Introduction to the Christmas Lunch problem

Thomas J. K. Stidsen

Overview

- The problem
- MIP formulation
- Solution representation
- Data files
- Program architecture

The Christmas Lunch Problem

The problem simple: Place people at tables at which other people with the same interests are.

Sets (indexes):

- Persons: $p, pp = 1, ..., 18$
- Wishes: $w = cars, ..., eating$
- Tables: $t = 1, ..., 3$

Parameters:

- A_w^p: 1 if person p has interest w

Decision variables:

- x_t^p: 1 if person p is seated at table t
- $y_{w,t}^{p,pp}$: 1 if person p and pp sit at the same table t and share interest w

Model

Objective to maximize: No of shared interests:

$$\sum_{p}\sum_{pp}\sum_{w}\sum_{t} y_{w,t}^{p,pp}$$

Constraints:

- Force seating of everybody:

$$\sum_t x_t^p = 1, \ \forall p$$

- Lower bound the number of number of people at the table

$$\sum_p x_t^p \geq \lfloor \frac{|p|}{|t|} \rfloor, \ \forall t$$

- Upper bound the number of number of people at the table

$$\sum_p x_t^p \leq \lceil \frac{|p|}{|t|} \rceil, \ \forall t$$

- Upper bound shared interests:

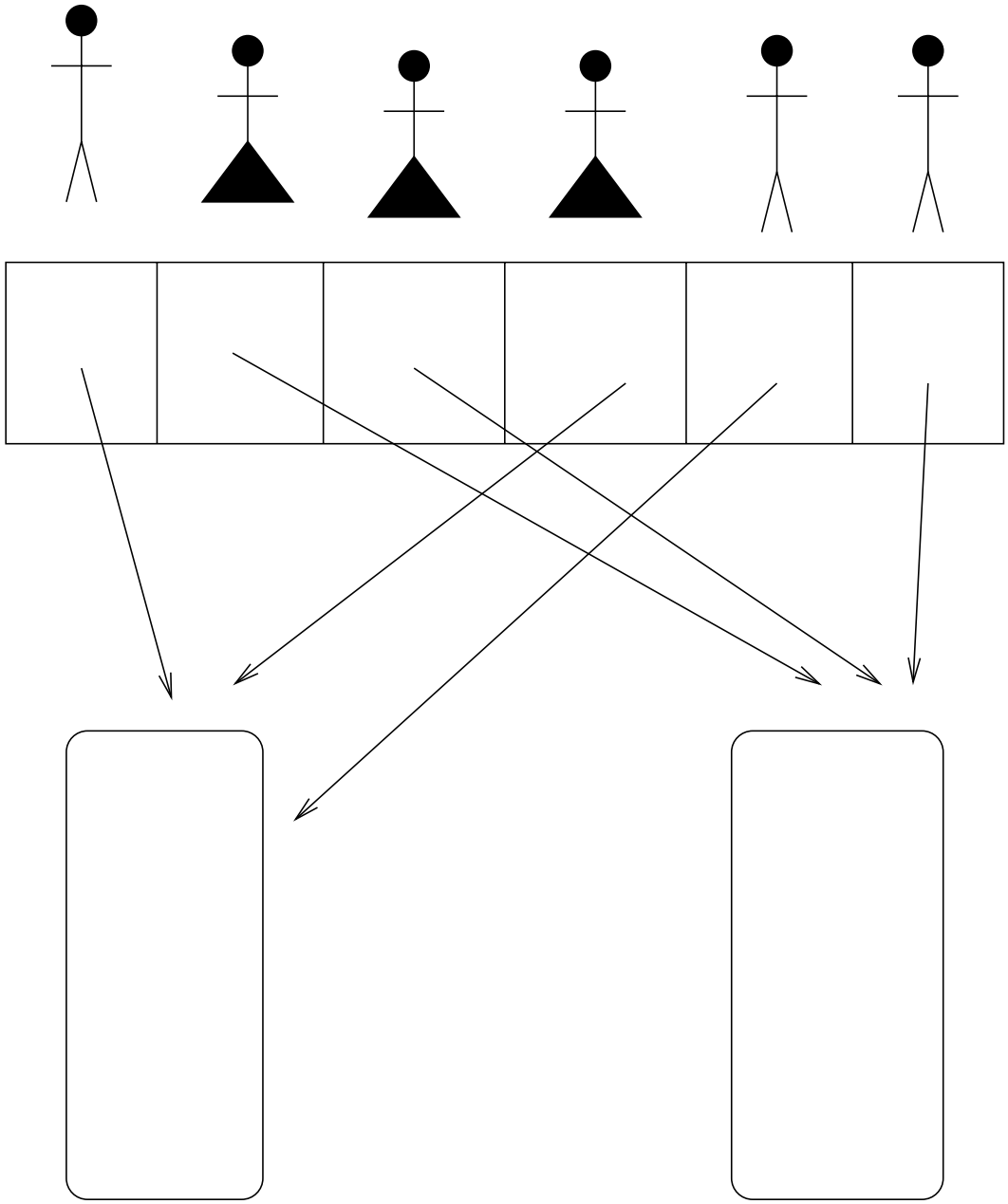
$$y_{w,t}^{p,pp} \leq \frac{1}{2}(A_w^p \cdot x_t^p + A_w^{pp} \cdot x_t^{pp}), \ \forall p, pp, w, t$$

Representation

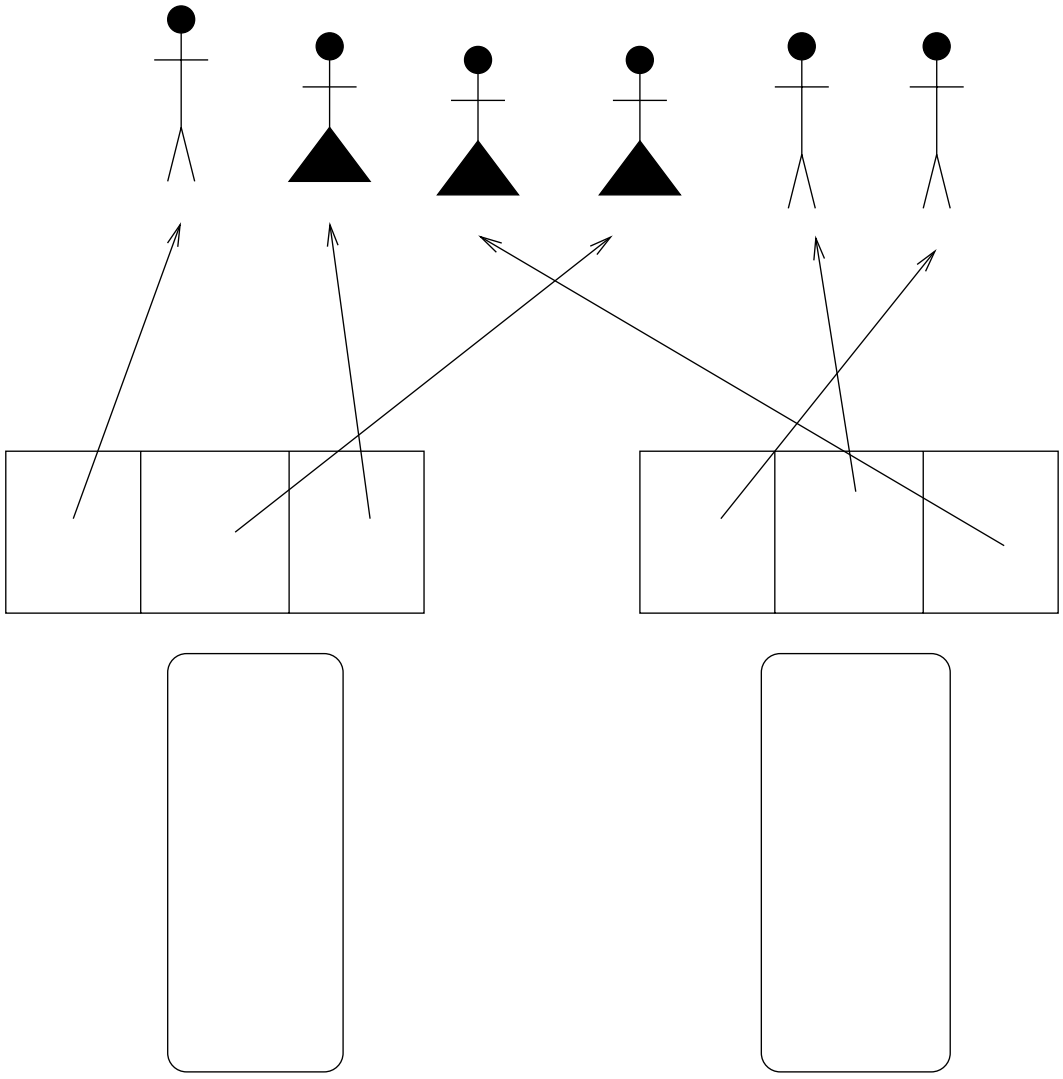
- Important lesson: You need another representation for meta-heuristics t-han in MIP models !

- Two possible representations:
  - Vector with one entry for each person. The number is the table the person is at
  - A vector for each table (hence a matrix).

Vector Representation



Matrix representation



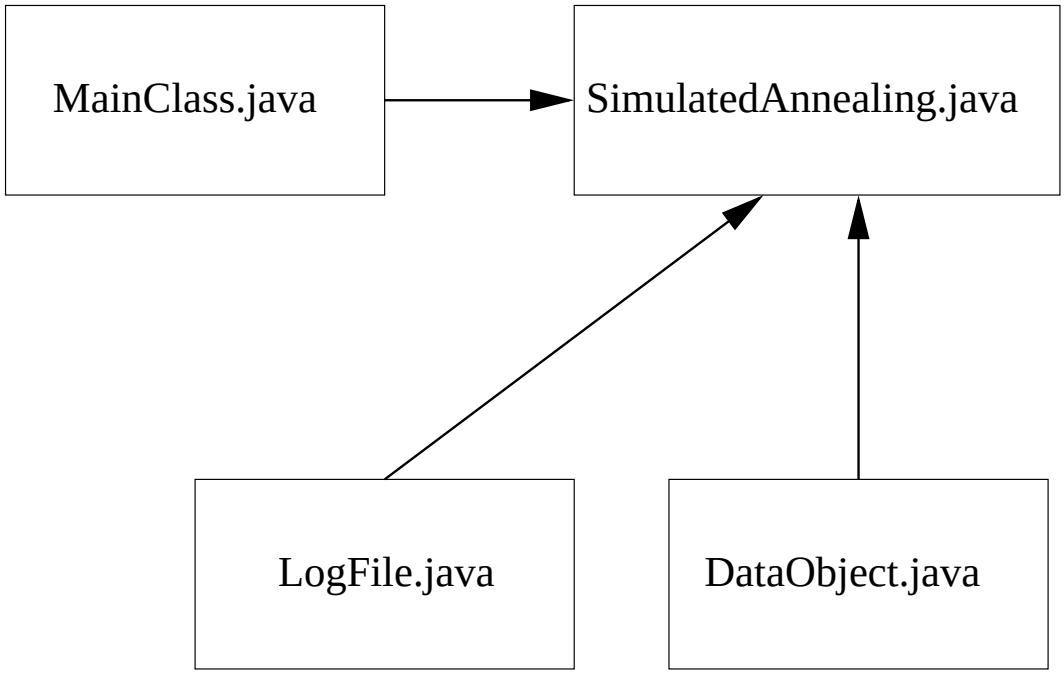
Representation

- Representation is more an art that a science ...

- A good representation is often more important that the right choice of meta-heuristic !

- A good representation should support:
  - Good correlation between neighbours
  - Fast evaluation (delta evaluation)
  - Good handling of infeasibility
  - Ease of programming

Overall Architecture



Program files:

- MainClass.java: Simple startup class
- metaheuristic.java: Main class where things happen ...
- LogFile.java: Class for sampling information
- DataObject.java: Class for reading problem related data **and** evaluating solutions ...

The data file

```
 110
  11
  10
  10
   1       2       2
   1       3       1
   1       4       2
....
 107     110       0
 108     109       0
 108     110       0
 109     110       1
```

Interpretation

- First line: No of persons
- Second line: No of tables
- Third line: Maximal no of persons at table
- Fourth line: Minimal no of persons at table
- Rest of lines: No. of common interests between Persons: First column is no of one person, second column is number of second person and third number is the number of shared interests.

The job

- Place all persons at a table
- Maxmimize the number of common interests

First questions for a problem

- What is a solution ?
- How can we evaluate a solution ?
- How can we „vary“ a solution ?
- Can a variation lead to infeasibility ?

What is a solution ?

All the information we need in order to be able to precisely define a solution.

- A list of integer values, where the position is the person:

```
integer[] currentSolution
```

- Notice that it is easy to create infeasible solutions ...

How can we evaluate a solution ?

Given a solution:

```
integer[] currentSolution
```

:

- Loop over all tables
- Loop over all persons (p1)
- Loop over persons (p2) (from current persons)
- If p1 and p2 are at the current table, add the joined value

How can we „vary“ a solution ?

- Simple: Swap two integer values (different)