Sensitivity Analysis
Marginal Price of Constraints

• The dual prices, or *marginal prices* allow us to put a value on “resources” (broadly construed).

• Alternatively, they allow us to consider the sensitivity of the optimal solution value to changes in the input.

• Consider the bond portfolio problem.

• By examining the dual variable for the each constraint, we can determine the value of an extra unit of the corresponding “resource”.

• We can then determine the maximum amount we would be willing to pay to have a unit of that resource.

• The so-called “reduced costs” of the variables are the marginal prices associated with the bound constraints.
Marginal Prices in AMPL

Again, recall the simple bond portfolio model from Lecture 3.

```
ampl: model bonds.mod;
ampl: solve;
...
ampl: display rating_limit, cash_limit;
rating_limit = 1
cash_limit = 2
```

- This tells us that the **optimal marginal cost** of the `rating_limit` constraint is 1.
- What does this tell us about the “cost” of improving the average rating?
- What is the return on an extra $1K$ of cash available to invest?
Advanced Modeling Techniques

AMPL: Displaying Auxiliary Values with Suffixes

• In AMPL, it's possible to display much of the auxiliary information needed for sensitivity using suffixes.

• For example, to display the reduced cost of a variable, type the variable name with the suffix .rc.

• Recall again the short term financing example (short_term_financing.mod).

    ampl: display credit.rc;
    credit.rc [*] :=
        0  -0.003212
        1   0
        2  -0.0071195
        3  -0.00315
        4   0
        5   0
    ;

• How do we interpret this?
**AMPL: Sensitivity Ranges**

- **AMPL** does not have built-in sensitivity analysis commands.
- **AMPL/CPLEX** does provide such capability, however.
- To get sensitivity information, type the following

  ```ampl
  ampl: option cplex_options 'sensitivity';
  ```

- Solve the bond portfolio model:

  ```ampl
  ampl: solve;
  ...
  suffix up OUT;
  suffix down OUT;
  suffix current OUT;
  ```
**AMPL: Accessing Sensitivity Information**

Access sensitivity information using the suffixes `.up` and `.down`. This is from the model `bonds.mod`.

```ampl
ampl: display cash_limit.up, rating_limit.up, maturity_limit.up;
cash_limit.up = 102
rating_limit.up = 200
maturity_limit.up = 1e+20

ampl: display cash_limit.down, rating_limit.down, maturity_limit.down;
cash_limit.down = 75
rating_limit.down = 140
maturity_limit.down = 350

ampl: display buy.up, buy.down;
: buy.up buy.down :=
A  6  3
B  4  2
;
```
AMPL: Sensitivity for the Short Term Financing Model

ampl: short_term_financing.mod;
ampl: short_term_financing.dat;
ampl: solve;
ampl: display credit, credit.rc, credit.up, credit.down;

: credit credit.rc credit.up credit.down :=
0 0 -0.00321386 0.00321386 -1e+20
1 50.9804 0 0.00318204 0
2 0 -0.00711864 0.00711864 -1e+20
3 0 -0.00315085 0.00315085 -1e+20
4 0 0 0 0 -1e+20
;

### AMPL: Sensitivity for the Short Term Financing Model (cont.)

```ampl
AMPL: display bonds, bonds.rc, bonds.up, bonds.down;
:    bonds    bonds.rc    bonds.up    bonds.down    :=
 0 150 0 0.00399754 -0.00321386
 1 49.0196 0 0 -0.00318204
 2 203.434 0 0.00706931 0
 3 0 0 0.00706931 0
 4 0 0 0 0
;
```
## AMPL: Sensitivity for the Short Term Financing Model (cont.)

```ampl
ampl: display invest, invest.rc, invest.up, invest.down;

: invest invest.rc invest.up invest.down :=
-1 0 0 0 0
0 0 -0.00399754 0.00399754 -1e+20
1 0 -0.00714 0.00714 -1e+20
2 351.944 0 0.00393091 -0.0031603
3 0 -0.00391915 0.00391915 -1e+20
4 0 -0.007 0.007 -1e+20
5 92.4969 0 1e+20 2.76446e-14
;
```
Sensitivity Analysis of the Dedication Model

Let's look at the sensitivity information in the dedication model

```ampl
ampl: model dedication.mod;
ampl: data dedication.dat;
ampl: solve;
ampl: display cash_balance, cash_balance.up, cash_balance.down;
: cash_balance cash_balance.up cash_balance.down :=
1  0.971429 1e+20 5475.71
2  0.915646 155010 4849.49
3  0.883046 222579 4319.22
4  0.835765 204347 3691.99
5  0.656395 105306 2584.27
6  0.619461 123507 1591.01
7  0.5327 117131 654.206
8  0.524289 154630 0
;
```

How can we interpret these?
Sensitivity Analysis of the Dedication Model

ampl: display buy, buy.rc, buy.up, buy.down;

: buy buy.rc buy.up buy.down :=
A 62.1361 -1.42109e-14 105 96.4091
B 0 0.830612 1e+20 98.1694
C 125.243 -1.42109e-14 101.843 97.6889
D 151.505 1.42109e-14 101.374 93.2876
E 156.808 -1.42109e-14 102.917 80.7683
F 123.08 0 113.036 100.252
G 0 8.78684 1e+20 91.2132
H 124.157 0 104.989 92.3445
I 104.09 0 111.457 101.139
J 93.4579 0 94.9 37.9011
;

# Sensitivity Analysis of the Dedication Model

```ampl
display cash, cash.rc, cash.up, cash.down;
cash   cash.rc  cash.up  cash.down  :=
0  0  0.02385714  1e+20  0.971429
1  0  0.0557823  1e+20  -0.0557823
2  0  0.0326005  1e+20  -0.0326005
3  0  0.0472812  1e+20  -0.0472812
4  0  0.17937  1e+20  -0.17937
5  0  0.0369341  1e+20  -0.0369341
6  0  0.0867604  1e+20  -0.0867604
7  0  0.0084114  1e+20  -0.0084114
8  0  0.524289  1e+20  -0.524289
;
```
Sensitivity Analysis in PuLP and Pyomo

• Both PuLP and Pyomo also support sensitivity analysis through suffixes.

• Pyomo
  – The option `--solver-suffixes='.*'` should be used.
  – The supported suffixes are `.dual`, `.rc`, and `.slack`.

• PuLP
  – PuLP creates suffixes by default when supported by the solver.
  – The supported suffixed are `.pi` and `.rc`. 
Sensitivity Analysis of the Dedication Model with PuLP

```python
for t in Periods[1:]:
    prob += (cash[t-1] - cash[t]
            + lpSum(BondData[b, 'Coupon'] * buy[b]
                for b in Bonds if BondData[b, 'Maturity'] >= t)
            + lpSum(BondData[b, 'Principal'] * buy[b]
                for b in Bonds if BondData[b, 'Maturity'] == t)
            == Liabilities[t]), "cash_balance_%s"%t

status = prob.solve()

for t in Periods[1:]:
    print 'Present of $1 liability for period', t,
    print prob.constraints["cash_balance_%s"%t].pi
```
Tradeoff Analysis
(Multiobjective Optimization)
Analysis with Multiple Objectives

- In many cases, we are trying to optimize multiple criteria simultaneously.
- These criteria often conflict (risk versus reward).
- Often, we deal with this by placing a constraint on one objective while optimizing the other.
- Extending the principles from the sensitivity analysis section, we can consider doing a parametric analysis.
- We do this by varying the right-hand side systematically and determining how the objective function changes as a result.
- More generally, we may want to find all non-dominated solutions with respect to two or more objectives functions.
- This latter analysis is called multiobjective optimization.
Parametric Analysis with PuLP

(FinancialModels.xlsx:Bonds-Tradeoff-PuLP)

- Suppose we wish to analyze the tradeoff between yield and rating in our bond portfolio.
- By iteratively changing the value of the right-hand side of the constraint on the rating, we can create a graph of the tradeoff.
Nonlinear modeling
Portfolio Optimization

- An investor has a fixed amount of money to invest in a portfolio of \( n \) risky assets \( S^1, \ldots, S^n \) and a risk-free asset \( S^0 \).
- We consider the portfolio's return over a fixed investment period \([0, 1]\).
- The random return of asset \( i \) over this period is

\[
R_i := \frac{S^i_1}{S^i_0}.
\]

- In general, we assume that the vector \( \mu = \mathbb{E}[R] \) of expected returns is known.
- Likewise, \( Q = \text{Cov}(R) \), the variance-covariance matrix of the return vector \( R \), is also assumed to be known.
- What proportion of wealth should the investor invest in asset \( i \)?
Formulating the Portfolio Optimization Problem

**Decision variables**: \( x_i \), proportion of wealth invested in asset \( i \).

**Constraints**:

- the entire wealth is assumed invested, \( \sum_i x_i = 1 \),
- if short-selling of asset \( i \) is not allowed, \( x_i \geq 0 \),
- bounds on exposure to groups of assets, \( \sum_{i \in G} x_i \leq b \), . . .

**Objective function**: In general, the investor wants to maximize expected return while minimizing “risk.” What to do?

- Let \( R = [R_1 \ldots R_n]^\top \) be the random vector of asset returns and \( \mu = \mathbb{E}[R] \) the vector of their expectations.

- Then the random return of the portfolio \( y \) is

\[
\frac{\sum_i y_i S^i_1 - \sum_i y_i S^i_0}{\sum_i y_i S^i_0} = \sum_i \frac{y_i S^i_0}{\sum_i y_i S^i_0} \cdot \frac{S^i_1 - S^i_0}{S^i_0} = R^\top x.
\]
Trading Off Risk and Return

- To set up an optimization model, we must determine what our measure of “risk” will be.
- The goal is to analyze the tradeoff between risk and return.
- One approach is to set a target for one and then optimize the other.
- The classical portfolio model of Henry Markowitz is based on using the variance of the portfolio return as a risk measure:

\[ \sigma^2(R^\top x) = x^\top Q x, \]

where \( Q = \text{Cov}(R_i, R_j) \) is the variance-covariance matrix of the vector of returns \( R \).
- We consider three different single-objective models that can be used to analyze the tradeoff between these conflicting goals.
Three Markowitz Models

(M1) \[ \min_{x \in \mathbb{R}^n} x^\top Q x \]
\[ \text{s.t.} \quad \mu^\top x \geq r, \]
\[ \sum_{i=1}^{n} x_i = 1, \]

where \( r \) is a targeted minimum expected portfolio return.

(M2) \[ \max_{x \in \mathbb{R}^n} \mu^\top x \]
\[ \text{s.t.} \quad x^\top Q x \leq \sigma^2 \]
\[ \sum_{i=1}^{n} x_i = 1, \]

where \( \sigma^2 \) is the maximum risk the investor is willing to take on.
Three Markowitz Models (cont.)

\[(M3) \quad \max_{x \in \mathbb{R}^n} \mu^\top x - \lambda x^\top Q x \]

\[\text{s.t.} \quad \sum_{i=1}^{n} x_i = 1,\]

where \(\lambda > 0\) is a risk-aversion parameter.

- All three models are examples of \textit{quadratic optimization problems},
- Also, since \(Q\) is a positive semidefinite symmetric matrix, then \(x \mapsto x^\top Q x\)
  is a convex function.
- Hence, these are actually \textit{convex quadratic programs}.
- Convex quadratic programs can generally be solved efficiently.
Modeling Nonlinear Programs

- Both AMPL and Pyomo support the inclusion of nonlinear functions in the model.
- In both cases, a wide range of built-in functions are available.
- By restricting the form of the nonlinear functions, we ensure that the Hessian can be easily calculated.
- The solvers ipopt, bonmin, and couenne can be used to solve the models.
- See
  - portfolio-*.mod,
  - portfolio-*.py,
  - FinancialModels.xlsx:Portfolio-AMPL, and
Getting the Data

• One of the most compelling reasons to use Python for modeling is that there are a wealth of tools available.

• Historical stock data can be easily obtained from Yahoo using built-in Internet protocols.

• Here, we use a small Python package for getting Yahoo quotes to get the price of a set of stocks at the beginning of each year in a range.

• See FinancialModels.xlsx:Portfolio-Pyomo-Live.

```python
for s in stocks:
    for year in range(1993, 2014):
        quote[year, s] = YahooQuote(s,'%s-01-01'%(str(year)),
                                     '%s-01-08'%(str(year)))
        price[year, s] = float(quote[year, s].split(', ')[5])
        break
```
The Efficient Frontier

• We can assume without loss of generality that $Q \succ 0$, so we have $\sigma_{\text{min}} > 0$, where

$$\sigma_{\text{min}}^2 := \min_x x^\top Q x$$

s.t. \quad \mu^\top x \geq r,$$

$$\sum_{i=1}^n x_i = 1,$$

• Let

$$(R) \quad r(\sigma) = \max_x \mu^\top x$$

s.t. \quad Ax \geq a$$

$$Bx = b$$

$$x^\top Q x \leq \sigma^2,$$

and note that for $\sigma \geq \sigma_{\text{min}}$ the function $r(\sigma)$ is well-defined.
The Efficient Frontier

Note that \( \mu^\top x \leq r(\sqrt{x^\top Qx}) \) for all feasible \( x \), and that it can never make sense to hold a portfolio \( x \) for which

\[
\mu^\top x < r(\sqrt{x^\top Qx}),
\]

since the portfolio \( x^* \) obtained from solving problem (R) with \( \sigma^2 = x^\top Qx \) would yield the more desirable expected return

\[
\mu^\top x^* = r(\sqrt{x^\top Qx}).
\]

Definition 1. **Portfolios that satisfy the relation**

\[
\mu^\top x = r(\sqrt{x^\top Qx})
\]

are called efficient. **The curve** \( \sigma \mapsto r(\sigma) \), defined for \( \sigma \geq \sigma_{\text{min}} \), is called the efficient frontier.
Efficient Frontier for the DJIA Data Set
Integer Programming
Constructing an Index Fund

- An index is essentially a proxy for the entire universe of investments.
- An index fund is, in turn, a proxy for an index.
- A fundamental question is how to construct an index fund.
- It is not practical to simply invest in exactly the same basket of investments as the index tracks.
  - The portfolio will generally consist of a large number of assets with small associated positions.
  - Rebalancing costs may be prohibitive.
- A better approach may be to select a small subset of the entire universe of stocks that we predict will closely track the index.
- This is what index funds actually do in practice.
A Deterministic Model

• The model we now present attempts to cluster the stocks into groups that are “similar.”

• Then one stock is chosen as the representative of each cluster.

• The input data consists of parameters $\rho_{ij}$ that indicate the similarity of each pair $(i, j)$ of stocks in the market.

• One could simply use the correlation coefficient as the similarity parameter, but there are also other possibilities.

• This approach is not guaranteed to produce an efficient portfolio, but should track the index, in principle.
An Integer Programming Model

- We have the following variables:
  - $y_j$ is stock $j$ is selected, 0 otherwise.
  - $x_{ij}$ is 1 if stock $i$ is in the cluster represented by stock $j$, 0 otherwise.

- The objective is to maximize the total similarity of all stocks to their representatives.

- We require that each stock be assigned to exactly one cluster and that the total number of clusters be $q$. 
An Integer Programming Model

Putting it all together, we get the following formulation

\[
\begin{align*}
\max & \quad \sum_{i=1}^{n} \sum_{j=1}^{n} \rho_{ij}x_{ij} \\
\text{s.t.} & \quad \sum_{j=1}^{n} y_j = q \\
& \quad \sum_{j=1}^{n} x_{ij} = 1 \quad \forall i = 1, \ldots, n \\
& \quad x_{ij} \leq y_j \quad \forall i = 1, \ldots, n, j = 1, \ldots, n \\
& \quad x_{ij}, y_j \in \{0, 1\} \quad \forall i = 1, \ldots, n, j = 1, \ldots, n
\end{align*}
\]

See `IndexFund-Pyomo.py` for model.
Interpreting the Solution

• As before, we let \( \hat{w} \) be the relative market-capitalized weights of the selected stocks

\[
\hat{w}_i = \frac{\sum_{j=1}^{n} z_i S^i x_{ij}}{\sum_{i=0}^{n} \sum_{j=1}^{n} z_i S^i x_{ij}},
\]

where \( z_i \) is the number of shares of asset \( i \) that exist in the market and \( S^i \) the value of each share.

• This portfolio is what we now use to track the index.

• Note that we could also have weighted the objective by the market capitalization in the original model:

\[
\max \sum_{i=1}^{n} \sum_{j=1}^{n} z_i S^i \rho_{ij} x_{ij}
\]
Effect of $K$ on Performance of Index Fund

- This is a chart showing how the performance of the index changes as its size is increased.
- This is for an equal-weighted index and the performance metric is sum of squared deviations.
Stochastic Programming
Building a Retirement Portfolio

- When I retire in 10 years or so :-), I would like to have a comfortable income.

- I’ll need enough savings to generate the income I’ll need to support my lavish lifestyle.

- One approach would be to simply formulate a mean-variance portfolio optimization problem, solve it, and then “buy and hold.”

- This doesn’t explicitly take into account the fact that I can periodically rebalance my portfolio.

- I may make a different investment decision today if I explicitly take into account that I will have recourse at a later point in time.

- This is the central idea of stochastic programming.
Modeling Assumptions

• In $Y$ years, I would like to reach a savings goal of $G$.

• I will rebalance my portfolio every $v$ periods, so that I need to have an investment plan for each of $T = Y/v$ periods (stages).

• We are given a universe $\mathcal{N} = \{1, \ldots, n\}$ of assets to invest in.

• Let $\mu_{it}, i \in \mathcal{N}, t \in T = \{1, \ldots, T\}$ be the (mean) return of investment $i$ in period $t$.

• For each dollar by which I exceed my goal of $G$, I get a reward of $q$.

• For each dollar I am short of $G$, I get a penalty of $p$.

• I have $B$ to invest initially.
Variables

- $x_{it}, i \in \mathcal{N}, t \in \mathcal{T}$: Amount of money to invest in asset $i$ at beginning of period $t$.
- $z$: Excess money at the end of horizon.
- $w$: Shortage in money at the end of the horizon.
A Naive Formulation

minimize

\[ qz + pw \]

subject to

\[ \sum_{i \in \mathcal{N}} x_{i1} = B \]

\[ \sum_{i \in \mathcal{N}} x_{it} = \sum_{i \in \mathcal{N}} (1 + \mu_{it}) x_{i,t-1} \quad \forall t \in \mathcal{T} \]

\[ \sum_{i \in \mathcal{N}} (1 + \mu_{iT}) x_{iT} - z + w = G \]

\[ x_{it} \geq 0 \quad \forall i \in \mathcal{N}, t \in \mathcal{T} \]

\[ z, w \geq 0 \]
A Better Model

• What are some weaknesses of the model on the previous slide?
• Well, there are many...
• For one, it doesn’t take into account the variability in returns (i.e., risk).
• Another is that it doesn’t take into account my ability to rebalance my portfolio after observing returns from previous periods.
• I can and would change my portfolio after observing the market outcome.
• Let’s use our standard notation for a market consisting of \( n \) assets with the price of asset \( i \) at the end of period \( t \) being denoted by the random variable \( S_t^i \).
• Let \( R_{it} = \frac{S_t^i}{S_{t-1}^i} \) be the return of asset \( i \) in period \( t \).
• As we have done previously, let’s take a scenario approach to specifying the distribution of \( R_{it} \).
Scenarios

- We let the scenarios consist of all possible sequences of outcomes.
- Generally, we assume that for a particular realization of returns in period \( t \), there will be \( M \) possible realizations for returns in period \( t + 1 \).
- We then have \( M^T \) possible scenarios indexed by a set \( S \).
- As before, we can then assume that we have a probability space \((P^t, \Omega^t)\) for each period \( t \) and that \( \Omega^t \) is partitioned into \( |S| \) subsets \( \Omega^t_s, s \in S \).
- We then let \( p^t_s = P(\Omega^t_s) \forall s \in S, t \in T \).
- For instance, if \( M = 4 \) and \( T = 3 \), then we might have...

<table>
<thead>
<tr>
<th>( t = 1 )</th>
<th>( t = 2 )</th>
<th>( t = 3 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 1 1</td>
<td>1 1 1</td>
<td>1 1 1</td>
</tr>
<tr>
<td>1 1 2</td>
<td>1 1 2</td>
<td>1 1 2</td>
</tr>
<tr>
<td>1 1 3</td>
<td>1 1 3</td>
<td>1 1 3</td>
</tr>
<tr>
<td>1 1 4</td>
<td>1 1 4</td>
<td>1 1 4</td>
</tr>
<tr>
<td>1 2 1</td>
<td>1 2 1</td>
<td>1 2 1</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>4 4 4</td>
<td>4 4 4</td>
<td>4 4 4</td>
</tr>
</tbody>
</table>

- \( |S| = 64 \)
- We can specify any probability on this outcome space that we would like.
- The time period outcomes don’t need to be equally likely and returns in different time periods need not be mutually independent.
• Essentially, we are approximating the continuous probability distribution of returns using a discrete set of outcomes.

• Conceptually, the sequence of random events (returns) can be arranged into a tree.
Making it Stochastic

- Once we have a distribution on the returns, we could add uncertainty into our previous model simply by considering each scenario separately.

- The variables now become
  - $x_{its}, i \in \mathcal{N}, t \in \mathcal{T}$: Amount of money to reinvest in asset $i$ at beginning of period $t$ in scenario $s$.
  - $z_s, s \in S$: Excess money at the end of horizon in scenario $s$.
  - $w_s, s \in S$: Shortage in money at the end of the horizon in scenario $s$.

- Note that the return $\mu_{its}$ is now indexed by the scenario $s$. 
A Stochastic Version: First Attempt

minimize

subject to

\[ \sum_{i \in N} x_{i1} = B \]
\[ \sum_{i \in N} x_{its} = \sum_{i \in N} (1 + \mu_{its}) x_{i,t-1,s} \quad \forall t \in T, \forall s \in S \]
\[ \sum_{i \in N} \mu_{iTs} x_{iTs} - z_s + w_s = G \quad \forall s \in S \]
\[ x_{its} \geq 0 \quad \forall i \in N, t \in T, \forall s \in S \]
\[ z_s, w_s \geq 0 \quad \forall s \in S \]
Easy, Huh?

• We have just converted a multi-stage stochastic program into a deterministic model.
• However, there are some problems with our first attempt.
• What are they?
One Way to Fix It

- What we did to create our \textit{deterministic equivalent} was to create copies of the variables for every scenario at every time period.

- One missing element is that we still have not have a notion of a probability distribution on the scenarios.

- But there’s an even bigger problem...

- We need to enforce \textit{nonanticipativity}...

- Let’s define $E_s^t$ as the set of scenarios with same outcomes as scenario $s$ up to time $t$.

- At time $t$, the copies of all the anticipative decision variables corresponding to scenarios in $E_s^t$ must have the same value.

- Otherwise, we will essentially be making decision at time $t$ using information only available in periods after $t$. 
A Stochastic Version: Explicit Nonanticipativity

minimize

$$\sum_{s \in S} p_s (qz_s - pw_s)$$

subject to

$$\sum_{i \in N} x_{i1} = B$$

$$\sum_{i \in N} x_{its} = \sum_{i \in N} (1 + \mu_{its}) x_{i,t-1,s} \quad \forall t \in T, \forall s \in S$$

$$\sum_{i \in N} \mu_{iTs} x_{iTs} - z_s + w_s = G \quad \forall s \in S$$

$$x_{its} = x_{its'} \quad \forall i \in N, \forall t \in T, \forall s \in S, \forall s' \in E_s^t$$

$$x_{its} \geq 0 \quad \forall i \in N, t \in T, \forall s \in S$$

$$z_s, w_s \geq 0 \quad \forall s \in S$$
Another Way

• We can also enforce nonanticipativity by using the “right” set of variables.
• We have a vector of variables for each node in the scenario tree.
• This vector corresponds to what our decision would be, given the realizations of the random variables we have seen so far.
• Index the nodes $\mathcal{L} = \{1, 2, \ldots, \mathcal{L}\}$.
• We will need to know the “parent” of any node.
• Let $A(l)$ be the ancestor of node $l \in \mathcal{L}$ in the scenario tree.
• Let $N(t)$ be the set of all nodes associated with decisions to be made at the beginning of period $t$. 
Another Multistage Formulation

maximize

\[ \sum_{l \in N(T)} p_l (qz_l + pw_l) \]

subject to

\[ \sum_{i \in N} x_{i1} = B \]

\[ \sum_{i \in N} x_{il} = \sum_{i \in N} (1 + \mu_{il})x_{i,A(l)} \quad \forall l \in \mathcal{L} \]

\[ \sum_{i \in N} \mu_{il}x_{il} - z_l + w_l = G \quad \forall l \in N(T) \]

\[ x_{il} \geq 0 \quad \forall i \in N, l \in \mathcal{L} \]

\[ z_l, w_l \geq 0 \quad \forall l \in N(T) \]

See `DE-PuLP.py` for full model.