Αλγόριθμοι και Προχωρημένες Δομές Δεδομένων

Γραμμικός και Ακέραιος Προγραμματισμός

Πανεπιστήμιο Ιωαννίνων, Τμήμα Πληροφορικής και Τηλεπικοινωνιών, Μεταπτυχιακό Πληροφορικής & Δικτύων (2019-2020)

Γκόγκος Χρήστος

Δεκέμβριος 2019

Τι είναι ο Γραμμικός Προγραμματισμός;

- Ο Γραμμικός Προγραμματισμός (Linear Programming = LP) είναι τεχνική που ασχολείται με το πρόβλημα της βέλτιστης κατανομής των περιορισμένων πόρων μεταξύ ανταγωνιστικών δραστηριοτήτων.
- Ως ανταγωνιστικές δραστηριότητες νοούνται εκείνες που ανταγωνίζονται μεταξύ τους στη κατανάλωση των διαθέσιμων πόρων.
- Μπορεί να είναι διαφορετικά επενδυτικά σχέδια που επιζητούν χρηματοδότηση, διαφορετικά προϊόντα που παράγονται από διαθέσιμες πρώτες ύλες, διαφορετικές διαδρομές που μπορεί να ακολουθήσουν προϊόντα που διακινούνται σε προορισμούς κλπ

Που στοχεύει ο Γραμμικός Προγραμματισμός;

- Ο Γραμμικός Προγραμματισμός (Linear Programming) στοχεύει στην **αποδοτική διαχείριση των διαθέσιμων πόρων** έτσι ώστε να επιτευχθεί το πλέον θετικό αποτέλεσμα για την επιχείρηση ή τον οργανισμό
- Αυτό μπορεί να σημαίνει αύξηση κέρδους, μείωση κόστους, βελτίωση ποιότητας προϊόντων και υπηρεσιών κ.λπ.

Παράδειγμα: ένα πρόβλημα παραγωγής

Η βιοτεχνία ΕΠΙΠΛΟΞΥΛ παράγει τραπέζια και καρέκλες. Τόσο τα τραπέζια όσο και οι καρέκλες απαιτούν εργασία στα τρία τμήματα της επιχείρησης: το ξυλουργείο, το βαφείο και το στιλβωτήριο. Ειδικότερα, η κατασκευή ενός τραπεζιού απαιτεί 8 ώρες στο ξυλουργείο, 4 ώρες στο βαφείο και 4 ώρες στο στιλβωτήριο. Αντίστοιχα η κατασκευή κάθε καρέκλας απαιτεί 8 στο ξυλουργείο, 2 στο βαφείο και 3 στο στιλβωτήριο. Οι διαθέσιμες ώρες εργασίας για την επόμενη περίοδο παραγωγής είναι 960 στο ξυλουργείο, 400 στο βαφείο και 420 στο στιλβωτήριο. Το κέρδος της επιχείρησης από την πώληση κάθε τραπεζιού είναι 140 ευρώ ενώ για κάθε καρέκλα το κέρδος είναι 100 ευρώ. Ποια θα πρέπει να είναι η παραγωγή σε τραπέζια και καρέκλες για την επόμενη περίοδο;

Βήματα κατάστρωσης μοντέλου Γραμμικού Προγραμματισμού

- Προσδιορισμός μεταβλητών απόφασης
- Προσδιορισμός αντικειμενικής συνάρτησης
- Προσδιορισμός περιορισμών

Οι μεταβλητές απόφασης

- Οι μεταβλητές απόφασης είναι μεγέθη που ο λήπτης αποφάσεων είναι σε θέση να προσδιορίσει
- Πρακτικοί κανόνες για τον εντοπισμό των μεταβλητών απόφασης:
  - Το αποτέλεσμα μπορεί να υπολογιστεί γνωρίζοντας τις τιμές των μεταβλητών απόφασης
  - Αντιπροσωπεύουν μεγέθη τις τιμές των οποίων μπορεί να προσδιορίσει ο λήπτης αποφάσεων
- Για το πρόβλημα της ΕΠΙΠΛΟΞΥΛ οι μεταβλητές απόφασης είναι ο αριθμός από τα τραπέζια $x1$ και ο αριθμός από καρέκλες $x2$

Η αντικειμενική συνάρτηση

- Η αντικειμενική συνάρτηση είναι η σχέση που συνδέει τις μεταβλητές του προβλήματος με το αποτέλεσμα για το οποίο επιζητούμε τη βελτιστοποίηση
- για το πρόβλημα της ΕΠΙΠΛΟΞΥΛ η αντικειμενική συνάρτηση είναι $140x1 + 100x2$

Οι περιορισμοί

- Οι περιορισμοί αποτυπώνουν τους περιορισμούς πόρων που υπάρχουν στο πρόβλημα
- Οι περιορισμοί καταγράφονται με την μορφή ανισοτήτων ή ισοτήτων
- Ένας πρακτικός κανόνας είναι ότι θα πρέπει να μπορούμε να περιγράψουμε λεκτικά τι αντιπροσωπεύει το αριστερό και τι το δεξί μέρος της κάθε ανισότητας

- Για το πρόβλημα της ΕΠΙΠΛΟΞΥΛ οι περιορισμοί είναι τρεις:
  - Περιορισμός Ξυλουργείου
    $$8x1 + 8x2 \leq 960$$
  - Περιορισμός Βαφείου
    $$4x1 + 2x2 \leq 400$$
  - Περιορισμός Στιλβωτηρίου
    $$4x1 + 3x2 \leq 420$$

Μαθηματικό μοντέλο προβλήματος παραγωγής

$x1$: αριθμός από τραπέζια που θα παραχθούν
$x2$: αριθμός από καρέκλες που θα παραχθούν

$$
\begin{array}{lrcrcr}
max & 140x1 & + & 100x2 & & \\
s.t. & 8x1 & + & 8x2 & \leq & 960 \\
 & 4x1 & + & 2x2 & \leq & 400 \\
 & 4x1 & + & 3x2 & \leq & 420 \\
 & x1 & , & x2 & \geq & 0
\end{array}
$$

Περιορισμός ξυλουργείου

Περιορισμός βαφείου

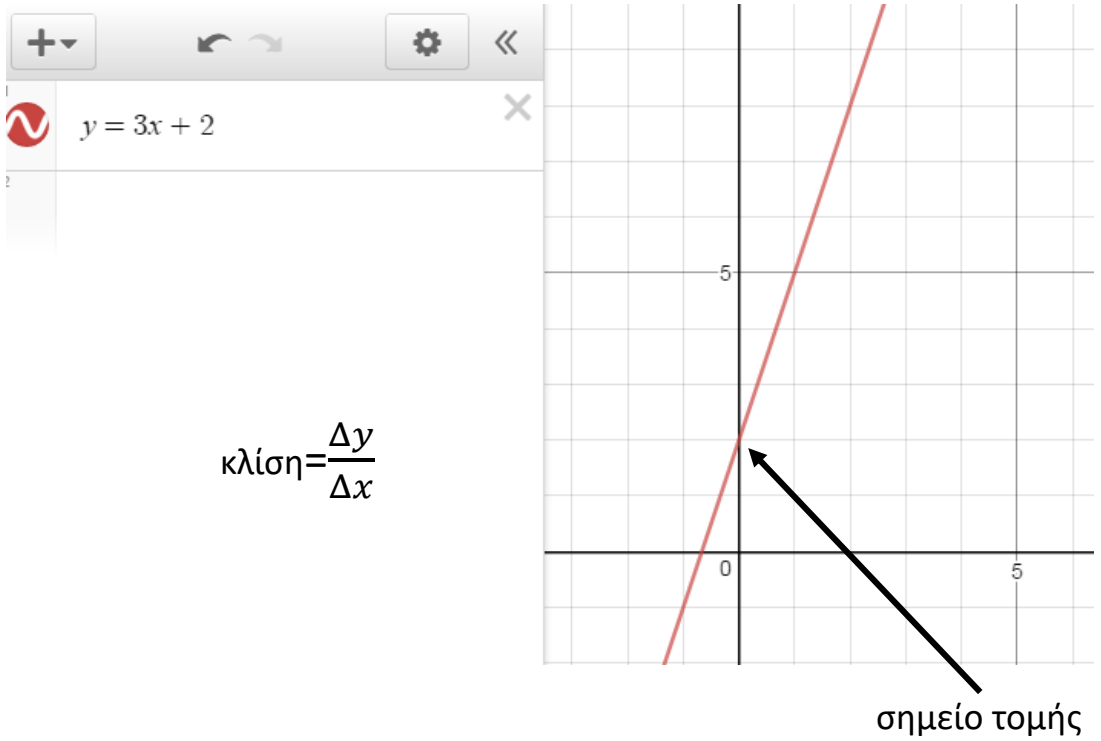
Περιορισμός στιλβωτηρίου

Ανάλυση ευαισθησίας

- Ένα σημαντικό πλεονέκτημα του Γραμμικού Προγραμματισμού είναι ότι μπορεί να υποστηρίξει τη διαδικασία λήψης αποφάσεων παρέχοντας επιπλέον πληροφορίες για την οικονομική ανάλυση του προβλήματος (ανάλυση ευαισθησίας)
- Εφόσον το πρόβλημα έχει λυθεί η ανάλυση ευαισθησίας μπορεί να δώσει απαντήσεις σε ερωτήσεις της μορφής:
  - Πως θα επηρεαστεί η λύση αν μεταβληθεί ο αριθμός των διαθέσιμων ωρών στο ξυλουργείο, στο βαφείο ή στο στιλβωτήριο;

Γραμμικές σχέσεις

- Μια σχέση ανάμεσα σε δύο μεγέθη είναι γραμμική όταν μια αλλαγή σε ένα μέγεθος προκαλεί ανάλογη αλλαγή στο άλλο μέγεθος
- Για παράδειγμα η σχέση $y = 3x + 2$ είναι γραμμική καθώς αύξηση 1 μονάδας στο x προκαλεί πάντα αύξηση 3 μονάδων στο y άσχετα με την αρχική τιμή του x
- Μια γραμμική σχέση όταν απεικονίζεται γραφικά είναι μια ευθεία γραμμή
- Μια γραμμική σχέση αναπαρίσταται μαθηματικά ως $y = mx + b$ όπου το m ονομάζεται κλίση (slope) και το b σημείο τομής (intercept)



Προϋποθέσεις εφαρμογής Γραμμικού Προγραμματισμού

- Οι προϋποθέσεις που πρέπει να ισχύουν για να διατυπωθεί ένα πρόβλημα Γραμμικού Προγραμματισμού είναι:
  - Γραμμικότητα (αναλογικότητα, προσθετικότητα)
  - Διαιρετότητα
  - Βεβαιότητα

Γραμμικότητα (linearity)

- Όλες οι συναρτήσεις του προβλήματος πρέπει να είναι γραμμικές ως προς τις άγνωστες μεταβλητές
- Σε κάποιες περιπτώσεις μπορεί να μην ισχύει απόλυτα η προϋπόθεση της γραμμικότητας αλλά οι γραμμικές συναρτήσεις να αποτελούν μια αρκετά καλή προσέγγιση της πραγματικότητας

- **Αναλογικότητα** (proportionality): η χρήση των πόρων και το κέρδος είναι ποσά ανάλογα με τις ποσότητες των μεταβλητών
- **Προσθετικότητα** (additivity): το κέρδος από τις επιμέρους μεταβλητές πρέπει να είναι ίσο με το άθροισμα των επί μέρους κερδών

$$f(x_1, x_2, \dots, x_n) = f(x_1) + f(x_2) + \cdots + f(x_n)$$

Διαιρετότητα (divisibility)

- Κάθε μεταβλητή θα πρέπει να είναι συνεχής και κατά συνέπεια άπειρα διαιρετή
- Άρα οι δραστηριότητες που αναπαρίστανται από τις μεταβλητές θα πρέπει να μπορούν να λαμβάνουν και δεκαδικές τιμές
- Αν δεν ισχύει η προϋπόθεση της διαιρετότητας τότε:
  - είτε αγνοείται η υπόθεση της διαιρετότητας και οι τιμές στρογγυλοποιούνται στην κοντινότερη ακέραια μονάδα
  - είτε χρησιμοποιούνται τεχνικές ακέραιου προγραμματισμού

Βεβαιότητα (deterministic property)

- Κάθε παράμετρος του προβλήματος θα πρέπει να είναι γνωστή με απόλυτη βεβαιότητα
- Αν κάποιες από τις παραμέτρους είναι τυχαίες μεταβλητές τότε το πρόβλημα λέγεται ότι είναι πρόβλημα στοχαστικού προγραμματισμού (**stochastic programming**) και οι μεταβλητές του θεωρούνται ότι ακολουθούν κάποιες κατανομές πιθανοτήτων

Ακέραιος Γραμμικός Προγραμματισμός

- Όταν για ένα πρόβλημα Γραμμικού Προγραμματισμού **δεν ισχύει** για κάποιες μεταβλητές απόφασης η **υπόθεση της διαιρετότητας** και οι μεταβλητές αυτές πρέπει υποχρεωτικά να λαμβάνουν ακέραιες τιμές τότε το πρόβλημα λέγεται ότι είναι πρόβλημα Ακέραιου Γραμμικού Προγραμματισμού (ILP=Integer Linear Programming)
- Όταν όλες οι μεταβλητές είναι ακέραιες τότε το πρόβλημα λέγεται αμιγώς ακέραιο (Pure Integer Program), ενώ αν μόνο κάποιες από τις μεταβλητές είναι ακέραιες τότε το πρόβλημα λέγεται μικτό ακέραιο πρόγραμμα (Mixed Integer Program)

Εφαρμογές Ακέραιου Γραμμικού Προγραμματισμού

Οι εφαρμογές του Ακέραιου Γραμμικού Προγραμματισμού εμπίπτουν σε δύο κατηγορίες:

- **Άμεσες**: Η φύση του προβλήματος καθιστά αδύνατη την εκχώρηση κλασματικών τιμών στις μεταβλητές του μοντέλου (π.χ. εύρεση του βέλτιστου πλήθους των μηχανών που απαιτούνται για την πραγματοποίηση μιας εργασίας)
- **Μετασχηματισμένες**: Προκύπτει η ανάγκη χρήσης βοηθητικών ακέραιων μεταβλητών για τη μοντελοποίηση του προβλήματος (π.χ. κατά την αλληλουχία εκτέλεσης δύο εργασιών Α και Β σε μια απλή μηχανή η εργασία Α πρέπει να προηγείται της εργασίας Β)

Παράδειγμα Ακέραιου Προγραμματισμού

Πέντε έργα αποτιμώνται για ένα χρονικό ορίζοντα προγραμματισμού διάρκειας τριών ετών. Στον ακόλουθο πίνακα δίνονται οι αναμενόμενες αποδόσεις για κάθε έργο και οι σχετιζόμενες με αυτό ετήσιες δαπάνες. Ποιο από τα έργα θα πρέπει να επιλεγεί για το χρονικό ορίζοντα των 3 ετών;

	Δαπάνες: εκατομμύρια € /έτος			
Έργο	**1**	**2**	**3**	**Αποδόσεις εκατομμύρια €**
1	**5**	**1**	**8**	**20**
2	**4**	**7**	**10**	**40**
3	**3**	**9**	**2**	**20**
4	**7**	**4**	**1**	**15**
5	**8**	**6**	**10**	**30**
Διαθέσιμα κεφάλαια εκατομμύρια €	**25**	**25**	**25**	

$$
\begin{array}{lrcrcrcrcrcr}
max & 20x1 & + & 40x2 & + & 20x3 & + & 15x4 & + & 30x5 & & \\
s.t. & 5x1 & + & 4x2 & + & 3x3 & + & 7x4 & + & 8x5 & \leq & 25 \\
 & x1 & + & 7x2 & + & 9x3 & + & 4x4 & + & 6x5 & \leq & 25 \\
 & 8x1 & + & 10x2 & + & 2x3 & + & x4 & + & 10x5 & \leq & 25 \\
 & x1 & & x2 & & x3 & & & & x5 & \geq & 0
\end{array}
$$

Η βέλτιστη ακέραια λύση είναι η:

$$x1 = x2 = x3 = x4 = 1, x5 = 0, z = 95$$

Η βέλτιστη λύση του γραμμικού προγραμματισμού είναι:

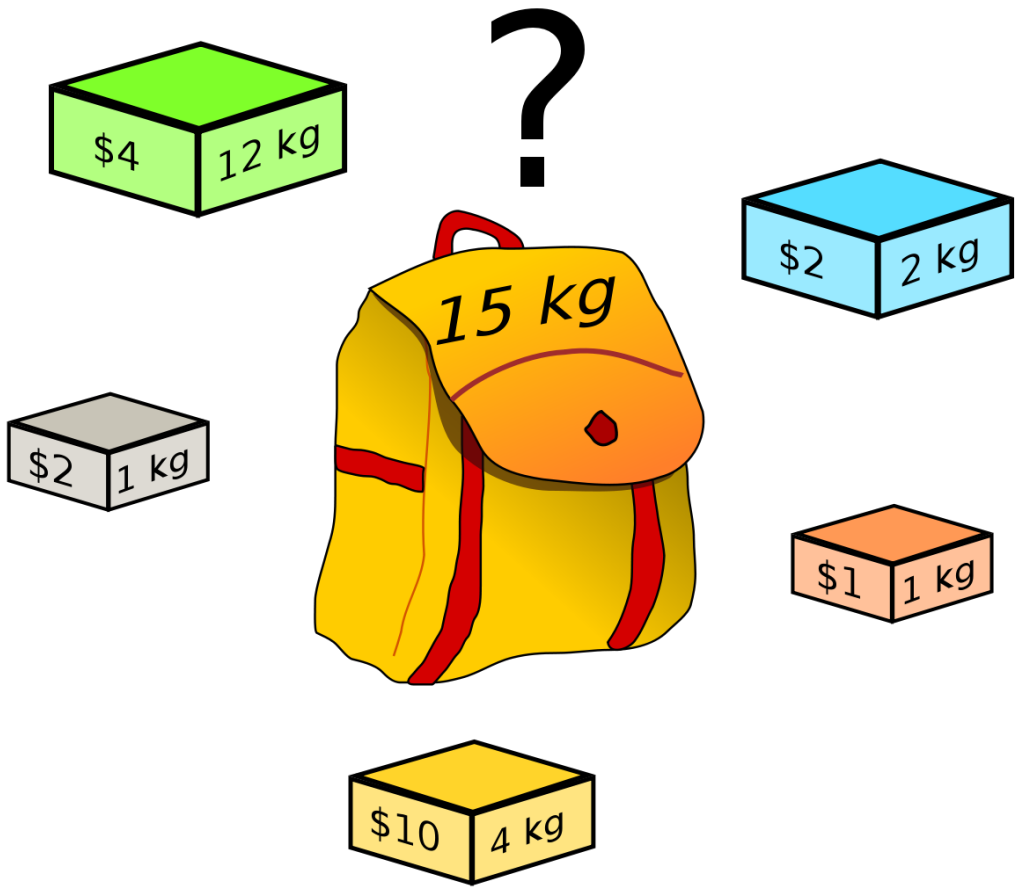
$x1 = 0.5789, x2 = x3 = x4 = 1, x5 = 0.7368, z = 108.68$

Η λύση αυτή δεν είναι έγκυρη λόγω του ότι οι δυαδικές μεταβλητές $x1, x5$ λαμβάνουν κλασματικές τιμές

Η στρογγυλοποίηση των μεταβλητών στον πλησιέστερο ακέραιο θα δώσει: $x1 = 1, x5 = 1$ αλλά τότε η λύση που προκύπτει παραβιάζει τους περιορισμούς

0-1 σακίδιο: ορισμός προβλήματος

- n αντικείμενα
- Αξίες αντικειμένων: $v_1, v_2, v_3, \dots, v_n$ (ακέραιες θετικές τιμές, αλλά μπορεί να είναι και πραγματικές θετικές τιμές)
- Βάρη αντικειμένων: $w_1, w_2, w_3, \dots, w_n$ (ακέραιες θετικές τιμές)
- Χωρητικότητα σάκου: C (ακέραια θετική τιμή)
- Ζητείται να βρεθεί το σύνολο $S \subseteq \{1,2,3, \dots, n\}$ και για το οποίο ισχύει ότι:
  - $\max V = \sum_{i \in S} v_i$
  - $\sum_{i \in S} w_i \leq C$



Μοντελοποίηση του προβλήματος 0-1 σακιδίου ως πρόβλημα IP

$$\max \sum_{i=1}^{n} v_i x_i$$

Subject to $\sum_{i=1}^{n} w_i x_i \leq C$

$\forall i \in \{1, \ldots, n\}\ x_i \in \{0,1\}$

Μοντελοποίηση στιγμιότυπου προβλήματος 0-1 σακιδίου

Αντικείμενο	Αξία	Βάρος
1	3	4
2	2	3
3	4	2
4	4	3
n=4, C=6		

$$\max \sum_{i=1}^{n} v_i x_i \rightarrow \mathbf{max}(\mathbf{3}\boldsymbol{x_1} + \mathbf{2}\boldsymbol{x_2} + \mathbf{4}\boldsymbol{x_3} + \mathbf{4}\boldsymbol{x_4})$$

$$\text{Subject to } \sum_{i=1}^{n} w_i x_i \leq C \rightarrow \mathbf{4}\boldsymbol{x_1} + \mathbf{3}\boldsymbol{x_2} + \mathbf{2}\boldsymbol{x_3} + \mathbf{3}\boldsymbol{x_4} \leq \mathbf{6}$$

$$\forall i \in \{1, \ldots, n\}\, x_i \in \{0,1\} \rightarrow \boldsymbol{x_1}, \boldsymbol{x_2}, \boldsymbol{x_3}, \boldsymbol{x_4} \in \{\mathbf{0}, \mathbf{1}\}$$

Λύση

Αντικείμενο	Αξία	Βάρος
1	3	4
2	2	3
3	4	2
4	4	3
n=4, C=6		

$$\boldsymbol{x_1 = 0, x_2 = 0, x_3 = 1, x_4 = 1}$$

$$\boldsymbol{\max(3x_1 + 2x_2 + 4x_3 + 4x_4) = 8}$$

$$\boldsymbol{4x_1 + 3x_2 + 2x_3 + 3x_4 = 5 \leq 6}$$

Προγραμματιστική λύση

Αντικείμενο	Αξία	Βάρος
1	3	4
2	2	3
3	4	2
4	4	3
n=4, C=6		

```
#include "../ortools/include/ortools/linear_solver/linear_solver.h"
namespace operations_research {
     void simple_knapsack_program() {
         MPSolver solver("IP KNAPSACK SOLVER", MPSolver::CBC_MIXED_INTEGER_PROGRAMMING);
         const double infinity = solver.infinity();
         MPVariable *const x1 = solver.MakeBoolVar("x1"); MPVariable *const x2 = solver.MakeBoolVar("x2");
         MPVariable *const x3 = solver.MakeBoolVar("x3"); MPVariable *const x4 = solver.MakeBoolVar("x4");
         MPConstraint *const c = solver.MakeRowConstraint(0, 6.0, "capacity_constraint");
         c->SetCoefficient(x1, 4); c->SetCoefficient(x2, 3);
         c->SetCoefficient(x3, 2); c->SetCoefficient(x4, 3);
         MPObjective *const objective = solver.MutableObjective();
         objective->SetCoefficient(x1, 3); objective->SetCoefficient(x2, 2);
         objective->SetCoefficient(x3, 4); objective->SetCoefficient(x4, 4);
         objective->SetMaximization(); solver.Solve();
         std::cout << "Objective value = " << objective->Value() << std::endl;
         std::cout << "x1 = " << x1->solution_value(); std::cout << " x2 = " << x2->solution_value();
         std::cout << " x3 = " << x3->solution_value();
         std::cout << " x4 = " << x4->solution_value() << std::endl;
     }
} // namespace operations_research
int main() {
    operations_research::simple_knapsack_program();
    return EXIT_SUCCESS;
}
```

https://github.com/chgogos/uoi_algorithms_and_complexity ortools_ip_solver_knapsack_demo.cc

Μεταγλώττιση και εκτέλεση (ORTools + Windows + VS 2017)

```
> tools\make run SOURCE=..\src\ortools_ip_solver_knapsack_demo.cc
```

Η εντολή μεταγλώττισης και εκτέλεσης δίνεται από τον κατάλογο ortools χρησιμοποιώντας το τερματικό x64 Native Tools Command Prompt for VS 2017

```
cl /EHsc /MD /nologo /D_SILENCE_STDEXT_HASH_DEPRECATION_WARNINGS -nologo /O2 -DNDEBUG /D__WIN32__
/DNOMINMAX /DWIN32_LEAN_AND_MEAN=1 /D_CRT_SECURE_NO_WARNINGS /DGFLAGS_DLL_DECL=
/DGFLAGS_DLL_DECLARE_FLAG= /DGFLAGS_DLL_DEFINE_FLAG= /Iinclude\\src\\windows /Iinclude /I. -DUSE_CBC -DUSE_CLP
-DUSE_BOP -DUSE_GLOP  -c ..\src\ortools_ip_solver_knapsack_demo.cc  /Foobjs\\ortools_ip_solver_knapsack_demo.obj
ortools_ip_solver_knapsack_demo.cc

cl /EHsc /MD /nologo /D_SILENCE_STDEXT_HASH_DEPRECATION_WARNINGS -nologo /O2 -DNDEBUG /D__WIN32__
/DNOMINMAX /DWIN32_LEAN_AND_MEAN=1 /D_CRT_SECURE_NO_WARNINGS /DGFLAGS_DLL_DECL=
/DGFLAGS_DLL_DECLARE_FLAG= /DGFLAGS_DLL_DEFINE_FLAG= /Iinclude\\src\\windows /Iinclude /I. -DUSE_CBC -DUSE_CLP
-DUSE_BOP -DUSE_GLOP  objs\\ortools_ip_solver_knapsack_demo.obj  lib\\ortools.lib psapi.lib ws2_32.lib
/Febin\\ortools_ip_solver_knapsack_demo.exe

bin\\ortools_ip_solver_knapsack_demo.exe
Objective value = 8
x1 = 0 x2 = 0 x3 = 1 x4 = 1
```

```
# ΟΡΓΑΝΩΣΗ ΣΕ ΚΑΤΑΛΟΓΟΥΣ
ortools
|
 → αποσυμπιεσμένη η εγκατάσταση του ORTOOLS
src
|
 → ortools_ip_solver_knapsack_demo.cc
```


https://github.com/chgogos/uoi_algorithms_and_complexity

Πηγές

- https://brilliant.org/wiki/linear-programming/
- http://people.brunel.ac.uk/~mastjjb/jeb/or/ip.html
- https://developers.google.com/optimization
- https://www.ibm.com/support/knowledgecenter/SSSA5P_12.8.0/ilog.odms.ide.help/OPL_Studio/opllanguser/topics/opl_languser_shortTour_IP_typical.html