Overview of the course

Topics and dates

- 30/9 - 14/10: Optimization in Telecommunication (TKS)
- 28/10 - 11/11: Discrete Location and Network design (SHG, focus on transportation networks)
- 18/11 - 2/12: Topic unknown (ADH)

Assignments – Course evaluation

There will be at least one assignment in GSPP and one in Telecommunication.

Overview

Why look at the Set Partitioning Problem?

- The Set Partitioning Problem has been used to model many different situations – and with success.
- For many problems the bounds on an SPP formulation is “very tight”.
- So in face of a new problem it could be a good idea to ask: “Can this be modelled as a Set Partitioning Problem?” If the answer is yes there is a good chance of success – also for practical applications.
- The amount of literature on the Set Partitioning Problem and applications is huge.....
The Set Partitioning Problem

Let \( I = \{1, 2, \ldots, m\} \) and \( S = \{S_1, S_2, \ldots, S_n\} \).

Let \( P \subset \{1, 2, \ldots, n\} \). Then \( P \) defines a partition of \( I \) iff

1. \( \bigcup_{j \in P} S_j = I \)
2. \( S_j \cap S_k = \emptyset \) \( \forall j, k \in P, j \neq k \)

Let \( c_j \) be the cost associated with \( S_j \). Then \( \sum_{j \in P} c_j \) is the cost of a partition \( P \).

In the Set Partitioning Problem (SPP) the objective is given \( S \) find the minimal cost partition \( P^* \) of \( I \).

The SPP can be described by a matrix representation.

First let a vector (column) represent a subset \( S_j \).

The vector contains only 0’s and 1’s. The size of the vector is equal to \( m \).

The \( i \)'th element in the vector is 1 if \( i \) is in \( S_j \) and 0 otherwise.
Let these columns form a matrix $A$

- There exists a one-to-one correspondence between $S_j$ and column $j$.
- Associate a 0-1 variable $x_j$ with column $j$.
- We can write the constraints as $Ax = e = (1, 1, 1, \ldots, 1)^T$.

The solution from the example can be written as $x = (0, 1, 0, 1, 0, 0, 1, 0, 0, 1, 1, 1)$.

A closely related problem to the SPP, that is also used in many practical applications is the Set Covering Problem (SCP):

$$
\begin{align*}
\text{min} & \quad z = c^T x \\
\text{subject to} & \quad Ax \geq e = (1, 1, 1, \ldots, 1)^T \\
& \quad x \in \{0, 1\}^n
\end{align*}
$$

Set Packing

Having $\leq$ instead of $\geq$ gives us the **Set Packing Problem**.
SPP in practice

Sometimes in practice:
- not all constraints are equalities – they might be covering (≥) or packing (≤).
- Some right-hand-side values might not be unit values (but still integer).
- A few entries in the constraint matrix might not be unit values (but small integers).
- Such deviations from the pure SPP will be denoted Generalised Set Partitioning Problems (GSPP).

Important observations
- The SPP is NP-hard.
- Even finding a feasible solution to a SPP is NP-hard.
- The SCP is in many ways more easy to solve – x = (1, 1, 1, ..., 1) is a (trivial) feasible solution.

Matrix reductions in Set Partitioning

If \( e_i^T A = e_k^T \) for some \( i \) and \( k \) then \( x_k = 1 \) in every feasible partition. We can remove \( x_k \) and column \( k \) from \( A \). We can also remove every row \( t \) such that \( a_{tk} = 1 \).

If \( e_i^T A \geq e_p^T A \) for some \( t \) and \( p \), we say that row \( p \) dominates row \( t \) since by covering row \( p \) we must automatically also cover row \( t \). In this case we can remove row \( t \) from the problem and in the SPP case, we can also remove all columns \( j \) for which \( a_{tj} = 1 \) and \( a_{pj} = 0 \) because \( x_j \) just be zero.

If for some set of columns \( S \) and some column \( k \) not included in \( S \), we have \( \sum A e_j = A e_k \) and \( \sum c_j \leq c_k \) then we can cover the rows covered by \( A e_k \) more cheaply by the columns in \( S \). We say the columns of \( S \) dominate \( A e_k \).

A scheduling problem

We have 6 assignments A, B, C, D, E and F, that needs to be carried out. For every assignment we have a start time and a duration (in hours).

<table>
<thead>
<tr>
<th>Assignment</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>Start</td>
<td>0.0</td>
<td>1.0</td>
<td>2.0</td>
<td>2.5</td>
<td>3.5</td>
<td>5.0</td>
</tr>
<tr>
<td>Duration</td>
<td>1.5</td>
<td>2.0</td>
<td>2.0</td>
<td>2.0</td>
<td>2.0</td>
<td>1.5</td>
</tr>
</tbody>
</table>
Objective

A Workplan

A workplan is a set of assignments. Now we want to formulate a mathematical model that finds the cheapest set of workplans that fulfill all the assignments.

Workplan rules

- A workplan can not consist of assignments that overlap each other.
- The length $L$ of a workplan is equal to the finish time of the last assignment minus start of the first assignments plus 30 minutes for checking in and checking out.
- The cost of a workplan is $\max(4.0, L)$.

A View of the Assignments

<table>
<thead>
<tr>
<th>Start</th>
<th>Duration</th>
</tr>
</thead>
<tbody>
<tr>
<td>A 0.0</td>
<td>1.5</td>
</tr>
<tr>
<td>B 1.0</td>
<td>2.0</td>
</tr>
<tr>
<td>C 2.0</td>
<td>2.0</td>
</tr>
<tr>
<td>D 2.5</td>
<td>2.0</td>
</tr>
<tr>
<td>E 3.5</td>
<td>2.0</td>
</tr>
<tr>
<td>F 5.0</td>
<td>1.5</td>
</tr>
</tbody>
</table>

Workplans starting with assignment A

Completing the model
The Mathematical Model

We get the following model:

\[
\begin{align*}
& \text{min} & \sum_{j=1}^{N} c_j x_j \\
& \text{s.t.} & \sum_{j=1}^{N} a_{ij} x_j = 1 & \forall i \\
& & x_j \in \{0, 1\}
\end{align*}
\]

- \(x_j = 1\) if we use workplan \(j\) and 0 otherwise (variable).
- \(c_j\) is equal to the cost of the plan (parameter).
- \(a_{ij}\) = 1 if assignment \(i\) is included in workplan \(j\) (parameter).

Solving the problem

- A feasible solution is: \(x_3 = 1, x_9 = 1, x_{13} = 1\) with a solution value of 16.
- An optimal solution is \(x_2 = 1, x_9 = 1, x_{14} = 1\) with a solution value of 13.
- In general the integrality constraint makes the problem much more difficult to solve than if it was a linear program.
- Another challenge is the number of variables/workplans. On this small example we only have 16 with is manageable, but real-life problems does not only have 6 assignments.

Column Generation

- If a problem has many variables/columns but relatively few constraints, then column generation may be beneficial.
- Column generation can be seen as a special case of Dantzig-Wolfe decomposition.
- But it is much easier to view column generation in view of the simplex algorithm.

Column Generation via the Simplex Algorithm

Simplex algorithm

- Recall from the simplex algorithm:
  - The dual values associated with a solution are given by the vector \(\alpha^T = c_B^{-1} B^{-1}\).
  - The reduced cost of a column \(A_j\) is given by \(\tilde{c}_j = c_j - c_B^{-1} B^{-1} A_j = c_j - \alpha^T A_j\).
  - So given \(\alpha\), we must determine whether there exists a column \(A_j\) such that \(\alpha^T A_j\) is “favorable”.

Specific for our SPP/SCP

- Translated into our case we get:
  - Let \(P_j\) be the set of indices for which we for column \(j\) has a 1 in the column.
  - Now the reduced cost can be written as \(\tilde{c}_j = c_j - \alpha^T A_j = c_j - \sum_{i \in P_j} \alpha_i\)
Column Generation as a Simplex Move

- Any column with negative reduced cost is “favorable”.
- Often we will look for the most negative one, which means that we end up solving an optimization problem.
- As we need an efficient way of solving the subproblem we might use heuristics instead of optimal methods.
- This optimization problem is called the pricing problem. Given a set of dual values, identify a column that has a favorable reduced cost or indicate that no such column exist.
- We need an initial feasible solution in order to compute the first set of duals.

The Column Generation Framework

```
MASTER PROBLEM

Duals [\rightarrow Columns

SUB PROBLEM

Integer Solutions
```

Finally in order to get integer solutions we need to “encapsulate” the column generation in a branch and bound framework. The total setup is often called Branch and Price.

A Network Design Problem

- Wired telecommunication networks are usually organized in a hierarchal structure based on two or more layers.
- The two layers in the network are denoted the backbone network and the access network.

```
A Network Design Problem

When designing hierarchal networks, a number of related questions have to be resolved:
  - Which nodes should be hubs,
  - how should we define the clusters,
  - and which interconnections should we allow.

Theses problems are interrelated, they should be addressed by an integrated approach in order to ensure an optimal solution.

We consider the joint selection of hubs and clustering of nodes of two-layered networks. In each of the layers, we assume the networks to be fully interconnected.
```
First model for FINDP I

Variables

- $x_{ij} = 1$ if $ij$ is a link in the access network.
- $y_{ij}$ correspondingly for the backbone.
- $h_i = 1$ if $i$ is a hub 0 otherwise.

Objective function

$$\min \sum_{ij \in E} c_{ij}x_{ij} + \sum_{ij \in E} c_{ij}y_{ij}$$

First model for FINDP II

Constraints

A link cannot be used in both the backbone and the access network.

$$y_{ij} + x_{ij} \leq 1 \ \forall ij \in E$$

If both $i$ and $j$ are hubs there can be no connection between them in the access network.

$$h_i + h_j + x_{ij} \leq 2 \ \forall ij \in E$$

If $k$ is a not hub there can be not backbone connections with $k$.

$$y_{ij} \leq h_k \ \forall ij \in E, k \in \{i, j\}$$

Ensure fully connectedness

$$x_{ik} + x_{jk} \leq x_{ij} + 1 \ \forall i, j, k \in V, i < j, k \neq i, k \neq j$$

$$y_{ik} + y_{jk} \leq y_{ij} + 1 \ \forall i, j, k \in V, i < j, k \neq i, k \neq j$$

First model for FINDP III

More constraints...

This formulation does however not ensure that each cluster contains a hub. Therefore we introduce $w_{ij} = h_i x_{ij}$, and then we can ensure by linear constraints:

$$w_{ij} \leq h_i \ \forall i, j \in V, i \neq j$$

$$w_{ij} \leq x_{ij} \ \forall i, j \in V, i \neq j$$

$$h_i + x_{ij} \leq 1 + w_{ij} \ \forall i, j \in V, i \neq j$$

$$h_j + \sum_{i \neq j} w_{ij} = 1 \ \forall ij \in E$$

First model for FINDP IV

...and finally the last constraints

$$b_{\min} \leq \sum_i h_i \leq b_{\max}$$

$$v_{\min} - 1 \leq \sum_j x_{ij} \leq v_{\max} - 1 \ \forall i \in V$$

$$x_{ij}, y_{ij}, w_{ij}, h_i \text{ binary}$$
Why not stick to this model?

### Euclidean

<table>
<thead>
<tr>
<th>Problem</th>
<th>LP-FINDP</th>
</tr>
</thead>
<tbody>
<tr>
<td>(n)</td>
<td>(B_d)</td>
</tr>
<tr>
<td>10</td>
<td>1</td>
</tr>
<tr>
<td>10</td>
<td>2</td>
</tr>
<tr>
<td>10</td>
<td>3</td>
</tr>
<tr>
<td>15</td>
<td>1</td>
</tr>
<tr>
<td>15</td>
<td>2</td>
</tr>
<tr>
<td>15</td>
<td>3</td>
</tr>
<tr>
<td>20</td>
<td>1</td>
</tr>
<tr>
<td>20</td>
<td>2</td>
</tr>
<tr>
<td>20</td>
<td>3</td>
</tr>
<tr>
<td>25</td>
<td>1</td>
</tr>
<tr>
<td>25</td>
<td>2</td>
</tr>
<tr>
<td>25</td>
<td>3</td>
</tr>
</tbody>
</table>

### Random

<table>
<thead>
<tr>
<th>Problem</th>
<th>LP-FINDP</th>
</tr>
</thead>
<tbody>
<tr>
<td>(n)</td>
<td>(B_d)</td>
</tr>
<tr>
<td>10</td>
<td>1</td>
</tr>
<tr>
<td>10</td>
<td>2</td>
</tr>
<tr>
<td>10</td>
<td>3</td>
</tr>
<tr>
<td>15</td>
<td>1</td>
</tr>
<tr>
<td>15</td>
<td>2</td>
</tr>
<tr>
<td>15</td>
<td>3</td>
</tr>
<tr>
<td>20</td>
<td>1</td>
</tr>
<tr>
<td>20</td>
<td>2</td>
</tr>
<tr>
<td>20</td>
<td>3</td>
</tr>
<tr>
<td>25</td>
<td>1</td>
</tr>
<tr>
<td>25</td>
<td>2</td>
</tr>
<tr>
<td>25</td>
<td>3</td>
</tr>
</tbody>
</table>

How do we formulate it as a SPP?

#### Key observations
- The key observation here is the independence of the cost of an access network from the backbone network and other access networks.
- The same observation can be made for the cost of the backbone network.
- We define two different types of columns: one representing an access network and one representing a backbone network.

Columns in the model

### Access network
A column for the access network must contain information on which nodes are in the access network and which of them is a hub.

### Backbone network
A column for the backbone network just need information on which nodes are in the backbone network.

### Linking
We need to link the chosen access networks to the backbone network.

A example of the final model

\[
\begin{align*}
a & : 1 \ 1 \ 1 \ 1 \ 1 \ 1 \ = 1 \\
b & : 1 \ 1 \ 1 \ 1 \ 1 \ 1 \ = 1 \\
c & : 1 \ 1 \ 1 \ 1 \ 1 \ 1 \ = 1 \\
d & : 1 \ 1 \ 1 \ 1 \ 1 \ 1 \ = 1 \\
a & : -1 \ -1 \ -1 \ 1 \ 1 \ 1 \ = 0 \\
b & : -1 \ -1 \ -1 \ -1 \ 1 \ 1 \ = 0 \\
c & : -1 \ -1 \ -1 \ -1 \ 1 \ 1 \ = 0 \\
d & : -1 \ -1 \ -1 \ -1 \ -1 \ 1 \ = 0
\end{align*}
\]
A SPP for the FINDP

\[
\begin{align*}
\min & \sum_{c \in C} c_c u_c + \sum_{b \in B} c_b v_b \\
\text{s.t.} & \sum_{c \in C} a_c^i u_c = 1 \quad i \in V \\
& \sum_{c \in C} s_c^i u_c + \sum_{b \in B} s_b^i v_b = 0 \quad i \in V \\
\end{align*}
\]

Variables are \( u_c = 1 \) if cluster \( c \) in \( C \) is selected, 0 otherwise and \( v_b = 1 \) if backbone \( b \) in \( B \) is selected, 0 otherwise.

Overview of CG-FINDP

Comparison of models

<table>
<thead>
<tr>
<th>Problem</th>
<th>LP-FINDP</th>
<th>CG-FINDP</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Seconds</td>
<td>Gap (%)</td>
</tr>
<tr>
<td>Euclidean</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10 1</td>
<td>0.15</td>
<td>52.6</td>
</tr>
<tr>
<td>10 2</td>
<td>0.12</td>
<td>61.1</td>
</tr>
<tr>
<td>10 3</td>
<td>0.15</td>
<td>69.9</td>
</tr>
<tr>
<td>15 1</td>
<td>1.15</td>
<td>31.3</td>
</tr>
<tr>
<td>15 2</td>
<td>1.31</td>
<td>60.0</td>
</tr>
<tr>
<td>15 3</td>
<td>0.38</td>
<td>73.3</td>
</tr>
<tr>
<td>20 1</td>
<td>1.69</td>
<td>57.0</td>
</tr>
<tr>
<td>20 2</td>
<td>1.16</td>
<td>65.6</td>
</tr>
<tr>
<td>20 3</td>
<td>1.36</td>
<td>76.0</td>
</tr>
<tr>
<td>25 1</td>
<td>6.39</td>
<td>27.1</td>
</tr>
<tr>
<td>25 2</td>
<td>4.73</td>
<td>56.5</td>
</tr>
<tr>
<td>25 3</td>
<td>5.07</td>
<td>70.8</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Problem</th>
<th>LP-FINDP</th>
<th>CG-FINDP</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Seconds</td>
<td>Gap (%)</td>
</tr>
<tr>
<td>Random</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10 1</td>
<td>0.16</td>
<td>66.5</td>
</tr>
<tr>
<td>10 2</td>
<td>0.14</td>
<td>74.4</td>
</tr>
<tr>
<td>10 3</td>
<td>0.13</td>
<td>78.5</td>
</tr>
<tr>
<td>15 1</td>
<td>1.74</td>
<td>49.6</td>
</tr>
<tr>
<td>15 2</td>
<td>1.29</td>
<td>76.0</td>
</tr>
<tr>
<td>15 3</td>
<td>0.47</td>
<td>80.4</td>
</tr>
<tr>
<td>20 1</td>
<td>1.81</td>
<td>56.8</td>
</tr>
<tr>
<td>20 2</td>
<td>1.09</td>
<td>80.2</td>
</tr>
<tr>
<td>20 3</td>
<td>1.12</td>
<td>88.9</td>
</tr>
<tr>
<td>25 1</td>
<td>6.61</td>
<td>48.5</td>
</tr>
<tr>
<td>25 2</td>
<td>4.97</td>
<td>76.2</td>
</tr>
<tr>
<td>25 3</td>
<td>4.64</td>
<td>83.7</td>
</tr>
</tbody>
</table>
The Berth Scheduling Problem

Background – Container shipping
- The world's first dedicated containership was put in service in 1951 in Denmark.
- Today the world's total fleet is 23.3 million TEUs.
- In total 440 million TEUs went through the container terminals in the world in 2007.

Masters project
The modelling and work presented here was done in a masters project by Clement Gram Christensen and Cecilie Holst.

The Berth Scheduling Problem

Problem definition
- When and where are the ships going to be berthed?
  - Minimize total weighted flow time (departure time minus arrival time).
  - All ships must be berthed at some time.
  - Each place of the berth can only handle one ship at a time.
  - Service times along the berth can vary.
- The problem comes in different “flavors”:
  - Static vs. Dynamic
  - Discrete vs. continuous

Container terminals
- On and off-loading of containers and also storage of containers.
- The world's largest terminal is Singapore that handled 27.9 million TEUs in 2007.
- Singapore has a total berth length of 17 km.

An SPP approach I

Berth and time layout
- Time and berth is discretized into smaller units.
- For the berth units of 25m seems to be very reasonable, and time units of 30min seems to be fair.
- Each berth unit × time unit defines a constraint as at most one ship can be at the position at any time.
### An SPP approach II

#### Columns in the model
- Each ship possible berthing in time and place can be modelled as a box.
- All corresponding berth unit × time units in that box gets a 1 in the column. The rest will be 0.
- In order to distinguish the columns for the different ships we add a GUB constraint for each ship to the model.

#### BCP and column generation
- For each constraint we have a dual variable.
- Let \( \alpha_i \) be the value of the dual variable of the \( i \)’th GUB constraint.
- Let \( \beta_j \) be the value of the dual variable of the \( j \)’th berth unit × time unit constraint.
- The problem can not be solved by an efficient search procedure where check each possible position.

#### The model

##### The IP Model

\[
\begin{align*}
\text{min} & \quad \sum_{j=1}^{N} L_j x_j \\
\text{s.t.} & \quad \sum_{j=1}^{N} a_{ij} x_j = 1 \quad \forall i \\
& \quad \sum_{j=1}^{N} b_{kj} x_j \leq 1 \quad \forall k \\
& \quad x_j \in \{0, 1\}
\end{align*}
\]

- \( x_j = 1 \) if we use berth plan \( j \) and 0 otherwise (variable).
- \( L_j \) is equal to the flow time.
- \( a_{ij} = 1 \) if berth plan \( i \) is for ship \( j \).
- \( b_{kj} = 1 \) if berth unit × time unit \( k \) is for ship \( j \).

#### Scheduling
Revisit the scheduling problem from earlier. A couple of extra issues needs to be integrated into the model.
- What if an assignment requires two persons?
- Given a list of pairs of people \( P \). If \( (p_1, p_2) \in P \) we say that \( p_1 \) and \( p_2 \) hates each other. Consequently they cannot work together on an assignment. Now include in the model that two people that hates each other cannot work on the same assignment.