complete master problem can be expressed as that to

$$\begin{array}{lll} \text{minimize } y & & \text{(60a)} \\ \text{s.t.} \quad y \geq \sum_{i \in \mathcal{N}} \sum_{l \in \mathcal{L}_i} v_{il}^{p^k} - \sum_{i \in \mathcal{N}} \sum_{t \in \mathcal{T}} u_{it}^{p^k} z_t + \sum_{t \in \mathcal{T}} d_t z_t, & k \in \{1, \ldots, K\}, & \text{(60b)} \\ \quad\quad 0 \geq \sum_{i \in \mathcal{N}} \sum_{l \in \mathcal{L}_i} v_{il}^{r^m} - \sum_{i \in \mathcal{N}} \sum_{t \in \mathcal{T}} u_{it}^{r^m} z_t, & m \in \{1, \ldots, M\}, & \text{(60c)} \\ \quad\quad z_t \in \{0, 1\}, & t \in \mathcal{T}, & \text{(60d)} \\ \quad\quad y \in R, & & \text{(60e)} \end{array}$$

where $(v_i^{p^k}, u_i^{p^k})$ denotes the extreme points of the polyhedron P and $(v_i^{r^m}, u_i^{r^m})$ denotes the extreme rays of the polyhedron P (see Section 2.5). K is the number of extreme points of the polyhedron defined by (55b)–(55d). This polyhedron is actually composed by $|\mathcal{N}|$ polyhedra, one for each $i \in \mathcal{N}$, and k and m, respectively, denote one Benders iteration.

The inequalities (60b)–(60c) are necessary and sufficient for the values z to be feasible, i.e., to admit feasible values of x_{it} in (52b)–(52c). In [10] there is a suggestion that by adding an artificial constraint, bounding the sum of all variables by a large positive number, the polyhedron can be made bounded, so the constraints corresponding to the extreme rays can be dropped. However there is an smarter way to avoid inequalities for extreme rays in our problem. The program (58) is feasible if and only if

$$\left\{ \bigcup_{i \in \mathcal{N}} \left\{ \bigcup_{l \in \mathcal{L}_i} \overline{\mathcal{T}}_{il} \right\} \right\} \bigcap \widetilde{\mathcal{T}} \neq \emptyset \quad \Longleftrightarrow \quad \overline{\mathcal{T}}_{il} \cap \widetilde{\mathcal{T}} \neq \emptyset,\ l \in \mathcal{L}_i,\ i \in \mathcal{N}, \tag{61}$$

which is in turn equivalent to the constraints

$$\sum_{t \in \overline{\mathcal{T}}_{il}} \tilde{z}_t \geq 1, \quad l \in \mathcal{L}_i, \quad i \in \mathcal{N} \tag{62}$$

to hold. The constraints (62) can be equivalently expressed as

$$\sum_{t=l+1}^{l+T_i} \tilde{z}_t \geq 1, \quad l \in \{0, \ldots, T - T_i\}, \quad i \in \mathcal{N}. \tag{63}$$

Defining $\overline{T} = \min_{i \in \mathcal{N}} \{T_i\}$ the (necessary and) sufficient feasibility cuts for the master problem (corresponding to the extreme rays of the feasible set of the dual subproblem (59)) are then given by

$$\sum_{t=l+1}^{l+\overline{T}} \tilde{z}_t \geq 1, \quad l \in \{0, \ldots, T - \overline{T}\}. \tag{64}$$

Hence, (60c) can be dropped as long as the condition (64) is enforced. Note that including all the constraints in (64) in (60c) ensures the problem (60) to have a bounded feasible set. The feasibility assumption on the opportunistic replacement problem (33) yields that the problem (60) is also feasible.

The Benders partitioning algorithm can be initiated with no constraints of the form (60b) and the inequalities of the form (64) in the problem (60). i.e., in each Benders step the problem to

$$\begin{aligned}
&\text{minimize } y && && (65a)\\
&\text{s.t.} \quad y \geq \sum_{i \in \mathcal{N}} \sum_{l \in \mathcal{L}_i} v_{il}^{p^k} - \sum_{i \in \mathcal{N}} \sum_{t \in \mathcal{T}} u_{it}^{p^k} z_t + \sum_{t \in \mathcal{T}} d_t z_t, && k \in \{1, \ldots, K\}, && (65b)\\
&\qquad \sum_{t=l+1}^{l+\overline{T}} \tilde{z}_t \geq 1, && l \in \mathcal{L}_i, && (65c)\\
&\qquad z_t \in \{0, 1\}, && t \in \mathcal{T}, && (65d)
\end{aligned}$$

$$y \in R, \tag{65e}$$

is solved, where, the number of constraints in (65b) is equal to the Benders iteration number.

Initiate the problem (65) with no constraints of the form (65b). Let the solution to the problem (65), be $(\tilde{y}, \tilde{z})$. Now with z being fixed at $\tilde{z}$, the dual problem (55) should be solved to obtain $(\tilde{v}_{il}, \tilde{u}_{it})$. Since the primal problem is bounded, the problem (55) is feasible. Let $\tilde{w}$ be the optimal objective value of the problem (55) and $\tilde{y}$ the optimal value obtained by solving the problem (65). If $\tilde{w} = \tilde{y} - \sum_{t\in\mathcal{T}} d_t \tilde{z}_t$, then by the optimality test, the current solution is optimal. Otherwise, we form a new constraint from $(\tilde{v}_{il}, \tilde{u}_{it})$ of the type (65b) and add it to the problem (65).

The summary of the Benders algorithm based on an iterative procedure is given in Algorithm 3.

Algorithm 3 (Benders Algorithm)

Step 0 **(Initialization)**: Set $\tilde{y} = -\infty$ and $r = 0$. Initiate the problem (65) with the constraints (65c) and (65d).

Step 1: Let $r = r + 1$. Solve the problem (65) to obtain a finite optimal solution $(\tilde{z}^r, \tilde{y}^r)$.

Step 2: Solve the dual linear program (55) with $\widetilde{\mathcal{T}} = \{t \in \mathcal{T} \mid \tilde{z}_t^r = 1\}$ to find $(\tilde{v}_{il}^r, \tilde{u}_{it}^r)$.

Step 3: If the optimal objective value in step 2 is equal to $\tilde{y}^r - \sum_{t\in\mathcal{T}} d_t \tilde{z}_t^r$, the solution $(\tilde{y}^r, \tilde{z}^r)$ solves (60). If $\tilde{x}^r$ solves the linear primal problem, then $(\tilde{x}^r, \tilde{z}^r)$ solves the corresponding opportunistic replacement problem (33). Stop!

Step 4: If the optimality test in step 3 is not passed, then

$$\tilde{y}^r < \sum_{i\in\mathcal{N}} \sum_{l\in\mathcal{L}_i} \tilde{v}_{il}^r - \sum_{i\in\mathcal{N}} \sum_{t\in T} \tilde{u}_{it}^r \tilde{z}_t^r + \sum_{t\in\mathcal{T}} d_t \tilde{z}_t^r, \tag{66}$$

holds, so the current solution to (65) does not satisfy the constraint

$$y \geq \sum_{i\in\mathcal{N}} \sum_{l\in\mathcal{L}_i} \tilde{v}_{il}^r - \sum_{i\in\mathcal{N}} \sum_{t\in\mathcal{T}} \tilde{u}_{it}^r z_t + \sum_{t\mathcal{T}} d_t z_t. \tag{67}$$

Add the constraint (67) to the problem (65) and return to Step 1.

Upper and lower bounds: As shown before, in the iterative procedure we will solve (65), which is a relaxation of (60) including only a subset of its constraints. Let $\tilde{y}^r$ be the optimal objective value to (65) at step r of the Bender's algorithm, i.e., the number of constraints (65b) is r in the current problem (65). Assume that y^* is the optimal objective value to problem (60), so $\tilde{y}^r$ is a lower bound on the optimal value y^* of the full master problem and it holds that

$$\tilde{y}^r \leq \tilde{y}^{r+1} \leq y^*. \tag{68}$$

The upper bound is generated by a sequence of feasible solutions to the original problem. To obtain the upper bound, let $(\tilde{y}^r, \tilde{z}^r)$ be the solution to (65) at step r. If $\tilde{x}^r$ solves the primal problem (54), then $(\tilde{x}^r, \tilde{z}^r)$ is feasible to (52), therefore it holds

that

$$\sum_{i\in\mathcal{N}}\sum_{t\in\mathcal{T}} c_{it}\tilde{x}^r_{it} + \sum_{t\in\mathcal{T}} d_t\tilde{z}^r_t \geq y^*. \tag{69}$$

At each iteration, lower and upper bounds are computed. From (68) and (69), at step r it holds that

$$\tilde{y}^r \leq y^* \leq \min_{1\leq s\leq r}\big(\sum_{i\in\mathcal{N}}\sum_{t\in\mathcal{T}} c_{it}\tilde{x}^s_{it} + \sum_{t\in\mathcal{T}} d_t\tilde{z}^s_t\big). \tag{70}$$

Hence, we can define the optimality gap at step r.

Definition 4.1. *: The optimality gap at step r is defined as*

$$\frac{\min_{1\leq s\leq r}\sum_{t\in\mathcal{T}}\big(\sum_{i\in\mathcal{N}} c_{it}\tilde{x}^s_{it} + d_t\tilde{z}^s_t\big)}{y^*} - 1.$$

The estimated optimality gap at step r is defined as

$$\frac{\min_{1\leq s\leq r}\sum_{t\in\mathcal{T}}\big(\sum_{i\in\mathcal{N}} c_{it}\tilde{x}^s_{it} + d_t\tilde{z}^s_t\big)}{\tilde{y}^r} - 1.$$

When the upper and lower bounds become equal, the algorithm terminates with an optimal solution. The hope is that termination will occur when r is considerably less than the number of extreme points of the polyhedron defining the problem (55). Benders algorithm may also be terminated when the computation time exceeds a specific time limit.

4.3 Computational experiment and results

In this section we present some numerical tests and their results for an implementation of Benders algorithm applied to some instances of the opportunistic replacement problem. The reference MILP solver used is `IBM ILOG CPLEX` 12.1. Algorithms are written in `MATLAB R2009b`. All numerical experiments are performed on a Linux desktop operating system with the processor Pentium(R), Dual-Core E5200 @ 2.50GHz. The CPU MHz is 1200.000 and it has a cache size of 2048 KB.

The solvers used for the numerical tests, are introduced as follows. `CPLEXMEX` is a `MEX` interface for the `cplex` callable library which enables us to use `cplex` from within `MATLAB`. The `CPLEXMEX` interface gives access to most of the `cplex` interactive mode functionality from within MATLAB. It is intended for solving linear programming (LP), mixed integer linear programming (MILP), and other related problems. `Cplexlp` is a function of `IBM ILOG CPLEX` toolbox in `MATLAB` which solves linear programming problems. Besides, `cplexmilp` is a function of `IBM ILOG CPLEX` toolbox in `MATLAB` which is intended to solve mixed integer linear programming problems.

For the computations, different instances of the opportunistic replacement problem are considered. Data describing these instances –called testbed problems– are shown in Table 1. For all testbed problems it is assumed that $c_{it} = c_i$, $i \in \mathcal{N}$, $t \in \mathcal{T}$, and $d_t = d$, $t \in \mathcal{T}$. Instance 1 in the testbed is a simple small size problem. Instance 2 is a middle sized problem intended to resemble a realistic problem. Instance 3 is a sparse problem in the sense that the components lives are short, it is designed to investigate the effects of a long planning horizon. Instance 4 is a dense problem in

Table 1: The problems in the testbed. In problem 4, the costs, c_i, are randomly chosen from $(0, 1]$.

instance	T	N	$\min T_i$	$\max T_i$	d	$\min c_i$	$\max c_i$
1	10	3	3	5	10	5	7
2	40	3	3	5	10	5	7
3	100	4	3	7	10	5	9
4	60	50	6	55	1	0.0089	1.0000
5 (HPT)	100	9	15	80	1	0.3613	4.0255
6 (LPT)	150	10	29	60	1	0.3171	1.5482
7	500	2	25	40	1	0.1324	0.7451
8	1000	2	25	40	1	0.1324	0.7451

Table 2: Parameter setting when solving testbed problems with `cplex`. Solutions reported in Table (3).

Directive	Value	Description
time limit	86400	`cplex` stops after 24 hours and return the current solution
nodefile	2	creates a compressed version of the node file in memory

a sense that the feasible solutions contain many maintenance occasions. The data in instances 5 and 6 are from real-world problems obtained from Volvo Aero and corresponding to two modules of an aircraft engine. Instance 5 corresponds to the data from a high pressure turbine (HPT) and instance 6 from a low pressure turbine (LPT). Note that the data from real world instances are scaled so that $d = 1$. Instances 7 and 8 are very big sparse problems. All the computation times are given in CPU seconds. The testbed problems solved with `cplex` are reported in Table 3. For solving the testbed problems using `cplex` [1], the parameter settings reported in Table 2 was used.

As the first experiment we solve instances chosen from the testbed, where `cplexlp` is used to solve the LP subproblems. `CPLEXMEX` is chosen as the solver for the MILP masterproblem. Then for the same instances, instead of solving the LP subproblems by `cplexlp`, the greedy rule, developed in Section 4.1, is implemented. The results are presented in Table 4.

Table 4 shows that the time needed to solve a medium size problem (e.g., HPT and LPT instances) is considerably huge, so it seems natural to investigate how much of the gap can be closed in 1 minute, 30 minutes and 24 hours for the different instances of the problem. Hence, the next experiments are to solve testbed problems in a time limit of 1 minute (60 CPU seconds), 30 minutes (1800 CPU seconds) and 24 hours (86,400 CPU seconds). In all these experiments, the estimated optimality gap is defined in Definition 4.1 and each Benders iteration is considered as one iteration and also LP subproblems solved using the greedy algorithm from Section 4.1. At first,

Table 3: Testbed problems solved with `cplex`. A † denotes that the computer memory was filled before the instance of the opportunistic replacement problem was solved to optimality. A * denotes that the problem was not solved to optimality when the time limit is reached.

instance	N	nodes	`cplex` iterations	time(s)	gap%
1	3	0	60	0.0222	0
2	3	210	5,476	0.4183	0
3	4	3,272,759	125,312,690	8941.7	0
4	50	1,943	461,183	125.6857	0
5 (HPT)	9	115	7,631	1.2449	0
6 (LPT)	10	1,132	80,185	13.8381	0
7	2	9,233,344	†122068477	17,505.01	5.64
8	2	3,508,639	*156343889	86,400.02	10.99

Table 4: Solving the testbed problems by Benders algorithm. The estimated optimality gap is defined in Definition 4.1 and each Benders iteration is considered as one iteration. The MILP master problems are solved using `CPLEXMEX` in all the computations reported in this Table. If Benders MILP master problems can not be solved because out of memory status it is marked in the table by a †. A time limit of 3 weeks (21 days) is put on test problem 3. Out of time statues is marked by a $*$. Computation times are given in CPU seconds

instance	subproblems solved using the simplex method			subproblems solved using Algorithm 2		
	time(s)	iter.	est.opt. gap%	time(s)	iter.	est.opt. gap%
1	3.9000	6	0	0.3400	5	0
2	540330	1897	0	12452	433	0
3	*($\geq$ 1814400)	161	33.52	†(1291200)	107	32.77
5 (HPT)	1955600	1430	0	129110	789	0
6 (LPT)	45031	279	0	1437900	580	0

testbed problems are solved for a time limit of 60 seconds by Benders algorithm. The MILP master problems are solved with `CPLEXMEX` and `cplexmilp`. For these solvers, comparing the results in Table 5 shows that `CPLEXMEX` is faster than `cplexmilp`. In Table 5 the initial integral gap for each instance is also reported. Tables 6 and 7 shows the results for the testbed problems in a 30 minutes and a 24 hours CPU-time limit respectively.

Problem 5 (HPT) is then solved by Benders method to observe how the properties of the solution process change when the time horizon varies. A CPU-time limit of 24 hours has been also considered. The results are written in Table 8. In all the problems in Table 8, the LP subproblems are solved by the greedy rule described in Algorithm 2. For having a better picture, the data presented in Table 8 are illustrated in the graphs in Figure 3.

It is known from Section 4.2 that in every Benders iteration a new constraint is added to the MILP master problem, so another question to answer is how this addition of constraints effect the solution speed. For answering this question we have chosen the middle size problem 2 and save the time spent for finding the solution of each MILP master problem. A computational difficulty is that, although `cplexmex` is slightly faster than the `cplex` toolbox function `cplexmilp` in `MATLAB`, but `cplexmex` like `cplexmilp` do not report the exact time spent by `cplex` to `MATLAB` terminal window. Computing time spend for finding each MILP solution in `MATLAB` is not reliable as the time of the data transferring between `cplex` and `MATLAB` will be added to this time. Since parallel solutions to the MILP master problems exist, `cplexmex` and `cplexmilp` might find different ones. Choosing `cplexmilp` instead of `cplexmex` leads to a different approach by the Benders method with 990 iterations and a gap closed of 94.84 in 24 hours; compare to what is reported in Table 7. Still it is a good example to observe how the solution time for each MILP master problem increases. The average time for solving LP subproblems for this particular example is 0.0200 seconds. Figure 4 shows the time reported from `cplex` for solving each MILP master problem.

Conclusions: When costs are non-increasing with time the greedy rule in Algorithm 2 can be used to solve the LP subproblems. The greedy rule solves the LP subproblems in a fraction of a second for middle sized problems while the solution time with the simplex algorithm can be considerably larger. As alternative solutions to LP subproblems exist, the greedy rule and the simplex algorithm may end up with different solutions that effects the behavior of Benders method, this can be seen in Table 4. In conclusion, for most cases in Table 4, solving Benders when the LP subproblems are solved using the greedy rule is faster with less number of iterations.

From comparing Tables 5, 6 and 7 one can conclude that most of the gap is closed in the first seconds (hours) of the computation time. Figure 4 indicates that the time needed by `cplex` to solve the MILP master problems increases linearly with the iteration number. According to Section 4.2, a lower bound of the problem at hand is the optimal objective value of the MILP master problem. Within the first iterations lower bound increases rapidly, making the bounds on the optimal value of the opportunistic replacement problem tighter. However, when the number of constraints of the MILP master problem with the Benders iteration number increases, the optimal value of the MILP master problem is barely different from that of the last

Table 5: Testbed problems are solved for a time limit of 1 minute (60s) by Benders algorithm where MILP master problems are solved with `CPLEXMEX` and `cplexmilp`. The LP subproblems solved using the Algorithm 2. In this table the initial optimality gap for each instance is also reported. The estimated optimality gap is defined in Definition 4.1 and each Benders iteration is considered as one iteration.

instance	init.opt. gap %	MILP master problems solved using `CPLEXMEX`		MILP master problems solved using `cplexmilp`	
		iterations	est.opt. gap%	iterations	est.opt gap%
1	60	5	0	5	0
2	154.65	69	20.11	35	24.29
3	254	15	41.83	12	47.01
4	27.42	19	22.22	14	22.54
5 (HPT)	45.45	35	14.75	33	11.07
6 (LPT)	57.04	29	35.86	24	50.22
7	93.41	15	55.73	11	57.61
8	87.79	4	63.95	4	63.95

Table 6: Testbed problems solved for at most 30 minutes (1800 sec) by Benders algorithm to observe the remaining optimality gap. The MILP master problems are solved with `CPLEXMEX` and the LP subproblems solved with the Algorithm 2. The estimated optimality gap is defined in Definition 4.1 and each Benders iteration is considered as one iteration.

instance	iterations	est.opt. gap%
1	5	0
2	202	17.61
3	32	41.58
4	63	21.25
5 (HPT)	185	14.31
6 (LPT)	89	23
7	45	56.90
8	9	60.97

Table 7: Testbed problems solved for at most 24 hours by Benders algorithm to observe the remaining optimality gap. The MILP master problems are solved with `CPLEXMEX` and the LP subproblems are solved using the Algorithm 2. The estimated optimality gap is defined in Definition 4.1 and each Benders iteration is considered as one iteration. If Benders MILP master problems can not be solved because of out of memory status it is marked in the table by a †.

instance	iterations	optimality gap %
1	5	0
2	433	0
3	61	39.56
4	174	15.05
5 (HPT)	690	4.977
6 (LPT)	274	18.73
7	88	55.67
8	† 10	60.97

Table 8: The opportunistic replacement problem solved by Benders algorithm with data from HPT in a 24 hour limit. The MILP master problems are solved with `CPLEXMEX` and the LP subproblems are solved using the Algorithm 2. The estimated optimality gap is defined in Definition 4.1 and each Benders iteration is considered as one iteration. In this table, $n = (N+1)T$ denotes the number of variables in the corresponding opportunistic replacement problem.

			`cplex`		Benders		
problem	T	n	time(s)	iterations	time(s)	iterations	est.opt gap%
1	50	300	0.02	206	0.8700	12	0
2	60	420	0.07	415	4.5500	41	0
3	68	476	0.07	536	10.7200	42	0
4	75	525	0.14	624	155.6200	117	0
5	85	850	0.21	1347	602.5100	119	0
6	95	950	2.32	21237	7799.4	308	0
7	100	1000	1.19	7631	86400	690	4.97
8	110	1100	9.74	49324	86400	373	3.01
9	120	1200	28.47	172920	86400	386	17.69
10	130	1300	37.30	207838	86400	265	17.32
11	140	1400	86.61	401851	86400	170	19.53
12	150	1500	58.38	259860	86400	170	31.04

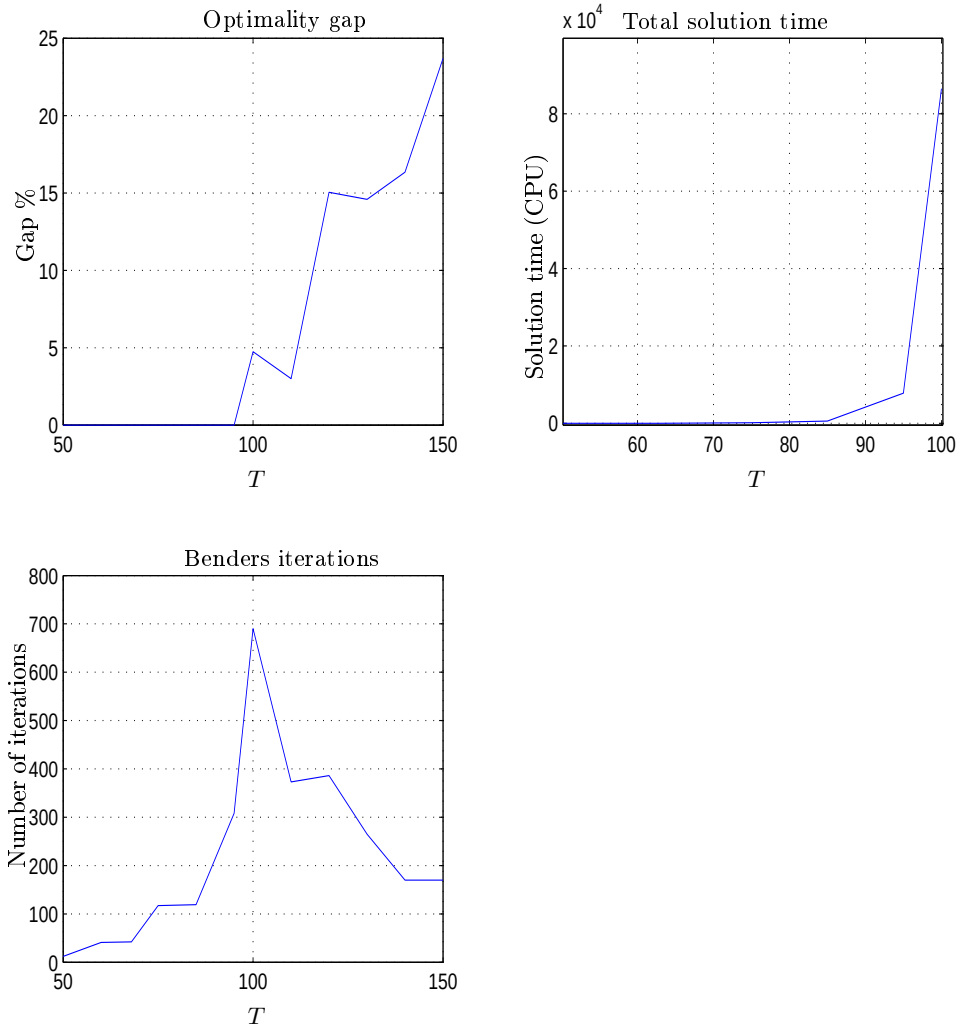

Figure 3: Optimality gap percentage, number of iterations, and solution time versus time horizon for the HPT problem solved by Benders method

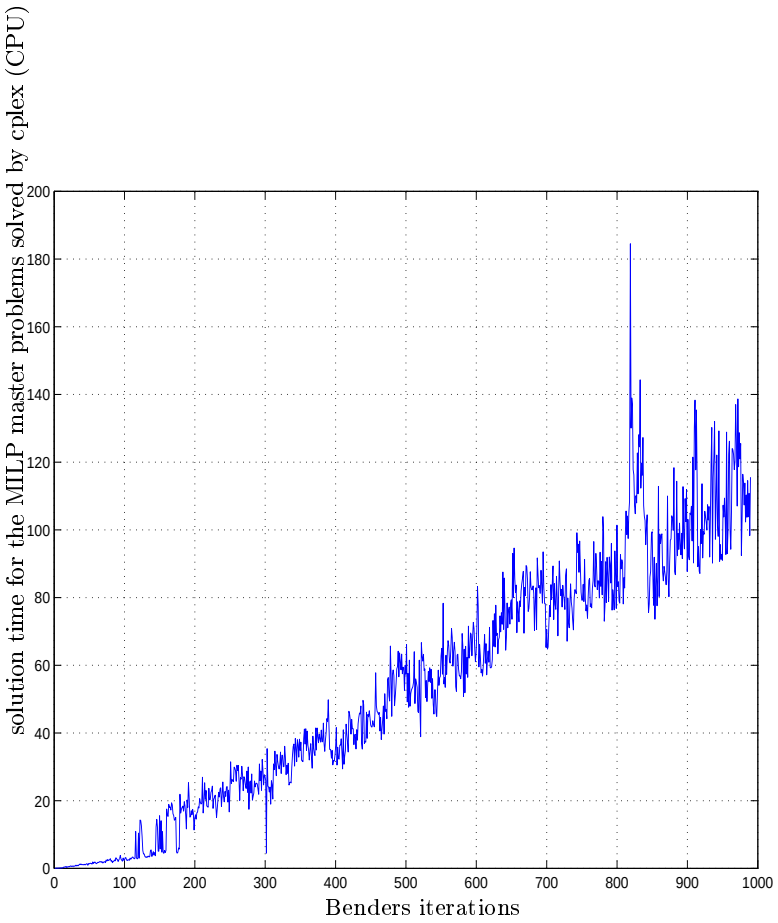

Figure 4: Solution times for the MILP master problems in Benders method for testbed problem 2 in a 24 hour limit.

Benders iteration. This is because a considerable addition of constraints is required to increase the MILP optimal objective value.

The report of the solutions for the HPT problem, when the time horizon (T) changes, in Table 8 and Figure 3 shows that when the problem size increases the time needed for Benders to solve it behave exponentially. The reason is that when the number of variables increases, the size of the LP subproblems and the MILP master problem increases. Moreover, the number of extreme points of the polyhedron $P = \{u \mid A'u \leq c, u \geq 0\}$ -defined by the constraint set of the dual problem- increases when the polyhedron dimension increase. Since Benders method searches among the extreme points of the polyhedron P, an increase in the number of extreme points results in an increase in the number of constraints defining the complete master problem. In Figure 3 it can be seen that the number of iterations increases rapidly, so a considerable gap closes in the first 24 hours. On the other hand, when T is large the number of iterations decreases. This is because, as the size of the MILP master problems increases, solving them requires more computation time.

In general one can say, since in each step a MILP minimization problem should be solved, Bender's decomposition algorithm is very slow for our opportunistic replacement problem. The solution time may increase exponentially when the problem size increases.

5 A rank-1 separation problem applied to the opportunistic replacement problem

Known from Section 2.3.1, given the polyhedron $P = \{x : \ x \in R^n_+, \ Ax \leq b\}$ and the set $S = P \cap Z^n$, facets of $conv(S)$ can be constructed iteratively utilizing integrality and the linear inequality description of P. This means that we start with the valid inequalities $Ax \leq b$ and if they are not enough to define $conv(S)$, i.e., the polyhedron P has non-integral extreme points, we progressively construct stronger valid inequalities as follows. We obtain valid inequalities for P by taking non-negative linear combinations of the inequalities defining P. For tightening the formulation of S, a strategy which can be taken is to examine the initial formulation, find a set of valid inequalities $\pi x \leq \pi_0$ for S, and add these to the original system, which gives a new formulation $P' = \{x \in R^n_+ : Ax \leq b, \ \pi x \leq \pi_0\} \subseteq P$ with $S = P' \cap Z^n$. If the valid inequalities are well chosen so that the set P' is significantly smaller than P, the bounds (see Section 2.1.3) should be improved and hence the integer programming techniques should be more effective. On the other hand, often the number of valid inequalities one would like to add is enormous. Increasing the number of constraints makes the time required to solve the linear programs increase. One should also note that finding the complete description of the convex hull is not of interest, but a good approximation of $conv(S)$ in the neighborhood of an optimal solution is desired.

In this chapter we address these general issues for the opportunistic replacement problem (33) by studying its $\mathcal{NP}$-hard separation problem. By modeling and solving the rank-1 Chvátal–Gomory separation problem we study how effective it is to optimize over the first Chvátal closure of the opportunistic replacement problem, i.e., which fraction of the integrability gap that can be closed by a branch and bound approach based on finding the most violated cuts for the replacement polytope. We also try to answer the question whether it is a benefit to generate rank-1 CG cuts until no more such inequality exists or one should better follow the strategy of generating Chvátal-Gomory inequalities of any rank. Since the opportunistic replacement problem is an MILP problem, here we introduce a projected version of the CG cuts and its associated separation problem and study its practical strength for the opportunistic replacement problem. Finally, we want to investigate how the addition of the generated CG cuts to the original formulation affect the general ILP branch and bound procedure to solve the opportunistic replacement problem.

5.1 The separation problem for the opportunistic replacement problem

In this section first a modified MILP model for the rank-1 Chvátal-Gomory separation problem is described, which can be solved by a general-purpose MILP solver. Then the projected Chvátal-Gomory separation problem is presented. This model is implemented in a pure cutting plane framework to generate several rank-1 CG-cuts in order to obtain a tighter bound on the value of the optimal solution of the opportunistic replacement problem. In the next section the computational assumptions needed are described. In section 5.2 the results of the numerical tests on some instances of the opportunistic replacement problem is reported.

5.1.1 The MILP model for solving the Chvátal-Gomory separation problem

Let $y = (x, z)$ to be the set of variables in the opportunistic replacement problem (33) and define A and b the matrix of the coefficients and the right-hand side vector in the constraints (33b)–(33c) respectively. Let $h' = (c', d')$. Then the opportunistic replacement problem can be simply written in the general form of a BLP, as

$$\min\{h'y : Ay \leq b,\ y \geq 0,\ y \in \{0,1\}^{(N+1)T}\}. \tag{71}$$

First let us describe our MILP model for CG separation of the integer linear programming problem (71). Consider the point $y^* \geq 0$ given, CG-SEP (Definition 2.16) finds a CG cut $\alpha' y \leq \alpha_0$ which is maximally violated by y^*, where $\alpha' = \lfloor u'A \rfloor$ and $\alpha_0 = \lfloor u'b \rfloor$ for a certain $u \in R^m_+$. The first CG-SEP model then is to

$$\begin{aligned}
\underset{(\alpha,\alpha_0,u)}{\text{maximize}}\ & \alpha' y^* - \alpha_0, && && \text{(72a)}\\
\text{s.t.}\ & \alpha_j \leq u' A_j, && j = 1,\dots,n, && \text{(72b)}\\
& \alpha_0 - \epsilon \geq u'b - 1, && && \text{(72c)}\\
& u_i \geq 0, && i = 1,\dots,m, && \text{(72d)}\\
& \alpha \in Z^n_+,\ \alpha_0 \in Z_+, && && \text{(72e)}
\end{aligned}$$

where $\alpha = (\alpha_1, \dots, \alpha_n)$, $u = (u_1, \dots, u_m)$, $n = (N+1)T$, $m = 2NT + N - \sum_{i=1}^{N} T_i$, and $\epsilon \in (0, 1)$. Note that n and m are determined by the information given by the instance of (33). In this model the u_i's are continuous variables while α_j and α_0 are integer valued variables. Here, the integer variables α_j and α_0 represent $\lfloor u'A_j \rfloor$ and $\lfloor u'b \rfloor$ in the CG-cut, respectively. The rounding conditions on α_j can be imposed through upper bound conditions on the variables α_j, $j = 1, \dots, n$, as in (72b), and with a lower bound condition on α_0, as in (72c). Since $\alpha_0 = \lfloor u'b \rfloor$ implies that $u'b - 1 \leq \alpha_0 \leq u'b$, the small constant $\epsilon > 0$ is introduced to ensure that when $u'b$ is integral then $\alpha_0 = u'b$ and not $u'b - 1$. Finally we should state that the objective function gives the amount of violation of the CG-cut evaluated.

As mentioned in Section 2.3.2 the CG-cut associated with any $u_i \geq 1$ is a dominated one, so we only consider $u_i < 1$. Also note that any variable y_j with $y^*_j = 0$ gives no contribution to the cut violation, so we use this property to reduce the size of the separation problem by not considering it explicitly in the separation model. Define the set $J(y^*) := \{j \in \{1, \dots, n\} :\ y^*_j > 0\}$ and slack variables $f_j = u'A_j - \lfloor u'A_j \rfloor$ for $j \in J(y^*)$ and $f_0 = u'b - \lfloor u'b \rfloor$. The slack variables f_j for $j \in J(y^*) \cup \{0\}$ are fractional and we require their values to be in the range $[0, 1 - \delta]$ for a small fixed value $\delta > 0$. Then the model (72) can be rewritten as to

$$\begin{aligned}
\underset{(\alpha,\alpha_0,u,f,f_0)}{\text{maximize}}\ & \sum_{j \in J(y^*)} \alpha'_j y^*_j - \alpha_0, && && \text{(73a)}\\
\text{s.t.}\ & f_j = u' A_j - \alpha_j, && j \in J(y^*), && \text{(73b)}\\
& f_0 = u'b - \alpha_0, && && \text{(73c)}\\
& 0 \leq f_j \leq 1 - \delta, && j \in J(y^*) \cup \{0\}, && \text{(73d)}\\
& 0 \leq u_i \leq 1 - \delta, && i = 1,\dots,m, && \text{(73e)}\\
& \alpha_j \in Z, && j \in J(y^*) \cup \{0\}. && \text{(73f)}
\end{aligned}$$

5.1.2 Projected Chvàtal-Gomory separation problem

Since by Proposition 3.1 in Section 3.2, the binary requirements on the variables x_{it} in the opportunistic replacement problem (33) can be relaxed, we can rewrite the problem (71) as the problem below:

$$\min\{c'x + d'z : A^1x + A^2z \leq b,\ x \geq 0,\ z \geq 0,\ x \in R^{NT},\ z \in \{0,1\}^T\} \tag{74}$$

In other words the opportunistic replacement problem can be treated both as an ILP and a MILP. This fact will give us the permission to project the problem onto the space of the integer variables z_t. Then we can derive CG-cuts for the projected polyhedron. The separation problem associated with the polyhedron projected into the z-dimension can be defined similarly to the problem (72)

$$\begin{aligned}
&\underset{(\alpha,\alpha_0,u)}{\text{maximize}}\ \alpha'z^* - \alpha_0, && && \text{(75a)}\\
&\text{s.t.} && \alpha_j \leq u'A^2_j, && j = 1,\dots,p, && \text{(75b)}\\
& && 0 \leq u'A^1_j, && j = 1,\dots,r, && \text{(75c)}\\
& && \alpha_0 - \epsilon \geq u'b - 1, && && \text{(75d)}\\
& && u_i \geq 0, && i = 1,\dots,m, && \text{(75e)}\\
& && \alpha \in Z^p_+,\ \alpha_0 \in Z_+, && &&
\end{aligned}$$

where $r = NT$, $p = T$ and $m = 2NT + N - \sum_{i=1}^{N} T_i$.

In [7] it is shown that for the matching problem, for which the convex hull and the first Chvátal closure of the problems coincide, solving the rank-1 separation problem is very effective. Our aim here is to investigate how tight is the first Chvátal-Gomory closure, i.e., P_1, defined in (9), for the opportunistic replacement problem (33) and how generating rank-1 CG inequalities can improve the bounds on the problem. We therefore in the next section, implement these models into a branch and bound procedure in which the separation model is solved and the most violated cuts for non-integral LP relaxation solutions of the opportunistic replacement problem is found. By a similar procedure we investigate how effective is the projected CG cuts on some instances of the opportunistic replacement problem. Also we try to find CG cuts for the opportunistic replacement problems of higher ranks. In the next section, first the computational procedures made to answer these questions are described in details, and then the computational results and conclusions are presented.

5.2 Computational tests

In this section we address the details of generating valid inequalities for the replacement polytope by iteratively solving the MILP models (73) and (75) in a pure cutting plane framework.

Implementation in a pure cutting framework: We have implemented our CG separating problems into a pure cutting plane framework. As a first test, we generate CG-cuts of rank-1 with respect to the original formulation of the opportunistic replacement problem at hand. The simple procedure for the first test is as follows. We solve the continuous relaxation of an instance of the opportunistic replacement problem using a general LP solver, then we try to find the most violated CG-cut for the LP optimal solution y^* by solving the separation problem (73) through a general MILP solver. We store the corresponding CG-cut in a pool. Then, these CG-cuts

is added to the current opportunistic replacement problem formulation. The continuous relaxation of the updated opportunistic replacement problem is solved again with the LP solver. In each step a new LP optimal solution y^* is obtained, and a CG-cut of rank-1 is obtained for that particular solution. Note that at each step, to ensure that all generated cuts are of rank 1, we stick to the original formulation of the opportunistic replacement problem when solving the separation problem (73), i.e., at each step the pair (A, b) is given by (71) and only y^* changes. We continue the generation of CG-cuts of rank-1 until either an integer solution is found by the LP solver or no such violated cut exists. In the latter case, if still there is a gap between the LP optimal value and the known integer optimal value, this means that we have optimized over the first Chávtal closure and for improving the solution higher rank CG-cuts are required. After the end of the separation phase, all the CG-cuts generated by this method can be added to the original ILP model of the opportunistic replacement problem to study how it can affect solving the opportunistic replacement problem through a general ILP solver.

The second test is designed to study the strength of the projected cuts. The procedure taken in the second test follows the same scheme as test one, except that in our cutting plane algorithm pro-CG-cuts of rank-1 are generated and saved. This means that at each step the MILP model (75) is solved to find pro-CG-cuts for the projected replacement polytope into the z_t variables space. We generate pro-CG-cuts iteratively to observe how a pro-cut can improve the optimal solution to the continuous relaxation of the opportunistic replacement problem.

The experiments above concern the optimization over the first Chvátal closure. For the next test we want to investigate whether producing CG-cuts of higher rank could close the integrality gap and compare it with the case when only rank-1 CG-cuts are generated. The new CG-cuts are found by combining the generated inequalities and original ones. The generated cuts are saved in a pool in order to add these inequalities to the original formulation when solving the opportunistic replacement problem with a general ILP solver.

Making the cut sparser and stronger by the penalty term: A major computational issue when solving the Chvtátal-Gomory separation problem is that several equivalent solutions of the separation problem typically exists. Some of these solutions produce very weak cuts for the opportunistic replacement problem which makes the strength of the cuts an issue. One need to produce a violated cut as strong as possible with respect to the first Chvátal closure. Therefore we look for a solution with as few nonzero elements as possible. To obtain such an answer we introduce the penalty term $-\sum_{i=1}^{m} \mu_i u_i$ in the objective function (73a) where $\mu_i = 10^{-4}$ for all $i = 1, \ldots, m$. The objective function is then to

$$\underset{(\alpha,\alpha_0,u,f,f_0)}{\text{maximize}} \Big(\sum_{j \in J(y^*)} \alpha_j^T y_j^* - \alpha_0 - \sum_{i=1}^{m} \mu_i u_i \Big).$$

We have defined the gap closed percentage as follows

Definition 5.1 (gap closed). *The proportion of the integrality gap closed is defined as*

$$\frac{\textit{optimal value}_{(P_1)} - \textit{optimal value}_{(P)}}{\textit{optimal value}_{(P_I)} - \textit{optimal value}_{(P)}},$$

where P_1 is defined in (9), and $P_I = conv(P \cap Z^n)$, as defined in Section 2.4. Furthermore, we choose $\delta = 0.01$ in the model (73) in our computations.

5.2.1 Numerical results

In this section we report the outcome of our experiments on a test-bed made of eight opportunistic replacement instances. The approach follows the scheme used in Section 5.2, i.e., we implemented a pure cutting plane algorithm where, at each iteration, CG-cuts are generated by solving the separation problems (73) and (75) using a standard MILP solver. In order to speed up the overall computation, the MILP solver is aborted when a certain time limit is reached. This time limit is chosen by considering the size of the problem instance at hand. Our implementation of the cutting-plane methods uses the commercial software `ILOG-Cplex 12.0` as LP solver, whereas the separation problems are solved by `ILOG-Cplex 12.0` with the parameter settings in Table 9. For a reference to the parameter settings see [1]. Algorithms are written in `MATLAB R2009b`. All numerical experiments are performed on a Linux desktop operating system with the processor Pentium(R), Dual-Core E5200 @ 2.50GHz. The CPU MHz is 1200.000 and it has a cache size of 2048 KB. All times are reported in CPU seconds.

Each testbed problem correspond to different instances of the opportunistic replacement problem. The data for these instances is reported in Table 10. Problems 1 and 2 in the testbed are very simple problems. The vectors d, c for problems 1 and 2 are reported in Table 10. For problems 3–8 in our testbed, it is assumed that $c_{it} = c_i$, for all $i \in \mathcal{N}$ and $t \in \mathcal{T}$, also $d_t = d$ for all $t \in \mathcal{T}$. Problem 3 is a middle sized problem intended to resemble a realistic problem. Problem 4 is a rather dense problem in the sense that it makes the optimal solution to contain many maintenance occasions. The data in problems 5 and 6 are from real-world instances obtained from Volvo Aero aircraft engine. Problem 6 is a sparse problem. Problem 5 and 6 are corresponding to the data from High Pressure Turbine (HPT) and Low Pressure Turbine (LPT) respectively. Note that the data from real world instances are scaled so that $d = 1$. Problems 7 and 8 are also rather dense with the same data in their objective function while problem 7 has a longer planning horizon.

Table 9: Parameter settings for solving the MILP separation problems with `cplex`

Directive	Value	Description
mipemphasis	4	indicates emphasis on finding very good feasible solutions.
nodefile	2	creates a compressed version of the node file in memory.

The testbed problems, while the integrality requirements on the variables x_{it} and z_t are relaxed, is solved with `cplex` and the results are presented in Table 11. The integer linear programming solutions, solution time in CPU seconds, and the number of `cplex` iterations are also reported in Table 11.

As stated in Section 5.2, the CG separating problem (73) is solved at each step of a pure cutting plane framework. CG-cuts of rank-1 with respect to the original

Table 10: Basic data for the testbed problems

instance	T	N	$\min T_i$	$\max T_i$	polyhedron dimension	d	c_1	c_2
1	4	2	3	4	12	$(3,3,1,3)$	$(1,1,2,1)$	$(1,5,5,1)$
2	4	2	2	3	12	$(3,3,1,3)$	$(1,1,2,1)$	$(1,5,5,1)$
							$\min c_i$	$\max c_i$
3	10	9	2	8	100	1	0.3613	4.0255
4	18	2	3	4	54	9	5	6
5(HPT)	100	9	15	80	1000	1	0.3613	4.0255
6(LPT)	110	10	29	60	1210	1	0.3171	1.5482
7	40	3	3	5	80	10	5	7
8	15	3	3	5	60	10	5	7

Table 11: Solution of the testbed problems. The continuous relaxations and the ILP's are solved by the general ILP solver `cplex` with default parameter settings.

instance	continuous relaxation		ILP solution		
	time (s)	opt. value	time (s)	# iterations	opt. value
1	0.0137	6.5	0.0160	10	7
2	0.0149	11	0.0360	12	12
3	0.0155	47.8397	0.0465	109	48.8397
4	0.0146	110	0.0271	96	114
5(HPT)	0.1116	57.5946	2.4483	28018	58.7463
6(LPT)	0.1332	23.7225	0.2659	1322	23.7225
7	0.0205	314.75	0.5799	7276	352
8	0.0171	118	0.0303	113	120

formulation of the testbed problems are generated. The computations are stop when either an integer solution is obtained or when no such cuts exists. The results are reported in Table 12, which shows that for some instances of the opportunistic replacement problem, by only generating rank-1 CG-cuts, the integrality gap can be significantly closed, while for testbed problem 5 with data from HPT it fails to increase the lower bound. Table 13 reports the results for the cutting plane algorithm using pro-CG-cuts for the testbed problems.

Table 12: Results when the rank-1 separation problem is implemented in a pure branch and cut framework and applied to the test bed problems.

instance	# cuts	time(s)	optimal value	% gap closed
1	1	2.27	7	100
2	3	10.61	12	100
3	4	2661.8	48.8397	100
4	8	2613.5	114	100
5(HPT)	28	71823.0	57.5946	0
6(LPT)	10	16710.0	23.7225	100
7	68	105800.60	335.9700	56.97
8	7	901.28	120	100

Table 13: Results when the projected separation problem is implemented in a pure branch and cut framework and applied to the testbed problems. The number of projected cuts is limited to 100.

instance	# cuts	time(s)	optimal	% gap closed
1	1	2.1700	7	100
2	2	3.7000	11	0
3	4	35.9900	48.8397	100
4	4	341.7400	111	25
5(HPT)	100	72452.0	57.5946	0
6(LPT)	7	674.8300	23.7225	100
7	19	51329.0	319.5000	12.75
8	13	813.6500	119	50

To observe how changing the time horizon T can affect the behavior of the solution procedure, we have considered the testbed problem 5 with data from HPT and varied T between 55 and 100. The LP relaxation and ILP solutions for instances of the HPT problem with various time horizons are reported in Table 14. The results for our implementations of the cutting plane method where the separation problems are solved with the MILP models (73) and (75) are reported in Table 15. For the HPT problem with various time horizons, it is obvious from Table 15 that the cutting plane method generally fails to close any integrality gap, except for the instance where $T = 60$.

Table 14: The continuous relaxation and the ILP solution of the opportunistic replacement problems with data from HPT and various values of the time horizons T reported from `cplex`

HPT	continuous relaxation solution		ILP solution		
T	time(s)	opt. value	time (s)	# iterations	opt. value
55	0.0277	24.6670	0.0842	440	24.7946
60	0.0318	31.6229	0.0934	466	31.6229
70	0.0480	35.1382	0.4303	4937	35.3195
80	0.0562	46.1809	1.2196	21748	47.1809
90	0.0755	53.4250	1.7091	14456	54.2643
100	0.1116	57.5946	2.4483	28018	58.7463

Table 15: Rank-1 and projected separation problem applied to the HPT instance with various values of T in a pure branch and cut framework.

HPT	Rank-1 separation problem				Projected CG-cuts			
T	#cuts	time(s)	opt. value	%gap clo.	#cuts	time(s)	opt. value	%gap clo.
55	46	13458.0	24.6926	20.06	2	409.7000	24.6670	0
60	13	8365.0	31.6229	100	23	1936.6	31.6229	100
70	19	20596.0	35.1382	0	100	8859.3	35.1382	0
80	10	6194.0	46.1809	0	100	35023.0	46.1809	0
90	15	23474.0	53.4250	0	100	67851.0	53.4250	0
100	28	71823.0	57.5946	0	100	72452.0	57.5946	0

Our previous experiments show that optimization over the first rank closure in some cases gives a very good approximation of the integer optimal value while in some other cases it fails to close any integrality gap. In our experiments we consider a case in which, in our cutting plane method, CG-cuts of higher ranks are generated. In this test beside, the original constraints of the opportunistic replacement problem (33), we consider the new generated CG-cuts from the separation problem (73). The results are reported in Table 16.

Table 16: Beyond the first Chvátal closure, CG-cuts of higher ranks has been generated. The separation problem to find the most violated cuts is limited to be solved 100 times for problem 7 and 150 times for problem 5.

instance	# cuts	time (s)	opt. value	% gap closed
2	2	13.7300	12	100
3	7	9533.1	48.8397	100
4	10	2664.5	114	100
5(HPT)	148	166460.0	57.8292	20.37
6(LPT)	8	1984.0	23.7225	100
7	100	109110.0	331.1521	44.03
8	9	2662.6	120	100

The CG-cuts which have been generated in the separation phase have been saved. We add the generated rank-1 cuts to the original formulations of the problem with integrality gap as reported in Tables 12 and 15. Then these new formulated problems are solved with `cplex`. The results are illustrated in Tables 17 and 18. We can compare the results with the ones in the Table 11 and 14.

Table 17: Rank-1 CG cuts are added to the original formulation and solved with `cplex` to obtain an integer solution for the instances for which the integrality gap is not completely closed (Table 12). We compare the results with those from Table 11.

instance	time (s)	# iterations
5(HPT)	1.5854	12869
7	0.7852	17311

5.2.2 Conclusions

The separation problem is $\mathcal{NP}$-hard in general, so as expected the computation time for solving the MILP models (73) and (75) is considerable. For many instances the cutting plane algorithm based on finding the most violated rank-1 CG-cuts for the LP solutions is effective to improve the lower bounds of the objective value. In other words this pure branch and bound procedure for rank-1 CG-cuts is capable of finding facets of the replacement polyhedron in the neighborhood of an optimal solution. By the results obtained in Table 12 one can conclude that the effectiveness

Table 18: Rank-1 CG-cuts are added to the original formulation and solved with `cplex` to obtain an integer solution for the problem HPT with various time horizons. We compare it with Table 14.

T	time (s)	# iterations
55	0.0541	243
70	0.1587	936
80	0.5849	7078
90	0.6739	4026
100	1.5854	12869

of generating the most violated CG-cuts is dependent on the sparseness of the constraint coefficients and the objective function rather than the size of the replacement polyhedron.

Finding the projected CG-cuts requires less computation time. This is because the MILP problem (75) is smaller than (73). For the strength of the projected CG-cuts it is of importance whether, in the general MILP problem at hand, the optimization of the integer variables, or optimizing over the continuous variables is the key. More precisely for our opportunistic replacement problem where there is a tie between the integer variables z_t and continuous variables x_{it}, a situation that may occur is that the projection z^* of the optimal solution (x^*, z^*) of the opportunistic replacement relaxation problem belongs to the first Chvátal closure $P_1(z)$. In this case, no pro-CG-cut can cut off that point, although there might be a huge gap between the optimal integer solution and its relaxation. This can be observed in Tables 13 and 15, in which for most instances in comparison with rank-1 CG-cuts, a smaller gap percentage is closed. Note that the percentage of gap closed is dependent on the objective function.

Producing CG-cuts of higher rank needs more computation time than only generating rank-1 CG-cuts. The reason is that in each step a new CG inequality is added to the matrix A in the model (73) which increases the size of the instance. Table 16 shows that it is beneficial to generate higher rank CG-cuts especially for dense and high dimensional problems. By generating higher rank CG-cuts more dominate valid inequalities can be generated and a tighter bound on the optimal solution with less number of valid inequalities can be obtained. Table 16 shows that producing higher rank cuts has the benefit of closing some of the integrality gap for the problem HPT while generating only rank-1 CG-cuts fails to do so.

Finally we have added the generated rank-1 CG-cuts to the original formulation of the opportunistic replacement instances where our cutting plane algorithm fails to report an integer solution. Comparing Tables 17 and 11 shows that although the generation of CG-cuts fails to increase the lower bound on the objective value for the HPT problem with $T = 100$, adding these new constraints to the original formulation simplifies the problem and decreases the computation time. However, for the smaller problem 7, although by generation the rank-1 CG-cuts 56.97 % of the integrality gap is closed, but adding the generated cuts to the original formulation makes the problem more complex and increases the computation time and the

number of simplex iterations. This is because adding the new inequalities makes the linear programs big which takes more time to solve, and so this is not so much of benefit for small size problems. Table 18 shows that the generated rank-1 CG-cuts for the HPT problem with various time horizons increases the computation time and the number of simplex iterations significantly.

In general, solving the separation problems (73) and (75) consumes huge amounts of computer memory.

6 Conclusions and future work

6.1 Conclusions

The focus of this thesis is to study the mathematical property and facial structure of the opportunistic replacement problem with deterministic component lives. The Benders decomposition method is implemented and a separation problem is modeled and solved in a branch and bound algorithm. The main results are as follows.

If the maintenance occasions are fixed, the remaining optimization model is a linear programming problem. The dual problem of this linear programming problem is presented. It is shown that an equivalent representation of the dual problems exists. Moreover, if the maintenance costs are non-increasing with time, the dual linear programming problem can be solved through a greedy algorithm. An implementation of the Benders decomposition method applied to the opportunistic replacement problem is discussed. Feasibility constraints for the master problem in the Benders algorithm are derived.

The computational experiments show that for the instances with non-increasing maintenance costs, using the greedy algorithm to solve the dual linear programming subproblems in the Benders algorithm often decreases the computational times. Besides, the computation time for solving the master problems in the Benders algorithm increases by each iteration. As an overall conclusion, when the size of the opportunistic replacement problem increases the computation time for solving the master problems in Benders algorithm behave exponentially, which makes this method inefficient for the opportunistic replacement problem. However, it is suggested that Benders decomposition method can be utilized to find a good feasible solution and an initial point for solving the opportunistic replacement problem.

This thesis also includes a branch and cut approach for solving the opportunistic replacement problem. A modified mixed integer linear programming model for the rank-1 Chvátal-Gomory separation problem is described. Then the projected Chvátal-Gomory separation problem is presented. The models are implemented in a pure cutting plane framework to generate the most violated first rank CG-cuts in order to obtain a tighter bound on the optimal solution. Then, Chvátal-Gomory cuts of higher ranks have been generated for middle sized instances of the opportunistic replacement problem. Since the separation problem is $\mathcal{NP}$-hard, the computation times and the memory usage are considerable. The effectiveness of generating most violated CG-cuts seems to be dependent on the sparseness of the constraint coefficients and the objective function rather than the size of the replacement polyhedron. However, generating the rank-1 Chvátal-Gomory cuts often yields a very tight approximation of the (integer) optimal value for the opportunistic replacement problem.

6.2 Future work

The work in this thesis shows that the Benders decomposition method is inefficient for the opportunistic replacement problem in general. However, the author's opinion is that Benders decomposition method can be utilized to find a good feasible solution and an initial point when solving the opportunistic replacement problem. Also generating the rank-1 Chvátal-Gomory cuts often gives a tight bound on the optimal solution of the opportunistic replacement problem. Obviously, one should

generate CG-cuts of any rank and search for new techniques to generate valid inequalities and find new classes of facets for the opportunistic replacement problem in order to obtain satisfactory results. Solving the separation problem and finding most violated valid inequalities for the replacement polytope could be useful as a tool to guess structures of some new classes of facets.

In this thesis work, a basic opportunistic replacement problem is considered in which the lives of all the components are deterministic. One may ask whether the work done generalizes to more realistic models for opportunistic maintenance or not.

In realistic situations, maintenance problems often include components with stochastic lives and it is important to apply the opportunistic replacement model to these problems as well. In [15], a two-stage stochastic programming approach for the opportunistic replacement problem with stochastic component lives, is developed and studied.

As an extension of the opportunistic replacement problem, one can consider a maintenance problem with different lives for different individuals of the same component. This problem is called the opportunistic replacement problem with individual lives. Solving a stochastic opportunistic replacement problem with perfect information about individual component lives leads to solving an opportunistic replacement problem with individual lives ([17]). Furthermore, a model of the opportunistic replacement problem with individual lives is the basis of a model of the current problem for the stochastic opportunistic replacement problem ([15]).

The scope of the future research work can be divided into three categories. One can work on realistic problems directly from the industry, study more complex real problems, and extend the current results to these kinds of problems. More extensions can be obtained by considering other defining factors (such as human work resource, etc). A second approach is to contribute to find effective methods in solving the multistage stochastic opportunistic replacement problem. A third possibility is to utilize the results of the facial structure of the opportunistic replacement problem, to the generalized cases such as stochastic opportunistic replacement problem and the opportunistic replacement problem with individual lives, in order to solve these generalized problems more efficiently. The goal of these research areas is the contribution of finding efficient solutions for larger and more complex maintenance problems.

References

[1] *ILOG AMPL CPLEX System, Version 11.0, Users Guide*, ILOG S.A. and ILOG, Inc., January 2008.

[2] T. Almgren, N. Andréasson, D. Anevski, M. Patriksson, A. Strömberg, and J. Svensson, *Optimization of opportunistic replacement activities: A case study in the aircraft industry*, preprint, Department of Mathematical sciences, Division of Mathematics, Chalmers University of Technology and University of Gothenburg, Göteborg, Sweden, 2008.

[3] M. S. Bazara, J. Jarvis, and H. D. Sherali, *Linear Programming and Network Flows, Fourth Edition*, John Wiley & Sons, New York, NY, USA, 2010.

[4] P. Bonami, G. Cornuéjols, S. Dash, M. Fischetti, and A. Lodi, *Projected Chvátal–Gomory cuts for mixed integer linear programs*, Mathematical Programming, 113 (2008), pp. 241–257.

[5] A. Caprara and M. Fischetti, $\{0, 1/2\}$*–Chvátal–Gomory cuts*, Mathematical Programming, 74 (1996), pp. 221–235.

[6] R. Dekker, R. E. Wildeman, and F. A. Van Der Duyn-Schouten, *A review of multi-component maintenance models with economic dependence*, Mathematical Methods of Operational Research, 45 (1997), pp. 441–435.

[7] M. Fischetti and A. Lodi, *Optimization over the first Chvátal closure*, Mathematical programming, Series A and B, 110 (Issue 1, 2007), pp. 3–20.

[8] C. A. Floudas and P. M. Pardalos, *Encyclopedia of Optimization, Volume 6*, Kluwer Academic Publishers, 2001.

[9] C. Gentile, P. Ventura, and R. Weismantel, *Mod-2 cuts generation yields the convex hull of bounded integer feasible sets*, SIAM Journal on Discrete Mathematics, 20 (Issue 4, 2006), pp. 913–919.

[10] P. A. Jensen and J. F. Bard, *Benders' Method, PDF supplements to the book "Operations Research Models and Methods"*, John Wiley & Sons, New York, NY, USA, 2003.

[11] L. S. Lasdon, *Optimization Theory for Large Systems*, The MACMILLAN COMPANY, New York, NY, USA, 1970.

[12] G. L. Nemhauser and L. A. Wolsey, *Integer and Combinatorial Optimization*, John Wiley & Sons, New York, NY, USA, 1988.

[13] M. Önnheim, *The facial structure of and efficient solution methods for the opportunistic replacement problem*, Master's thesis, Chalmers University of Technology, Department of Mathematical Sciences, March 2010.

[14] M. W. Padberg, *On the facial structure of set packing polyhedra*, Mathematical Programming, 5 (1973), pp. 199–215.

[15] M. Patriksson, A.-B. Strömberg, and A. Wojciechowski, *The stochastic opportunistic replacement problem: A two stage solution approach*, to be submitted to Annals of Operations research.

[16] C. Raack and A. Werner, *Introduction to Porta and Polymake*, ZIB, Berlin Mathematical School, Berlin, 2009.

[17] A. Wojciechowski, *On the optimization of opportunistic maintenance activities*, licentiate thesis, Chalmers University of Technology, University of Gothenburg, Gothenburg, Sweden, 2010.

[18] A. Wojciechowski, T. Almgren, N. Andréasson, M. Patriksson, and A. Strömberg, *The opportunistic replacement problem: Analysis and case studies*, tech. rep., Department of Mathematical sciences, Division of Mathematics, Chalmers University of Technology and University of Gothenburg, Göteborg, Sweden, May, 2010.

[19] L. A. Wolsey, *Integer Programming*, John Wiley & Sons, New York, NY, USA, 1998.