A Multi-Period Renewal equipment problem

Xiaokang Cao, Antoine Jouglet, Dritan Nace

A multi-period renewal equipment problem

Xiaokang Cao, Antoine Jouglet and Dritan Nace

UMR CNRS 6599 Heudiasyc, Université de Technologie de Compiègne
Centre de Recherches de Royallieu, 60200 Compiègne, France
xiaokang.cao@hds.utc.fr, antoine.jouglet@hds.utc.fr, dritan.nace@hds.utc.fr

Abstract

This paper looks at a Multi-Period Renewal equipment problem (MPR). It is inspired by a specific real-life situation where a set of hardware items is to be managed and their replacement dates determined, given a budget over a time horizon comprising a set of periods. The particular characteristic of this problem is the possibility of carrying forward any unused budget from one period to the next, which corresponds to the multi-periodicity aspect in the model. We begin with the industrial context and deduce the corresponding knapsack model that is the subject of this paper. Links to certain variants of the knapsack problem are next examined. We provide a study of complexity of the problem, for some of its special cases, and for its continuous relaxation. In particular, it is established that its continous relaxation and a special case can be solved in (strongly) polynomial time, that three other special cases can be solved in pseudo-polynomial time, while the problem itself is strongly *NP*-hard when the number of periods is unbounded. Next, two heuristics are proposed for solving the MPR problem. Experimental results and comparisons with the Martello&Toth and Dantzig heuristics, adapted to our problem, are provided.

Keywords: scheduling, knapsack, multi-period, complexity, heuristic methods

1. Introduction

The *Knapsack Problem* (KP) has a significant place in the study of integer programming models with binary variables. In the standard knapsack problem the quantity $\sum_{i \in N} p_i x_i$ is to be maximized subject to the constraint $\sum_{i \in N} w_i x_i \leq b$, where $x_i \in \{0, 1\}$, $N = \{1, 2, \ldots, n\}$, p_i is the *value* or *profit* of item i, w_i is the *weight* of item i and b is the knapsack *capacity*, all assumed to be non-negative. In addition to the standard problem, a number of different variants of the problem have been put forward and investigated by researchers over the last decades. This paper looks at one of these variants that we have called the **Multi-Period Renewal equipment problem** (MPR). It is inspired by a specific real-life situation where a set of hardware items is to be managed and their replacement dates determined, given a budget over a time horizon comprising a set of periods. The particular characteristic of this variant of the problem is the possibility of carrying forward any unused budget from one period to the next, corresponding to the multi-periodicity aspect in the statement of the problem. The contribution of this paper is a new knapsack model originating from a real industrial context, together with a complete theoretical examination of the problem and its relations to other knapsack problems, and a set of efficient heuristics for solving it.

The paper is organized as follows. In Section 2 we present the specific industrial context that gave rise to our problem and the corresponding mathematical model. Section 3 looks at links to other knapsack problems. In Section 4 we investigate the computational complexity of the MPR problem as well as some of its special cases and its continuous relaxation. We show in particular that the MPR problem is strongly NP-hard when the number of periods is unbounded and weakly NP-hard for the bounded case. Finally, in Section 5, we propose two new heuristics for the MPR problem and recall how two other well-known heuristics, that is to say the Dantzig

and Martello&Toth heuristics, are also suitable for solving MPR. We provide a comparative experimental study of all these heuristics.

2. From an industrial problem to a theoretical model

In some countries it is usual for a city, town or municipality to contract certain public utilities (water supply, electricity, etc.) out to private companies, usually under concessions, leases or management contracts. Under these arrangements, the public entity delegates the provision of the service for a time horizon M typically ranging from 15 to 25 years, while the private entity remains under a contractual obligation to spend a given amount of money (B) on the maintenance and renewal of equipment. The company's maintenance strategy is based on continuous renewal so as to ensure continuity of service and to avoid problems with antiquated plant. For the application in hand (water supply network), equipments have lifetimes that range from 50 to 100 years which is largely greater than the considered time horizon. This implies that at most one replacement occurs over time horizon M. However, in practice more than one replacement could happen for one equipment during the time horizon due to unpredictable failures. These situations are handled by the daily maintenance process, while this paper deals only with the strategic maintenance process. In line with an internal budgeting policy the company allots an annual budget b_j to such expenditure, that is $\sum_{j \in M} b_j = B$. Given that the entire budget has to be used up, the company carries any unused budget at year j over to the following years. For each piece of equipment the replacement cost is assumed to be constant over the time horizon M, because in pratice there is no reliable information of the variation of these costs over time. On the other hand, the profit attributable to the replacement, is calculated according to a formula based on such elements as the probability of failure, the expected lifetime of the equipment, its importance in the industrial process, etc. Hence, the profit change along with time and this

change corresponds to a certain deterioration process [1]. Thus the related cost, profit and budget coefficients are assumed to be known with certainty, in contrast to conventional renewal theory which relies on probability theory, (see for instance Cox [2]). From this point of view the problem is a simplified deterministic version of conventional renewal problems. However, there is one particular property that increases the difficulty of problems, that is to say the property of multi-periodicity. More specifically, any decision made in some period j impacts those made in subsequent periods.

Before formulating the mathematical model of the MPR problem, let us give the notation used throughout the paper. Let N be a set of n pieces of equipment, and M a horizon of m periods.

- $x_{i,j}$ is the assignment decision variable, that is to say $x_{i,j} = 1$ if equipment i is replaced in period j, and 0 otherwise;
- $p_{i,j}$ is the *profit* obtained when replacing equipment i at period j;
- w_i is the *replacement cost* of equipment i (it remains unchanged over periods);
- b_j gives the *budget* alloted to period j.

All these data are assumed to be non-negative integers. Our problem can be mathematically

formulated as follows:

$$
\begin{aligned}
&\max \sum_{i\in N}\sum_{j\in M} p_{i,j}x_{i,j} \\
&\sum_{i\in N} w_i x_{i,1} \le b_1, \\
\sum_{i\in N} w_i x_{i,2} &\le b_2 + b_1 - \sum_{i\in N} w_i x_{i,1}, \\
&\dots \\
\sum_{i\in N} w_i x_{i,j} \le b_j + \sum_{t=1}^{j-1} b_t - \sum_{t=1}^{j-1}\sum_{i\in N} w_i x_{i,t}, &\qquad \forall j \in M, \\
\sum_{j\in M} x_{i,j} \le 1, &\qquad \forall i \in N, \\
x_{i,j} \in \{0,1\}, &\quad \forall i \in N, j \in M.
\end{aligned}
$$

In the above formulation, the term $b_1 - \sum_{i\in N} w_i x_{i,1}$ gives the unused budget at the end of the first year. This is added to the allotted budget for the second year, and so on. Let B_j denote the cumulative budget from period 1 to period j, that is $B_j = \sum_{t=1}^{j} b_t$. Since all b_j are assumed to be non-negative, we have the following relation: $B_1 \le B_2 \le \cdots \le B_m$. The problem can then be rewritten as follows:

$$
\begin{aligned}
(P) \qquad &\max \sum_{j\in M}\sum_{i\in N} p_{i,j}x_{i,j} && (1)\\
&\sum_{t=1}^{j}\sum_{i\in N} w_i x_{i,t} \le B_j, \qquad \forall j \in M, && (2)\\
&\sum_{j\in M} x_{i,j} \le 1, \qquad \forall i \in N, && (3)\\
&x_{i,j} \in \{0,1\}, \quad \forall i \in N, j \in M. && (4)
\end{aligned}
$$

Note that when $B_j = B_{j+1}$ for two consecutive periods j and $j+1$, it can easily be shown that only the capacity constraint related to period $j+1$ needs to be retained, and for each item

the choice having the lower profit over these two periods may simply be discarded. Hence, we assume that $B_1 < B_2 < ... < B_m$.

From now on, in line with the notation commonly used for knapsack models, we shall use the term *item* for equipment, *weight* instead of replacement cost, and *capacity* for budget. Hence, item i assigned to period j reads equipment i replaced during period j. In the above model the sum of the weights of all items chosen from period 1 to period j cannot exceed capacity B_j for all $j \in M$ (2). Each item i can be assigned to at most one period j (3). The multi-period aspect lies in the fact that each constraint involves the current period and all preceding ones. The total profit is to be maximized (1). As far as we know, we were the first to model the problem in [3]. In the following we establish and exhibit links with some other problems studied in the literature.

3. Literature review and links with other knapsack problems

Let us first look at the *Multi-Period Knapsack problem* (MPK) introduced by Faaland in 1981 [4]. Faaland considers a set N of items and a set M of periods. To each period $j \in M$ there corresponds a subset $N_j = \{j \in 1, ..., m\}$ of items that can be assigned to this period. Note that $\bigcup_{j=1}^{m} N_j = N$ and $N_k \cap N_j = \emptyset$ for each pair $(j, k)|j \neq k$ of items. For each item i, a *profit* p_i and a *weight* w_i are given. The cumulative weight of all items chosen from period 1 to period j cannot exceed the *capacity* B_j associated with period j. The total profit has to be maximized by selecting items in their associated periods. The decision variables in this problem are unbounded: an item i may be chosen more than once in period j such that $i \in N_j$. Faaland proposed a polynomial algorithm to solve exactly the continuous relaxation of MPK, and in so doing to compute an upper bound of MPK. He also proposed a *branch and bound* algorithm using this upper bound.

The **binary version** of this problem, in which an item is chosen at most once in its associated

period, is called BMPK. Given the above notation, BMPK can be formulated as follows:

$$\max \sum_{j \in M} \sum_{i \in N_j} p_i x_i \tag{5}$$

$$\sum_{t=1}^{j} \sum_{i \in N_t} w_i x_i \leq B_j, \quad \forall j \in M, \tag{6}$$

$$x_i \in \{0, 1\}, \quad \forall i \in N. \tag{7}$$

Thus the weight of any chosen item will impact subsequent periods, given that in each period the cumulative weight is considered. The overall profit is maximized by choosing items in each period (5), without violating the cumulative capacity constraints (6). It now becomes apparent that BMPK and MPR have some similarities. What is different is that items in BMPK can be only assigned to a single (i.e. its associated) period. As we shall show, BMPK may be viewed as special case of MPR (see Section 4.3), in which an item may be chosen on at most one occasion. BMPK is shown to be weakly *NP*-hard in Section 4.3, and its complexity does not depend on the number of periods.

The *Generalized Assignment Problem* (GAP) is another problem related to knapsack problems. GAP is known to be strongly *NP*-hard and has been widely studied, (see for instance Martello and Toth in 1990 [5], or Nauss in [6] and Oncan in [7] for a survey on resolution methods and applications). GAP can be formulated as follows. We are given a set N of n items and a horizon M of m periods. For each item i, a *profit* $p_{i,j}$ and a *weight* $w_{i,j}$ are given for each period j. In addition, a *capacity* c_j is associated with each period j. The mathematical formulation of

GAP is as follows:

$$\begin{aligned}
\max \sum_{j\in M}\sum_{i\in N} p_{i,j}x_{i,j} & \\
\sum_{i\in N} w_{i,j}x_{i,j} \le c_j, & \qquad \forall j \in M, \\
\sum_{j\in M} x_{i,j} \le 1, & \qquad \forall i \in N, \\
x_{i,j} \in \{0,1\}, & \quad \forall i \in N, \forall j \in M.
\end{aligned}$$

Note that unlike the MPR problem, in the GAP problem no capacity may be carried over to subsequent periods and weights vary over time. A comparison with these two knapsack problems therefore shows that the problem studied in this paper is a problem in its own right, although it contains elements from each of the two above problems, taking from GAP the fact that profits vary over items and periods, while the multi-periodicity aspect is present in BMPK. Moreover, as previously discussed, MPR generalizes BMPK.

Finally, another problem related to multi-period knapsacks is the *Multiple-Choice-Multi-Period Knapsack problem* (MCMPK) studied by Lin and Wu in [8]. This problem is similar to BMPK, except that in MCMPK only one item in each period can be chosen. Lin and Wu proposed a heuristic approach for obtaining a strong lower bound, along with two branch-and-bound procedures for finding the optimal solution. More recently, the same authors have proposed a dynamic programming approach for MCMPK in [9].

4. Complexity

The formulation *P* describes the general case of *MPR*. We focus particularly on complexity aspects related to our problem and discuss in detail its continuous relaxed version and some of its special cases with respect to their computational complexity:

- (P^r): the continuous relaxation of P, *i.e.* $x_{i,j} \in [0,1]$, for all $i \in N$, $j \in M$;
- (P^1): $w_i = w$, for all $i \in N$;
- (P^2): $p_{i,j}$ values are all 0 except for one period, for all $i \in N$;
- (P^3): $p_{i,j}$ values are *non-increasing* during the time horizon, for all $i \in N$.
- (P^4): $p_{i,j}$ values are *non-decreasing* during the time horizon, for all $i \in N$.

We show that P^r and P^1 can be solved in (strongly) polynomial time. We also show that solving P^2 leads to a solution to the BMPK problem (see Section 3) and that both problems can be solved in pseudo-polynomial time. Finally, we show that the version of P^3 when the number of periods is bounded and that P^4 can be solved in pseudo-polynomial time, while P^3, and MPR as well, are strongly NP-hard when the number of periods is unbounded.

4.1. The continuous relaxation of MPR problem

Let us begin with the continuous relaxed version of the problem and show that it can be solved in polynomial time. We note first that its continous relaxation cannot be solved through a simple application of the Dantzig rule: choosing the item with the largest $(p_{i,j}/w_i)$ value. On the other hand, a simple change of variables, i.e. $y_{i,j} = w_i x_{i,j}$, allows us to rewrite it as follows:

$$(P^r) \qquad \max \sum_{j \in M} \sum_{i \in N} \frac{p_{i,j}}{w_i} y_{i,j} \tag{8}$$

$$\sum_{t=1}^{j} \sum_{i \in N} y_{i,t} \leq B_j, \qquad \forall j \in M, \tag{9}$$

$$\sum_{j \in M} y_{i,j} \leq w_i, \qquad \forall i \in N, \tag{10}$$

$$y_{i,j} \in [0, w_i], \quad \forall i \in N, j \in M. \tag{11}$$

Now we show that P^r can be expressed through, what we call, a maximum-profit flow model, by analogy with the minimum-cost flow model. We start by constructing a graph composed of the following elements: a source node S, a destination node D, a set of item nodes $n_i, i \in N$, a set of period nodes $t_j, j \in M$. Arcs are of several types: $E_1(i)$ represents the arc from source node S to item node n_i, for all $i \in N$. Its capacity is set to w_i and its profit to 0. $E_2(i, j)$ represents the arc from item node n_i to period node t_j, for all $i \in N$ and $j \in M$. Its capacity is set to w_i and its profit to $p_{i,j}/w_i$. $E_3(j)$ represents the arc from period node t_j to period node t_{j+1}, for all $j \in \{1, \ldots, m-1\}$. Its capacity is set to B_j and its profit to 0. $E_3(m)$ represents the arc from period node t_m to destination node D. Its capacity is set to B_m and its profit to 0.

The objective is to find a flow from the source node S to the destination node D such that the cumulative profit over all arcs is maximized. To make things clearer, let us examine the following example.

Example. Consider an instance of P^r with 3 items with respectively weights 1, 1 and 2, and 2 periods with respectively profits taken $(3, 1)$, $(4, 3)$ and $(4, 2)$. The mathematical formulation is

as follows:

$$\max 3y_{1,1} + 1y_{1,2} + 4y_{2,1} + 3y_{2,2} + 2y_{3,1} + 1y_{3,2}$$

$$y_{1,1} + y_{2,1} + y_{3,1} \leq 1,$$

$$y_{1,1} + y_{2,1} + y_{3,1} + y_{1,2} + y_{2,2} + y_{3,2} \leq 3,$$

$$y_{1,1} + y_{1,2} \leq 1,$$

$$y_{2,1} + y_{2,2} \leq 1,$$

$$y_{3,1} + y_{3,2} \leq 2,$$

$$y_{1,j} \in \{0, 1\}, \quad j \in \{1, 2\}$$

$$y_{2,j} \in \{0, 1\}, \quad j \in \{1, 2\}$$

$$y_{3,j} \in \{0, 2\}, \quad j \in \{1, 2\}$$

Figure 1 gives the corresponding maximum-profit flow model of this instance. The notation c, p is used to indicate an arc with capacity c and profit p. For any instance of P^r we can build an

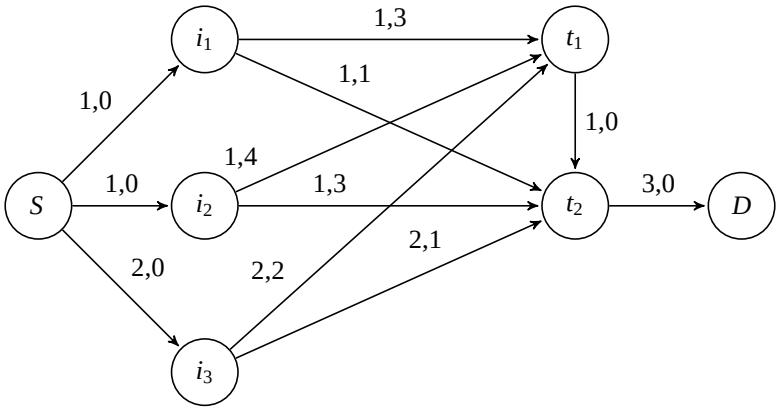

Figure 1: Maximum-profit flow model

instance of the maximum profit flow model. The following result (the proof is immediate) holds:

Proposition 1. *Any optimal solution to P^r is an optimal solution to the corresponding maximum profit flow problem and vice-versa.*

Furthermore, P^r may be reduced to a minimum cost flow problem as follows: obviously the maximum profit is achieved when flow attains its maximum value B_m (note that we assume without loss of generality that $\sum_{j\in M} w_i \geq B_m$). Now let us consider only those arcs (i, j) that correspond to $y_{i,j}$ variables in the relaxed MPR, and again we denote their flow values as $y_{i,j}$. We should have $\sum_{i\in N}\sum_{j\in M} y_{i,j} = B_m$ for the optimal solution of the maximum profit flow problem. Hence, we can restrict ourselves to cases when $y_{i,j}$ variables give admissible flow values for solutions of the maximum profit flow problem such that $\sum_{i\in N}\sum_{j\in M} y_{i,j} = B_m$. Let $p' = \max\{p_{i,j}/w_i | i \in N, j \in M\}$. Then, in the formulation of P^r above we have: $\max\{\sum_{i\in N}\sum_{j\in M}\frac{p_{i,j}}{w_i}y_{i,j}\} \Leftrightarrow \max\{\sum_{i\in N}\sum_{j\in M}\frac{p_{i,j}}{w_i}y_{i,j} - p'B_m\} \Leftrightarrow \max\{\sum_{i\in N}\sum_{j\in M}\frac{p_{i,j}}{w_i}y_{i,j} - p'\sum_{i\in N}\sum_{j\in M}y_{i,j}\} \Leftrightarrow \max\{\sum_{i\in N}\sum_{j\in M}(\frac{p_{i,j}}{w_i} - p')y_{i,j}\} \Leftrightarrow \min\{\sum_{i\in N}\sum_{j\in M}(p' - \frac{p_{i,j}}{w_i})y_{i,j}\}$. Thus, the same flow solution applicable to the above maximum profit flow problem also solves the minimum cost flow problem defined with costs set to $p' - (p_{i,j}/w_i)$ for all above arcs (i, j) and flow equal to B_m and vice-versa. As the minimum cost flow problem can be solved in polynomial time, we conclude that the continous relaxation of MPR (*i.e.* P^r) can be solved in polynomial time.

4.2. Complexity of P^1

When weights are all the same, the problem can be reduced to the special case $w_i = 1$, for all $i \in N$. For this we need to replace all B_j with $\lfloor B_j/w \rfloor$ for all $j \in M$. Let us look now at the special case $w_i = 1$, for all $i \in N$ and show that it can be modeled in the same way as the above problem P^r, and thus can be solved in polynomial time. Obviously, this special case without any variable change leads to P^r and hence can be modeled as a min-cost flow problem. Next, the integrality

of B_j implies that flow variables take integer values and that the corresponding solution of P^1 is also an integer solution.

4.3. Complexity of P^2 and BMPK

Let us first look at BMPK and compare it with P^2. Consider an instance of P^2 in which for each i, $p_{i,j}$ are all 0 except for one period. Now consider an optimal solution of this instance. In this solution, any item i assigned to period j for which $p_{i,j} = 0$ can be removed from the solution without decreasing the profit. Note that the obtained solution is also an optimal solution for the instance of BMPK, which is obtained in the following way:

- each item i has the same weight w_i and each period j has the same capacity B_j in both instances.
- in the BMPK instance, each item i belongs to set N_j such that j is the only period for which $p_{i,j} \neq 0$ in P^2 instance.

Hence, for any optimal solution for P^2 an optimal solution for the corresponding BMPK problem can be deduced. Obviously, there is a bijection between instances of BMPK and those of P^2. Furthermore, the following proposition is straightforward.

Proposition 2. *Any optimal solution for BMPK is also optimal for the corresponding problem P^2.*

Let us now consider the complexity of BMPK.

Proposition 3. *The BMPK problem (and hence the P^2) can be solved in pseudo-polynomial time $O(nB_m)$.*

Proof. This problem can be solved through a dynamic programming schema slightly adapted from the classical schema used for the standard knapsack problem.

First, let $A_i(C)$ denote the maximum profit that can be obtained when considering only the first i items and keeping the total cumulative weight lower than or equal to C. There is one restriction in our dynamic programming, namely that the items are introduced in the order determined by sets N_j ($1 \leq j \leq m$): that is to say that items are indexed from 1 to n such that if $i \in N_j$ and $k \in N_l$ with $j < l$ then $i < k$.

Initially, we set $A_0(C) = 0$ for all C in $[0, B_m]$.

Then, the following recursive formula is applied for any i and $C \in \left[0, B_j\right]$, where j is the period in which i has to be assigned, (i.e., $i \in N_j$):

- $A_i(C) = A_{i-1}(C)$, for $C < w_i$;
- $A_i(C) = \max\{A_{i-1}(C), A_{i-1}(C - w_i) + p_i\}$, for $w_i \leq C \leq B_j$;
- $A_i(C) = A_i(B_j)$, for $B_j < C \leq B_m$.

The value of optimal solution is then given from $A_n(B_m)$. Hence the computational complexity is in $O(nB_m)$ time. □

4.4. Complexity of P^3

We consider in this paragraph the special case P^3, where $p_{i.j}$ decrease throughout time horizon for any item i, $i \in N$. Note first that the problem is obviously NP-complete as the standard knapsack problem can be reduced to the case with $m = 1$ of P^3. We show that P^3 is weakly NP-complete when the number of periods is bounded, and strongly NP-complete when it is unbounded.

4.4.1. The case of bounded number of periods

We first consider the case where the number of periods is taken to be bounded by a constant number m'. The following result holds:

Theorem 1. *For a number of periods m bounded from above by a positive integer constant m', the P^3 problem may be solved in pseudo-polynomial time $O(nm'{B_m}^{m'})$.*

Proof. Let $A_i(C_1, C_2, \ldots, C_m)$ denote the maximal profit obtained when only the i first items are used and the cumulative weight corresponding to period j is less than or equal to C_j. Initially, we set:

$$A_0(C_1, C_2, \ldots, C_m) = \begin{cases} 0 & C_j \in [0, B_j], \forall j \in \{1, \ldots, m\}, \\ -\infty & \textit{otherwise}. \end{cases}$$

Moreover we assume that $A_i(C_1, C_2, \ldots, C_m) = -\infty$ for any case when $C_j < 0$ for some $j \in \{1, \ldots, m\}$.

In the following recursive formula we only consider $C_j \in [0, B_j]$, $j \in \{1, \ldots, m\}$ such that $C_1 \leq C_2 \leq \cdots \leq C_m$. We consider capacities in the decreasing order of periods. Before examining a new item i, we set $A_{i-1}(C_1, C_2, \ldots, C_m) = A_{i-1}(C'_1, C'_2, \ldots, C'_{m-1}, C_m)$ where $C'_j = \min\{C_j, C_{j+1}\}$, for all remaining $(C_1, C_2, \ldots, C_m)$. Then, the recursive condition on item i is as follows:

$$A_i(C_1, C_2, \ldots, C_m) = \max \begin{cases} A_{i-1}(C_1, C_2, \ldots, C_m) \\ A_{i-1}(C_1 - w_i, C_2 - w_i, \ldots, C_m - w_i) + p_{i,1} \\ A_{i-1}(C_1, C_2 - w_i, \ldots, C_m - w_i) + p_{i,2} \\ \cdots, \\ A_{i-1}(C_1, C_2, \ldots, C_{m-1}, C_m - w_i) + p_{i,m} \end{cases}$$

The value of optimal solution is then given from $A_n(B_1, B_2, \ldots, B_m)$. Moreover, since $B_j \leq B_m$ for all j and $m \leq m'$, we obtain a time complexity of $O(nm' {B_m}^{m'})$ time. □

4.4.2. The case of an unbounded number of periods

For the case of an unbounded number of periods m, we show that the 3-Partition problem can be reduced to a special case of P^3.

The P^3 decision problem. Any solution of P^3 can be presented as an assignment of items to m periods. Let $N_t = \{i | x_{i,t} = 1\}$ be the subset of items assigned to period t. The capacity constraints are $\sum_{t=1}^{j} \sum_{i \in N_t} w_i \leq B_j$, for all $j \in M$. Then, an instance can be defined as follows:

Instance. We have a finite set of items $\{1, 2, \ldots, n\}$, positive integers V, m, and an increasing sequence $B_1, B_2, \ldots, B_m$. For each item i, some positive weight w_i and a set of positive profits $p_{i,1}, p_{i,2}, \ldots, p_{i,m}$ are given.

Question. Do m disjoint subsets $N_1, N_2, \ldots, N_m$ of items exist such that $\sum_{t=1}^{m} \sum_{i \in N_t} p_{i,t} \geq V$ and for all $j \in \{1, \ldots, m\}$ we have $\sum_{t=1}^{j} \sum_{i \in N_t} w_{i,t} \leq B_j$?

The 3-Partition decision problem. 3-Partition is known to be a strongly NP-hard problem ([10]). The decision problem can be defined as follows:

Instance. We have a finite set $\mathrm{A} = \left\{a_1, a_2, \ldots, a_{3q}\right\}$ of $3q$ numbers, a positive integer bound B such that for each $a_i \in A$, $B/4 < a_i < B/2$ and $\sum_{i=1}^{3q} a_i = qB$.

Question. Can A be partitioned into q disjoint sets $A_1, A_2, \ldots, A_q$ such that for any t, $1 \leq t \leq q$, we have $|A_t| = 3$ and $\sum_{a_i \in A_t} a_i = B$?

Theorem 2. *The P^3 decision problem is NP-complete in the strong sense.*

Proof. First, the P^3 decision problem is clearly in NP, because a nondeterministic algorithm needs only to guess the subsets N_t of items of N and check in polynomial time that the cumulative

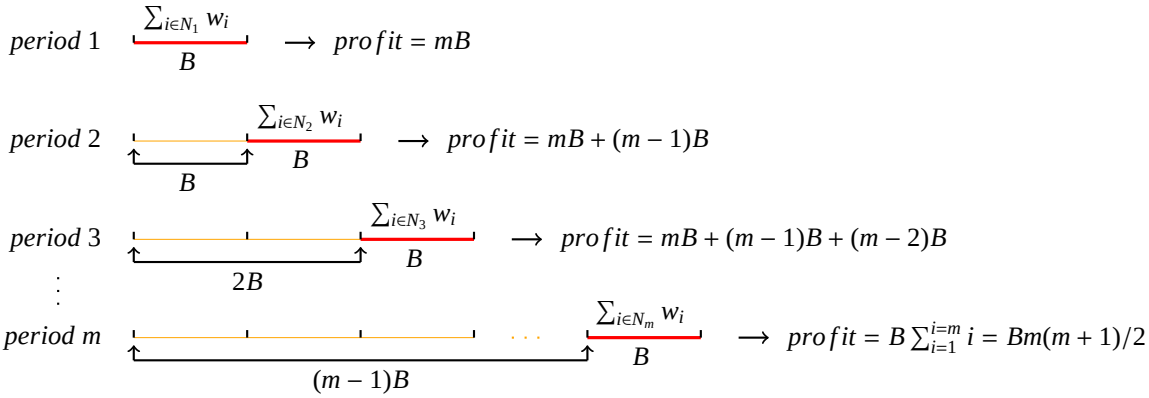

Figure 2: Distribution of items among periods on P^3 corresponding to the 3-Partition instance.

profit for all items in all these subsets is greater than or equal to V.

An instance of P^3 can be built from an instance of 3-Partition in polynomial time as follows. Given an arbitrary instance I of 3-Partition, an instance $f(I)$ of P^3 can be defined as follows: the number of periods $m = q$; the number of items $n = 3q$; $w_i = a_i$, for all $i \in N$; profit $p_{i,t} = (m + 1 - t)a_i$ for all $i \in N$ and $t \in M$; capacity $B_j = jB$; $V = B\sum_{t=1}^{m} t = Bm(m+1)/2$. Note here that $p_{i,t}$ values are then non-increasing for $t \in \{1, ..., M\}$ and integers.

Let us first suppose that the answer to I is *yes*, which means that A can be partitioned into q disjoint subsets $A_1, A_2, \ldots, A_q$ such that for all $1 \leq t \leq q$, $|A_t| = 3$ and $\sum_{a_i \in A_t} a_i = B$. Then, for $f(I)$, let $N_t = \{i | a_i \in A_t\}$, for all $t \in M$. It can easily be shown that:

1. For $1 \leq j \leq m$, $\sum_{t=1}^{j} \sum_{i\in N_t} w_i = \sum_{t=1}^{j} \sum_{i\in N_t} a_i = \sum_{t=1}^{j} B = jB \leq B_j$. This ensures that the assignment is valid;
2. The total profit $\sum_{t=1}^{m} \sum_{i\in N_t} p_{i,t} = \sum_{t=1}^{m} \sum_{i\in N_t} (m + 1 - t)a_i = B\sum_{t=1}^{m} (m + 1 - t) = B\sum_{t=1}^{m} t = Bm(m + 1)/2 = V$.

This shows that the answer to $f(I)$ is also *yes*.

Inversely, let us suppose that the answer to $f(I)$ is *yes*, meaning that there exists an assignment

of items such that the total profit $\sum_{t=1}^{m}\sum_{i\in N_t} p_{i,t} \geq V$, where N_t contains the items assigned to period t. Given that $\sum_{t=1}^{j}\sum_{i\in N_t} w_i \leq B_j = jB$ for all $j \in M$, it can be shown by recurrence on the number of periods j that $\sum_{t=1}^{j}\sum_{i\in N_t} p_{i,t} \leq B\sum_{t=1}^{j}(m+1-t) = B\sum_{t=m+1-j}^{m} t$ for any $j \in M$. Thus, we have for $j = m$, $\sum_{t=1}^{m}\sum_{i\in N_t} p_{i,t} \leq B\sum_{t=1}^{m} t = V$. Combining this with $\sum_{t=1}^{m}\sum_{i\in N_t} p_{i,t} \geq V$, we obtain $\sum_{t=1}^{m}\sum_{i\in N_t} p_{i,t} = V$. Notice here that this maximum total profit is only reached when $\sum_{i\in N_t} w_i = B$ for all $t \in M$. This is obvious since otherwise any capacity non used at some period j but used later will bring less profit and will not allow to reach the value V. Furthermore, since $B/4 < w_i < B/2$ and $\sum_{i\in N_t} w_i = B$ for all $t \in M$, there can only be three items assigned in each period t. Last, we put in A_t all items assigned to period t, that is N_t, and we have a valid 3-Partition. Hence, the answer to I is also *yes*.

It can be concluded that the P^3 decision problem is NP-complete in the strong sense, which means that the original P^3 problem is strongly NP-hard. This proof is illustrated in Figure 2. □

4.5. Complexity of MPR

From above, we can now easily deduce the complexity for the problem MPR.

Theorem 3. *The MPR problem is strongly NP-hard.*

Proof. Given that the P^3 problem, which is a special case of MPR, is strongly NP-hard, it follows that MPR is also strongly NP-hard. □

We will go further and show that a method that solves P^3 would solve any instance of MPR. Hence we need to demonstrate that any instance I of the MPR problem can be reduced to an instance of the P^3 problem. For this we propose a preprocessing procedure for profit values and show that any instance I of the MPR problem can be reduced to an instance of the P^3 problem by applying this preprocessing procedure. This procedure, which we shall refer to as

μ, is described in Algorithm 1 and runs in $O(nm)$. Its main objective is to convert the $p_{i,j}$ values so that they are non-increasing over the time horizon. This may be achieved as follows, the $p'_{i,j}$ values representing the profit values in the new instance $\mu(I)$:

Algorithm 1: Procedure profit transformation μ

Data: an instance I
for $i \in N$ **do**
  $p'_{i,m} \leftarrow p_{i,m}$;
  for $j \leftarrow m-1$ *to* 1 **do**
    $p'_{i,j} \leftarrow \max(p_{i,j}, p'_{i,j+1})$

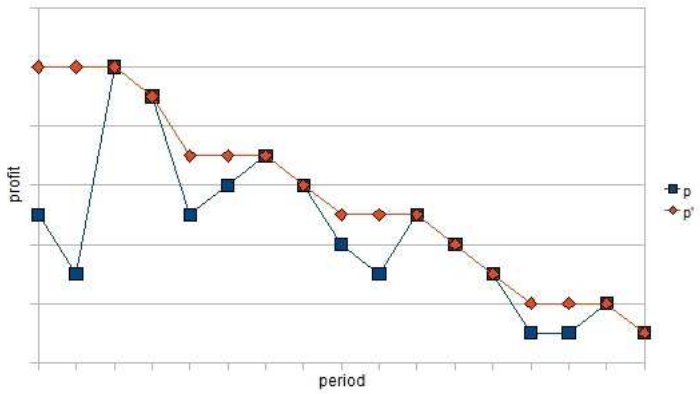

Figure 3: An example of the profit transformation with the preprocessing procedure

In Figure 3, curve p gives an example of original profit values for one item in instance I. Curve p' shows the corresponding profit values (*non-increasing*) for this item following the preprocessing procedure.

Proposition 4. *Any optimal solution for instance I is also optimal for instance $\mu(I)$, and from any optimal solution for instance $\mu(I)$ an optimal solution can be constructed for instance I.*

Proof. The proof is straightforward. Any feasible solution for instance I is outperformed by some other solution obtained by delaying the decision to a later period that yields a larger profit.

Note that the latter solution is also a solution for instance $\mu(I)$. On the other hand, some optimal solution for instance $\mu(I)$ gives rise to an optimal solution for instance I obtained by delaying the decision for any item to the latest period yielding an identical profit. □

As a consequence, since $\mu(I)$ is also an instance of P^3, any algorithm solving problem P^3 can be used to solve problem P.

In particular, when profit values $p_{i,j}$ are *non-decreasing* during the time horizon for any item i, the above procedure will result in a system of m identical constraints. Hence, all decisions may be delayed until the last period and this case can be reduced to a simple knapsack problem formulated as follows: max $\sum_{i \in N} p_{i,m} x_{i,m}$ subject to $\sum_{i \in N} w_i x_{i,m} \leq B_m$. Consequently, this case may be solved in pseudo-polynomial time $O(nB_m)$.

5. Heuristic methods and computational results for MPR

Recall first that any instance of MPR can be simplified to an instance of P^3. In this section we present methods specifically designed for solving the P^3 problem, bearing in mind that since this problem is strongly *NP*-hard, optimal solutions are unlikely to be found for large instances. Notice also that this solution strategy was initially intended to be integrated in a simulation tool, and computational efficiency in terms of CPU time was therefore an important requirement. Typically a CPU time less than 30 seconds for this application were required. In order to evaluate the effectiveness of the above heuristics, we tested instances of both practical and theoretical types.

5.1. Heuristics

In this section, two well-known heuristics from the literature, namely the Dantzig greedy heuristic proposed for the knapsack problem and the Martello&Toth heuristic proposed for GAP,

adapted by Dao et al. [3] for MPR, are recalled. Next, two new heuristic methods, called Back&Forth and CombMT&D, are described.

The Dantzig heuristic (DZ). The Dantzig heuristic is a greedy type heuristic proposed for the unbounded knapsack problem in [11]. The principle was adapted for the MPR problem in [3]. First a list containing all ratios $\{p_{i,j}/w_i\}$ for each item i and period j sorted in non-increasing order is created. At each iteration of the algorithm, the first ratio $\{p_{i,t}/w_i\}$ which corresponds to a feasible assignment (according to the previous decided assignments) of item i in period t is chosen. Next, all ratios $\{p_{i,j}/w_i\}$, for all $j \in \{1, \ldots, m\}$ are removed from the list. The process is repeated until no more items can be chosen.

The Martello&Toth heuristic (MT). The Martello&Toth heuristic is also a greedy type heuristic proposed by Martello and Toth in 1990 for GAP [5]. The principle was also adapted and presented in [3]. In this algorithm items are assigned iteratively. At each iteration we assign as soon as possible the item i maximizing the difference $\{(p_{i,t} - p_{i,t+1})/w_i\}$ where t is the earliest period in which i is assignable. Items which are assignable only at the last period are not considered at this step of the algorithm. At the end of the algorithm, when only such items remain, they are assigned iteratively in decreasing order of $\{p_{i,m}/w_i\}$.

Discussion. Clearly there are two aspects of MPR that an approximate solution method should take into account. The first one is related with the rate profit-weight and the second is concerned with the variation of profit over time. With respect to above, and as already confirmed by the numerical results, the Dantzig heuristic captures very well the benefits coming when prefering items with high profit-weight ratios, but it fails for scenarios with important changes in profits over time. For the later ones, Martello and Toth heuristic could be a better choice but, still

this heuristic encounter difficulties as it neglects the importance of items with high profit-weight ratios.

These drawbacks are illustrated in the following example with 3 items and 3 periods for which the optimal total profit is 22: $B_1 = 1$, $B_2 = 2$, $B_3 = 3$, $w_1 = w_2 = w_3 = 1$, $p_{11} = 11$, $p_{12} = 10$, $p_{13} = 1$, $p_{21} = 9$, $p_{22} = 6$, $p_{23} = 1$, $p_{31} = 8$, $p_{32} = 3$, $p_{33} = 3$.
As it can be verified, *DZ* assigns item 1 to period 1, item 2 to period 2 and item 3 to period 3. The obtained total profit is then $p_{11} + p_{22} + p_{33} = 20$. The drawback of *DZ* here is to prefer assigning item 1 first without taking into account that item 2 will be largely devaluated from period 1 to period 2. The assignment of *MT* is item 3 to period 1, item 1 to period 2 and item 2 to period 3. The obtained total profit is then $p_{31} + p_{12} + p_{23} = 19$. The drawback of *MT* is to prefer assigning item 3 on period 1 because of its high devaluation from period 1 to period 2. The consequence is to neglect too much high rate profit-weight items.

These drawbacks are our motivation to look for heuristics that take into account both aspects, as the two ones given below:

The Back&Forth heuristic. This heuristic is inspired by the Dantzig heuristic for MPR. It is implemented in two steps. The main idea behind this heuristic is to assign temporarily the *best* items in the first step and to improve the assignments in the second. In contrast to the Dantzig heuristic, an item is not assigned to the earliest possible period, but it is assigned temporarily to a later period where the profit is not significantly diminished. In the second step the assignment is moved forward if capacity constraints allow. During the algorithm, B'_j gives the remaining capacity values, with respect to assigned items. Initially, $B'_j = B_j$, for all $j \in \{1, \ldots, m\}$. For each unassigned item i, we denote as epp_i the earliest possible period to which i can be assigned, according to the current capacity constraints $B'_j, j \in M$, i.e. $epp_i = \text{argmin}_j\{w_i \leq B'_t, \forall t \in$

$\{j, \ldots, m\} | j \in M\}$. Otherwise, if $w_i > B'_m$ (i cannot be assigned to any of the periods), set $epp_i = m + 1$. Heuristic Back&Forth proceeds as follows:

1. Let $\bar{N}$ be the collection of the unassigned items. Find the item $z = \text{argmax}_i\{p_{i,epp_i}/w_i | i \in \bar{N} \wedge epp_i \leq m\}$. Next, determine the period t_z ($t_z \geq epp_z$) in which item z will be temporarily assigned. t_z corresponds to the largest period where the reduced profit ($p_{z,epp_z} - p_{z,t_z}$) does not exceed a certain value $\lambda(p_{z,epp_z} - p_{z,m}), 0 \leq \lambda \leq 1$. Remove z from $\bar{N}$ and update all concerned B'_j. Update epp_i, for all $i \in \bar{N}$. Repeat this step until the assignment is no longer possible;
2. Let $N_0 = N \setminus \bar{N}$ be the collection of items assigned in the first step. Find item $x = \text{argmax}_i\{(p_{i,epp_i} - p_{i,S_i})/w_i | i \in N_0 \wedge epp_i \neq S_i\}$, where S_i is the current period in which item i is assigned. Assign item i to period epp_i and update all concerned B'_j. Update epp_i, for all $i \in N_0$. Repeat this step until no further items can be brought forward.

The CombMT&D heuristic. This heuristic is a combination of the Martello&Toth and Dantzig heuristics. It is implemented in two steps. In the first step, items are iteratively assigned to periods until $m - 1$, respecting a particular order combining the Dantzig and Martello&Toth heuristics. In the second step, the unassigned items are assigned iteratively to the last period m according to the Dantzig heuristic. Each assignment satisfies the capacity constraints. Heuristic CombMT&D proceeds as follows:

1. Let $\bar{N}$ be the collection of unassigned items. Find item $z = \text{argmax}_i\{(p_{i,epp_i}/w_i)^\alpha (p_{i,epp_i} - p_{i,epp_i+1})/w_i | i \in \bar{N} \wedge epp_i \leq m - 1\}$ and assign item z to period epp_z. Remove item z from $\bar{N}$ and update all concerned B'_j. Update epp_i, for all $i \in \bar{N}$. Repeat this step until the assignment is no longer possible;
2. Consider the set $\bar{N}$ of remaining unassigned items. Find item $x = \text{argmax}_i\{(p_{i,m}/w_i) | i \in$

$\bar{N} \wedge epp_i \leq m\}$ and assign it to period m. Update B'_m et epp_i, for all $i \in \bar{N}$. Repeat this step until the assignment is no longer possible.

It can be easily deduced that the complexity time of each of them is $O(n^2m)$. Moreover, for the same values of α and λ in the numerical results given in the next section (*i.e.* $\alpha = 3$ and $\lambda = 0.25$), both heuristics provide an optimal solution for the previous example by assigning item 2 to period 1, item 1 to period 2 and item 3 to period 3, leading to a total profit of $p_{21} + p_{12} + p_{33} = 22$.

5.2. Computational experiments

In order to evaluate the effectiveness of the above heuristics, we tested instances of both practical and theoretical types. The main difference between practical and theoretical types is that for practical instances the profits for an item i correspond to a function which attains its maximum value at a certain period r_i and then progressively decreases over the subsequent periods. Therefore, in accordance with Proposition 4, item i will not be assigned before r_i. This reduces the number of periods to which it is possible to assign item i, thus making instances easier to solve, and this is confirmed in Table 1 by the small gap values of this practical type compared with other types on instances of identical or similar sizes. Otherwise, for theoretical instances, in order to test a more general case and evaluate the robustness of each heuristic, the maximum profit for all items is deemed to occur in the first period, and the manner in which profits decrease over subsequent periods is random. For each type and for each size (n, m), 20 instances are generated.

Four types of instances are tested. Type 1 are practical ones. For each item i, the highest profit is attained at a certain period r_i. After period r_i, profits decrease in accordance with certain rules (see the details below). The preprocessing procedure is applied to transform an instance of this type to an instance of P^3(profits in non-increasing order from the first period). The size of tested instances for this type are $n = [500, 1000, 5000, 10000]$ and $m = [15, 20, 25]$. Type

2, type 3 and type 4 are theoretical types. For type 2 and type 3, profits $p_{i,j}$ decrease randomly with a certain distribution for each item i. For type 4, profits $p_{i,j}$ decrease linearly from the first period with different slopes for each item i. The size of tested instances for the last three types are $n = [50, 100, 500, 1000]$ and $m = [10, 30, 50]$. In addition, two capacity scenarios are considered for each type of instance, namely *similar capacities* and *dissimilar capacities* (see the details below). Notice that in order to stick to the application, the practical instances have different values for m and n.

Instance type 1 (practical). R is set to 1000 (R takes the same value for all types). For each item i, the weight w_i is generated in $[1, R]$ based on a uniform distribution. A parameter r_i is generated in $[1, m]$ based on uniform distribution. r_i represents the period where i has the highest profit. This highest profit p_{i,r_i} is generated in $[1, R]$ based on a uniform distribution. For each item i, a variable b is first randomly chosen from among three values $\{0.01, 0.015, 0.02\}$. Then, profits $p_{i,j}$ are set as follows: for all $j \in [1, r_i - 1]$, $p_{i,j} = 0$; For all $j \in [r_i + 1, m]$, $p_{i,j} = p_{i,j-1} - p_{i,r_i} * (b/2 + b * (j - r_i)/10)$. As we can see from this equation, the decreased profit between two consecutive periods augments along with the increase of j, since the slope $b/2 + b * (j - r_i)/10$ increases along with the increase of j. Surely, $p_{i,j}$ are guaranteed to be positive in all cases.

With respect to the capacity values, we consider that the total capacity B_m is around $\sum_{i \in N} w_i/2$. *similar capacities* and *dissimilar capacities* [5] are two different capacity scenarios that we considered for all types of instances. For *similar capacities*, the capacity B_1 and all augmented capacities $(B_j - B_{j-1})$ for all $j \in \{2, \ldots, m\}$ are generated in $[0.4 \sum_{i \in N} w_i/m,\ 0.6 \sum_{i \in N} w_i/m]$ based on uniform distribution. For *dissimilar capacities*, B_m is first set to $\sum_{i \in N} w_i/2$ and then B_j, $\{j \in 1, \ldots, m-1\}$ is generated based on B_{j+1} with the following distribution. There is a 2%

probability that $B_j = B_{j+1}$, a 3% probability that $B_j \in [0.98B_{j+1}, B_{j+1})$, a 15% probability that $B_j \in [0.95B_{j+1}, 0.98B_{j+1})$, a 30% probability that $B_j \in [0.9B_{j+1}, 0.95B_{j+1})$, a 30% probability that $B_j \in [0.85B_{j+1}, 0.9B_{j+1})$, a 10% probability that $B_j \in [0.8B_{j+1}, 0.85B_{j+1})$, a 5% probability that $B_j \in [0.65B_{j+1}, 0.8B_{j+1})$, a 2% probability that $B_j \in [0.4B_{j+1}, 0.65B_{j+1})$, a 2% probability that $B_j \in [0.3B_{j+1}, 0.4B_{j+1})$, and a 1% probability that $B_j \in [0.1B_{j+1}, 0.3B_{j+1})$.

Instance type 2. Values w_i, $p_{i,1}$ are generated in $[1, R]$ based on a uniform distribution (w_i and $p_{i,1}$ are uncorrelated as proposed in [12]). For all $j \in \{2, \ldots, m\}$, $p_{i,j}$ is generated based on $p_{i,j-1}$ with the following distribution. There is a 10% probability that $p_{i,j} = p_{i,j-1}$, a 15% probability that $p_{i,j} \in [0.95p_{i,j-1}, p_{i,j-1})$, a 25% probability that $p_{i,j} \in [0.9p_{i,j-1}, 0.95p_{i,j-1})$, a 20% probability that $p_{i,j} \in [0.8p_{i,j-1}, 0.9p_{i,j-1})$, a 10% probability that $p_{i,j} \in [0.7p_{i,j-1}, 0.8p_{i,j-1})$, a 10% probability that $p_{i,j} \in [0.6p_{i,j-1}, 0.7p_{i,j-1})$, and a 10% probability that $p_{i,j} \in [0.3p_{i,j-1}, 0.6p_{i,j-1})$.

Instance type 3. Values w_i are generated in the same way as for type 2, while $p_{i,1} = w_i + R/10$ (w_i and $p_{i,1}$ are strongly correlated, see [12]). Values $p_{i,j}$ for all $j \in \{2, \ldots, m\}$ are generated in the same way as for type 2.

Instance type 4. Values w_i, $p_{i,1}$ are generated in $[1, R]$ based on a uniform distribution (w_i and $p_{i,1}$ are uncorrelated); For each item i, a variable b is first randomly chosen from among three values $\{1/m, 1/(2m), 1/(3m)\}$. Then, for the other profits we take $p_{i,j} = p_{i,1} - p_{i,1}(j-1)b$, for all $j \in \{2, \ldots, m\}$ (profit values decrease linearly).

Experimental results. All tests were run on a PC with a 2 GHz microprocessor and 2 GB of Ram. To evaluate the performances of our heuristics, we have measured the gap with the optimal solution (or the best integer solution) provided by the MIP. We have run the algorithm of CPLEX 11 with a time limit fixed to 600 seconds to obtain optimal solutions or near optimal solutions

of the instances. There are 54.3% of instances which are not solved to optimality at the end of the time limit. However, the results of MIP are generally very close to the optimal solutions. Based on the informations provided by CPLEX, the average final gap of MIP over all the tested instances is 0.04%. The results of the heuristics are then evaluated by the best integer solution obtained by the MIP algorithm. While it is obvious that the results of the heuristics are not really comparable with these results, note that if we let to CPLEX the same time which is taken in the worst case by the heuristics, CPLEX does not provide better solutions in most of cases.

The parameters λ and α used in our two heuristics have been chosen empirically. We have set λ to 0.25 and α to 3. In Table 1, we provide the average relative distance ("gap") between the result of each heuristic and the best integer solution of MIP as well as the average CPU time ("time"). The gap is computed as *(the best integer solution of MIP - the value of the heuristic)/(the best integer solution of MIP)*. The CPU time in the tables are in *milliseconds*. The 0.0 values for *time* means that the CPU time is less than 0.1 millisecond. *MT* represents the Martello&Toth heuristic; *DZ* represents the Dantzig heuristic; *BF* represents the Back&Forth heuristic and *Comb* represents the CombMT&D heuristic.

We tested all methods on 960 instances (240 instances for each type). Table 1 provides the gap values and the CPU time for instances of each size and of each type as well as the average values by rows and by column ("AVG"). Table 2 provides for each type of instance, the percentage of instances for which each heuristic obtain the best results among all the tested heuristics ("best") and the average gap. Notice that for some instances, more than one heuristic yields the best result. During our experiments, we have noticed that there is no relevant difference of performance (gap values and CPU times) of the algorithms with respect to the nature of capacities (similar and dissimilar) of generated instances. That is why all results are merged and shown in a common table.

As shown in the tables, the obtained results demonstrate the effectiveness of the proposed heuristics. Both our heuristics often perform better than the two other adapted heuristics. Although our heuristics are comparable with each other in terms of the number of successful instances and the relative distances with best integer solutions of MIP for instances of theoretical types, Back&Forth is more robust for all tested instances, especially for instances of the practical type. Moreover, combining the Back&Forth and CombMT&D heuristics allows the best results to be obtained in more than 93% of instances of similar and dissimilar capacities. On the other hand, the Dantzig heuristic performs quite well for a large number of instances while Martello&Toth encounters some difficulties.

We notice that the CPU times for all heuristics are very satisfactory for the application in hand. Even for the BF heuristic, which takes the largest computational times among them, the CPU times for the largest instances (10000 items and 25 periods) is at most 12 seconds. We believe that the proposed heuristics achieve a good trade-off between the calculation time and the performance in terms of quality of the provided solutions.

6. Conclusion

This paper investigates a new knapsack variant, the Multi-Period Renewal equipment problem. This problem can be seen as a simplified deterministic version of conventional renewal problems. Several special cases of this problem are enumerated and studied. Analysis of complexity have shown the NP-hardness in the strong sense of the general case, while some special cases can be handled in pseudo polynomial and polynomial time. As a perspective of this work we propose to investigate a general version of the MPR problem where item weights vary over time.

References

[1] R. Fenner, Approaches to sewer maintenance: a review, Urban Water 2(4) (2000) 343–356.

[2] D. Cox, Renewal theory, Wiley London, New York, 1962.

[3] T.-T. Dao, A. Nace, D. Nace, X. Cao, The multi-period renewal equipment problem, in: Proceedings of the International Conference on Metaheuristics and Nature Inspired Computing, 2008.

[4] B. Faaland, The multiperiod knapsack problem, Operations Research 29 (3) (1981) 612–616.

[5] S. Martello, P. Toth, Knapsack problems: algorithms and computer implementations, John Wiley & Sons, Inc., New York, NY, USA, 1990.

[6] R. Nauss, The generalized assignment problem, integer programming: theory and practice, Taylor and Francis group, 2006.

[7] T. Oncan, A survey of the generalized assignment problem and its applications, INFOR: Information Systems and Operational Research 45(3) (2007) 123–141.

[8] E. Lin, C. Wu, The multiple-choice multi-period knapsack problem, Journal of the Operational Research Society 55 (2004) 187–197(11).

[9] E. Lin, C. Wu, A dynamic programming approach to the multiple-choice multi-period knapsack problem and the recursive apl2 code, International Journal of Operations and Quantitative Management.

[10] M. R. Garey, D. S. Johnson, Computers and Intractability : A Guide to the Theory of NP-Completeness (Series of Books in the Mathematical Sciences), W. H. Freeman, 1979.

[11] G. Dantzig, Discrete-variable extremum problems, Operations Research 5 (1957) 266–277.

[12] D. Pisinger, Where are the hard knapsack problems?, Computers & Operations Research 32 (9) (2005) 2271–2284.

Type	m	Algo	n=500		n=1000		n=5000		n=10000		AVG	
			gap	time	gap	time	gap	time	gap	time	gap	time
1	15	MT	2.34	11.70	2.47	44.40	2.40	1089.95	2.47	5487.80	2.42	1658.46
		DZ	0.19	9.55	0.21	38.30	0.22	953.60	0.17	4748.90	0.20	1437.59
		BF	0.09	21.30	0.15	83.70	0.14	2143.10	0.12	9320.35	0.13	2892.11
		Comb	0.42	13.40	0.44	51.75	0.41	1290.35	0.33	6471.00	0.40	1956.63
	20	MT	3.38	14.70	3.11	51.30	3.04	1123.85	3.25	5719.15	3.20	1727.25
		DZ	0.40	9.60	0.39	38.00	0.45	973.85	0.39	4752.40	0.41	1443.46
		BF	0.19	24.40	0.23	94.30	0.24	2328.50	0.22	10534.40	0.22	3245.40
		Comb	0.44	14.10	0.44	54.60	0.49	1335.80	0.41	6860.45	0.44	2066.24
	25	MT	4.52	12.35	4.19	48.45	3.95	1140.65	4.12	6072.40	4.19	1818.46
		DZ	0.72	9.65	0.67	38.35	0.71	987.15	0.66	4797.20	0.69	1458.09
		BF	0.33	26.30	0.37	103.55	0.39	2593.00	0.32	12028.30	0.35	3687.79
		Comb	0.70	13.90	0.64	54.55	0.62	1360.85	0.51	7350.30	0.62	2194.90
	AVG	MT	3.41	12.92	3.26	48.05	3.13	1118.15	3.28	5759.78	3.27	1734.73
		DZ	0.44	9.60	0.42	38.22	0.46	971.53	0.41	4766.17	0.43	1446.38
		BF	0.20	24.00	0.25	93.85	0.26	2354.87	0.22	10627.68	0.23	3275.10
		Comb	0.52	13.80	0.51	53.63	0.50	1329.00	0.42	6893.92	0.49	2072.59
			n=50		n=100		n=500		n=1000		AVG	
2	10	MT	5.90	0.25	6.11	0.60	6.19	11.70	5.93	43.15	6.03	13.93
		DZ	5.07	0.15	4.63	0.60	4.64	9.35	5.11	36.30	4.86	11.60
		BF	2.75	0.30	2.86	0.75	2.62	18.25	2.89	68.85	2.78	22.04
		Comb	3.28	0.30	2.31	0.60	1.97	13.55	1.93	52.00	2.37	16.61
	30	MT	9.70	0.45	7.22	0.95	7.20	12.15	7.19	46.75	7.83	15.08
		DZ	5.72	0.10	5.40	0.65	5.86	9.40	5.65	36.90	5.66	11.76
		BF	3.63	0.55	2.65	1.55	2.82	31.05	2.88	112.55	2.99	36.43
		Comb	3.85	0.15	2.89	0.55	2.26	12.25	2.19	48.05	2.80	15.25
	50	MT	7.96	0.40	7.22	1.00	7.52	11.50	6.95	37.80	7.41	12.68
		DZ	3.23	0.15	4.77	0.55	5.13	9.05	5.13	34.50	4.57	11.06
		BF	2.31	1.10	2.98	2.15	3.05	35.75	2.82	135.80	2.79	43.70
		Comb	2.82	0.30	2.92	0.40	2.50	10.30	2.27	37.60	2.63	12.15
	AVG	MT	7.85	0.37	6.85	0.85	6.97	11.78	6.69	42.57	7.09	13.89
		DZ	4.67	0.13	4.94	0.60	5.21	9.27	5.30	35.90	5.03	11.48
		BF	2.90	0.65	2.83	1.48	2.83	28.35	2.86	105.73	2.85	34.05
		Comb	3.32	0.25	2.71	0.52	2.24	12.03	2.13	45.88	2.60	14.67
3	10	MT	12.92	0.25	12.30	0.70	13.41	11.45	12.47	42.20	12.78	13.65
		DZ	6.49	0.20	5.55	0.40	5.67	9.15	5.85	37.95	5.89	11.93
		BF	4.43	0.25	3.30	0.90	3.57	18.45	3.68	78.65	3.75	24.56
		Comb	4.41	0.20	3.95	0.65	3.48	14.10	3.40	53.70	3.81	17.16
	30	MT	15.90	0.50	16.71	0.90	16.16	11.40	15.77	43.15	16.13	13.99
		DZ	5.61	0.05	6.92	0.50	6.41	9.35	6.48	35.70	6.35	11.40
		BF	3.82	0.55	4.08	1.35	3.35	28.80	3.29	113.20	3.64	35.98
		Comb	4.86	0.15	3.88	0.50	3.92	11.90	3.86	44.85	4.13	14.35
	50	MT	14.22	0.55	14.25	1.10	15.99	10.40	16.59	34.90	15.26	11.74
		DZ	5.65	0.05	5.70	0.50	5.11	8.65	5.38	33.45	5.46	10.66
		BF	4.23	0.75	4.34	2.15	3.29	33.90	3.16	128.75	3.76	41.39
		Comb	3.77	0.10	3.80	0.60	3.47	9.35	3.50	34.90	3.64	11.24
	AVG	MT	14.35	0.43	14.42	0.90	15.19	11.08	14.94	40.08	14.72	13.13
		DZ	5.92	0.10	6.06	0.47	5.73	9.05	5.90	35.70	5.90	11.33
		BF	4.16	0.52	3.91	1.47	3.40	27.05	3.38	106.87	3.71	33.98
		Comb	4.35	0.15	3.87	0.58	3.63	11.78	3.59	44.48	3.86	14.25
4	10	MT	5.49	1.25	5.67	0.55	5.39	12.45	5.44	47.60	5.50	15.46
		DZ	3.33	0.20	3.01	0.55	3.31	9.25	3.18	36.45	3.21	11.61
		BF	2.52	0.25	2.21	0.75	2.10	19.50	1.93	76.95	2.19	24.36
		Comb	2.16	0.15	1.57	0.55	1.40	14.20	1.32	59.40	1.61	18.58
	30	MT	8.39	0.90	9.05	2.60	8.54	15.50	8.61	52.50	8.65	17.88
		DZ	3.75	0.20	3.93	0.30	4.28	9.60	4.25	37.40	4.05	11.88
		BF	2.29	0.55	2.45	1.65	2.61	34.70	2.71	133.50	2.52	42.60
		Comb	2.01	0.40	2.04	0.70	2.00	15.00	1.97	58.40	2.00	18.63
	50	MT	9.79	0.25	10.68	0.95	10.16	17.75	10.33	53.05	10.24	18.00
		DZ	3.27	0.25	3.46	0.40	4.11	9.85	3.97	38.20	3.70	12.18
		BF	2.03	1.00	2.20	2.75	2.60	47.30	2.46	178.20	2.32	57.31
		Comb	2.36	0.20	1.98	0.80	1.99	15.10	1.96	58.90	2.07	18.75
	AVG	MT	7.89	0.80	8.47	1.37	8.03	15.23	8.13	51.05	8.13	17.11
		DZ	3.45	0.22	3.47	0.42	3.90	9.57	3.80	37.35	3.66	11.89
		BF	2.28	0.60	2.29	1.72	2.44	33.83	2.37	129.55	2.34	41.43
		Comb	2.18	0.25	1.86	0.68	1.80	14.77	1.75	58.90	1.90	18.65

Table 1: Gap values and CPU times (ms) for instances of each size and of each type

Type	MT		DZ		BF		Comb		Max(BF, Comb)	
	best	gap	best	gap	best	gap	best	gap	best	gap
1	1.25	3.27	24.58	0.43	74.58	0.23	10.42	0.49	85.00	0.21
2	4.58	7.09	7.08	5.03	32.92	2.85	61.67	2.60	93.33	2.16
3	1.67	14.72	5.42	5.90	51.25	3.71	46.25	3.86	95.42	3.06
4	1.25	8.13	0.42	3.66	18.33	2.34	80.00	1.90	98.33	1.82
Overall	2.19	8.30	9.38	3.75	44.27	2.29	49.58	2.21	93.02	1.81

Table 2: The best heuristics and the average gap values