UCLouvain

École polytechnique de Louvain

Optimization of production planning with resource allocation

Author: **Florian KNOP**
Supervisors: **Pierre SCHAUS, Charles THOMAS**
Reader: **Hélène VERHAEGHE**
Academic year 2018–2019
Master [120] in Computer Science

Contents

Chapter 1

Introduction

Chapter 2

The Village n°1 problem

Village n°1 is a belgian company employing persons with disabilities. They offer services to companies and private individuals such as industrial jobs. They are currently in the process of automating the way they schedule these jobs. The company has internal work for their employees but also receives demands from external clients. They need to assign workers for one or more time periods to those demands. They also need to assign other resources such as vehicles, machines, locations to these demands. Moreover, persons with disabilities have more constraints related to their working capabilities and availabilities. The workers need specific skills and may have some restrictions which prevent them to work for certain jobs (e.g. a worker can't lift more than five kilograms).

2.0.1 Problem Description

Demands

The problem consists of solving resource allocation to various demands. Each demand has a set of required workers with specific skills, a list of required machines and eventual possible locations. A demand may be an external demand, which means an external company asked for workers for a given task. For this problem, demands are already assigned a given set of time periods that the workers need to be assigned to. It does not take into account the start and end date of a demand.

Workers

Workers are the employees with disabilities that work for Village n°1. They each have a set of availabilities which need to be respected. Each worker has a set of skills that can be put to use for various demands. Workers can also have some incompatibilities with other workers or clients making them unable to work with or for those people.

Machines

Machines also need to be assigned to demands. For this problem, we consider vehicles to also be machines.

Locations

Locations are working places that workers need to go to for their daily work. A demand may need to have a location assigned to it but it is not always the case as some external demands might already have the locations decided by the client.

Chapter 3

State of the art

3.1 Mixed Integer Programming

The most common *Mixed Integer Programming* (MIP) problems are of the form:

$$\begin{aligned} \min \quad & \boldsymbol{c}^T \boldsymbol{x} && (3.1)\\ \text{s.t} \quad & A\boldsymbol{x} = \boldsymbol{b} && (3.2)\\ & \boldsymbol{l} \leq \boldsymbol{x} \leq \boldsymbol{u} && (3.3)\\ & \text{Some or all } x_i \text{ must take integer values} && (3.4) \end{aligned}$$

(3.1) is the problem objective. $\boldsymbol{c}^T$ is the vector of coefficient, $\boldsymbol{x}$ is the vector of variables. (3.2) are the linear constraints. $\boldsymbol{b}$ is a vector of bounds while A is a matrix of coefficients for the constraints. (3.3) are the bound constraints. Each x_i can only take values between l_i and u_i. And finally, (3.4) states the integrality constraints over some or all variables.

MIP problems are usually solved using a branch-and-bound algorithm [1]. The process is as follow: we start with the MIP formulation and remove all integrality constraints to create a resulting linear-programming (LP) relaxation to the original problem. The relaxation can be solved easily compared to the original problem. The result might satisfy all integrality constraints and be a solution to the original problem. But more often than not, a variable has a fractional value. We can then solve two relaxations by imposing two additional constraints. For example, if x takes value 5.5, we add the following linear constraints: $x \leq 5.0$ and $x \geq 6.0$. This process is repeated throughout the search until a valid solution is found. More techniques are used to find solution more efficiently. Each solver uses its own algorithm (e.g Gurobi Optimizer [1]).

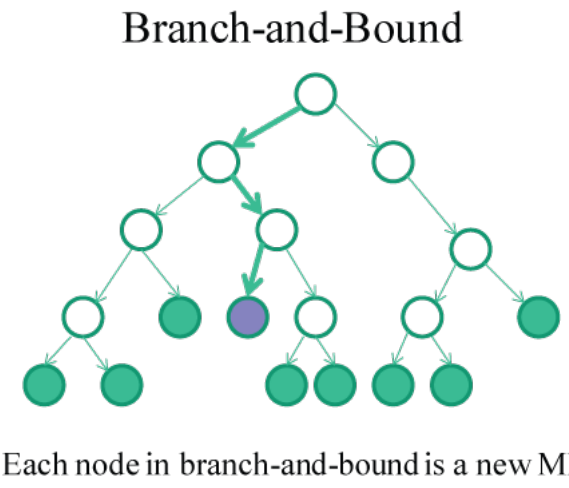

Figure 3.1: MIP Branch & Bound search tree [1]

3.1.1 Gurobi Optimizer

The *Gurobi Optimizer* [2] is a state-of-the-art commercial solver for mathematical programming. Gurobi includes multiple solvers, among those:

- Linear Programming (LP)
- Mixed Integer Linear Programming (MILP), abbreviated as MIP.

The Gurobi Optimizer is used by more than 2100 companies in over 40 industries at this time. It allows describing business problems as mathematical models. It also supports a lot of programming interfaces in a variety of programming languages like C++, Java, Python, C#.

3.2 Constraint Programming

A *Constraint Satisfaction Problem* (CSP) consists of a set of n variable, $\{x_1, \ldots, x_n\}$; a domain D_i of possible values for each variable x_i, $1 \leq i \leq n$; and a collection of m constraints $\{C_1, \ldots, C_m\}$. Each constraint C_j, $1 \leq j \leq m$, is a constraint over some set of variables called the scheme of the constraint. The size of this set is known as the arity of the constraint. A solution to a CSP is an assignment of values $a_i \in D_i$ to x_i, that satisfies all of the constraints. [3]

3.2.1 OscaR

OscaR [4] is a Scala toolkit for solving Operations Research problems. OscaR has multiple optimization techniques available:

- Constraint Programming
- Constraint Based Local Search (CBLS)
- Derivative Free Optimization
- Visualization

The project is mainly developped by UCLouvain and the research group of Pierre Schaus. But some companies like *N-Side* and *CETIC* allocate resources to improve it.

The library of OscaR in which this project is interested in is the Constraint Programming library. It offers a lot of existing constraints and abstractions. Some black-box searches are also implemented but we can bring our own heuristics to drive the search forward.

Chapter 4

Models for the Village n°1 problem

In this chapter, we present models for both Mixed Integer Programming and Constraint Programming. We first start by presenting constraints that need to be respected. We then present formal notations used by both models. The mathematical (MIP) model is presented first for reference followed by the CP model which is an adaptation of the mathematical model.

4.1 Notations

- Set of periods:
$$T = \{0, \ldots, n \mid n \in \mathbb{N}\}$$
- Set of workers:
$$W = \{w_0, \ldots, w_n \mid n \in \mathbb{N}\}$$
  - $w^T \subseteq T$: Availabilities of a worker:
- Set of machines:
$$M = \{m_0, \ldots, m_n \mid n \in \mathbb{N}, m_n \in \mathbb{N}\}$$

  Let's also define the set of machines for a given machine value (i.e name). M_i is the set of machines that takes the value i.

$$M_i = \{m_j \mid m_j = i, \forall j \in M\}$$

- Set of vehicles:
$$V = \{v_0, \ldots, v_n \mid n \in \mathbb{N}\}$$
This could also be expressed as a subset of machines:
$$V \subseteq M$$
- Set of zones:
$$Z = \{z_0, \ldots, z_n \mid n \in \mathbb{N}\}$$
- Set of demands:
$$D = \{d_0, \ldots, d_n \mid n \in \mathbb{N}\}$$
  - $d^w \in \mathbb{N}$: Required number of workers for this demand
  - $d^T \subseteq T$: Possible periods for a demand
  - $d^Z \subseteq Z$: Possible zones for a demand
  - $d^M \subseteq M$: List of required machines by the demand
  - $d^c \in C$: Client for that demand
  - $d^S \in S$: List of skill required by the demand (each skill need to have a different worker)
  - d^{s_0}: The first skill in d^S
  - d^{S^+}: Set of additional skills that can be satisfied by any worker in that demand
  - $d^{S^+_0}$: The first skill in d^{S^+}
  - $d^P \in \{0, \ldots, d^w - 1\}$: List of positions
  - $d^O \subseteq D$: Set of overlapping demands in time for that demand. e.g. the overlapping demands for demand 1 is d^O_1
- Set of clients:
$$C = \{c_0, \ldots, c_n \mid n \in \mathbb{N}\}$$
- Set of skills:
$$S = \{s_0, \ldots, s_n \mid n \in \mathbb{N}\}$$
Let's also define the set of workers that satisfy a skill or skill set:
$$W_s \subseteq W, s \in S$$
- Set of restrictions:
$$R = \{r_0, \ldots, r_n \mid n \in \mathbb{N}\}$$

- Set of incompatibilities between workers:

$$I_{ww} = \{(i,j) \in \mathbb{N} \times \mathbb{N} \mid w_i, w_j \in W, w_i \neq w_j\}$$

- Set of incompatibilities between workers and clients:

$$I_{wc} = \{(i,j) \in \mathbb{N} \times \mathbb{N} \mid w_i \in W, c_j \in C\}$$

4.2 Mixed Integer Programming Model

4.2.1 Variables

To represent our problem in MIP, we will need three types of variables, one per resource.

$$w_{ijkl} = \begin{cases} 1 & \text{if worker } i \text{ is working at time } j \text{ for demand } k \text{ at position } l \\ 0 & \text{otherwise} \end{cases}$$

$$m_{ij} = \begin{cases} 1 & \text{if machine } i \text{ is used for demand } j \\ 0 & \text{otherwise} \end{cases}$$

$$z_{ij} = \begin{cases} 1 & \text{if zone } i \text{ is used for demand } j \\ 0 & \text{otherwise} \end{cases}$$

This is in fact a binary Integer Programming model as every variables is a $\{0,1\}$ integer.

4.2.2 Complete Model

$$\min \quad \sum_{k\in D}\sum_{l\in d_k^P}\sum_{i\in W} min(\sum_{j\in T} w_{ijkl}, 1) \tag{4.1}$$

$$\text{s.t} \quad \sum_{i\in W} w_{ijkl} = 1, \qquad \forall k \in D, j \in d_k^T, l \in d_k^P \tag{4.2}$$

$$\sum_{k\in D}\sum_{l\in d_k^P} w_{ijkl} \leq 1, \qquad \forall i \in W, j \in T \tag{4.3}$$

$$t_j \notin d_k^T \implies \forall i, l \; w_{ijkl} = 0, \qquad \forall j \in T, k \in D \tag{4.4}$$

$$t_j \notin w_i^T \implies \forall k, l \; w_{ijkl} = 0, \qquad \forall j \in T, i \in W \tag{4.5}$$

$$\sum_{l\in d_k^P} w_{ajkl} + w_{bjkl} < 2, \qquad \forall (a,b) \in I_{ww}, j \in T, k \in D \tag{4.6}$$

$$d_k^c = c \implies \forall l \; w_{ijkl} = 0, \qquad \forall (i,c) \in I_{wc}, j \in T, k \in D \tag{4.7}$$

$$w_{ijkl} = 0, \qquad \forall j \in T, k \in D, l \in d_k^P, \; i \in W \setminus W_{d_k^{sl}} \tag{4.8}$$

$$\sum_{l\in d_k^P} w_{ijkl} \geq 1, \qquad \forall j \in T, k \in D, s \in d_k^{S^+}, i \in W_{d_k^{S^+}} \tag{4.9}$$

$$z_i \notin d_j^Z \implies z_{ij} = 0, \qquad \forall i \in Z, j \in D \tag{4.10}$$

$$|d_j^Z| > 0 \implies \sum_{i\in Z} z_{ij} = 1, \qquad \forall j \in D \tag{4.11}$$

$$z_{ij} + z_{ik} \leq 1, \qquad \forall j \in D, k \in d_j^O, i \in Z \tag{4.12}$$

$$m_i \notin d_j^M \implies m_{ij} = 0, \qquad \forall i \in M, j \in D \tag{4.13}$$

$$\sum_{i\in M_k} m_{ij} = |d_j^{M_k}|, \qquad j \in D, k \in d_j^M \tag{4.14}$$

$$m_{ij} + m_{ik} \leq 1, \qquad \forall j \in D, k \in d_j^O, i \in M \tag{4.15}$$

$$w_{ijkl} \in \{0,1\}, \qquad \forall i \in W, j \in T, k \in D, l \in d_k^P \tag{4.16}$$

$$m_{ij} \in \{0,1\}, \qquad \forall i \in M, j \in D \tag{4.17}$$

$$z_{ij} \in \{0,1\}, \qquad \forall i \in Z, j \in D \tag{4.18}$$

The objective function is stated in (4.1), it minimizes the number of different workers for every position between periods of that demand. $min(\sum_{j\in T} w_{ijkl}, 1)$ is one if the worker i is working for that position at that time, 0 otherwise. Hence, the sum of that value for all worker will be equal to the number of worker for that shift.

Constraint (4.2) ensures that each position is filled by only one worker. Constraint (4.3) ensures that no worker works for multiple demands at the same time period. The constraints (4.4) and (4.5) ensures that no worker is working for a

demand that is not occurring or when is himself not available. Constraint (4.6) ensures that no incompatible workers work together while (4.7) ensures that no incompatible pair of worker and client work together. Constraint (4.8) ensures that no worker work for a position in which they are not qualified to work at. Constraint (4.9) ensures that for each additional skills, at least one worker in the group has that skill.

Constraint (4.10) ensures that no zone is assigned to a demand in which this zone is not a possible assignment. (4.11) ensures that only one zone is assigned to this demand if this demand is in need of a zone. Constraint (4.12) ensures that no zone is assigned to two overlapping demands in time.

Constraint (4.13) ensures that no machine is assigned to a demand not in need of that machine. Constraint (4.14) ensures that the required number for each machine is satisfied. And again, (4.15) ensures that no machine is assigned to two overlapping demands in time.

Finally (4.16), (4.17) and (4.18) ensure the variables only takes binary values.

4.3 Constraint Programming Model

The translation to the mathematical (MIP) model to the CP model is fairly straightforward. Binary variables are translated to integer variables, each value representing one resource (i.e. worker, zone or machines). For example, binary variables $w_{0jkl}, \ldots, w_{njkl}$ are transformed to a single variable $w_{jkl} \in \{0, \ldots, n\}$

4.3.1 Variables

First, we need to express the set of workers for each demand at each time period in which that demand occurs.

$$w_{ijk} \in W \tag{4.19}$$

(4.19) is the worker working at time i for demand j at the k^{th} position with $t_i \in T$, $d_i \in D$, $t_i \in d_j^T$ and $k \in d_j^P$. This is done by using a 3-dimensional array of variables. The first dimension being the indices of the time periods, the second dimension is the indices of the demands while the last dimension is the list of worker variables. This last dimension has the size of the number of required workers for that demand.

The same reasoning is used for zones and machines:

$$m_{ij} \in M \tag{4.20}$$
$$z_i \in Z \tag{4.21}$$

(4.20) is the j^{th} machine used for demand i while (4.21) is the zone used for demand i

Some constraints are already satisfied by the modeling of the variables, like the number of required resources (i.e. worker, location, machine) per demand.

4.3.2 Constraints

A worker can only work for one demand at a time

Let $X_i = \{w_{ijk} \mid j \in D, k \in d_j^P\}$ be the set of worker variables for all the demands accross time period i. The constraint `allDifferent`(X_i) states that all workers working at time i need to be different. For each time period i and set X_i, we need an `allDifferent` constraint.

All workers can only work when they're available

Let $W_i = \{w \mid t_i \notin w^T\}$ be the set of workers not available at t_i. The constraint `notEqual` can be used to remove values from the worker variables w_{ijk}. For each time period i and demand, remove the values contained in the set W_i. This can however be done at the domain initialization time by omitting those values instead of having an additional constraint.

Incompatibilities between workers

We can solve this constraint using a `negativeTable` constraint. For each pair of variables (a, b) in the set $\{w_{ij0}, \ldots, w_{ijn|n=d_j^w-1}\}$, we add `negativeTable`$(a, b, I_{ww})$. This prevents workers to work with incompatible other workers. Note that we need to cover the two directions of incompatibilities from $I_w w$. We can do this by adding another `negativeTable`(b, a, I_{ww}) or simply by appending the reversed direction to the I_{ww} table.

Incompatibilities between workers and clients

This constraint is much simpler than the one described above. Contrary to the worker-worker incompatibilities, the clients are fixed values. We simply need to remove values from the domain of worker variables. For each $(w, c) \in I_{wc}$, we remove the value w to each variable on demand d where $d^c = c$. This can be done by a `notEqual` constraint or by removing the value at initialization.

Skill requirements

The variable modeling makes this constraint simple. Each position is described by one worker variable. Each position is assigned a set of skills needed by one worker. Again, we need to remove values from the domain of the worker variables when those workers do not meet the skill requirements of that position.

For example, if a demand has a requirement of two workers, one `lifter` and another one with no particular skill. The first worker variable for that demand will remove all workers that do not have the `lifter` skill while the second variable will remain untouched.

Additional skill requirement

Additionnal skills are the skills that can be satisfied by any worker in the group. TODO: explain gcc + sum occurences

No zone should be used by two overlapping demands

Let $Z_i^O = \{z_j \mid j \in d_i^O\}$ be the set of zone variables for demands that overlap in time with demand i. The constraint $\texttt{allDifferent}(Z_i^O)$ states that all zones for overlapping demands should be different. For each demand i and set Z_i^O, we add an `allDifferent` constraint.

No machine should be used by two overlapping demands

Let $M_i^O = \left\{m_{jk} \mid j \in d_i^O, k \in \{0, \ldots, |d_j^M| - 1\}\right\}$ be the set of machine variables for demands that overlap in time with demand i. The constraint $\texttt{allDifferent}(M_i^O)$ states that all machines for overlapping demands should be different. For each demand i and set M_i^O, we add an `allDifferent` constraint.

Objective function

The objective function expresses the minimization of the sum of different worker for each shift. Let $W_{jk} = \{w_{ijk} \mid i \in d_j^T\}$ be the set of all worker variables for demand j at position k accross all time periods for that demand. We use this set to compute the number of different workers for a given shift with the constraint $\texttt{atLeastNValue}(W_{jk}, N_{jk})$ with N_{jk} being the number of different workers for shift k of demand j. The objective can be expressed by $\sum_{j,k} N_{jk}$ which is the sum of different workers over all shifts.

Chapter 5

Implementation (TODO title)

In this chapter, we describe our implementation for the models presented in chapter 4. The implementation is done in Scala using *OscaR* (3.2.1) for the Constraint Programming model and *Gurobi Optimizer* (3.1.1) for the Mixed Integer Programming model.

Chapter 6

Experiments

Chapter 7

Conclusion

Bibliography

[1] "Gurobi - MIP Basics." http://www.gurobi.com/resources/getting-started/mip-basics. Accessed: 2019-03-13.

[2] L. Gurobi Optimization, "Gurobi optimizer reference manual," 2018.

[3] P. van Beek and X. Chen, "Cplan: A constraint programming approach to planning," in *AAAI/IAAI*, 1999.

[4] OscaR Team, "OscaR: Scala in OR," 2012. Available from https://bitbucket.org/oscarlib/oscar.

UNIVERSITÉ CATHOLIQUE DE LOUVAIN
École polytechnique de Louvain
Rue Archimède, 1 bte L6.11.01, 1348 Louvain-la-Neuve, Belgique | **www.uclouvain.be/epl**