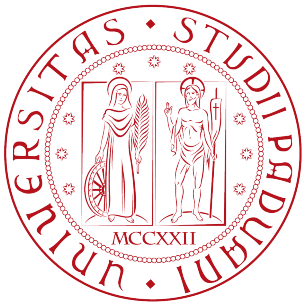


UNIVERSITÀ DEGLI STUDI DI PADOVA

Dipartimento di Tecnica e Gestione dei Sistemi Industriali

CORSO DI LAUREA TRIENNALE IN INGEGNERIA GESTIONALE (DM 270/04)

Tesi di Laurea di Primo Livello

Elaborazione di un problema complesso di routing mediante SCIP con risoluzione tramite Cplex

Relatore:
Prof. GIORGIO ROMANIN JACUR

Laureando:
DAVIDE GASPARI
n° matr. 612569/IG

ANNO ACCADEMICO 2012 – 2013

SOMMARIO

Introduzione

Il seguente lavoro di tesi è svolto con l'intento di elaborare un problema di programmazione lineare con un metodo alternativo. L'obiettivo proposto è quello di scrivere il codice di risoluzione del problema in un linguaggio alto, ovvero più vicino o familiare alla logica del nostro linguaggio naturale, come il linguaggio SCIP per trasformarlo poi in un linguaggio più basso ma più comune ed eseguibile da ogni calcolatore come il linguaggio Cplex. Come linguaggio di partenza è stato scelto SCIP vista la sua velocità di esecuzione e per l'uso gratuito, anche se non ancora molto diffuso. Attraverso una funzione del linguaggio ZIMPL ci sarà possibile tradurre il nostro problema scritto in SCIP nel linguaggio Cplex. Questa tecnica verrà testata su un problema riguardante il trasporto di pelli di bovino fresche dai macelli alle concerie con specifiche finestre temporali e vincoli di capacità.

Partiremo da una presentazione introduttiva su SCIP, sui comandi e su come utilizzarli per scrivere il codice di risoluzione del problema, una spiegazione di ZIMPL e sul come tradurre il codice dal linguaggio SCIP al linguaggio Cplex. In secondo luogo si presenterà il problema da risolvere con relativo codice, traduzione in Cplex, risoluzione e risultati. Il tutto sarà presentato anche con delle schermate relative alla schermata video che appare utilizzando i programmi di scrittura per l'elaborazione del problema, e la shell dei comandi per visualizzare il funzionamento di ZIMPL nella fase di traduzione e il programma Cplex nella fase di lettura, ottimizzazione e stampa delle soluzioni.

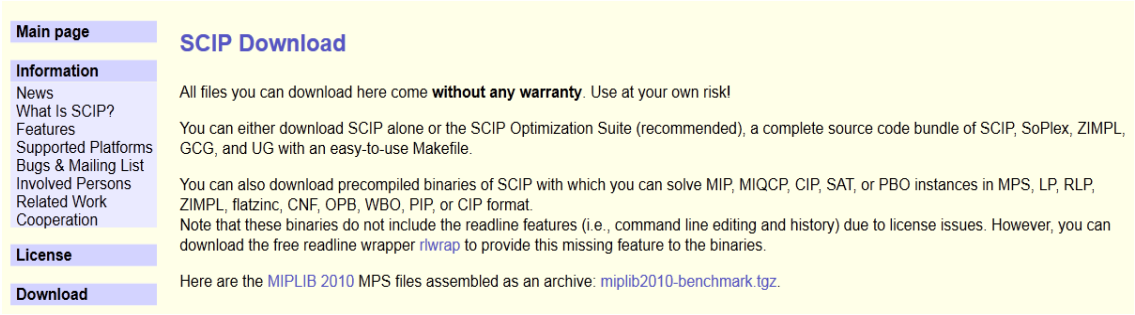
1. SCIP e ZIMPL

1.1 cos'è SCIP

SCIP (**S**olving **C**onstraint **I**nteger **P**rograms) è attualmente uno dei più veloci risolutori non commerciali di programmazione mista intera (MIP). La motivazione principale per cui ci accingiamo ad effettuare questo studio sta nel fatto che è un programma gratuito ad uso non commerciale. Sviluppato dai ricercatori del centro di ricerca per le tecnologie dell'informazione Konrad Zuse di Berlino, è un programma che si concentra per lo più sulla programmazione a vincoli interi orientato alle necessità degli esperti di programmazione matematica che vogliono avere un controllo totale del processo risolutivo e un accesso dettagliato alle informazioni che stanno all'interno del risolutore. SCIP può anche essere usato come un puro risolutore di programmazione mista intera oppure come un programma per i Branch-and-Cut e Branch-and-Price. Esso è implementato come una libreria C e fornisce delle classi wrapper C++ per gli utenti. Può anche essere utilizzato come programma autonomo per risolvere programmi misti interi del tipo MPS, LP e CIP. Oltre a ciò SCIP può leggere correttamente i modelli in linguaggio ZIMPL. Ed è proprio su quest'ultimo punto che si concentrerà la nostra analisi del software risolutore, poiché nel suo utilizzo in modalità stand-alone e grazie all'utilizzo del linguaggio ZIMPL, il programma non richiede un background elevato di conoscenze informatiche come nel suo utilizzo con le librerie e le classi, ma invece può essere considerato come ottima alternativa ai più comuni software risolutori (GPLK, CPLEX (IBM), Gurobi) con il vantaggio però di essere completamente gratuito per le attività non commerciali. *[1]*

1.2 SCIP, come usarlo

Scip è un programma open source completamente gratuito per uso non commerciale ma non concede alcun supporto e garanzia all'utente.



Main page

Information
News
What Is SCIP?
Features
Supported Platforms
Bugs & Mailing List
Involved Persons
Related Work
Cooperation

License

Download

SCIP Download

All files you can download here come **without any warranty**. Use at your own risk!

You can either download SCIP alone or the SCIP Optimization Suite (recommended), a complete source code bundle of SCIP, SoPlex, ZIMPL, GCG, and UG with an easy-to-use Makefile.

You can also download precompiled binaries of SCIP with which you can solve MIP, MIQCP, CIP, SAT, or PBO instances in MPS, LP, RLP, ZIMPL, flatzinc, CNF, OPB, WBO, PIP, or CIP format.
Note that these binaries do not include the readline features (i.e., command line editing and history) due to license issues. However, you can download the free readline wrapper rlwrap to provide this missing feature to the binaries.

Here are the MIPLIB 2010 MPS files assembled as an archive: miplib2010-benchmark.tgz.

Pagina di DOWLOAD di SCIP: è specificato l'uso senza garanzia [6] http://scip.zib.de/download.shtml]

Con i plugins inclusi nel programma, SCIP può essere usato come un risolutore stand-alone per problemi MIP. Aprendo il programma si accede alla shell di DOS (o di altri programmi in base al sistema operativo usato) e qui si possono utilizzare i comandi relativi alle possibili operazioni che il programma può eseguire. Nel caso del nostro problema ci riferiremo al programma SCIP (scip.exe) e al programma per la scrittura ZIMPL (zimpl.exe) entrambi all'interno nella cartella \scip\ con percorso C:\scip\. Il computer utilizzato per la risoluzione del nostro problema è un Asus U36SD con processore Intel i7-2620M con 8.00 GB di RAM e sistema operativo Windows 7 64 bit. Il programma per la scrittura da zero dei problemi è il Blocco Note fornito dal sistema operativo Windows 7.

La pagina che appare all'avvio del programma è questa:

```
SCIP version 2.1.1 [precision: 8 byte] [memory: block] [mode: optimized] [LP solver: SoPlex 1.6.0]
[GitHash: 8ab125b] #utilizzo dei parametri di default con il risolutore Soplex 1.6.0
Copyright (c) 2002-2011 Konrad-Zuse-Zentrum fuer Informationstechnik Berlin (ZIB)
External codes:
SoPlex 1.6.0      Linear Programming Solver developed at Zuse Institute Berlin (soplex.zib.de) [GitHash:
41efc2]
CppAD 20110101.5  Algorithmic Differentiation of C++ algorithms developed by B. Bell
www.coinr.org/CppAD)
Ipopt 3.10.1      Interior Point Optimizer developed by A. Waechter et.al.(www.coin-or.org/Ipopt)
user parameter file <scip.set> not found - using default parameters
```

mentre questi sono i comandi principali che possono esser dati in SCIP

```
SCIP> help (  *)      * il comando help illustra le possibili funzioni del programma SCIP
<change>              change the problem
<display>             display information
<set>                 load/save/change parameters
<write>               write information to file
checksol              double checks best solution w.r.t. original problem
conflictgraph         writes binary variable implications of transformed problem as conflict graph to file
count                 count number of feasible solutions
countpresolve         presolve instance before counting number of feasible solutions
free                  free current problem from memory
help                  display this help
newstart              reset branch and bound tree to start again from root
optimize              solve the problem
presolve              solve the problem, but stop after presolving stage
quit                  leave SCIP
read                  read a problem
```

Grazie al comando **read** è possibile andare a leggere i file in formato .lp,.mps o .tbl che sono però praticamente incomprensibili per l'uomo in quanto programmi scritti in linguaggio di basso livello. Dopo aver letto e caricato il programma col comando **read** bisogna dare il comando **optimize** che risolve il problema, per vedere la soluzione bisogna dare il comando **display solution** per visualizzare i risultati trovati.

1.3 ZIMPL

Il problema ora è trovare un modo per scrivere il codice non in un linguaggio incomprensibile all'uomo ma in un linguaggio più alto come il linguaggio ZIMPL, permettendo così la scrittura in una maniera lineare e logica il problema con dei comandi forniti attraverso un editor di testo come Wordpad o Blocco Note. Permettendo poi la traduzione in formati come .mps, .lp o .tbl .

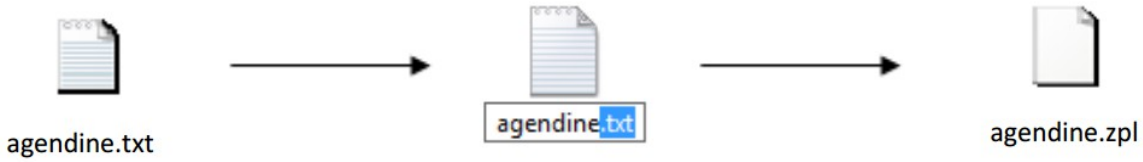
ZIMPL (**Z**use **I**nstitute **M**athematical **P**rogramming **L**anguage) è un semplice linguaggio in grado di tradurre il modello matematico di un problema in un programma lineare o misto - intero scritto in un formato *.lp* o *.mps* che può essere facilmente letto e risolto da un risolutore LP o MIP.

E' stato sviluppato dal dottor Thorsten Koch del centro di ricerca per le tecnologie dell'informazione Konrad Zuse presso l'Università di Berlino nel anno 2004. L'utilità del linguaggio è quella di poter scrivere un problema seguendo un modello matematico comprensibile per l'uomo ma che possa essere facilmente tradotto e letto anche dai risolutori LP e MPS. In assenza di ciò si avrebbero dei problemi scritti dall'uomo ma non leggibili dai software, o viceversa dei linguaggio *.mps* o *.lp* risolvibili dal software, ma praticamente incomprensibili per l'uomo e perciò di scarsa utilità.

Il programma ZIMPL si può recuperare sempre gratuitamente dal sito http://zimpl.zib.de

Una volta trascritto il modello matematico nel linguaggio ZIMPL tramite un editor di testo, il formato di output è un .txt che viene convertito rinominandolo cambiando il formato in un formato .zpl.

Ad esempio con il problema agendine:



una volta creato il file .zpl il programma ZIMPL traduce automaticamente il file in .lp e .tbl con un semplice comando:

```
C:\scip>zimpl.exe agendine.zpl

*****************************************************
* Zuse Institute Mathematical Programming Language  *
* Release 2.07  Copyright (C)2006 by Thorsten Koch  *
*****************************************************
*   This is free software and you are welcome to    *
*   redistribute it under certain conditions        *
*   ZIMPL comes with ABSOLUTELY NO WARRANTY         *
*****************************************************
Reading agendine.zpl

Instructions evaluated: 255

Name: agendine.zpl   Variables: 2   Constraints: 2   Non Zeros: 4

Writing [agendine.tbl]

Writing [agendine.lp]
```

ora abbiamo il file in formato .lp leggibile dal programma SCIP e dal programma Cplex.

Istruzioni più utilizzate in ZIMPL:

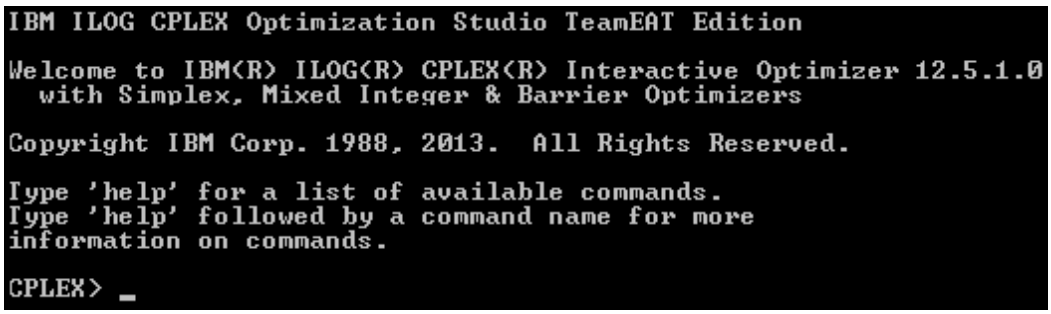
- SET: l'istruzione set è molto importante nel linguaggio zimpl e permette di creare una lista ordinata di elementi. I componenti sono o numeri o stringhe e ogni set specifico ha lo stesso numero di componenti. Il tipo del componente di ogni set deve essere lo stesso (i. e. devono essere o tutti i numeri o tutte le stringhe)
- PARAM: è l'istruzione per dichiarare i parametri. I parametri, però, possono essere dichiarate con o senza un set di indice. Senza l'indicizzazione di un parametro è solo un singolo valore, che è un numero o una stringa. Per i parametri indicizzati c'è un valore per ogni membro del set ma èpossibile anche dichiarare un valore predefinito.
- VAR: come i parametri, le variabili possono essere indicizzate all'inizio. Una variabile può essere di possibili tipi: continuo (chiamato reale), binari o interi. Il tipo predefinito è reale. Le variabili possono avere limiti inferiore e superiore, ma standard hanno come limite inferiore zero e l'infinito come limite superiore. Variabili binarie sono sempre delimitate tra zero e uno.
- MINIMIZE o MAXIMIZE: all'interno di un problema ci deve essere una dichiarazione oggettiva in un modello. L'obiettivo può essere minimizzare o massimizzare un termine lineare che esprime la funzione obiettivo.
- SUBTO: è l'istruzione base per la creazione di vincoli. I vincoli possono esser costruiti con disuguaglianze (strette e non) oppure con uguaglianze.
- FORALL: viene usata per la costruzione di molti vincoli generati con un’unica istruzione

1.4 Cplex

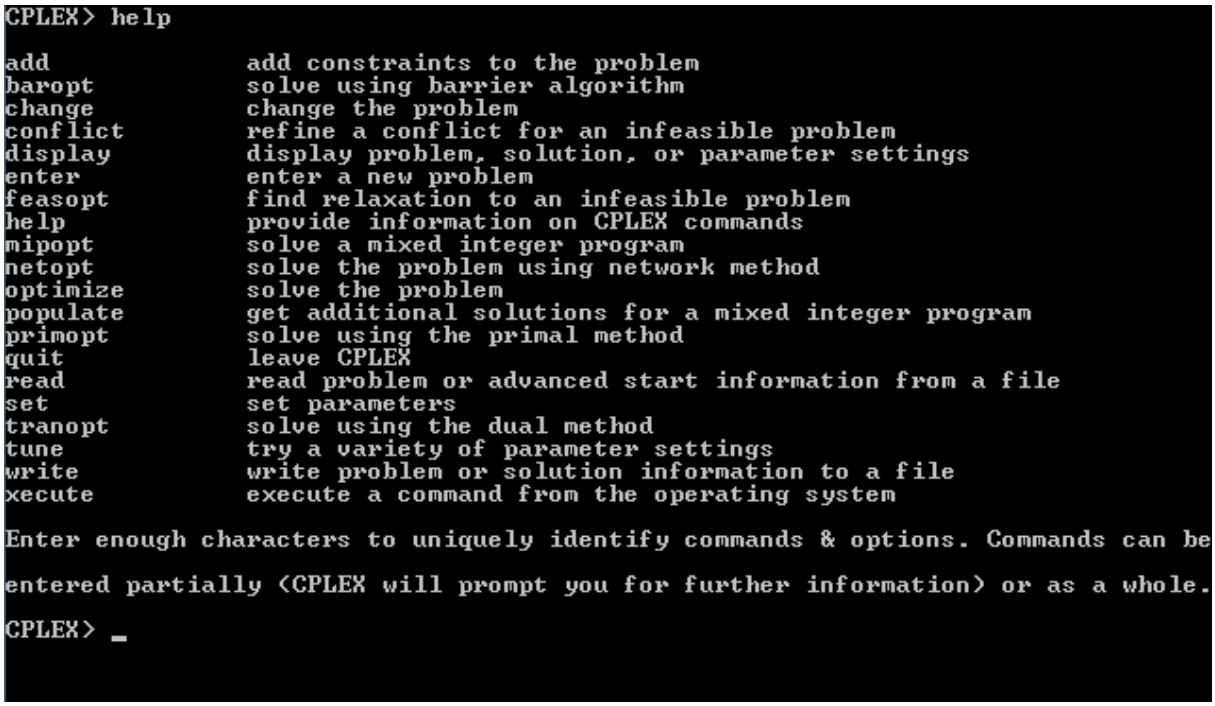
Cplex è un programma creato dalla IBM che risolve problemi di programmazione lineare intera, problemi di programmazione lineare anche di notevoli dimensioni, utilizzando le varianti primale o duale del metodo del simplesso o metodi di punto interno; risolve inoltre problemi di programmazione quadratica convessa, e problemi con vincoli quadratici convessi, utilizzando tecniche di Second-order cone programming (SOCP).

E' un programma a pagamento ma concede una licenza gratuita per l'uso accademico previa registrazione scaricabile direttamente dal sito IBM

All'apertura del programma cplex.exe ci appare questa schermata

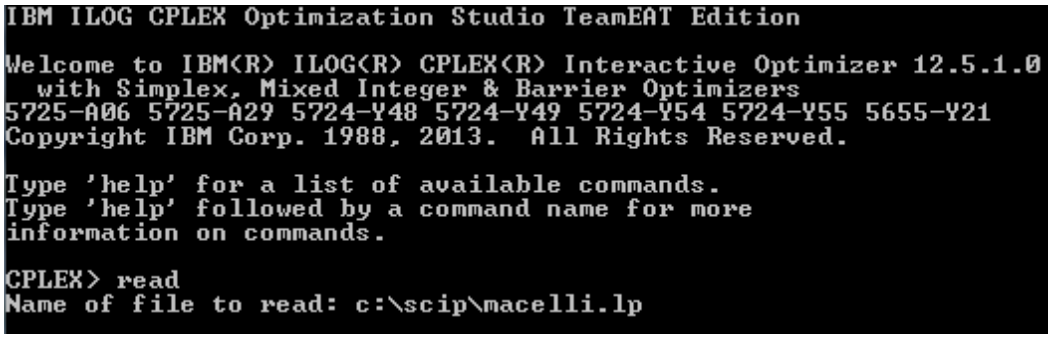


Digitando il comando HELP si trova



ovvero tutti i comandi possibili del programma Cplex

per far leggere un problema bisogna procedere in questa modo:



scrivere il comando **read**, premere invio, scrivere il nome del file con relativo percorso ed estensione .lp. Apparirà poi una schermata che conferma la lettura:

successivamente andrà poi fatto risolvere il problema col comando **optimize** e premere invio.

Sarà poi possibile vedere i risultati digitando il comando **display solution**

2 Descrizione del problema

TRASPORTO DI PELLI DI BOVINO FRESCHE DAI MACELLI ALLE CONCERIE CON SPECIFICHE FINESTRE TEMPORALI E VINCOLI DI CAPACITA'

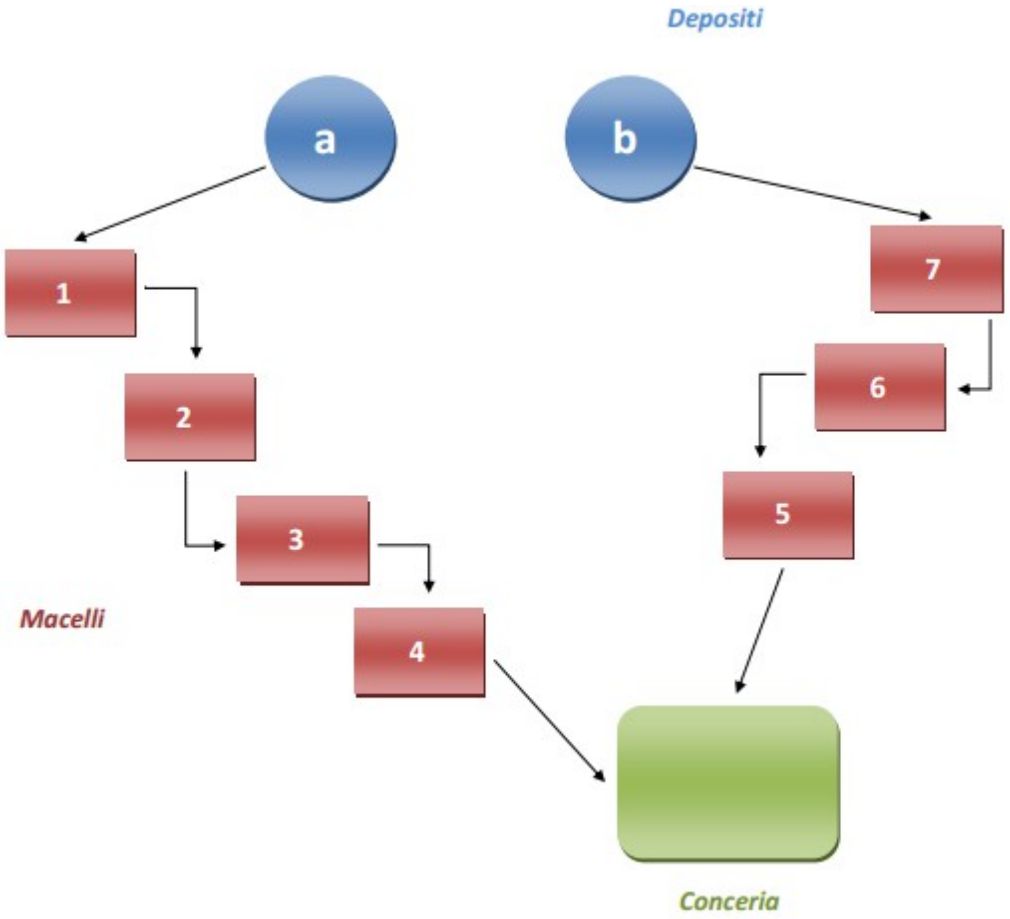
2.1 Introduzione al problema

Le concerie possono trattare sia pelli fresche che pelli trattate al sale. Le pelli trattate al sale hanno lunga durata a temperatura ambiente (mesi), ma presentano problemi di atrofia del sale (danni ambientali). Le pelli fresche non presentano questo problema, ma devono essere trattate o portate a bassa temperature entro breve tempo altrimenti saranno danneggiate irreparabilmente. Dato che le concerie preferiscono trattare pelli fresche, hanno bisogno di organizzare il loro trasporto dai macelli, tenendo conto che ogni singolo macello ha la propria (variabile) produzione giornaliera e la proprio finestra temporale di consegna delle pelli. [2]

2.2 Ambiente

Si considerino:

- Un territorio con una rete stradale
- Una conceria in un posto fissato
- Dei macelli in posti fissati
- Dei depositi di camion in posti fissati



- Camion eterogenei partono dai depositi per raccogliere le pelli fresche dai macelli e trasportarle alla conceria.

- Ogni camion parte dal proprio deposito o vi rimane.

- Ogni macello deve essere servito da un camion; i tempi e i costi di caricamento sono dati per ogni coppia camion-macello.

- Ogni camion, se partito dal suo deposito, deve terminare il suo giro alla conceria; tempi e costo di scarico sono dati per ogni camion.

- Tempi e costi di percorrenza sono dati per ogni connessione, per ogni camion.

- I camion devono rispettare le finestre temporali per ogni deposito, per ogni macello e per la conceria.

- Un'ulteriore finestra temporale per la conceria è imposta dai macelli a causa della durata massima di conservazione delle pelli .

- Il carico totale di ogni camion non deve superare il limite massimo di capacità di carico del camion.

Il problema considerato è un problema di routing con finestre temporali e parco automezzi eterogeneo (*capacitated vehicle routing problem with time windows and heterogeneous vehicle fleet*), che presenta alcune complessità peculiari rispetto a quelli normalmente considerati, come ad esempio la minimizzazione del costo totale con vincoli di tempo, finestre diverse a seconda dei percorsi seguiti, numero variabile di veicoli, numero di clienti visitati che può variare.

2.3 Obiettivo e vincoli del problema

2.3.1 Obiettivo

L'obiettivo è quello di trovare il giro che per ogni camion che parte, minimizzi i costi totali. I costi totali sono composti da:

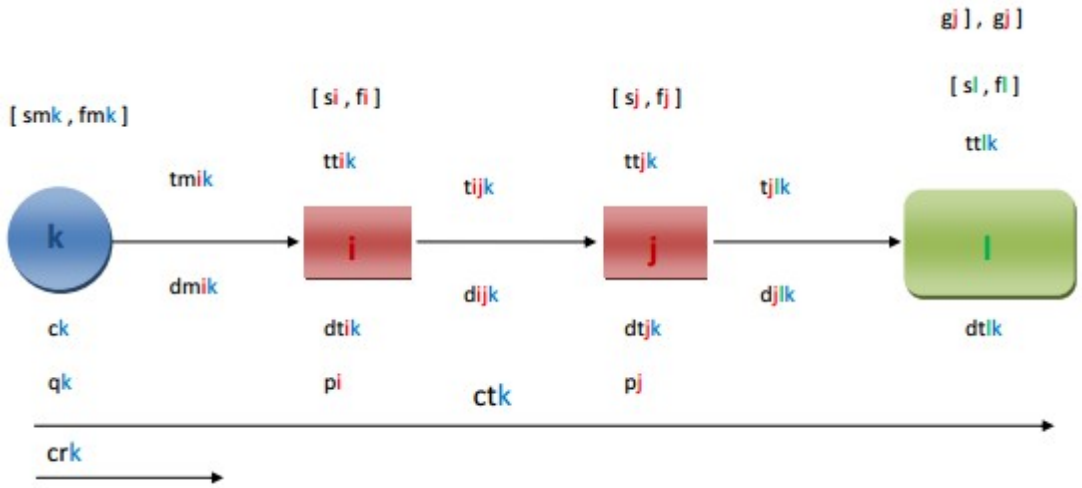
- Costo fisso di uscita per ogni camion che parte
- Costo totale di viaggio per ogni camion
- Costi totali di carico delle pelli
- Costi di scarico in conceria
- Costi totali di percorrenza
- Eventuali costi dovuti a partenze in ritardo

MIMIZE:

$$\sum_k \quad \{ +\text{exit cost} + \text{total travel cost} + \text{total loading cost} + \text{unloading cost} + \text{total spent time cost} + \text{start delay cost} \}$$

2.3.2 Dati del problema

r camion indicizzati da *k* ; ogni camion è univocamente associato ad un deposito *mk*;
n macelli indicizzati da *i,j* ;
la conceria è denominata *l* .



LEGENDA:

smk,fmk: *finestra di partenza per ogni camion [start-finish];*

si/sj,fi/fj: *finestra di carico per ogni macello [start-finish];*

sl,fl: *finestra di scarico in conceria [start-finish];*

gi/gj: *tempo minimo di fine scarico in conceria imposto da ogni macello;*

qk: *portata massima per ogni camion;*

pi/pj: *quantità caricata in ogni macello;*

ck: *costo di uscita per ogni camion;*

tmik: *tempo di trasporto dalla partenza ad ogni macello per ogni camion;*

dmik: *costo di trasporto dalla partenza ad ogni macello per ogni camion;*

ttik/ttjk: *tempo di carico di ogni macello per ogni camion;*

dtik/dtjk: *costo di carico di ogni macello per ogni camion;*

tijk: *tempo di trasporto da ogni macello ad ogni altro macello per ogni camion;*

dijk: *costo di trasporto da ogni macello ad ogni altro macello per ogni camion;*

tjlk: *tempo di trasporto da ogni macello alla conceria per ogni camion;*

djlk: *costo di trasporto da ogni macello alla conceria per ogni camion;*

ttlk: *tempo di scarico in conceria per ogni camion;*

dtlk: *costo di scarico alla conceria per ogni camion;*

ctk: *costo totale del tempo impiegato;*

crk: *costo dell'eventuale ritardo alla partenza di ogni camion;*

2.3.3 Vincoli

Vincoli di viaggio:

- visita di un camion per ogni macello
- sequenzialità
- il giro termina alla conceria

Vincoli di finestre temporali:

- finestre temporali rigide per ogni camion, per ogni macello e per la conceria
- finestre temporali imposte dai macelli che vengono serviti (per mantenere fresche le

pelli)

Vincoli di capacità massima di carico dei camion

- I camion devono rispettare le finestre temporali per ogni deposito, per ogni macello e per la conceria.
- Il carico totale di ogni camion non deve superare il limite massimo di capacità di carico del camion.

2.3.4 Variabili del problema

- Camion in partenza
- Per ogni camion:
  - percorso seguito*
  - tempo di visita per ogni macello servito

**Il percorso seguito può essere dato sia dalle connessioni coperte dal camion, sia dalla sequenza dei macelli serviti.*

2.4 Il modello matematico

EQUAZIONI DEL MODELLO II:

MIN $\sum$ (costl(k) | k=1,r)

$\sum$ (q(k)*y(i,k) | k=1,r) $\geq$ p(i) per ogni i ;

$\sum$ (y(i,k) | k=1,r) $\leq$ 1 per ogni i ;

$\sum$ (p(i)*y(i,k) | i=1,n) $\leq$ q(k)*ym(k) per ogni k ;

z(i) $\geq$ zm(k)+(tm(i,k)+tt(i,k))*(ym(k)+y(i,k)-1) per ogni i ;

z(j) $\geq$ z(i)+tt(j,k)+t(i,j,k)-M*(1-x(i,j))-M*(2-y(i,k)-y(j,k)) per ogni i<j ;

z(i) $\geq$ z(j)+tt(i,k)+t(j,i,k)-M*x(i,j)-M*(2-y(i,k)-y(j,k)) per ogni i<j ;

zl(k) $\geq$ z(i)+tl(i,k)+ttl(k)-M*(1-y(i,k)) per ogni k ;

cost(i) $\geq$ (c(k)+dm(i,k))*(ym(k)+y(i,k)-1) per ogni i,k ;

cost(j) $\geq$ cost(i)+d(i,j,k)+dt(j,k)-M*(1-x(i,j))-M*(2-y(i,k)-y(j,k)) per ogni i<j, per ogni k ;

cost(i) $\geq$ cost(j)+d(j,i,k)+dt(i,k)-M*x(i,j)-M*(2-y(i,k)-y(j,k)) per ogni i<j, per ogni k ;

costl(k) $\geq$ cost(i)+dl(i,k)+dtl(k)+ct(k)*(zl(k)-zm(k))+cr(k)*(zm(k)-sm(k))-M*(2-y(i,k)-ym(k))
per ogni i,k ;

zm(k) $\geq$ sm(k) per ogni k ;

zm(k) $\leq$ fm(k) per ogni k ;

z(i) $\geq$ s(i)+tt(i,k)-M*(1-y(i,k)) per ogni i,k ;

z(i) $\leq$ f(i)+M*(1-y(i,k)) per ogni i,k ;

zl(k) $\geq$ sl+ttl(k) per ogni k ;

zl(k) $\leq$ fl per ogni k ;

zl(k) $\leq$ g(i)+M*(1-y(i,k)) per ogni i,k ;

Il modello ottenuto è con un numero di variabili abbastanza ridotto e permette di operare in maniera molto efficiente se non si superano determinate condizioni, ovvero non avere più di 13 macelli e 3 camion. Un modello che può ritenersi adatto al nostro caso. Per il nostro obiettivo è corretto utilizzare questo modello in quanto l'implementazione del codice sarà sì lunga ma permette di ridurre sensibilmente il numero delle variabili utilizzate.

3 Risoluzione del problema mediante SCIP

3.1 La scrittura del problema con il linguaggio ZIMPL

Per la scrittura del programma utilizziamo un editor di testo come blocco note e rimanendo congrui con il codice e le regole del linguaggio ZIMPL cominciamo la stesura del codice.

Le righe di codice che comincano per # son commenti e il programma non le legge.

```
#macelli
set I := {1..13};
#camion
set K := {1..3};
#apertura (start) finestra di scarico in conceria
param sl := 420;
#chiusura (finish) finestra di scarico in conceria
param fl := 1740;
#big M
param M := 100000;
#apertura (start) finestra di partenza per ogni camion
param sm[K] := <1>10,<2>30,<3>20;
#chiusura (finish) finestra di partenza per ogni camion
param fm[K] := <1>100,<2>130,<3>120;
#tempo di scarico in conceria per ogni camion
param ttl[K] := <1>10,<2>10,<3>10;
#apertura (start) finestra di carico in ogni macello
param s[I] :=
<1>120,<2>380,<3>180,<4>450,<5>210,<6>500,<7>880,<8>590,<9>110,<10>720,<11>100,<
12>260,<13>80;
#chiusura (finish) finestra di carico in ogni macello
param f[I] :=
<1>540,<2>960,<3>490,<4>840,<5>900,<6>1500,<7>1400,<8>1000,<9>710,<10>1060,<11>1
000,<12>800,<13>440;
#tempo minimo di fine scarico in conceria imposto da ogni macello
param g[I] :=
<1>1700,<2>1800,<3>1800,<4>1900,<5>1701,<6>1600,<7>1900,<8>1500,<9>1600,<10>1800
,<11>1700,<12>1700,<13>1500;

#quantità caricata in ogni macello
param p[I] :=
<1>250,<2>530,<3>800,<4>300,<5>390,<6>340,<7>420,<8>390,<9>480,<10>620,<11>400,<
12>550,<13>270;
#portata massima di ogni camion
param q[K] := <1>3700,<2>2700,<3>2700;
#costo di uscita per ogni camion
param c[K] := <1>200,<2>220,<3>180;
#costo di scarico alla conceria per ogni camion
param dtl[K] := <1>5,<2>6,<3>4;
#costo del tempo totale di viaggio per ogni camion
param ct[K] := <1>0.1,<2>0.1,<3>0.1;
#costo dovuto al ritardo di partenza per ogni camion
param cr[K] := <1>0.05,<2>0.05,<3>0.05;
#tempo di carico di ogni macello per ogni camion
param tt[I*K] :=     |1    , 2      , 3      |
            |1   |   20,      22,      24|
            |2   |   25,      28,      30|
            |3   |   32,      35,      37|
            |4   |   18,      19,      20|
            |5   |   30,      34,      36|
```

```
            |6   |   22,      26,      28|
            |7   |   19,      21,      22|
            |8   |   41,      42,      44|
            |9   |   23,      24,      26|
            |10  |   44,      46,      47|
            |11  |   32,      34,      35|
            |12  |   29,      31,      32|
            |13  |   34,      35,      37|;
#costo di carico di ogni macello per ogni camion
param dt[I*K] :=     |1   , 2      , 3      |
            |1   |   2,       2,       2|
            |2   |   3,       3,       3|
            |3   |   3,       4,       4|
            |4   |   2,       2,       2|
            |5   |   3,       3,       4|
            |6   |   2,       3,       3|
            |7   |   2,       2,       3|
            |8   |   4,       4,       4|
            |9   |   2,       3,       4|
            |10  |   4,       3,       3|
            |11  |   3,       3,       3|
            |12  |   4,       4,       4|
            |13  |   4,       3,       3|;
#tempo di trasporto dalla partenza ad ogni macello per ogni camion
param tm[I*K] :=     | 1    , 2      , 3      |
            |1   |   125,     130,     132|
            |2   |   137,     140,     143|
            |3   |   151,     157,     160|
            |4   |   134,     144,     148|
            |5   |   129,     133,     138|
            |6   |   120,     121,     121|
            |7   |    90,      92,      96|
            |8   |   112,     120,     124|
            |9   |   106,     109,     115|
            |10  |   149,     153,     157|
            |11  |   120,     121,     122|
            |12  |   123,     129,     133|
            |13  |   141,     144,     146|;
#costo di trasporto dalla partenza ad ogni macello per ogni camion
param dm[I*K] :=     | 1   , 2      , 3      |
            |1   |   25,      30,      27|
            |2   |   37,      40,      38|
            |3   |   51,      57,      53|
            |4   |   34,      44,      38|
            |5   |   41,      43,      42|
            |6   |   33,      33,      32|
            |7   |   28,      29,      29|
            |8   |   31,      34,      32|
            |9   |   33,      35,      34|
            |10  |   29,      32,      30|
            |11  |   40,      43,      41|
            |12  |   35,      37,      36|
            |13  |   32,      35,      33|;

#tempo di trasporto da ogni macello alla conceria per ogni camion
param tl[I*K] :=     | 1    , 2      , 3      |
            |1   |   128,     131,     133|
            |2   |   139,     142,     144|
            |3   |   153,     157,     160|
            |4   |   139,     146,     150|
            |5   |   141,     142,     143|
            |6   |   128,     129,     130|
            |7   |   158,     162,     165|
            |8   |   160,     162,     164|
            |9   |   136,     139,     141|
            |10  |   124,     125,     126|
            |11  |   151,     154,     156|
            |12  |   144,     147,     149|
            |13  |   137,     140,     142|;
#costo di trasporto da ogni macello alla conceria per ogni camion
```

```
param dl[I*K] :=     | 1    , 2     , 3      |
            |1   |   28,      31,      29|
            |2   |   39,      42,      40|
            |3   |   53,      57,      55|
            |4   |   39,      46,      42|
            |5   |   37,      39,      38|
            |6   |   36,      35,      35|
            |7   |   41,      43,      42|
            |8   |   58,      59,      59|
            |9   |   36,      37,      37|
            |10  |   26,      28,      27|
            |11  |   49,      52,      50|
            |12  |   40,      41,      41|
            |13  |   42,      44,      43|;

#tempo di trasporto da ogni macello ad ogni altro macello per ogni camion
param t[I*I*K] :=     |1   ,   2    ,  3     |
            |1,2  | 145,     147,     146|
            |1,3  | 161,     166,     163|
            |1,4  | 156,     159,     157|
            |1,5  | 122,     123,     122|
            |1,6  | 125,     127,     126|
            |1,7  | 111,     112,     111|
            |1,8  | 137,     139,     138|
            |1,9  | 108,     111,     109|
            |1,10 | 149,     154,     151|
            |1,11 | 166,     166,     166|
            |1,12 | 139,     142,     140|
            |1,13 | 122,     119,     121|
            |2,1  | 142,     143,     142|
            |2,3  | 150,     154,     152|
            |2,4  | 160,     163,     161|
            |2,5  | 167,     165,     166|
            |2,6  | 156,     156,     157|
            |2,7  | 163,     166,     164|
            |2,8  | 123,     125,     124|
            |2,9  | 175,     177,     176|
            |2,10 | 141,     144,     142|
            |2,11 | 111,     115,     112|
            |2,12 | 157,     158,     158|
            |2,13 | 152,     154,     153|
            |3,1  | 158,     163,     160|
            |3,2  | 149,     151,     150|
            |3,4  | 168,     171,     169|
            |3,5  | 168,     166,     167|
            |3,6  | 141,     144,     142|
            |3,7  | 128,     131,     129|
            |3,8  | 181,     187,     184|
            |3,9  | 148,     151,     149|
            |3,10 | 124,     128,     125|
            |3,11 | 154,     159,     156|
            |3,12 | 146,     148,     147|
            |3,13 | 139,     141,     140|
            |4,1  | 166,     168,     167|
            |4,2  | 167,     170,     168|
            |4,3  | 158,     165,     150|
            |4,5  | 171,     170,     171|
            |4,6  | 152,     154,     153|
            |4,7  | 181,     190,     184|
            |4,8  | 114,     116,     115|
            |4,9  | 166,     170,     167|
            |4,10 | 116,     119,     117|
            |4,11 | 139,     141,     140|
            |4,12 | 112,     114,     113|
            |4,13 | 148,     150,     149|
            |5,1  | 156,     157,     156|
            |5,2  | 181,     178,     179|
            |5,3  | 145,     147,     146|
            |5,4  | 132,     133,     133|
            |5,6  | 120,     120,     121|
```

5,7	100, 101, 101
5,8	161, 170, 164
5,9	173, 175, 174
5,10	146, 148, 147
5,11	182, 186, 184
5,12	151, 155, 153
5,13	116, 119, 117
6,1	112, 115, 113
6,2	107, 109, 108
6,3	119, 117, 118
6,4	137, 135, 136
6,5	113, 115, 114
6,7	115, 116, 115
6,8	101, 107, 104
6,9	150, 153, 151
6,10	161, 165, 163
6,11	143, 148, 145
6,12	138, 141, 139
6,13	144, 146, 145
7,1	109, 110, 110
7,2	111, 113, 112
7,3	133, 140, 136
7,4	112, 112, 113
7,5	100, 102, 101
7,6	128, 132, 129
7,8	98, 99, 98
7,9	146, 149, 147
7,10	151, 153, 152
7,11	105, 109, 106
7,12	167, 170, 168
7,13	150, 153, 151
8,1	131, 134, 132
8,2	140, 143, 141
8,3	129, 132, 130
8,4	107, 109, 108
8,5	119, 121, 120
8,6	170, 170, 169
8,7	159, 161, 160
8,9	121, 123, 122
8,10	177, 180, 178
8,11	133, 136, 134
8,12	149, 151, 150
8,13	101, 105, 102
9,1	103, 105, 104
9,2	177, 183, 179
9,3	145, 147, 146
9,4	170, 175, 172
9,5	138, 142, 140
9,6	152, 155, 153
9,7	149, 153, 151
9,8	119, 123, 121
9,10	133, 135, 134
9,11	126, 131, 128
9,12	160, 164, 162
9,13	175, 178, 176
10,1	152, 156, 154
10,2	138, 141, 139
10,3	120, 124, 121
10,4	115, 118, 116
10,5	143, 147, 144
10,6	166, 171, 168
10,7	153, 155, 154
10,8	174, 177, 175
10,9	131, 134, 132
10,11	168, 172, 169
10,12	123, 126, 124
10,13	169, 171, 170
11,1	168, 174, 170
11,2	113, 118, 115
11,3	157, 161, 159

```
             |11,4 | 136,     142,     139|
             |11,5 | 184,     190,     186|
             |11,6 | 147,     151,     149|
             |11,7 | 107,     112,     109|
             |11,8 | 129,     132,     130|
             |11,9 | 130,     134,     131|
             |11,10| 124,     128,     126|
             |11,12| 157,     160,     158|
             |11,13| 160,     162,     161|
             |12,1 | 134,     137,     135|
             |12,2 | 160,     165,     162|
             |12,3 | 144,     147,     145|
             |12,4 | 115,     118,     116|
             |12,5 | 148,     152,     150|
             |12,6 | 136,     138,     137|
             |12,7 | 172,     177,     174|
             |12,8 | 151,     154,     152|
             |12,9 | 164,     167,     165|
             |12,10| 170,     175,     172|
             |12,11| 154,     156,     155|
             |12,13| 120,     123,     121|
             |13,1 | 154,     157,     155|
             |13,2 | 137,     140,     138|
             |13,3 | 178,     184,     181|
             |13,4 | 151,     153,     152|
             |13,5 | 118,     119,     119|
             |13,6 | 146,     150,     148|
             |13,7 | 149,     151,     150|
             |13,8 | 103,     104,     103|
             |13,9 | 172,     175,     174|
             |13,10| 169,     174,     171|
             |13,11| 158,     162,     160|
             |13,12| 121,     123,     122|;
#costo di trasporto da ogni macello ad ogni altro macello per ogni camion
param d[I*I*K] :=     |    1,       2,       3|
             |1,2   |   45,      47,      46|
             |1,3   |   61,      66,      63|
             |1,4   |   56,      59,      57|
             |1,5   |   49,      51,      50|
             |1,6   |   51,      50,      50|
             |1,7   |   46,      47,      47|
             |1,8   |   39,      42,      40|
             |1,9   |   36,      35,      35|
             |1,10  |   48,      46,      47|
             |1,11  |   35,      33,      34|
             |1,12  |   51,      50,      50|
             |1,13  |   46,      44,      45|
             |2,1   |   42,      43,      42|
             |2,3   |   50,      54,      52|
             |2,4   |   60,      63,      61|
             |2,5   |   56,      58,      57|
             |2,6   |   60,      58,      59|
             |2,7   |   51,      54,      52|
             |2,8   |   57,      56,      56|
             |2,9   |   63,      60,      61|
             |2,10  |   47,      45,      46|
             |2,11  |   55,      53,      54|
             |2,12  |   44,      43,      44|
             |2,13  |   51,      48,      49|
             |3,1   |   58,      63,      60|
             |3,2   |   49,      51,      50|
             |3,4   |   68,      71,      69|
             |3,5   |   64,      64,      65|
             |3,6   |   51,      50,      50|
             |3,7   |   60,      59,      59|
             |3,8   |   42,      41,      41|
             |3,9   |   40,      38,      39|
             |3,10  |   31,      29,      30|
             |3,11  |   55,      53,      54|
             |3,12  |   50,      49,      49|
```

3,13	67, 64, 65
4,1	66, 68, 67
4,2	67, 70, 68
4,3	58, 65, 60
4,5	71, 69, 70
4,6	66, 63, 64
4,7	72, 73, 72
4,8	61, 62, 61
4,9	66, 63, 64
4,10	43, 41, 42
4,11	50, 47, 48
4,12	31, 29, 30
4,13	44, 42, 43
5,1	45, 45, 45
5,2	58, 61, 59
5,3	49, 49, 50
5,4	58, 57, 57
5,6	61, 64, 62
5,7	59, 63, 61
5,8	77, 75, 76
5,9	44, 41, 42
5,10	53, 50, 51
5,11	69, 66, 67
5,12	52, 49, 50
5,13	36, 35, 35
6,1	59, 60, 60
6,2	71, 70, 70
6,3	49, 52, 50
6,4	61, 62, 61
6,5	80, 78, 79
6,7	67, 68, 67
6,8	63, 61, 62
6,9	52, 50, 51
6,10	75, 72, 73
6,11	40, 38, 39
6,12	62, 60, 61
6,13	45, 42, 43
7,1	75, 74, 74
7,2	65, 64, 64
7,3	49, 52, 50
7,4	51, 56, 52
7,5	49, 47, 48
7,6	39, 37, 38
7,8	29, 27, 28
7,9	63, 61, 62
7,10	42, 40, 41
7,11	42, 41, 41
7,12	76, 73, 74
7,13	38, 36, 37
8,1	53, 54, 53
8,2	45, 51, 47
8,3	71, 72, 71
8,4	47, 46, 46
8,5	69, 66, 67
8,6	70, 68, 69
8,7	54, 59, 56
8,9	36, 34, 35
8,10	64, 61, 62
8,11	54, 53, 53
8,12	57, 55, 55
8,13	38, 37, 37
9,1	35, 34, 34
9,2	65, 63, 64
9,3	42, 41, 41
9,4	67, 64, 65
9,5	43, 41, 41
9,6	54, 52, 53
9,7	61, 58, 59
9,8	38, 37, 37
9,10	51, 47, 48

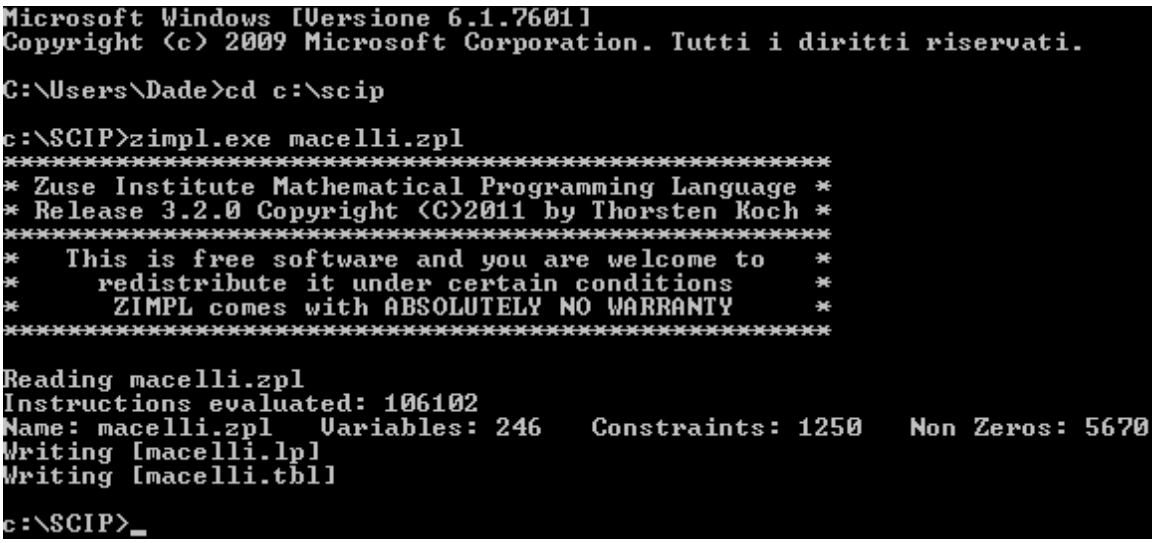
```
              |9,11 |   40,      38,      39|
              |9,12 |   57,      55,      56|
              |9,13 |   72,      66,      68|
              |10,1 |   49,      46,      47|
              |10,2 |   46,      43,      44|
              |10,3 |   30,      29,      29|
              |10,4 |   42,      40,      41|
              |10,5 |   55,      53,      54|
              |10,6 |   73,      71,      72|
              |10,7 |   44,      42,      43|
              |10,8 |   65,      63,      64|
              |10,9 |   52,      49,      50|
              |10,11|   63,      60,      61|
              |10,12|   45,      43,      44|
              |10,13|   72,      68,      69|
              |11,1 |   33,      31,      32|
              |11,2 |   74,      72,      73|
              |11,3 |   56,      54,      55|
              |11,4 |   49,      48,      48|
              |11,5 |   68,      66,      67|
              |11,6 |   39,      37,      38|
              |11,7 |   41,      39,      40|
              |11,8 |   54,      51,      52|
              |11,9 |   41,      39,      40|
              |11,10|   64,      62,      63|
              |11,12|   52,      50,      51|
              |11,13|   55,      53,      54|
              |12,1 |   52,      51,      51|
              |12,2 |   42,      40,      41|
              |12,3 |   49,      47,      48|
              |12,4 |   32,      31,      31|
              |12,5 |   53,      51,      52|
              |12,6 |   63,      60,      61|
              |12,7 |   77,      74,      75|
              |12,8 |   56,      54,      55|
              |12,9 |   55,      54,      54|
              |12,10|   42,      41,      41|
              |12,11|   56,      54,      55|
              |12,13|   30,      30,      30|
              |13,1 |   45,      42,      43|
              |13,2 |   53,      51,      52|
              |13,3 |   68,      65,      66|
              |13,4 |   42,      39,      40|
              |13,5 |   35,      33,      34|
              |13,6 |   46,      44,      45|
              |13,7 |   39,      37,      38|
              |13,8 |   37,      36,      36|
              |13,9 |   71,      67,      68|
              |13,10|   74,      70,      71|
              |13,11|   57,      55,      56|
              |13,12|   29,      27,      28|;

  var y[I*K] binary;
  var ym[K] binary;
  var x[I*I] binary;

  var z[<i> in I] real >= 0;
  var zm[<k> in K] real >= 0;
  var zl[<k> in K] real >= 0;
  var cost[<i> in I] real >= 0;
  var costl[<k> in K] real >= 0;
  #var ff[I*I] real >= 0;

  minimize costot: sum <k> in K:costl[k]*1;

  subto vmac1: forall <i> in I do sum <k> in K: q[k]*y[i,k] >= p[i];

  subto vmac2: forall <i> in I do sum <k> in K: y[i,k] <= 1;

  subto port: forall <k> in K do sum <i> in I: p[i]*y[i,k] <= q[k]*ym[k];
```

```
subto temini: forall <k> in K do
     forall <i> in I do z[i] >= zm[k]+(tm[i,k]+tt[i,k])*(ym[k]+y[i,k]-1);

subto temfin: forall <k> in K do
     forall <i> in I do  zl[k] >= z[i]+tl[i,k]+ttl[k]-M*(1-y[i,k]);

subto finini1: forall <k> in K do zm[k] >= sm[k];

subto finini2: forall <k> in K do zm[k] <= fm[k];

subto fin1: forall <k> in K do
     forall <i> in I do z[i] >= s[i]+tt[i,k]-M*(1-y[i,k]);

subto fin2: forall <k> in K do
     forall <i> in I do z[i] <= f[i]+M*(1-y[i,k]);

subto finl1: forall <k> in K do zl[k] >= sl+ttl[k];

subto finl2: forall <k> in K do zl[k] <= fl;

subto finl3: forall <k> in K do
     forall <i> in I do zl[k] <= g[i]+M*(1-y[i,k]);

subto cini: forall <k> in K do
     forall <i> in I do cost[i] >= (c[k]+dm[i,k])*(ym[k]+y[i,k]-1);

subto cfin: forall <k> in K do
     forall <i> in I do costl[k] >= cost[i]+dl[i,k]+dtl[k]+ct[k]*(zl[k]-zm[k])
+cr[k]*(zm[k]-sm[k])-M*(2-y[i,k]-ym[k]);


subto tem1:forall <k> in K do
     forall <i> in I do
            forall <j> in I  with i<j do
                   z[j] >= z[i]+tt[j,k]+t[i,j,k]-M*(1-x[i,j])-M*(2-y[i,k]-
y[j,k]);

subto tem2:forall <k> in K do
     forall <i> in I do
            forall <j> in I  with i<j do
                  z[i] >= z[j]+tt[i,k]+t[j,i,k]-M*x[i,j]-M*(2-y[i,k]-y[j,k]);


subto c1:forall <k> in K do
     forall <i> in I do
            forall <j> in I with i<j do
                  cost[j] >= cost[i]+d[i,j,k]+dt[j,k]-M*(1-x[i,j])-M*(2-y[i,k]-
y[j,k]);


subto c2:forall <k> in K do
     forall <i> in I do
            forall <j> in I with i<j do
                  cost[i] >= cost[j]+d[j,i,k]+dt[i,k]-M*x[i,j]-M*(2-y[i,k]-
y[j,k]);
```

Come spiegato in precedenza ora si salva il file in formato .txt, il file viene così chiamato *macelli.txt,* successivamente rinominato in .zpl così da farlo leggere al programma zimpl.exe e viene messo nel percorso **C:\scip .**

3.2 Compilazione del programma

Apriamo poi la shell di DOS, il prompt dei comandi, e ci spostiamo nella directory **C:\Scip** tramite il comando: **cd c:\Scip**. Ora possiamo eseguire il comando **zimpl.exe macelli.zpl** che, se non ci son errori, permetterà la creazione dei file *macelli.tbl* e, il più importante per noi, *macelli.lp* che verrà poi usato nella risoluzione con Cplex.



Come si vede dallo screen il programma ZIMPL fa una lettura, riconoscendo il numero delle variabile e il numero dei vincoli creando poi i file *macelli.tbl* e *macelli.lp*.

Il file .tbl è un file di tabelle in cui sono inclusi i dati scritti nel file .zpl, estrapolati e resi leggibili dal pc, per esempio una parte di codice è:

```
zimpl    v    0 y#1#1    "y#1#1"
zimpl    v    1 y#1#2    "y#1#2"
zimpl    v    2 y#1#3    "y#1#3"
zimpl    v    3 y#2#1    "y#2#1"
zimpl    v    4 y#2#2    "y#2#2"
zimpl    v    5 y#2#3    "y#2#3"
zimpl    v    6 y#3#1    "y#3#1"
zimpl    v    7 y#3#2    "y#3#2"
zimpl    v    8 y#3#3    "y#3#3"
```

```
zimpl          v       9 y#4#1              "y#4#1"
```

Il file .lp è un file di programmazione lineare scritto in un linguaggio di programmazione più basso, ovvero seguendo la logica del computer invece che quella umana come fatto nel file .zpl, un esempio di codice di un file .lp è il seguente:

```
set K := {1..3};
param ttl[K] := <1>10,<2>10,<3>10;
subto vmac1: forall <i> in I do sum <k> in K: q[k]*y[i,k] >= p[i];
vmac2_1:
 + y#1#3 + y#1#2 + y#1#1 <= 1
vmac2_2:
 + y#2#3 + y#2#2 + y#2#1 <= 1
vmac2_3:
 + y#3#3 + y#3#2 + y#3#1 <= 1
temini_38:
 -165 y#12#3 -165 ym#3 - zm#3 + z#12 >= -165
temini_39:
 -183 y#13#3 -183 ym#3 - zm#3 + z#13 >= -183
cfin_1:
 -100000 ym#1 -100000 y#1#1 +0.05 zm#1 -0.1 zl#1 - cost#1 + costl#1
 >= -199967.5
cfin_2:
 -100000 ym#1 -100000 y#2#1 +0.05 zm#1 -0.1 zl#1 - cost#2 + costl#1
 >= -199956.5
0 <= x#4#10 <= 1
0 <= x#4#11 <= 1
0 <= x#4#12 <= 1
0 <= x#4#13 <= 1
```

3.3 Risoluzione tramite Cplex

Riassumendo abbiamo scritto il modello in ZIMPL corretto, tradotto in .lp pronto per la lettura in Cplex e per poi, tramite il programma, esser ottimizzato e risolto.

Cominciamo con il far partire il programma *cplex.exe* e tramite il comando read facciamo leggere il file *macelli.lp.*

```
CPLEX> read
Name of file to read: c:\scip\macelli.lp
Problem 'c:\scip\macelli.lp' read.
Read time = 0.00 sec. (0.13 ticks)
CPLEX> _
```

Una volta letto il programma non ci resta altro che risolvere il problema digitando il comando ***optimize***. Il programma inizia ad elaborare e alla fine ci mostra in output l'esito dei calcoli. Ci avverte se la soluzione ottimale è stata trovata e il tempo impiegato per farlo.

```
Problem 'c:\scip\macelli.lp' read.
Read time = 0.00 sec. (0.13 ticks)
Tried aggregator 1 time.
MIP Presolve eliminated 40 rows and 0 columns.
MIP Presolve modified 278 coefficients.
Reduced MIP has 1210 rows, 155 columns, and 5550 nonzeros.
Reduced MIP has 120 binaries, 0 generals, 0 SOSs, and 0 indicators.
Presolve time = 0.02 sec. (2.66 ticks)
Probing time = -0.00 sec. (1.42 ticks)
Cover probing fixed 0 vars, tightened 55 bounds.
Tried aggregator 1 time.
MIP Presolve eliminated 132 rows and 0 columns.
MIP Presolve modified 2864 coefficients.
Reduced MIP has 1078 rows, 155 columns, and 4956 nonzeros.
Reduced MIP has 120 binaries, 0 generals, 0 SOSs, and 0 indicators.
Presolve time = 0.02 sec. (3.86 ticks)
Probing time = 0.00 sec. (0.50 ticks)
Clique table members: 52.
MIP emphasis: balance optimality and feasibility.
MIP search method: dynamic search.
Parallel mode: deterministic, using up to 4 threads.
Root relaxation solution time = 0.00 sec. (3.25 ticks)
```

```
      Nodes                                         Cuts/
   Node  Left     Objective  IInf  Best Integer    Best Bound    ItCnt     Gap

      0     0        0.0000    38                    0.0000       80
*     0+    0                         1782.4500        0.0000       80  100.00%
      0     0        0.0000    32     1782.4500      Cuts: 46      149  100.00%
*     0+    0                         1657.2000        0.0000      149  100.00%
      0     0        0.0000    35     1657.2000     Cuts: 111      212  100.00%
      0     0        0.0000    38     1657.2000      Cuts: 38      298  100.00%
*     0+    0                         1654.2000        0.0000      298  100.00%
*     0+    0                         1649.9500        0.0000      298  100.00%
*     0+    0                         1607.8000        0.0000      298  100.00%
      0     2        0.0000    16     1607.8000        0.0000      298  100.00%
Elapsed time = 0.33 sec. (86.16 ticks, tree = 0.01 MB, solutions = 5)
*    93    86      integral     0     1365.9500        0.0000     2838  100.00%
*   318+  259                          1330.3500        0.0000     6966  100.00%
*   714+  536                          1249.4000        0.0000    12299  100.00%
*   714+  536                          1246.5000        0.0000    12299  100.00%
    714   538      347.4063    27     1246.5000        0.0000    12299  100.00%
   1484   992      669.4378    26     1246.5000      311.5000    23522   75.01%
   2467  1603      400.0422    39     1246.5000      415.8465    42640   66.64%
   2767  1252      854.8438    31     1246.5000      513.7487    46505   58.78%
   3534  1109        cutoff           1246.5000      776.5257    58604   37.70%
   4452  1160     1057.3939    10     1246.5000      809.0000    72828   35.10%
   5122  1503      849.7000    23     1246.5000      829.7273    84794   33.44%
   5954  1898     1064.0329    31     1246.5000      852.2000    98108   31.63%
   8592  2971      997.0000    18     1246.5000      900.9875   150886   27.72%
Elapsed time = 6.72 sec. (3454.25 ticks, tree = 1.23 MB, solutions = 14)
  11243  3795        cutoff           1246.5000      933.2234   202534   25.13%
  14206  4523     1245.9000     4     1246.5000      958.2711   253283   23.12%
  17373  5180     1027.8000    10     1246.5000      981.3334   306106   21.27%
  20613  5877     1091.6881    11     1246.5000      996.3058   355156   20.07%
  23825  6527     infeasible           1246.5000     1010.6045   402405   18.92%
  26877  7067        cutoff           1246.5000     1024.7634   445703   17.79%
  30427  7488     1085.2241    10     1246.5000     1039.1500   495263   16.63%
  34070  7819     1117.3500     7     1246.5000     1049.1000   541561   15.84%
  37653  8002     1059.9000     9     1246.5000     1059.2644   589351   15.02%
* 37805  8005      integral     0     1220.1000     1060.2000   591317   13.11%
Elapsed time = 24.10 sec. (12081.47 ticks, tree = 4.42 MB, solutions = 15)
  41504  7344     1122.3000    11     1220.1000     1071.4292   640024   12.19%
  45401  6726        cutoff           1220.1000     1086.0000   691199   10.99%
  49487  5872        cutoff           1220.1000     1099.7791   743062    9.86%
  53707  4768     1212.5000     7     1220.1000     1115.3357   795748    8.59%
  58132  3174        cutoff           1220.1000     1135.8931   846974    6.90%
```

```
GUB cover cuts applied:  1
Cover cuts applied:  50
Implied bound cuts applied:  28
Flow cuts applied:  113
Mixed integer rounding cuts applied:  176
Gomory fractional cuts applied:  12

Root node processing (before b&c):
  Real time             =    0.33 sec. (85.73 ticks)
Parallel b&c, 4 threads:
  Real time             =   35.24 sec. (17417.10 ticks)
  Sync time (average)   =    0.00 sec.
  Wait time (average)   =    0.00 sec.
                          ------------
Total (root+branch&cut) =   35.57 sec. (17502.83 ticks)

Solution pool: 15 solutions saved.

MIP - Integer optimal, tolerance (0.0001/1e-006):  Objective = 1.2201000000e+003
Current MIP best bound = 1.2200000000e+003 (gap = 0.1, 0.01%)
Solution time =   35.57 sec.  Iterations = 882462  Nodes = 62687 (2)
Deterministic time = 17502.83 ticks  (492.09 ticks/sec)
```

Come si vede dall'output del programma, il nostro problema è stato risolto, e la soluzione ottimale è stata trovata in 35.57 secondi. Un tempo molto corto, considerato che il tempo impiegato per risolverlo in GAMS è stato all'incirca intorno ai 900 secondi e di 1267,18 secondi con SCIP.

3.4 Interpretazione della soluzione

L'ultimo passo consiste nel visualizzare ed interpretare la soluzione digitando il comando ***display solution***.

```
objective value:                1220.09999998668
y#1#1                              1   (obj:0)
y#2#1                              1   (obj:0)
y#3#1                              1   (obj:0)
y#4#3                              1   (obj:0)
y#5#3                              1   (obj:0)
y#6#1                              1   (obj:0)
y#7#1                              1   (obj:0)
y#8#1                              1   (obj:0)
y#9#1                              1   (obj:0)
y#10#3                             1   (obj:0)
y#11#1                             1   (obj:0)
y#12#3                             1   (obj:0)
y#13#3                             1   (obj:0)
ym#1                               1   (obj:0)
ym#2                               1   (obj:0)
ym#3                               1   (obj:0)
x#1#2                              1   (obj:0)
x#1#3                              1   (obj:0)
x#1#6                              1   (obj:0)
x#1#7                              1   (obj:0)
x#1#8                              1   (obj:0)
x#1#9                              1   (obj:0)
x#1#11                             1   (obj:0)
x#2#6                              1   (obj:0)
x#2#7                              1   (obj:0)
x#2#11                             1   (obj:0)
x#3#6                              1   (obj:0)
x#3#7                              1   (obj:0)
x#3#8                              1   (obj:0)
x#3#11                             1   (obj:0)
x#4#7                              1   (obj:0)
x#4#8                              1   (obj:0)
x#4#10                             1   (obj:0)
x#5#10                             1   (obj:0)
```

```
x#5#12                          1   (obj:0)
x#5#13                          1   (obj:0)
x#6#12                          1   (obj:0)
x#7#12                          1   (obj:0)
x#8#10                          1   (obj:0)
x#8#11                          1   (obj:0)
x#9#11                          1   (obj:0)
x#9#13                          1   (obj:0)
x#11#12                         1   (obj:0)
x#11#13                         1   (obj:0)
z#3                           463   (obj:0)
z#4                           690   (obj:0)
z#5                           246   (obj:0)
z#6                          1269   (obj:0)
z#7                          1119   (obj:0)
z#8                           685   (obj:0)
z#9                           286   (obj:0)
z#10                          854   (obj:0)
z#11                          993   (obj:0)
z#12                          554   (obj:0)
z#13                          400   (obj:0)
zm#2                          130   (obj:0)
zm#3                           72   (obj:0)
zl#1                         1407   (obj:0)
zl#2                         1740   (obj:0)
zl#3                          990   (obj:0)
cost#1          224.999999998079   (obj:0)
cost#2          401.999999995343   (obj:0)
cost#3          307.999999998079   (obj:0)
cost#4          324.999999999593   (obj:0)
cost#5          222.000000000233   (obj:0)
cost#6          543.999999994412   (obj:0)
cost#7          502.999999994063   (obj:0)
cost#8          353.999999995343   (obj:0)
cost#9          262.999999998079   (obj:0)
cost#10         369.999999992258   (obj:0)
cost#11         459.999999994063   (obj:0)
```

cost#12	292.000000000233 (obj:0)
cost#13	260.000000000233 (obj:0)
z#1	155 (obj:0)
costl#3	495.399999992258 (obj:1)
costl#1	724.699999994424 (obj:1)
zm#1	10 (obj:0)
z#2	850 (obj:0)

La prima riga ci mostra il nostro valore di *object value* ovvero il valore della funzione obiettivo del problema che nel nostro caso di minimizzazione risulta 1220.09999998668 ed è il risultato della somma dei *costl#k*, cioè i costi totali che ogni camion sostiene per ogni giro. 725,7 per il camion 1, 495,4 per il camion 3 e non essendoci *costl#2,* il camion 2 non ha alcun costo. Ovvero, il camion 2 sta fermo, non partendo dal deposito e non passando per alcun macello.
Le variabili *y#i#k* confermano i risultati già analizzati, le *i* rappresentano i macelli mentre le *k* il camion da cui è visitato. Il camion 1 passa dai macelli 1, 2, 3, 6, 7, 8, 9, 11, il camion 3 visita i macelli 4 ,5, 10, 12, 13 mentre, come previsto, il camion 2 non visita nessun macello.

Le variabili *x#i#j* invece rappresentano l'ordine con cui i camion visitano i macelli, ove la *i* rappresenta il macello precedente e la le lettera *j* il macello successivo. Si vede come il macello 1 precede i macelli 2, 3, 6, 7, 8, 9, 11; il macello 2 precede i macelli 6, 7, 11; il macello 3 precede i macelli 6, 7, 8, 11; il macello 4 precede il 10; il macello 5 precede i macelli 10, 12, 13; Il macello 8 precede l'11; il macello 9 precede l'11.

Le variabili non contemplate nell'analisi sono quelle che legano i macelli non visitati dallo stesso camion, quindi, per noi, senza significato.

Elaborando i dati possiamo definire i viaggi dei due camion:

1. k1 – i1 – i9 – i3 – i8 – i2 – i11 – i7 – i6 – l

2. k3 – i5 – i13 – i12 – i4 – i10 – l

Questi percorsi sono confermati anche dalle variabili *zm, z, zl*. Le *zm#k* segnalano il tempo di partenza del camion *k* dal deposito, le *z#i* rappresentano il tempo di arrivo del camion in ogni macello visitato; le *zl#k* indicano il tempo di arrivo del camion *k* alla conceria I.

I tempi, ovviamente, sono pari a quelli trovati dai percorsi precedentemente:

1. k1(10) - i1(155) – i9(286) – i3(463) – i8(685) – i2(850) – i11(993) – i7(1119) – i6(1269) – l(1407)

2. k3(72) – i5(246) – i13(400) – i12(554) – i4(690) – i10(854) – l(990)

Un altra conferma la si ha controllando i costi sostenuti e il correlato aumento ad ogni macello visitato con la variabile *cost#i:*

K1: 225(cost#1) - 263(cost#9) - 308(cost#3) - 354(cost#8) - 402(cost#2) - 460(cost#11) - 503(cost#7) - 554(cost#6) – 724,7(costl#1);
K3: 222(cost#5) – 260(cost#13) – 292(cost#12) – 325(cost#4) – 370(cost#10) – 495,4(costl#3);

3.5 Tabelle riassuntive:

Camion 1:

Percorso	K1	i1	i9	i3	i8	i2	i11	i7	i6	1
Tempi	10	155	286	463	685	850	993	1119	1269	1407
Costi	0	225	263	308	354	402	460	503	544	724,7

Camion 2:

Percorso	K3	i5	i13	i12	i4	i10	l
Tempi	72	246	400	554	690	854	990
Costi	0	222	260	292	325	370	495,4

4. Conclusioni

L'obiettivo della tesi è stato raggiunto sotto ogni punto di vista.
E' stato trovato un modo molto semplice ed efficace, tramite questa terna di programmi, Scip, ZIMPL e Cplex, per poter scrivere il programma risolutore di un problema complesso in un linguaggio alto, facilmente scrivibile dall'utente, e per poter esser trasformato poi in un linguaggio più basso elaborabile da ogni calcolatore tramite il più famoso Cplex. Con questo metodo si ha la possibilità inoltre di risparmiare tempo sia nella stesura che nella compilazione del programma. E' un metodo efficiente e che potrebbe avere grandi risvolti in future applicazioni.

Per quanto riguarda la risoluzione del problema *macelli*, dopo aver scritto un modello matematico opportuno, non ci son state grandi difficoltà nella stesura del testo del programma e, una volta elaborato, abbiamo trovato risultati concordi e veritieri. Le difficoltà maggiori possibili sono nella comprensione del problema e nell'evidenziare quali sono le variabili, i dati, i vincoli e la funzione obiettivo. Successivamente non serve altro che scrivere correttamente il problema in linguaggio ZIMPL, stando attenti alle regole di stesura, aiutandosi anche col manuale ZIMPL.
L'utilizzo di tale programmazione, attraverso questa triade di programmi è consigliata a utenze con un minimo di competenze informatiche, in particolar modo di programmazione C++. Tuttavia come utilizzo stand-alone è abbordabile per chiunque voglia risolvere dei problemi di programmazione lineare, intera o mista-intera, che abbia affinità con la ricerca operativa, basandosi sul manuale.

Spero, inoltre, che questa tesi possa servire a chi vorrà cimentarsi per la prima volta nell'uso di questi tre programmi, sia nell'uso del singolo e che nelle varie combinazioni.

Bibliografia

[1] Cornelius Schwarz: *An Introduction to SCIP* University of Bayreuth September 28, 2008

[2] Giorgio ROMANIN-JACUR (Department of Management and Engineering University of Padova, Italy), Carlo FILIPPI (Department of Quantitative Methods, University of Brescia, Italy): *FRESH BOVINE SKIN TRANSPORTATION FROM SLAUGHTERS TO TANNERY WITH SPECIAL TIME WINDOWS AND CAPACITY CONSTRAINTS* (*EURO 2012 VILNIUS)*

[3] Marc Pfetsch: *Introduction to SCIP* DFG Research Center MATHEON Konrad-zuse-Zentrum für Infromationstechnik Berlin November 10, 2007

[4] Thorsten Koch: *ZIMPL User Guide (Zuze Institute Mathematical Programming Language)* Konrad-zuse-Zentrum für Infromationstechnik Berlin October 24, 2011

Siti web

[5] http://scip.zib.de

[6] http://scip.zib.de/download.shtml

[7] http://zimpl.zib.de

[8] http://www-01.ibm.com/software/commerce/optimization/cplex-optimizer/

Ringraziamenti

Per un traguardo così importante le persone da ringraziare sono veramente tante. Innanzitutto il mio relatore, il professor Giorgio Romanin Jacur, per la Sua disponibilità e gentilezza durante il periodo di lavoro, sempre pronto a recuperare altro materiale e ad informarsi per aiutarmi nello svolgimento. Il ringraziamento più importante va alla mia famiglia, in particolare ai miei genitori e a mia sorella che in questi anni di università mi hanno sempre sostenuto e incoraggiato. Spero con questo risultato di ripagare tutti gli sforzi che hanno compiuto per me in questi anni e di renderli orgogliosi. Ringrazio tutti i miei amici che in questi anni mi son sempre stati accanto, in ogni modo possibile immaginabile, cercando di tirarmi su di morale in ogni momento ce ne fosse stato bisogno. Ringrazio inoltre i miei parenti, in particolar modo i miei nonni Sergio e Silvana e mio zio Fabio che mi son sempre stati accanto, sempre pronti a gioire con me e a spingermi a dare il massimo delle mie capacità. Ringrazio per ultima, ma sicuramene non meno importante (anzi), la mia ragazza Arianna, che mi ha sempre sostenuto, sempre pronta ad ascoltare i miei sfoghi, le mie lamentele e, fortunatamente, anche sempre pronta a gioire con me, per me.

Davide Gaspari

21 Marzo 2014