budget_optimization

October 8, 2020

[1]:
```
%reload_ext ampl
```

1 1. budget optimization

Un budget di b Euro è disponibile per realizzare n progetti. Ogni progetto è descritto da un costo di c_i Euro e un profitto atteso di p_i Euro. Quali progetti devono essere selezionati per massimizzare il profitto atteso nei limiti del budget disponibile?

Partiamo con questa piccola istanza: budget di $b = 63$ Euro e progetti potenziali riportati in tabella

progetto	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
profitto	1	2	3	7	10	15	16	17	19	21	22	23	25	27	28
costo	2	4	6	7	9	10	12	13	14	16	17	19	20	22	23

giochiamoci un po'...

[2]:
```
import os
os.startfile("zaino.xlsx")
```

Qual è la soluzione migliore?

1.1 2. Metodi di soluzione

Proviamo ad utilizzare algoritmi e modelli:

- strategia 1: scegli prima il progetto più economico (euristico?);
- strategia 2: scegli prima il progetto più vantaggioso, cioè con il profitto atteso più alto (euristico?);
- strategia 3: scegli prima il progetto con il *profitto specifico* più alto (euristico?);
- enumerazione completa (esatto)
- programmazione matematica

[2]:
```
%%ampl
reset;

#instance ==========================
set projects ordered;
param profits{projects} >= 0;
```

```
param costs{projects} >= 0;
param budget;
#===================================

param curr_b;                  #working variable
param i_iter;                  #working variable
param ratios{projects};        #working variable
set b_argmax ordered;


var x{projects}, binary;       #project selection

data;

param budget := 63;
param: projects: profits costs:=
1 1 2
2 2 4
3 3 6
4 7 7
5 10 9
6 15 10
7 16 12
8 17 13
9 19 14
10 21 16
11 22 17
12 23 19
13 25 20
14 27 22
15 28 23;

display profits, costs;
```

```
:  profits costs    :=
1       1       2
2       2       4
3       3       6
4       7       7
5      10       9
6      15      10
7      16      12
8      17      13
9      19      14
10     21      16
11     22      17
12     23      19
```

```
13    25     20
14    27     22
15    28     23
;
```

1.1.1 strategia 1: scegli prima il progetto più economico

(osservazione: i progetti sono ordinati per valori crescenti sia di profitto sia di costo)

[3]:
```
%%ampl
include AMPL_cheapest_project_first.run;
```

```
selected projects:
[1] p = 1 c = 2
[2] p = 2 c = 4
[3] p = 3 c = 6
[4] p = 7 c = 7
[5] p = 10 c = 9
[6] p = 15 c = 10
[7] p = 16 c = 12
[8] p = 17 c = 13

tot: p = 71 c = 63
```

1.1.2 strategia 2: scegli prima il progetto più remunerativo

[4]:
```
%%ampl
include AMPL_most_profitable_project_first.run;
```

```
selected projects:
[15] p = 28 c = 23
[14] p = 27 c = 22

tot: p = 55 c = 45
```

1.1.3 strategia 3: scegli prima il progetto con il profitto specifico più alto

seleziona i progetti in ordine non crescente del rapporto p_i/c_i.

[5]:
```
%%ampl
include AMPL_Dantzig.run;
```

```
ratios [*] :=
 1  0.5
 2  0.5
 3  0.5
```

```
 4  1
 5  1.11111
 6  1.5
 7  1.33333
 8  1.30769
 9  1.35714
10  1.3125
11  1.29412
12  1.21053
13  1.25
14  1.22727
15  1.21739
;


selected projects:
[6] p = 15 c = 10
[9] p = 19 c = 14
[7] p = 16 c = 12
[10] p = 21 c = 16
[5] p = 10 c = 9
[1] p = 1 c = 2

tot: p = 82 c = 63
```

1.1.4 algoritmo di enumerazione completa (esatto)

[18]:
```
import os
os.startfile("knap_15.exe")
```

1.1.5 Programmazione Matematica

$$z = \max \sum_{i=1}^{n} p_i x_i$$
$$\sum_{i=1}^{n} c_i x_i \leq b$$
$$x_i \in \{0,1\} \qquad i = 1, \ldots, n$$

[6]:
```
%%ampl

maximize profit_tot:  sum {i in projects} profits[i] * x[i];
subject to knap: sum {i in projects} costs[i]*x[i] <= budget;

option solver cplex;
solve;
display x;
```

```
CPLEX 12.10.0.0: optimal integer solution; objective 83
6 MIP simplex iterations
```

```
0 branch-and-bound nodes
x [*] :=
 1  0
 2  0
 3  0
 4  0
 5  1
 6  1
 7  0
 8  1
 9  1
10  0
11  1
12  0
13  0
14  0
15  0
;
```

Risultati:

- strategia 1: scegli prima il progetto più economico: soluzione 71, sì è euristico
- strategia 2: scegli prima il progetto più profittevole: soluzione 55, sì è decisamente euristico
- strategia 3: scegli prima il progetto con il profitto specifico più alto: soluzione 82, euristico, ma abbastanza buono
- algoritmo di enumerazione completa (esatto): soluzione ottima 83
- programmazione matematica: soluzione ottima 83

1.2 3. Efficacia e efficienza

1.2.1 3.1. Efficacia, ossia qualità delle soluzioni

La Strategia 1 è sempre migliore della strategia 2?

[10]:
```
%%ampl
include AMPL_efficacy_heuristics.run;
```

```
option randseed 1601914073;

          Cheapest           Most_profit             Dantzig         Opt
--------------------------------------------------------------------------
      2889 ( 64.52%)       2558 ( 57.12%)       4478 (100.00%)       4478
      2518 ( 69.77%)       2127 ( 58.94%)       3602 ( 99.81%)       3609
      2520 ( 61.63%)       2869 ( 70.16%)       4089 (100.00%)       4089
      2911 ( 66.90%)       2323 ( 53.39%)       4343 ( 99.82%)       4351
      2562 ( 61.44%)       2425 ( 58.15%)       4170 (100.00%)       4170
      2645 ( 63.23%)       2352 ( 56.23%)       4183 (100.00%)       4183
      2487 ( 63.41%)       2433 ( 62.03%)       3922 (100.00%)       3922
```

```
2834 ( 62.46%)      2689 ( 59.27%)      4536 ( 99.98%)      4537
2521 ( 65.40%)      2403 ( 62.33%)      3845 ( 99.74%)      3855
2592 ( 63.84%)      2310 ( 56.90%)      4059 ( 99.98%)      4060
3169 ( 73.87%)      2408 ( 56.13%)      4290 (100.00%)      4290
2085 ( 54.03%)      2582 ( 66.91%)      3859 (100.00%)      3859
2126 ( 51.51%)      2958 ( 71.67%)      4114 ( 99.69%)      4127
2730 ( 66.02%)      2542 ( 61.48%)      4134 ( 99.98%)      4135
2239 ( 55.26%)      2550 ( 62.93%)      4052 (100.00%)      4052
2046 ( 53.66%)      2501 ( 65.59%)      3808 ( 99.87%)      3813
3202 ( 71.57%)      2278 ( 50.92%)      4461 ( 99.71%)      4474
2596 ( 64.24%)      2426 ( 60.03%)      4039 ( 99.95%)      4041
2257 ( 60.44%)      2365 ( 63.34%)      3728 ( 99.84%)      3734
2562 ( 67.74%)      2315 ( 61.21%)      3779 ( 99.92%)      3782
2285 ( 56.35%)      2614 ( 64.46%)      4055 (100.00%)      4055
2863 ( 67.64%)      2442 ( 57.69%)      4212 ( 99.50%)      4233
2304 ( 55.69%)      2615 ( 63.21%)      4135 ( 99.95%)      4137
2532 ( 61.86%)      2504 ( 61.18%)      4085 ( 99.80%)      4093
2806 ( 72.96%)      1799 ( 46.78%)      3840 ( 99.84%)      3846
2287 ( 55.48%)      2584 ( 62.69%)      4122 (100.00%)      4122
2489 ( 57.94%)      2726 ( 63.45%)      4290 ( 99.86%)      4296
2637 ( 66.74%)      2278 ( 57.66%)      3940 ( 99.72%)      3951
2469 ( 64.01%)      2315 ( 60.02%)      3857 (100.00%)      3857
2603 ( 59.69%)      2765 ( 63.40%)      4354 ( 99.84%)      4361
2734 ( 59.22%)      3040 ( 65.84%)      4617 (100.00%)      4617
2953 ( 69.19%)      2238 ( 52.44%)      4265 ( 99.93%)      4268
2356 ( 55.09%)      2902 ( 67.85%)      4277 (100.00%)      4277
2273 ( 53.67%)      2796 ( 66.02%)      4235 (100.00%)      4235
2540 ( 61.95%)      2732 ( 66.63%)      4082 ( 99.56%)      4100
2634 ( 65.83%)      2319 ( 57.96%)      3994 ( 99.83%)      4001
2431 ( 57.69%)      2628 ( 62.36%)      4207 ( 99.83%)      4214
2529 ( 67.24%)      2097 ( 55.76%)      3748 ( 99.65%)      3761
2318 ( 53.09%)      3034 ( 69.49%)      4362 ( 99.91%)      4366
2633 ( 62.11%)      2751 ( 64.90%)      4239 (100.00%)      4239
2483 ( 60.59%)      2366 ( 57.74%)      4091 ( 99.83%)      4098
2475 ( 62.95%)      2317 ( 58.93%)      3920 ( 99.69%)      3932
2486 ( 58.80%)      2842 ( 67.22%)      4224 ( 99.91%)      4228
2590 ( 65.35%)      2188 ( 55.21%)      3961 ( 99.95%)      3963
2746 ( 63.95%)      2483 ( 57.82%)      4289 ( 99.88%)      4294
2552 ( 60.12%)      2692 ( 63.42%)      4240 ( 99.88%)      4245
2550 ( 59.85%)      2967 ( 69.63%)      4250 ( 99.74%)      4261
2828 ( 70.38%)      2187 ( 54.43%)      4006 ( 99.70%)      4018
2657 ( 67.11%)      2395 ( 60.50%)      3952 ( 99.82%)      3959
2587 ( 66.13%)      2264 ( 57.87%)      3912 (100.00%)      3912
```

La strategia 3 sembra essere sempre quasi-ottima. E' vero in generale?

[12]:
```
%%ampl
reset;

#instance ==========================
set projects ordered;
param profits{projects} >= 0;
param costs{projects} >= 0;
param budget;
#====================================

param curr_b;                 #working variable
param i_iter;                 #working variable
param ratios{projects};       #working variable
param z_dantzig;
set b_argmax ordered;


var x{projects}, binary;      #project selection

/*
data;
param budget := 100;
param: projects: profits costs:=
1 9 10
2 80 100
3 8 9
4 7 8
5 6 7
;
*/

# things can getting worse!!


data;
param budget := 100;
param: projects: profits costs:=
1 0.1 0.2
2 99.9 100
3 0.1 0.3
4 0.1 0.4
5 0.1 0.5
6 0.1 0.1
7 0.1 0.2
8 0.1 0.3
9 0.1 0.4
10 0.1 0.3
```

```
;


include AMPL_dantzig.run;
let z_dantzig := sum{i in projects}profits[i]*x[i];


maximize profit_tot:  sum {i in projects} profits[i] * x[i];
subject to knap: sum {i in projects} costs[i]*x[i] <= budget;

option solver cplex;
solve > log.txt;


printf "\n\n\n      Dantzig         opt";
printf "\n------------------------------";
printf "\n%6f (%6.2f%%) %6ld", z_dantzig, 100 - 100*(profit_tot - z_dantzig)/
 ↪profit_tot,profit_tot;
```

```
ratios [*] :=
 1  0.5
 2  0.999
 3  0.333333
 4  0.25
 5  0.2
 6  1
 7  0.5
 8  0.333333
 9  0.25
10  0.333333
;


selected projects:
[6] p = 0 c = 0
[1] p = 0 c = 0
[7] p = 0 c = 0
[3] p = 0 c = 0
[8] p = 0 c = 0
[10] p = 0 c = 0
[4] p = 0 c = 0
[9] p = 0 c = 0
[5] p = 0 c = 0

tot: p = 0 c = 2
```

```
      Dantzig         opt
-----------------------------
0.900000 (  0.90%)     99
```

1.2.2 3.2. Efficienza

Le strategie 1,2 e 3 sono molto veloci ma sono solo euristiche.

Qual è l'efficienza degli approcci esatti?

Proviamo con questa (piccola) istanza con 50 progetti.

[13]:
```
%%ampl
reset;

#instance ==========================
set projects ordered;
param profits{projects} >= 0;
param costs{projects} >= 0;
param budget;
#====================================

param curr_b;                 #working variable
param i_iter;                 #working variable
param ratios{projects};       #working variable
set b_argmax ordered;


var x{projects}, binary;      #project selection

data;

param budget := 153;
param: projects: profits costs:=
1 1 2
2 2 4
3 3 6
4 7 7
5 10 9
6 15 10
7 16 12
8 17 13
9 19 14
10 21 16
11 22 17
12 23 19
13 25 20
14 27 22
15 28 23
```

```
16 30 24
17 31 25
18 33 28
19 34 29
20 36 30
21 38 32
22 39 33
23 41 35
24 42 36
25 43 37
26 45 38
27 47 39
28 48 40
29 49 41
30 50 42
31 51 43
32 52 44
33 53 46
34 54 47
35 55 48
36 56 49
37 58 50
38 59 51
39 60 53
40 62 54
41 63 55
42 64 57
43 66 58
44 67 59
45 68 60
46 69 62
47 70 63
48 72 64
49 73 65
50 75 67;

display costs;
display profits;
```

```
costs [*] :=
 1  2    6 10   11 17   16 24   21 32   26 38   31 43   36 49   41 55   46 62
 2  4    7 12   12 19   17 25   22 33   27 39   32 44   37 50   42 57   47 63
 3  6    8 13   13 20   18 28   23 35   28 40   33 46   38 51   43 58   48 64
 4  7    9 14   14 22   19 29   24 36   29 41   34 47   39 53   44 59   49 65
 5  9   10 16   15 23   20 30   25 37   30 42   35 48   40 54   45 60   50 67
;
```

```
profits [*] :=
 1  1    6 15   11 22   16 30   21 38   26 45   31 51   36 56   41 63   46 69
 2  2    7 16   12 23   17 31   22 39   27 47   32 52   37 58   42 64   47 70
 3  3    8 17   13 25   18 33   23 41   28 48   33 53   38 59   43 66   48 72
 4  7    9 19   14 27   19 34   24 42   29 49   34 54   39 60   44 67   49 73
 5 10   10 21   15 28   20 36   25 43   30 50   35 55   40 62   45 68   50 75
;
```

[14]:
```
%%ampl
option solver_msg 1;

maximize profit_tot:  sum {i in projects} profits[i] * x[i];
subject to knap: sum {i in projects} costs[i]*x[i] <= budget;

option solver cplex;
solve;
display x;
```

```
CPLEX 12.10.0.0: optimal integer solution; objective 198
2 MIP simplex iterations
0 branch-and-bound nodes
x [*] :=
 1 0    6 1   11 1   16 1   21 0   26 0   31 0   36 0   41 0   46 0
 2 0    7 1   12 0   17 1   22 0   27 0   32 0   37 0   42 0   47 0
 3 0    8 1   13 0   18 0   23 0   28 0   33 0   38 0   43 0   48 0
 4 0    9 1   14 1   19 0   24 0   29 0   34 0   39 0   44 0   49 0
 5 0   10 1   15 0   20 0   25 0   30 0   35 0   40 0   45 0   50 0
;
```

[15]:
```
import os
os.startfile("knap_50.exe")
```

[16]:
```
1000000/60
```

[16]: 16666.666666666668

[17]:
```
16666/24
```

[17]: 694.4166666666666

[ ]:
```

```