International Journal of Applied Information Systems (IJAIS) – ISSN : 2249-0868
Foundation of Computer Science FCS, New York, USA
Volume 12 – No. 11, February 2018 – www.ijais.org

Solving Knapsack Feasibility Problem via Nonnegative Least-Squares Approach

Omar Kettani
Scientific Institute, Physics of the Earth Laboratory
Mohamed V- University
Rabat, Morocco

Faical Ramdani
Scientific Institute, Physics of the Earth Laboratory
Mohamed V- University
Rabat, Morocco

ABSTRACT
In the present paper, a dynamic programming algorithm based on nonnegative least-squares approach is proposed to tackle the NP-hard knapsack feasibility problem. Some examples are presented to show the effectiveness of the proposed approach and a Matlab code implementation is provided in the appendix.

General Terms
Heuristics, Algorithms

Keywords
Knapsack feasibility problem, NP-hard, nonnegative least-squares, dynamic programming

1. INTRODUCTION
We consider the n-dimensionnal NP-hard knapsack feasibility problem of finding an n-dimensionnal vector $x \in \{0, 1\}^n$ such that:

$$ax=b, \quad (1)$$

where a is an n-dimensional vector of positive integers, and b is a positive integer.

This problem is often called the integer knapsack problem and is well known to be NP-complete (Karp [1]). In [2], Mangasarian establishes an equivalence between the knapsack feasibility problem and an absolute Value equation then proposed to solve this problem via a Concave Quadratic Program and a Successive Linear Programming. A comprehensive survey on all aspects of knapsack problem was given by Kellerer et al. in [3].

The rest of this paper is organized as follows; in the next section, a dynamic programming algorithm for solving this problem is proposed. Section 3 provides some examples to illustrate the effectiveness of this approach. Conclusion of the paper is summarized in Section 5. Finally, a Matlab [4] code implementation is provided in the appendix.

2. PROPOSED APPROACH
The system of linear equations Ax=b, with binary variables $x \in \{0, 1\}^n$ can be solved using the following heuristic algorithm (in the present case, A is nx1 matrix b is an integer):

Input: n,A,b

Output: $x \in \{0, 1\}^n$

```
B=b'
A ← sort(A)
FOR j=n downto 1 DO
    C=B-A(:,j)
    A(:,j) ← NIL
    e0=norm(A*lsqnonneg(A,B)-B)
    e1=norm(A*lsqnonneg(A,C)-C)
    IF  e0< e1 THEN x(j) ← 0
            ELSE x(j)  ← 1
                B←C
    END IF
END FOR
FOR i=1 to DO
   IF  x(i)=1  THEN  output A(i,1)
 END FOR
END IF
```

Where lsqnonneg is the non negative least square function, x(j) denotes the j^{th} component of vector x, A(:,j) denotes the j^{th} column of A and norm is the Euclidean norm.

Notice that since the main loop requires n iterations which mainly computes matrices multiplications in $O(n^3)$, then the time complexity of the proposed approach is $O(n^4)$.

Table 1. Table captions should be placed above the table

Graphics	Top	In-between	Bottom
Tables	End	Last	First
Figures	Good	Similar	Very well

3. EXAMPLES
In this section, some randomly generated examples are presented, in order to illustrate the effectiveness of the proposed approach.

Example 1

b=16

n=8

16

1 2 6 2 6 1 7 2

16=1+2+6+7

Example 2

n=20

b=100

18 5 15 15 7 11 1 1 10 15 18 2 11 9 0

6 3 15 6 10

100=1+3+15+15+15+15+18+18

3.

Example 3

b=9843

n=200

163 173 16 79 51 160 86 182 36 52 29 27 173 115 109

28 170 124 70 102 80 15 47 24 36 47 83 9 180 188

98 97 67 180 73 22 156 77 48 80 19 26 188 191 115

11 46 70 164 3 8 33 129 146 129 90 109 59 148 37

137 36 73 125 156 16 185 155 97 87 89 61 101 102 163

158 128 75 162 106 70 187 175 110 124 117 41 60 94 46

168 38 45 34 45 87 62 184 86 36 180 195 87 22 51

81 118 52 120 142 44 23 59 63 84 101 17 52 160 5

185 146 97 115 47 91 192 109 104 46 97 124 135 79 73

197 7 177 182 159 19 52 67 135 27 144 21 130 98 155

143 180 178 66 139 39 6 148 100 95 180 121 123 171 161

115 36 47 177 5 97 33 195 142 100 94 11 136 8 14

104 19 163 163 144 29 131 103 194 129 160 90 86 165 16

26 34 78 166 160

9843=3+129+129+129+130+131+135+135+136+137+139+142+143+144+144+146+146+148+148+155+155+156+158+159+160+160+160+160+161+163+164+165+166+168+170+171+173+173+175+177+177+178+180+180+180+180+180+182+182+184+185+185+187+188+188+191+192+194+195+195+197

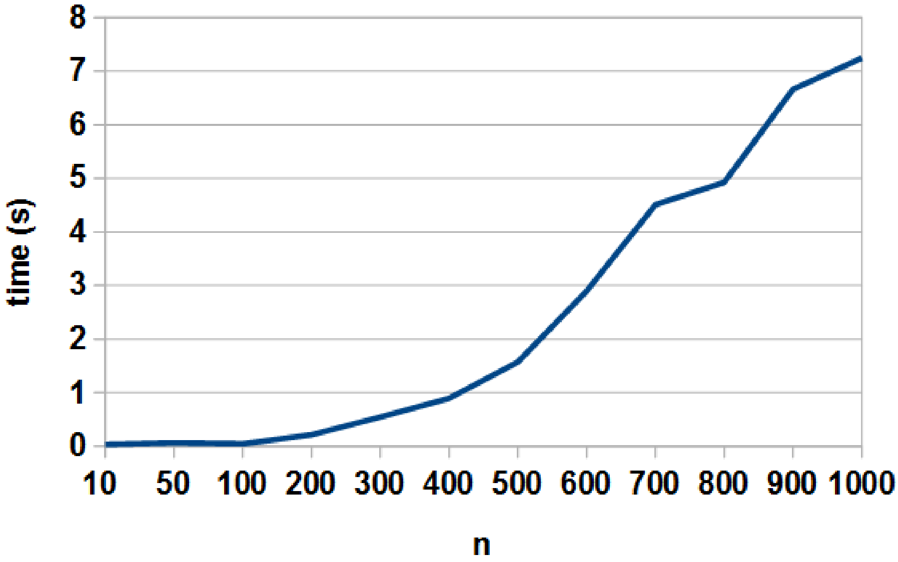

Fig 1: Chart of the elapsed time as function of n (the number of items)

4. CONCLUSION

In this work, a dynamic programming approach was proposed for solving the knapsack feasibility problem which is know to be an NP-hard problem.

In future work, this method will be compared to other existing approaches for solving this problem.

5. ACKNOWLEDGMENTS

Our thanks to the experts who have contributed towards development of the template.

6. REFERENCES

[1] R. M. Karp, Reducibility among combinatorial problems, in Complexity of Computer Computations, R. E. Miller and J. W. Thatcher, eds, Plenum, New York, 1972, pp. 85–103.

[2] O. L. Mangasarian, Knapsack Feasibility as an Absolute Value Equation Solvable by Successive Linear Programming Data Mining Institute Technical Report 08-03, September 2008. Optimization Letters 3(2) March 2009, 161-170

[3] H. Kellerer, U. Pferschy, and D. Pisinger. Knapsack Problems. Springer, 2004.

[4] https://www.mathworks.com/

7. APPENDIX

A Matlab code implementation, using the lsqnonneg, lsmr and pseudo-inverse inbuilt Matlab functions.

```
clear all;

n=input('input n=');

m=1

aa=randint(m,n,n)

xr=randint(n,1)
```

```
b=aa*xr;

b=b'

tic

[C,I]=sort(aa)

disp(aa)

rep=input('VU')

disp(b)

rep=input('VU')

A=C

B=b'

tic

 for i=n:-1:1

    B1=B-A(:,i)

    A(:,i)=[]

    if norm(A*lsqnonneg(A,B)-B,2)<
norm(A*lsqnonneg(A,B1)-B1,2)

        x(i)=0

    else

        x(i)=1

        B=B1
```

International Journal of Applied Information Systems (IJAIS) – ISSN : 2249-0868
Foundation of Computer Science FCS, New York, USA
Volume 12 – No. 11, February 2018 – www.ijais.org

```
    end;
end
toc
 disp(' Solution found via lsqnonneg:');
disp(b)
disp(sort(aa))
disp('x=');
disp(x);
y=C*x';
disp('checking lsqnonneg');
disp(y);
disp('TIME lsqnonneg');
t=toc
disp(' Elapsed time is:');
disp(t);
disp('seconds');
rep=input('VU')

%==========================================
tic
A=C
B=b'
for i=n:-1:1
    B1=B-A(:,i)
    A(:,i)=[]

    if norm(A*lsmr(A,B)-B,2)< norm(A*lsmr(A,B1)-B1,2)
        x(i)=0
    else
        x(i)=1
        B=B1
```

```
    end;
end
toc
disp(b)
disp(sort(aa))
disp(' Solution found via lsmr:');
disp('x=');
disp(x);
y=C*x';
disp('checking lsmr');
disp(y);
disp('TIME lsmr');
t=toc
disp(' Elapsed time is ');
disp(t);
disp(' seconds\n');
rep=input('VU')
tic
A=C
B=b'
for i=n:-1:1
    B1=B-A(:,i)
    A(:,i)=[]
    if norm(A*(max(A\B,zeros(1,i-1)'))-
B,2)<norm(A*(max(A\B1,zeros(1,i-1)'))-B1,2)
        x(i)=0
    else
        x(i)=1
        B=B1
    end;
end
```

```
toc
disp(' Solution found via Pinv:');
 disp(b)
disp(sort(aa))
disp('x=');
disp(x);
y=C*x';
disp('checking Pinv');
disp(y');
disp('TIME Pinv');
t=toc
disp(t);
z=[]
z=strcat(z,int2str(b))
 z=strcat(z,'=')
 for i=1:n
    if  (x(i)>0)&(aa(I(i))>0)
        z=strcat(z,int2str(aa(I(i))))
        z=strcat(z,'+')
        end;
 end


 z(length(z))=[]
  disp(y'-b);
 disp(b)
 disp(z)
```