A Partition Algorithm for 0-1 Unconstrained Hyperbolic Programs

Anass Nagih *Gérard Plateau*

Université Paris 13, LIPN, UPRES-A CNRS 7030
Institut Galilée, 99, Avenue J-B Clément
93430 Villetaneuse, France
fax: +33 1 48 26 07 12
{anass.nagih, gerard.plateau}@lipn.univ-paris13.fr

Abstract
The 0-1 unconstrained hyperbolic program is a key tool for solving Lagrangean decomposition or relaxation of the 0-1 constrained hyperbolic programs. It has to be solved numerous times in any subgradient algorithm. In this paper, we propose a revisited partition algorithm whose computation times dominate those of the best known linear time complexity algorithm.

Keywords: Hyperbolic Programming, 0-1 Integer Programming.

1 Introduction

Linear or nonlinear fractional programming problems, with discrete or continuous variables, occur in various fields such as computer science [11], mathematical programming [5, 10, 19]), stochastic programming [2], and economies [6, 12, 20].

We are concerned in this paper with the 0-1 unconstrained hyperbolic program:

$$(P) \qquad \begin{cases} \max & \dfrac{c_0 + \sum_{j=1}^{n} c_j x_j}{d_0 + \sum_{j=1}^{n} d_j x_j} \\ \\ \text{s.t.} & x_j \in \{0,1\} \quad j = 1, ..., n \end{cases}$$

which is a key tool for solving Lagrangean decomposition or relaxation of the 0-1 constrained hyperbolic programs [14, 16].

Classically we assume that:

- $d_0 + \sum_{j=1}^{n} d_j x_j > 0 \ \ \forall x_j \in \{0,1\}, j = 1, ..., n$

- $c_j > 0, \forall j \in \{1, ..., n\}$ and $d_j > 0, \forall j \in \{0, ..., n\}$.

Note that the second hypothesis is not restrictive since it can be deduced by the following transformations assuming the first hypothesis holds (Hansen et al [11]):

(i) we obtain $d_j \geq 0, j = 1, ..., n$ by replacing x_j by its complement $1 - x_j$, for each j such that $d_j < 0$ (note that the first hypothesis implies that $d_0 > 0$),

(ii) the second hypothesis is then obtained in two steps (by denoting $x^\star$ an optimal solution of (P)):

step 1 : fix $x_j^\star$ at 0 for all j such that $c_j \leq 0$,
step 2 : fix $x_j^\star$ at 1 for all j such that $d_j = 0$.

The first works about problem (P) have been developed by Hammer and Rudeanu in their book devoted to boolean methods in operations research [9]. This problem is NP-hard [11], but for the class of instances where data are nonnegative, polynomial exact algorithms have been proposed by Hammer and Rudeanu [9], Robillard [17] and Hansen et al [11].

Section 2 recalls briefly these algorithms with some basic properties (see [13, 15] for a complete survey devoted to fractional programs). Section 3 deals with a revisited version of the linear time complexity algorithm of Hansen et al. The computational results of section 4 show the efficiency of our algorithm.

2 Algorithms survey

All the algorithms described here are based on the two following fundamental properties:

Property 1:

$$\min\left\{\frac{c_j}{d_j}, \frac{c_k}{d_k}\right\} \leq \frac{c_j + c_k}{d_j + d_k} \leq \max\left\{\frac{c_j}{d_j}, \frac{c_k}{d_k}\right\} \quad \forall j, k \in \{1, ..., n\}.$$

Property 2: **(elimination of variables)**

$$x_j^\star = 0 \quad \text{for all} \quad j \in J = \{1, ..., n\} \quad \text{such that} \quad \frac{c_j}{d_j} \leq \frac{c_0}{d_0}$$

Proof: Let $J_0 = \left\{ j \in J \mid \frac{c_j}{d_j} \leq \frac{c_0}{d_0} \right\}$. The result is deduced directly from property 1 and the relationship

$$\frac{c_0 + \sum_{j \in J} c_j x_j}{d_0 + \sum_{j \in J} d_j x_j} \leq \frac{c_0 + \sum_{j \in J - J_0} c_j x_j}{d_0 + \sum_{j \in J - J_0} d_j x_j}, \quad \forall x \in \{0, 1\}^n. \qquad \boxtimes$$

Remark 1: This elimination of variables is a preprocessing phase for all the algorithms. It is henceforth supposed that data of (P) are such that $\frac{c_j}{d_j} > \frac{c_0}{d_0}$ for all $j \in J = \{1, ..., n\}$.

2.1 Hammer and Rudeanu's algorithm

```
{initialization}
    $J \leftarrow \{1, ..., n\}$;
{begin}
    repeat
        determine $k$ such that $\frac{c_k}{d_k} = \max\left\{\frac{c_j}{d_j} \mid j \in J\right\}$;
        $x_k^\star \leftarrow 1$;   $c_0 \leftarrow c_0 + c_k$;   $d_0 \leftarrow d_0 + d_k$;
        $J_0 \leftarrow \left\{j \in J \mid \frac{c_j}{d_j} \leq \frac{c_0}{d_0}\right\}$;   $J \leftarrow J - J_0$;
        fix $x_j^\star$ at 0 for all $j$ in $J_0$
    until fixation of all variables $\{J = \emptyset\}$
    $\lambda^\star \leftarrow \frac{c_0}{d_0}$   {optimal value of (P)}
{end}
```

Fig. 1: HR68: Algorithm of Hammer and Rudeanu [9]

These authors propose an algorithm, denoted by HR68 (figure 1) with a quadratic time complexity based on the following property:

Property 3 ([9]):

$$x_k^\star = 1 \text{ for all } k \in \{1, ..., n\} \text{ such that } \frac{c_k}{d_k} = \max\left\{\frac{c_j}{d_j} \mid j \in \{1, ..., n\}\right\}.$$

Proof: This result comes from the relationship

$$\frac{c_0 + \sum_{j \in J - \{k\}} c_j x_j}{d_0 + \sum_{j \in J - \{k\}} d_j x_j} \leq \frac{c_0 + c_k + \sum_{j \in J - \{k\}} c_j x_j}{d_0 + d_k + \sum_{j \in J - \{k\}} d_j x_j} \qquad \forall x_j \in \{0, 1\},\ j \in J - \{k\}. \boxtimes$$

2.2 Robillard's algorithm

This algorithm (figure 2) has also a quadratic time complexity. It is based on the following characterization of an optimal solution.

Property 4 ([17]):

Suppose that $\frac{c_0}{d_0} < \frac{c_1}{d_1} \leq \frac{c_2}{d_2} \leq ... \leq \frac{c_n}{d_n}$, let k an integer in $\{1, ..., n\}$ such that:

$$\frac{c_0 + \sum_{j=k}^n c_j}{d_0 + \sum_{j=k}^n d_j} < \frac{c_i}{d_i} \qquad \textit{for all } i \geq k$$

```
{initialization}
    $J \leftarrow \{1, ..., n\}$ ;   $J_0 \leftarrow \emptyset$;   $J_1 \leftarrow \emptyset$;
{begin}
    for  $j = 1$ to $n$  do
        if  $\frac{c_j}{d_j} > \frac{c_0}{d_0}$  then
            $J_1 \leftarrow J_1 \cup \{j\}$ ;   $c_0 \leftarrow c_0 + c_j$ ;   $d_0 \leftarrow d_0 + d_j$ ;
            repeat
                $J_0 \leftarrow \left\{ i \in J_1 \mid \frac{c_i}{d_i} \leq \frac{c_0}{d_0} \right\}$;   $J_1 \leftarrow J_1 - J_0$;
                $c_0 \leftarrow c_0 - \sum_{i \in J_0} c_i$ ;   $d_0 \leftarrow d_0 - \sum_{i \in J_0} d_i$
            until  $J_0 = \emptyset$
        endif
    endfor ;
    for  $j = 1$ to $n$  do
        if  $j \in J_1$  then  $x_j^\star \leftarrow 1$
        else  $x_j^\star \leftarrow 0$
        endif
    endfor
{end}
```

Fig. 2: Algorithm of Robillard [17]

and

$$\frac{c_0 + \sum_{j=k}^{n} c_j}{d_0 + \sum_{j=k}^{n} d_j} \geq \frac{c_i}{d_i} \qquad \textit{for all } i < k$$

then $x^\star \in \{0,1\}^n$ defined by: $x_j^\star = 1$ if $j \geq k$
$x_j^\star = 0$ if $j < k$

is an optimal solution of (P) whose optimal value is given by $\dfrac{c_0 + \sum_{j=k}^{n} c_j}{d_0 + \sum_{j=k}^{n} d_j}$.

2.3 Hansen-Poggi-Ribeiro's algorithm

The relationships of property 4 binding the value and an optimal solution of a 0-1 unconstrained hyperbolic program can be rewritten in the following form [11]:

Property 5:

If we denote $\lambda^\star$ the optimal value of (P), then any optimal solution $x^\star$ of (P) is such that for all $j \in \{1, ..., n\}$:

$x_j^\star = 1$ if $\frac{c_j}{d_j} > \lambda^\star$

$x_j^\star = 0$ if $\frac{c_j}{d_j} < \lambda^\star$

$x_j^\star = 0$ or 1 if $\frac{c_j}{d_j} = \lambda^\star$.

```
{initialization}
    $\widetilde{c} \leftarrow c_0$;   $\widetilde{d} \leftarrow d_0$;   $J \leftarrow \{1, ..., n\}$;
{begin}
    $stop \leftarrow false$ ;
    repeat
        find the median $\frac{c_k}{d_k}$ of the list $\left\{\frac{c_j}{d_j} \ , \ j \in J\right\}$;
        $J_0 \leftarrow \left\{j \in J \mid \frac{c_j}{d_j} < \frac{c_k}{d_k}\right\}$; $J_1 \leftarrow \left\{j \in J \mid \frac{c_j}{d_j} > \frac{c_k}{d_k}\right\}$; $J_2 \leftarrow \left\{j \in J \mid \frac{c_j}{d_j} = \frac{c_k}{d_k}\right\}$;
        $\widetilde{c} \leftarrow c_0 + \sum_{j \in J_1 \cup J_2} c_j$ ;  $\widetilde{d} \leftarrow d_0 + \sum_{j \in J_1 \cup J_2} d_j$ ;  $\lambda \leftarrow \frac{\widetilde{c}}{\widetilde{d}}$ ;
        if $\lambda > \frac{c_k}{d_k}$ then
            $x_j^\star \leftarrow 0 \ \forall j \in J_0 \cup J_2$ ;   $J \leftarrow J_1$
        else
            $x_j^\star \leftarrow 1 \ \forall j \in J_1 \cup J_2$ ;   $stop \leftarrow true$ ;
            if $J_0 \neq \emptyset$ then
                find the largest element $\frac{c_k}{d_k}$ of the list $\left\{\frac{c_j}{d_j}, \ j \in J_0\right\}$ ;
                if $\lambda < \frac{c_k}{d_k}$ then
                    $J \leftarrow J_0$ ;  $c_0 \leftarrow \widetilde{c}$ ;  $d_0 \leftarrow \widetilde{d}$ ;   $stop \leftarrow false$
                else
                    $x_j^\star \leftarrow 0 \ \forall j \in J_0$
                endif
            endif
        endif
    until $stop$
    $\lambda^\star \leftarrow \lambda$ {optimal value of (P)}
{end}
```

Fig. 3: HPR91: Algorithm of Hansen-Poggi-Ribeiro [11]

Based on this characterization, Hansen et al [11] propose a dichotomic procedure, denoted by HPR91 (figure 3) with a linear time complexity.

Remark 2: The median search embedded in this algorithm (figure 3) can be realized for instance by the linear time algorithm of Blum et al [3].

3 A revisited partition algorithm

We recall that a geometrical interpretation of this problem is [18]: given two points of $\mathbb{R}_+^2$ with coordinates (d_i, c_i) and (d_j, c_j), the ratio c_j/d_j (respectively $(c_i + c_j)/(d_i + d_j)$) represents the slope of the vector (d_j, c_j) (respectively the vector addition $(d_i + d_j, c_i + c_j)$). Thus, the resolution of (P) consists in determining among vectors (d_j, c_j), $j \in J$, those whose addition with the vector (d_0, c_0) gives a maximal slope.

Therefore, solving a 0-1 unconstrained hyperbolic program can be view as the search of a target $\lambda^\star$ verifying property 5. In other words, given the list of ratios c_j/d_j, we have to determine a pivot value around which J can be partitioned into two subsets J_0 and J_1 such that:

$$\forall i \in J_0 \qquad \frac{c_i}{d_i} \;\leq\; \frac{c_0 + \sum_{j\in J_1} c_j}{d_0 + \sum_{j\in J_1} d_j} \;\leq\; \frac{c_k}{d_k} \qquad \forall k \in J_1 \tag{1}$$

The induced optimal solution $x^\star$ of (P) is:

$$\forall j \in J_0 \quad x_j^\star = 0 \qquad \text{and} \qquad \forall j \in J_1 \quad x_j^\star = 1 \tag{2}$$

In practice, the algorithm consists in finding a real number $\lambda^\star$ (pivot value) which constructs a bipartition of J:

$$\forall i \in J_0 \quad \forall k \in J_1 \qquad \frac{c_i}{d_i} \;\leq\; \lambda^\star \;\leq\; \frac{c_k}{d_k} \tag{3}$$

satisfying the relationship (1).

We propose to construct a partition algorithm, denoted by NPUH96, according to a process already used to solve the relaxation of the 0-1 linear Knapsack problem (Balas and Zemel [1], Bourgeois and Plateau [4], Fayard and Plateau [7, 8]). It generates a sequence of pivot r and for each of them, the distribution given by the relationship 3 is performed.

If for the real value

$$\lambda(r) \;=\; \frac{c_0 + \sum_{j\in J_1(r)} c_j}{d_0 + \sum_{j\in J_1(r)} d_j}$$

where $J_1(r)$ denotes the subset produced by this distribution, there exists $j \in J_1(r)$ such that

$$\lambda(r) \;>\; \frac{c_j}{d_j}$$

the process is iterated by choosing a new pivot greater than r. In the opposite case, i.e. when there exists $j \in J_0(r)$ such that

$$\lambda(r) \;<\; \frac{c_j}{d_j}$$

the process is iterated by choosing a new pivot smaller than r, otherwise the current pivot is optimal ($\lambda^\star = \lambda(r)$), and an optimal solution is given by relationship 2 (see figure 4).

The linear time complexity of the algorithm in the worst of case or in the mean case depends on the quality of the pivot whose choice has to bring to a balanced

```
{initialization }
    $\widetilde{c} \leftarrow c_0$;   $\widetilde{d} \leftarrow d_0$;   $J \leftarrow \{1, ..., n\}$;
{begin }
    repeat
        calculate a **pivot** $r$ ;
        $J_0 \leftarrow \left\{ j \in J \mid \frac{c_j}{d_j} < r \right\}$ ;   $J_1 \leftarrow \left\{ j \in J \mid \frac{c_j}{d_j} \geq r \right\}$;
        $\widetilde{c} \leftarrow c_0 + \sum_{j \in J_1} c_j$ ;   $\widetilde{d} \leftarrow d_0 + \sum_{j \in J_1} d_j$ ;   $\lambda \leftarrow \frac{\widetilde{c}}{\widetilde{d}}$ ;
        if $\lambda \geq r$ then
            $x_j^\star \leftarrow 0 \;\; \forall j \in J_0$;   $J \leftarrow J_1$
        else
            $x_j^\star \leftarrow 1 \;\; \forall j \in J_1$;
            $x_j^\star \leftarrow 0 \;\; \forall j \in J_0$ such that $\frac{c_j}{d_j} \leq \lambda$;   $J_0 \leftarrow J_0 - \left\{ j \in J_0 \mid \frac{c_j}{d_j} \leq \lambda \right\}$;
            $c_0 \leftarrow \widetilde{c}$;   $d_0 \leftarrow \widetilde{d}$;   $J \leftarrow J_0$
        endif
    until $|J| \leq p$
    repeat
        determine $k$ such that $\frac{c_k}{d_k} = \max \left\{ \frac{c_j}{d_j} \mid j \in J \right\}$;
        $x_k^\star \leftarrow 1$;   $c_0 \leftarrow c_0 + c_k$;   $d_0 \leftarrow d_0 + d_k$;
        $J_0 \leftarrow \left\{ j \in J \mid \frac{c_j}{d_j} \leq \frac{c_0}{d_0} \right\}$;   $J \leftarrow J - J_0$;
        fix $x_j^\star$ at 0 for all $j$ in $J_0$
    until fixation of all variables $\{J = \emptyset\}$
    $\lambda^\star \leftarrow \frac{c_0}{d_0}$   {optimal value of (P)}
{end }
```

Fig. 4: NPUH96: Partition algorithm

partition, i.e. to generate two sublists with closed sizes.

The best known algorithm designated by Hansen et al [11] (algorithm 3) is based on a pivot equal to the median ratio of the list c_j/d_j. Thus the goal is the balancing of the partition of the list.

We propose to take into account, in addition, the Euclidean norm of the vector associated with the pivot because it realizes a compromise between a balanced and optimal partition of data. The following small example shows the strong importance of this idea.

Let us consider the two following instances:

$$(P1) \qquad \begin{cases} \max & \dfrac{1+13x_1+20x_2}{1+6x_1+8x_2} \\ \text{s.t.} & x_1, x_2 \in \{0,1\} \end{cases}$$

and

$$(P2) \qquad \begin{cases} \max & \dfrac{1+13x_1+5x_2}{1+6x_1+2x_2} \\ \text{s.t.} & x_1, x_2 \in \{0,1\} \end{cases}$$

Although the ratios associated respectively with x_1 and x_2 have the same values, these two instances have different optimal solutions: $x^1 = (0,1)$ for (P1) and $x^2 = (1,1)$ for (P2).

We propose therefore to replace the notion of median ratio by a pivot which is a "mean ratio". Namely, given a list $\left\{\dfrac{c_j}{d_j} \mid j \in J = \{1,...,n\}\right\}$, we consider as a pivot value

$$r = \frac{\sum_{j\in J'} c_j}{\sum_{j\in J'} d_j} \tag{4}$$

with $J' \subseteq J$.

4 Computational experiments

All runs were executed on a Sun Sparc station 5 with a C implementation of the HR68, HPR91 and NPUH96 algorithms for solving 0-1 unconstrained hyperbolic program.

To avoid trivial reductions, all ratios c_j/d_j are greater than c_0/d_0 (see proposition 2).

Three classes of problems have been distinguished: a class with a great proportion (more than 80%) of ratios c_j/d_j close to c_0/d_0, i.e. a class of instances with solutions with a great proportion of components equal to 0 (class 1). In the opposite, a second class with a great proportion (more than 80%) of ratios distant from c_0/d_0. This induces solutions with a great proportion of components equal to 1 (class 2). The third class constitutes an intermediate distribution between the two others (proportion of components equal to 1 between 40% and 60%) (class 3).

Ten instances are generated for each class (1, 2 and 3) and each size (number of variables n= 100, 1000 or 5000) data.

A first series of experiments have for purpose the comparisons between algorithm HR68 (figure 1), algorithm HPR91 (figure 3) and our algorithm NPUH96 (figure 4) in which pivot is the global mean ratio (i.e. $J' = J$ in relationship 4).

Table 1 gives the average running CPU time (in hundredth of seconds) of the ten running. The last column gives the gain realized by using our revisited method.

class	n	*HR68* t_1	*HPR91* t_2	*NPUH96* t_3	*gain (in %)* $= 100 \times \frac{t_2 - t_3}{t_2}$
1	*100*	*4.5*	*1.6*	*1.2*	*25*
	1000	*82.3*	*17.1*	*13.1*	*23*
	5000	*4000*	*92*	*72*	*22*
2	*100*	*9.8*	*1.9*	*1.4*	*26*
	1000	*1034.2*	*21.2*	*15.2*	*28*
	5000	*20000*	*110*	*82*	*25*
3	*100*	*22.7*	*2*	*1.2*	*40*
	1000	*2412.9*	*22.5*	*16.3*	*28*
	5000	*60000*	*120*	*90*	*25*

Tab. 1: Average running CPU time (in hundredth of seconds) computed over 10 problem instances for the algorithms HR68 (figure 1), HPR91 (figure 3) and NPUH96 (figure 4)

Two others variants of the partition algorithms (HPR91 and NPUH96) are proposed. For both algorithms, they differ in the choice of the pivot: given a list J to be partitioned, the pivot is now chosen taking into account only a sublist J' of J ("partial pivot"), namely such that $|J'| = 3$. These three elements are located at the beginning, the end and the middle of the list J. This choice has been used in the algorithm NKR dedicated to the 0-1 Knapsack problem (Fayard and Plateau [7]).

A comparison of the efficiency of these four implementations (two versions for HPR91, and two for NPUH96) is given by table 2.

This points out the gain realized by using "partial pivot" instead of "global pivot".

5 Conclusion

The 0-1 unconstrained hyperbolic program is a key tool for solving Lagrangean decomposition or relaxation of the 0-1 constrained hyperbolic programs. It has to be solved numerous times in any subgradient algorithm. Thus a special attention has to be made in order to construct a really efficient tool. This paper shows that our revisited partition algorithm reaches this goal.

n	*class*	*HPR91 algorithm*			*NPUH96 algorithm*		
		global t_2	*partial* t'_2	*gain (in %)* $100 \times \frac{t_2 - t'_2}{t_2}$	*global* t_3	*partial* t'_3	*gain (in %)* $= 100 \times \frac{t_3 - t'_3}{t_3}$
100	*1*	*1.6*	*1.6*	*0*	*1.2*	*1.2*	*0*
	2	*1.9*	*1.6*	*16*	*1.4*	*1.2*	*14*
	3	*2*	*1.6*	*20*	*1.2*	*1.2*	*0*
1000	*1*	*17.1*	*12.7*	*26*	*13.1*	*11.2*	*15*
	2	*21.2*	*15.1*	*29*	*15.2*	*13.6*	*11*
	3	*22.5*	*15.3*	*32*	*16.3*	*13.8*	*15*
5000	*1*	*92*	*75*	*18*	*72*	*66*	*8*
	2	*110*	*89*	*19*	*82*	*74*	*10*
	3	*120*	*91*	*24*	*90*	*89*	*1*

Tab. 2: Average running CPU time (in hundredth of second) computed over 10 problem instances for the four versions of partition algorithm

References

[1] E. Balas et E. Zemel "An algorithm for large zero-one Knapsack problems", *Operations Research* 28 (1980) 1130-1145.

[2] B. Bereanu, "Decision regions and minimum risk solutions in linear programming," in : A. Prekopa, Ed., Colloquium on applications of mathematics to economics, Budapest, 1963, (Publ. house of the Hungarian academy of sciences, Budapest, 1965, 37-42).

[3] M. Blum, R. W. Floyd, V. Pratt, R. L. Rivest et R. E. Tarjan "Time bounds for selection", *Journal of Computer and System Sciences* 7 (1973) 448-461.

[4] P. Bourgeois and G. Plateau, "BPK92 : A Revisited Hybrid Algorithm for 0-1 Knapsack Problem", EURO XI, Helsinki, June 1992.

[5] B. D. Craven, "Fractional Programming," Helderman, Berlin, 1988.

[6] C. Derman, "On sequential decisions and Markov chains," *Management Science* 9 (1962) 16-24.

[7] D. Fayard et G. Plateau "An algorithm for the solution of the 0-1 Knapsack problem", *Computing* 28 (1982) 269-287.

[8] D. Fayard et G. Plateau "An exact algorithm for the 0-1 collapsing Knapsack problem", *Discrete Applied Mathematics* 49 (1994) 175-87.

[9] P. L. Hammer et S. Rudeanu "Boolean methods in operations research and related areas", (Springer, Berlin - New York, 1968).

[10] P. Hansen, M. Minoux and M. Labbé, "Extension de la programmation linéaire généralisée au cas des programmes mixtes," *Comptes Rendus de l'Académie des Sciences de Paris* 305 (series I) (1987) 569-572.

[11] P. Hansen, M. V. Poggi de Aragao and C. C. Ribeiro, "Hyperbolic 0-1 programming and query information retrieval," *Mathematical Programming* 52 (1991) 255-263.

[12] M. Klein, "Inspection - maintenance - replacement schedule under Markovian deterioration," *Management Science* 9 (1963) 25-32.

[13] A. Nagih et G. Plateau "Programmes fractionnaires : tour d'horizon sur les applications et méthodes de résolution", *RAIRO - Operations Research* Vol. 33 (4) (1999) 383-419.

[14] A. Nagih et G. Plateau "A Lagrangean Decomposition For 0-1 Hyperbolic Programming Problems", **To appear** in *International Journal of Mathematical Algorithms.*

[15] A. Nagih et G. Plateau "Méthodes Lagrangiennes pour les problèmes hyperboliques en variables 0-1", FRANCORO : Rencontres Francophones de Recherche Opérationnelle, Mons, Belgique, june 1995.

[16] A. Nagih and G. Plateau "An exact Method for the 0-1 Fractional Knapsack Problem", INFORMS, New Orleans, USA, october 29 - novembre 1, 1995.

[17] P. Robillard "0-1 hyperbolic programming", *Naval Research Logistic Quarterly* 18 (1971) 47-58.

[18] Alan L. Saipe "Solving a (0, 1) hyperbolic program by branch and bound", *Naval Research Logistic Quarterly* 22 (1975) 397-416.

[19] H. M. Wagner and J. S. C. Yuan, "Algorithmic equivalence in linear fractional programming," *Management Science* 14 (1968) 301-306.

[20] W. T. Ziemba, C. Parkanand and R. Brooks-Hill, "Calculation of investment portfolios with risk free borrowing and lending," *Management Science* 21 (1974) 209-222.