EasyChair Preprint

№ 7319

The Office Scheduling Problem

Minh Vinh Nguyen Phuoc Bao

January 10, 2022

The Office scheduling problem

Vinh Nguyen Phuoc Bao Minh

January 7, 2022

Abstract

The Coronavirus Disease 2019 has disrupted many aspects of our daily life, such as going to work. To cope with this rapid changes, offices are required to make a more flexible schedule. In this paper, we introduce an approach to this problem using constraint programming.

1 Introduction

The Coronavirus Disease 2019 (COVID-19), which was first identified in Wuhan, China, is an ongoing pandemic (Liu et al., 2020). According to the study conducted by the World Health Organization (The World Health Organization, 2019), COVID-19 can cause serious respiratory illness that could become fatal, especially to those with underlying health conditions such as cardiovascular disease, diabetes, chronic respiratory disease, or cancer. Following the current situation, multiple measures have been executed in order to fight the spread of the virus while limiting the negative economic consequences. As far as the workplace is concerned, reducing the number of employees in the workplace by encouraging teleworking plays a crucial role in protecting them, their families, and society as a whole.

The mid-COVID-19 pandemic is forcing businesses all over the world (for example, Hong Kong (Vyas and Butakhieo, 2021), Vietnam (Dezan Shira et. al., 2021)), to shift its paradigm from onsite- to online-working in the hope of limiting the exposure of the virus while retaining the productivity and workload. In universities such as the Vietnamese-German University, although students and teaching staff (e.g. professors, lecturers) are encouraged to work from home, administrative staff (e.g. lab-engineers, faculty assistants) must commute to the university on a daily basis. Therefore, the university administrators

Person	Alice		Bob		Charlie		David		Eve	
Time	AM	PM	AM	PM	AM	PM	AM	PM	AM	PM
1		✓				✓	✓			✓
2				✓		✓			✓	✓
3				✓			✓			✓
4	✓						✓		✓	
5					✓		✓			
6			✓				✓			✓
7		✓					✓	✓	✓	
8					✓					
9		✓	✓	✓		✓				✓
10		✓	✓		✓	✓			✓	
11						✓		✓	✓	
12										

Table 1: Availability of staff members (✓: available)

need to schedule a specific work plan to assure safety to its staff while maintaining efficiency. In order to resolve the *Office scheduling problem*, let us assume that employees are allowed to have flexible working schedules, i.e., non-traditional arriving and leaving time.

In combinatorial optimization (Korte and Vygen, 2018), a problem setting can be quite similar to another. The *Office scheduling problem* is not an exception since it is based on the *Production scheduling problem* (Graves, 1981), with respect to the optimization objective, while differing from particular constraints and assumptions. The *Office scheduling problem* takes into account a group of staff who are required to work for a number of hours. Moreover, their tasks are independent, i.e., they do not need to wait for another staff member to finish his/her work. Because every staff member may need to stay home with a child due to school and childcare closures or other reasons, they will be assigned a time slot that fits their own schedule. For the sake of simplicity, let us assume that only one person is allowed to be physically in the office at a time. Inspired by (Graves, 1981; Guerriero and Guido, 2021), in this paper, our objectives are

- to determine a working schedule that fits every staff preferences,
- and more importantly, to determine an optimal working schedule, i.e., the hour that the office can close the earliest.

Example 1. Let us consider a case of 5 staff members: *Alice*, *Bob*, *Charlie*, *David* and *Eve* working in a department at an office. As previously stated, only one person at most

	1	2	3	4	5	6	7	8	9	10	11	12
AM	David	Eve	David	Eve	Charlie	Bob			Bob	Bob	Eve	
PM	Alice					Eve	Alice					

Table 2: A possible schedule

	1	2	3	4	5	6	7	8	9	10	11	12
AM	David	Eve	David	Alice	Charlie	Bob	Eve		Bob	Eve	Eve	
PM	Alice	Bob										

Table 3: A more optimal schedule

is permitted to be in the room. Each person has a number of hours they need to work. Respectively, *Alice*, *Bob*, *Charlie*, *David* and *Eve* need to work for 2, 3, 1, 2 and 4 hours. Each person has registered a preference time that they can work, so as not to disrupt their home activities, reported in Table 1.

One scheduling solution could be depicted in Table 2: in which the office will be closed after 7 P.M.. Another feasible answer could be viewed in Table 3: in which the office will be closed after 2 P.M.. The solution in the latter table is better than the former table. In fact, Table 3 displays the most optimal schedule. [1]

In this thesis, we will implement a program using an open-source constraining programming language, `MiniZinc`, which is used to model constraint satisfaction and optimization problems (Nethercote et al., 2007). The thesis is organized as follows, Section 2 introduces the *Office scheduling problem* and its formulation mathematically, Section 3 will talk about programming language and tool, and then implementation for the problem, Section 4 describes some potential variants concerning this problem. Finally, Section 5 features the concluding state of affairs.

2 Mathematical model and formulation

The section starts with the definition of the *Office scheduling problem*.

Definition 1. An *Office scheduling problem* $\mathscr{O}$ is defined as a quadruple

$$\mathscr{O} = \langle P, T, \text{free}, \text{hour} \rangle$$

where:

[1] We will leave the proof of this claim for interested readers.

- P be a set of individuals,
- T be a set of 24-hour time window: i.e., $T = \{t \mid t \geq 1 \wedge t \leq 24\}$,
- free() : $P \times T \rightarrow \texttt{Bool}$ be a function that takes a person and a time slot to return a boolean value that expresses whether this person is available to work at that particular time window,
- hour() : $P \rightarrow \mathbb{N}$ be a function that takes a person and returns a number of hours that person has to work.

Example 2. Consider the *Office scheduling problem* in Example 1.

- $P = \{\texttt{Alice}, \texttt{Bob}, \texttt{Charlie}, \texttt{David}, \texttt{Eve}\}$,
- $T = \{t \mid t \geq 1 \wedge t \leq 24\}$,
- $$\text{free}(p,t) = \begin{cases} 1 & \text{if } (p = \texttt{Alice} \wedge t \in \{4,13,19,21,22\}) \\ & \vee (p = \texttt{Bob} \wedge t \in \{6,9,10,14,15,21\}) \\ & \vee (p = \texttt{Charlie} \wedge t \in \{5,8,10,13,14,21,22,23\}) \\ & \vee (p = \texttt{David} \wedge t \in \{1,3,4,5,6,7,19,23\}) \\ & \vee (p = \texttt{Eve} \wedge t \in \{2,4,7,10,11,13,14,15,18,21\}), \\ 0 & \text{otherwise} \end{cases}$$
- $\text{hour}(\texttt{Alice}) = 2$
  $\text{hour}(\texttt{Bob}) = 3$
  $\text{hour}(\texttt{Charlie}) = 1$
  $\text{hour}(\texttt{David}) = 2$
  $\text{hour}(\texttt{Eve}) = 4$

Definition 2. Given an *Office scheduling problem* $\mathscr{O} = \langle P, T, \text{free}, \text{hour} \rangle$
A solution of $\mathscr{O}$ is a function:

$$\text{assign}() : P \times T \rightarrow \texttt{Bool},$$

that satisfies the following constraints:

- a staff member can only work at the time at which he/she prefers,

$$\forall p,t.\text{assign}(p,t) \Rightarrow \text{free}(p,t)$$

- the number of working hours of a staff member equals to his/her required hours,

$$\forall p. \sum_{t=1}^{24} \text{assign}(p,t) = \text{hour}(p)$$

- the number of staff members working at a time is at most 1.

$$\forall t. \sum_{p}^{P} \text{assign}(p,t) \leq 1$$

Example 3. The solution in Table 2 can be represented as the following function assign() : $P \times T \rightarrow$ `Bool`

- $\text{assign}(Alice,t) = \begin{cases} 1 & t \in \{13,19\}, \\ 0 & \texttt{otherwise} \end{cases}$
- $\text{assign}(Bob,t) = \begin{cases} 1 & t \in \{6,9,10\}, \\ 0 & \texttt{otherwise} \end{cases}$
- $\text{assign}(Charlie,t) = \begin{cases} 1 & t \in \{5\}, \\ 0 & \texttt{otherwise} \end{cases}$
- $\text{assign}(David,t) = \begin{cases} 1 & t \in \{1,3\}, \\ 0 & \texttt{otherwise} \end{cases}$
- $\text{assign}(Eve,t) = \begin{cases} 1 & t \in \{2,4,11,18\}, \\ 0 & \texttt{otherwise} \end{cases}$

All constraints and assumptions in Definition 2 are satisfied.

Definition 3. Given an *Office scheduling problem* in Definition 1. We define t_{max} of function assign() as the last hour to be scheduled in the office. Then, assign*() is said to be the optimal solution if and only if for every solution assign(), $t^\star_{max} \leq t_{max}$.

Example 4. The solution in Table 3 is optimal.

3 MiniZinc and computational experiments

As mentioned earlier, `MiniZinc` language is specified in constrained optimization and programming. `MiniZinc` lets users write models in such a way that is close to a mathematical formulation of the problem, using familiar notation such as existential and universal

quantifiers, sums over index sets, or logical connectives like implications and if-then-else statements (Peter J. Stuckey, Kim Marriott, Guido Tack, 2020).

```
1  enum P;
2  array[P] of int: hour;
3  set of int: T = 1..24;
4  array[P,T] of bool: free;
5  array[P,T] of var int: assign;
6  constraint forall (t in T, p in P) (
7    assign[p,t] in (0..1)
8  );
9  constraint forall (t in T) (
10   1 >= sum (p in P)(assign[p,t])
11 );
12 constraint forall (p in P) (
13   hour[p] = sum (t in T)(assign[p,t])
14 );
15 constraint forall (p in P, t in T) (
16   (assign[p,t] = 1) -> free[p,t]
17 );
18 var int: t_max;
19 constraint forall (p in P, t in T) (
20     (assign[p,t] = 1) -> (t <= t_max)
21 );
22 constraint exists (p in P) (
23   assign[p,t_max] = 1
24 );
25 solve minimize t_max;
```

Listing 1: The MiniZinc implementation of the *Office scheduling problem*

Listing 1 is the implementation of the *Office scheduling problem*. From line 1–4, we define the *Office scheduling problem* according to Definition 1. Line 5–17, we constrain the solution according to Definition 2. Line 18–24, the program focus on constraining t_{max} according to Definition 3. And finally, line 25 tries to find the optimal value of t_{max} using a special command provided by MiniZinc language.

The program takes a data input file and returns the optimal working schedule, if plausible; UNSATISFIABLE, otherwise. Interested readers can find the implementation, some sample datasets and its solutions in (Minh Vinh, 2022).

4 Variants of the Office scheduling problem

The purpose of this section of the thesis is to introduce some new concepts of the *Office scheduling problem*.

4.1 Priority working

Recall that staff members are equal and work individually. Here, the problem is expanded such that some staff work dependent on others. For example, Alice may have to wait until Bob finishes his work.

This variant can be defined by adding a function

$$\text{dependent}() : P \times P \rightarrow Bool,$$

that takes 2 people and return a boolean value that expresses if the former person is dependent on the latter person. In addition, few constraints need to be appended.

4.2 "New normality"

Another variant to this problem can be that it is not limited to just one person in the office at a certain time. For example, there is no restriction for the people who have been vaccinated, while the ones who have not, still have to work separately.

This variant can be defined with another function

$$\text{vaccine}() : P \rightarrow Bool,$$

that takes a person and returns whether he/she is vaccinated or not. In addition, few constraints need to be appended.

4.3 Office hour restriction

Recall that our *Office scheduling problem* is fixed between time 1–24. In pratice, offices open from 8A.M. to 5P.M., so everything needs to be finalized between that time. Therefore, two variables need to be input that expresses the starting and ending time.

5 Conclusion

The COVID-19 pandemic has forced the world to adopt a new paradigm of worksite, one that must take into account school closures and other issues of social distancing, while

still maintaining the same efficiency as before. The optimization model can be used to address the university/company scheduling, in which, all members' needs are considered. It belongs to the class problem of Production scheduling. In this paper, we have proven that it is possible for staff to carry on production, have a better work/life balance schedule and manage to continue working amidst the COVID-19 pandemic.

References

Dezan Shira et. al. (2021). Vietnam's New Normal: The Hybrid Work Strategy. https://www.vietnam-briefing.com/news/. Online; accessed 07 January 2022.

Graves, S. C. (1981). A review of production scheduling. *Oper. Res.*, 29(4):646–675.

Guerriero, F. and Guido, R. (2021). Modeling a flexible staff scheduling problem in the era of covid-19. *Optimization Letters*.

Korte, B. and Vygen, J. (2018). *Combinatorial Optimization: Theory and Algorithms*.

Liu, Y.-C., Kuo, R.-L., and Shih, S.-R. (2020). Covid-19: The first documented coronavirus pandemic in history. *Biomedical Journal*, 43(4):328–333.

Minh Vinh (2022). StaffSchedulingProblem GitHub repository. `https://github.com/minhnguyen1312/StaffSchedulingProblem`. Online; accessed 06 January 2022.

Nethercote, N., Stuckey, P. J., Becket, R., Brand, S., Duck, G. J., and Tack, G. (2007). Minizinc: Towards a standard CP modelling language. In Bessiere, C., editor, *Principles and Practice of Constraint Programming - CP 2007, 13th International Conference, CP 2007, Providence, RI, USA, September 23-27, 2007, Proceedings*, volume 4741 of *Lecture Notes in Computer Science*, pages 529–543. Springer.

Peter J. Stuckey, Kim Marriott, Guido Tack (2020). The MiniZinc Handbook. `https://www.minizinc.org/doc-2.5.5/en/intro.html`. Online; accessed 06 January 2022.

The World Health Organization (2019). Coronavirus disease (COVID-19). `https://www.who.int/health-topics/coronavirus#tab=tab_1`. Online; accessed 06 January 2022.

Vyas, L. and Butakhieo, N. (2021). The impact of working from home during covid-19 on work and life domains: an exploratory study on hong kong. *Policy Design and Practice*, 4(1):59–76.