Since January 2020 Elsevier has created a COVID-19 resource centre with free information in English and Mandarin on the novel coronavirus COVID-19. The COVID-19 resource centre is hosted on Elsevier Connect, the company's public news and information website.

Elsevier hereby grants permission to make all its COVID-19-related research that is available on the COVID-19 resource centre - including this research content - immediately available in PubMed Central and other publicly funded repositories, such as the WHO COVID database with rights for unrestricted research re-use and analyses in any form or by any means with acknowledgement of the original source. These permissions are granted for free by Elsevier for as long as the COVID-19 resource centre remains active.
25th International Conference on Knowledge-Based and Intelligent Information & Engineering Systems

Modeling Employee Flexible Work Scheduling As A Classification Problem

Fred N. Kiwanuka*, Louay Karadsheh, Ja’far alqatawna, Anang Hudaya Muhamad Amin

Higher Colleges of Technology, Dubai Men’s College UAE

Abstract

Many organizations have adapted flexible working arrangements during COVID19 pandemic because of restrictions on the number of employees required on site at any time. Unfortunately, current employee scheduling methods are more suited for compressed working arrangements. The problem of automating compressed employee scheduling has been studied by many researchers and is widely adopted by many organizations in an attempt to achieve high quality scheduling. During process of employee scheduling many constraints may have to be considered and may require negotiating a large dimension of constraints like in flexible working. These constraints make scheduling a challenging task in these working arrangements. Most scheduling algorithms are modeled as constraint optimization problems and suited for compressed work but for flexible working with large constraint dimensions, achieving accurate scheduling is even more challenging. In this research, we propose a machine learning approach that takes advantage of mining user-defined constraints or soft constraints and transform employee scheduling into a classification problem. We propose automatically extracting employee personal schedules like calendars in order to extract their availability. We then show how to use the extracted knowledge in a multi-label classification approach in order to generate a schedule for faculty staff in a University that supports flexible working. We show that the results of this approach are comparable to that of a constraint satisfaction and optimization method that is commonly used in literature. Results show that our approach achieved accuracy of 93.1% of satisfying constraints as compared to 92.7% of a common constraint programming approach.

© 2021 The Authors. Published by Elsevier B.V.
This is an open access article under the CC BY-NC-ND license (https://creativecommons.org/licenses/by-nc-nd/4.0)
Peer-review under responsibility of the scientific committee of KES International.

Keywords: Employee Scheduling, Constraint Programming, Data mining, Machine Learning

1. Introduction

The ability to have satisfied employee on duty at the right time is an important factor for any organization to achieve its goals and objectives. This can translate into improved employee performance and productivity [16]. This calls for

*E-mail address: (fkiwanuka,lkaradsheh,jaqlqatawna,aamin)@hct.ac.ae
effective and proper employee scheduling. Employee scheduling problems have received so much attention over the past decades. The scheduling problem has been studied using several approaches, including operational research techniques and artificial intelligence [5], [9], [3], [7], [13]. These methods largely aim at constraint satisfaction or optimization and this usually comes with high computational burden in finding optimal solutions when the number of constraints is high. The performance of these methods deteriorates with large dimension constraints like those found in flexible working environments.

But also, current scheduling methods are more suited for compressed working arrangements like hospitals, but many organizations especially non manufacturing have increasingly adopted flexible working especially during this COVID19 pandemic where companies have restrictions on the number of workers required on site. Flexible work is evolving rapidly and its permutations are proliferating to many organizations. Under a flextime schedule, employees exercise a decision regarding the time of day they will arrive at and leave from work. For a review of about flexible working see [6].

The rapidly growing field of data mining and machine learning has the potential of improving performance of existing scheduling systems especially flexible working arrangements. For example, deep learning models have been able to perform to almost human level accuracy in many tasks like image processing and analysis and many more applications [4]. Nowadays, many applications are generating large amounts of data about our daily lives, which is often not utilized to its potential. It is possible to extract knowledge from these applications and use it to improve current scheduling practice.

In this research, we aim to demonstrate the potential of data mining and machine learning in employee scheduling tasks. Data mining has shown in many domains its potential in discovering unexpected knowledge and insights that can be used in further improvements. To the best of our knowledge, this is the first attempt to model employee scheduling as a classification problem. Our approach involves processing of personal schedules like calendars into appropriate data formats, discovery of key scheduling concepts, and representation of the data mining results in a way that enables its use for employee scheduling. Through this data-driven approach, we show how mined personal schedules can be applied to scheduling to generate schedules through classification.

The paper is organized as follows. Section 2 introduces related work on scheduling problem. Section 3 presents data mining framework including the data mining, preprocessing, post-processing as well as the modeling. Section presents the results and discussion. Section 5 concludes the research and future directions.

2. The Scheduling Problem

The task of building computational methodologies to automatically generate solutions to employee scheduling problems has been well presented in literature for a number of decades. Timetabling for employees approaches are categorized using different criteria such as heuristics, mathematical and linear programming, Particle swarm optimization, colony optimization, genetic approaches, reasoning approaches, Mathematical programming approaches, simulated annealing, methods based on graph coloring algorithm, constraint logic programming. For a comprehensive discussion of the various approaches see [2], [5], [9], [3], [7], [13].

However, it is the area of planning and industrial scheduling that has embraced machine learning and data mining. Most of these systems use historical data for the prediction. Artificial intelligence based methods for scheduling for planning and scheduling of tasks has recently received considerable attention. [15] provides a review on its usage in industrial planning tasks and shop scheduling. The work of [8], used timed Petri nets to describe the dispatching processes of the job shop scheduling scenarios. On this basis, a data mining based scheduling knowledge extraction framework is developed to mine the expected scheduling knowledge from the solutions generated by traditional optimization or approximation method for job shop scheduling. They also show how to use the extracted knowledge as a new dispatching rule to generate complete schedules.

There is little in terms of literature in usage of machine learning and data mining in employee scheduling. Some attempt to use machine learning was made in university course time tabling in [12]. They proposed hyper-heuristics learning based on an iterated local search procedure that autonomously combines a set of move operators.
3. Methodology

In this section, we describe our approach of machine learning as well as constraint optimization scheme that is commonly used in scheduling and time tabling. There are six stages to the approach as seen in figure 1.

![High level data mining framework for Employee schedules](image)

Fig. 1. A high level data mining framework for Employee schedules

3.1. The data

We collected 80 faculty personal schedules in a university setting. We can note here that our approach does not consider scheduling for courses and rooms of the university but faculty working times. The choice of the university setting was made due to the fact that most University professors have a flexible work schedule. We requested this group to share their personal schedules like calendars, and any other personal schedules of other responsibilities during the work week. We then extracted the data from the personal schedules through python based scripts. An example of how a raw personal schedule looks is shown in figure 2.

3.2. Pre-processing

This stage handled data cleaning. Personal schedules and calendars were messy, had incomplete entries, inconsistent formatting and entry errors. Data cleaning became a necessary stage. During the preprocessing phase, we divided the timeslots into 2 hours each for the days Sunday to Thursday between 8 AM to 8 PM making 6 shifts per day. We then extracted from personal schedules these times, the days, Sunday to Thursday, 8AM to 8PM. For each time slot without any entry assigned 1 otherwise 0. A time slot of 1 meant a faculty is available to teach at that time. More formally,

\[ f_i = \begin{cases} 
1, & \text{if } f_i \text{ is available} \\
0, & \text{otherwise} 
\end{cases} \] (1)
3.3. Post-processing

During the post-processing, the pre-processed schedule is made consistent with the 6 shift work day. If a preprocessed schedule does not have a particular shift then it meant that the faculty is available for that missing shift see figure 3. Formally, post-processing stage consider two sets A and B of elements such that $A \in B$, the union of $A$ and $B$ can be defined as:

$$A \cup B = \{ x : x \in A \| x \in B \}$$

Where $A$ represents faculty preprocessed schedule and $B$ represents a standard 6 shift work day where all elements are equal to 1. The result of $A \cup B$ is the post-processing knowledge.

3.4. Data Preparation: Labeling

To develop a classification system, we need a training data set with labels. We had two options for this: manually and painstakingly develop a schedule or use a constraint optimization method generated schedule. For the soft constraints, we had faculty calendars showing their availability. This was achieved by data mining personal shared schedules. The following are the hard constraints(organizational based work policies) for our case study:

- Faculty teaching hours per day is maximum of 2 hrs, once a day. This is to enable faculty have enough time for research as well as student advising.
- The teaching should be spread out to all the 5 working days

![Fig. 2. Left: raw personal schedule, right: Pre-processed](image1)

![Fig. 3. Left: Pre-processed, right: Post-processed](image2)
• Working day starts at 8 AM and ends at 8 PM
• Tuesday 12-2 PM is reserved for departmental meetings
• A faculty must teach at least once but not more than thrice a week
• There must be at least one faculty per shift

With those hard constraint in mind we manually inspected each faculty schedule (soft constraint) and assigned at most 3 shifts depending on the faculty duty responsibilities. We defined the number of shifts in a week to be 29 with each work day 8AM to 8 PM having 6 shifts each of 2 hours, multiplied by 5 working days gives 30 minus a Tuesday time slot for departmental meeting.

For the manual schedule, it took us approximately 7 hours to generate a fair schedule that satisfied all the hard constraints and had minimum contradictions to faculty requests for a division of 20 faculty. We then requested faculty to give feedback on its accuracy in terms of satisfying their requests and received positive response.

But we needed more data for our training and we got it from other divisions of the University. But rather than doing it manually, this time we first used the a constraint optimization approach as explained in section 3.5 and noted where constraints were not satisfied. We then manually intervened to try to correct those constraints by either asking the faculty which alternative timeslot would work for them or simply through inspection. This reduced the time needed to generate a schedule to less than an hour. In total we had 80 faculty schedules for training.

Definition 1. A label is defined as the shift assigned to a faculty satisfying all hard constraints and as many soft constraint as possible.

A faculty can have more than one label. From Table 1, the label for the faculty in this example is the set $S = \{d1s6, d3s6, d5s2\}$ where the $d$ is day and $s$ is shift. $d3s6$ means on day 3 of the week, the faculty is assigned shift 6. It must be noted that there are number of faculty with almost similar personal calendars and determining who is assigned the shift is a very time consuming process when using the manual process.

Table 1. An Example of manual shift assignment to a faculty: Left is a personal schedule, Right is a manual shift assignment (colored cells) based on hard constraints of the University and personal schedule

|   | d1  | d2  | d3  | d4  | d5  |
|---|-----|-----|-----|-----|-----|
| s1| 0   | 0   | 1   | 0   | 0   |
| s2| 1   | 0   | 1   | 0   | 1   |
| s3| 1   | 1   | 0   | 0   | 0   |
| s4| 0   | 0   | 0   | 0   | 0   |
| s5| 0   | 0   | 0   | 0   | 0   |
| s6| 1   | 0   | 1   | 0   | 0   |

|   | d1  | d2  | d3  | d4  | d5  |
|---|-----|-----|-----|-----|-----|
| s1| 0   | 0   | 1   | 0   | 0   |
| s2| 1   | 0   | 1   | 0   | 1   |
| s3| 1   | 1   | 0   | 0   | 0   |
| s4| 0   | 0   | 0   | 0   | 0   |
| s5| 0   | 0   | 0   | 0   | 0   |
| s6| 1   | 0   | 1   | 0   | 0   |

3.5. Constraint Satisfaction Problem: Constraint Programming

Its clear that for each faculty there is a True/False availability for a shift. There are 6 shifts of 2 hours Sunday to Thursday. For, a 20 faculty team without constraints, each day there are at least $(20!)$ possible assignment. Each week there are $(20!)^5$ possible assignments which in total comes to $8.5 \times 10^{91}$ possible faculty assignments. This would be computationally demanding to find an optimal solution. However, with the hard and soft constraints, the number of feasible solutions to our scheduling problem reduces to hundredth of thousands which are manageable even for a low end CPU.

Given the above problem with homogeneous collection of finite constraints $C_i$ as given, each with a finite number of variables $x_i$ belonging to their respective domains $D_i$. A turple (X,D,C), is a constraint satisfaction problem and is defined as:
1. A set of \( n \) variables such that \( x_1, x_2, \ldots, x_n = X \)
2. A set of domains \( D_1, D_2, \ldots, D_n = D \) for \( x_i \in D_i \) with \( i = 1, \ldots, n \)
3. A set of constraints \( C_i = C, i = 1, \ldots, n \) where each constraint defines an equation involving any number of variables imposing some relation between them.

The aim is to solve or satisfy every constraint in \( C \) in respect with the variables \( X \). The satisfiability problem then becomes a boolean function of determining if a solution exists. A solution to our problem is an assignment of either 1 or 0 to every variable, in such a way that all constraints are satisfied at once. The goal is to find such solution that satisfies all the constraints and maximise the objective function. In our case, the objective function can be defined. Let \( f \) be faculty personal schedule, \( d \) is day of the week and \( s \) is shift. Let \( S \) represent a shift such that

\[
S_{(f,d,s)} = \begin{cases}
1, & \text{if } f \text{ is assigned a shift } s \text{ on day } d \\
0, & \text{otherwise}
\end{cases}
\] (2)

Furthermore, let \( R \) represent faculty request resulting from post-processed requests in section 3.3.

\[
R_{(f,d,s)} = \begin{cases}
1, & \text{if } f \text{ is available} \\
0, & \text{otherwise}
\end{cases}
\] (3)

Let \( \psi \) be our objective function. We want to optimize this objective function such that:

\[
\psi = \text{Max} \sum_{i=0}^{n} \sum_{j=0}^{d} \sum_{k=0}^{s} R_{(f,d,s)} \times S_{(f,d,s)}
\] (4)

\( \psi_{f,d,s} \) is 1 if shift \( s \) is assigned to faculty \( n \) on day \( d \) and that faculty requested that shift (and 0 otherwise), the objective is to maximize the number of shifts of assignments that meet requests. In this research we employ a backtracking based a Boolean satisfiability (SAT) [17]. The SAT algorithm is basically is about finding whether the variables of a propositional formula can be assigned in such a way that the formula evaluates to true. In this research through the pre-processing and post-processing of personal schedules we were able to standardize the problem as a clausification. We then used the SAT Algorithm using Davis-Logemann-Loveland (DLL) search based [10] as described in Algorithm 1.

For the implementation of constraint programming approach, any constraint programming solver can be used once the process of clausification of the problem has been achieved. Each input to model is \( 6 \times 5 \) as described in section 3.3 with hard constraints hard-coded in the modeling.

4. Modeling

In this section, we describe the modeling approaches used in this research. From Table 1, our scheduling problem is now a multi-label classification problem where a faculty input has more than one label. The origins of multi-label classification can be traced from the text categorisation problem [11], where each document may belong to several predefined topics simultaneously. Multi-label classification [14] is a supervised learning problem that involves an instance being associated with multiple labels unlike the traditional classification task of single-label classification of multi-class, or binary where each instance is only associated with a single class label.

As can be seen in our data each faculty at a particular time slot on a specific data has multiple labels simultaneously associated with a single instance. The faculty can be available or not available at any time slot. In multi-label
Algorithm 1: SAT using DLL search based Algorithm

1 Procedure: SATiterate();
2 status = preprocess();
3 if (status != UNKNOWN) then
4     return status
5 end
6 while (1) do
7     decideNextBranch();
8     while (true) do
9         status = deduce();
10        if (status == CONFLICT) then
11           blevel = analyzeConflict();
12           if (blevel < 0) then
13               return UNSATISFIABLE;
14           else
15               backtrack(blevel);
16           end
17        else if (status == SATISFIABLE) then
18          return SATISFIABLE;
19        else
20          break;
21      end
22 end
23

classification tasks, the training set (faculty processed personal schedules) is composed of instances each associated
with a set of labels, and the task becomes a prediction of the label sets of unseen instances through analyzing training
instances with known label sets.

Definition 2. Let \( F^d \subset \mathbb{R} \) be the input domain of all possible faculty features values as extracted from their personal
schedules. An instance is represented as a vector of \( d \) feature values \( F = [s_1, \ldots, s_d] \) where \( F \in [0, 1] \). The set
\( \Lambda = \{1, \ldots, L\} \) is the output domain of possible labels. This set is represented by an \( L \)-vector \( s = [s_1, \ldots, s_L] \), where \( s_l = 1 \) if and only if label \( k \) is associated with instance \( f \), and 0 otherwise.

For the classification of a new instance \( f \), this method outputs as a set of labels the union of the labels that are output
by the \( L \) classifier.

In our case employing multi-label classification required problem transformation, whereby a multi-label problem
is transformed into one or more binary problem. Then we employed these single-label classifiers and their predictions
transformed into multi-label predictions.

4.1. The Transformation process

We used problem transformation as described in[1]. We transformed our multi-label problem into single-label problems
and for this task we used binary relevance [1]. Binary relevance is simple and given the nature of our features which do not have correlation, it treats each label as a separate single class classification problem. In this
case an ensemble of single-label binary classifiers is trained, one for each class. Each classifier predicts either the
membership or the non-membership of one class. The union of all classes that were predicted is taken as the multi-
label output. For example, let us consider a case in Table 2. In the dataset \( F \) is the independent feature and \( S \) ’s are the
target variable. In our case, binary relevance breaks this problem into 29 different single class classification problems.
Each faculty schedule is modeled as 29 different single class where the class in our context is the shift on a particular
day as explained earlier in section 3.3. The input $F$ is $6 \times 5$ dimensional which is the processed calendar for a faculty. The goal now becomes using a processed faculty personal calendar, can you predict which shifts can be assigned to a faculty.

Table 2. Binary relevance decomposition $f$ is the input (faculty schedule), $s$ is the shift availability as extracted from calendar: Left is the data frame of faculty with there corresponding processed calendar, Right is binary decomposition

| f   | s1 | s2 | s3 | .. | sn |
|-----|----|----|----|----|----|
| f1  | 1  | 0  | 0  | 0  | 1  |
| f2  | 0  | 1  | 1  | 0  | 1  |
| f3  | 1  | 1  | 1  | 1  | 1  |
| f4  | 1  | 0  | 1  | 0  | 1  |
| ..  | 0  | 0  | 1  | 1  | 1  |
| fn  | 1  | 0  | 0  | 0  | 1  |

| f   | s1 | f1  | f2  | f3  | f4  | ..  | fn  |
|-----|----|-----|-----|-----|-----|-----|-----|
| f1  | 1  | 1   | 0   | 1   | 0   | ... | 1   |
| f2  | 0  | 1   | 1   | 0   | 1   | ..  | 0   |
| f3  | 1  | 1   | 1   | 1   | 1   | ..  | 1   |
| f4  | 1  | 0   | 1   | 0   | 1   | ..  | 1   |
| ..  | 0  | 0   | 1   | 1   | 1   | ..  | 1   |
| fn  | 1  | 0   | 0   | 0   | 1   | ..  | 1   |

4.2. Implementation

In total we had 80 schedules. We trained our classifiers on 80% of the data and tested on 20%. Each input is, $6 \times 5$ as described in section 3.3 and the target are the multi-label shift as described in section 3.4. For the binary relevance, we used logistic regression with grid search parameter tuning. We utilized a parameter grid that consists of running the model with variations with respect to type of regularization, size of penalty, and type of solver in order to select the best model. In practice any machine learning model can be used including gradient boosting machines classifiers or deep learning models. However, because of the binary nature of our features, tree based classifiers do not perform well. But for a reasonable trade-off between speed and accuracy we could also opt for ensemble models. Deep learning would do a great job in terms of accuracy if our sample was larger. Logistic regression has been shown to perform well on binary based features.

The main challenge in multi-label classification is data imbalance where some classes in the dataset are more frequent than others. In our case, we did not have to worry about this because there was a small variation of less that 20% in the classes. For the evaluation, measures for single-label are usually different from multi-label. Precision, recall as used in single class classification do not work in multi-label classification, we therefore used accuracy.

5. Results and Discussion

Binary relevance transformation when implemented with logistic regression classifier achieved an accuracy of 93.1%. This is very good performance. The performance should improve with more training.

To further evaluate the performance of these techniques we used the predictions of the test set to generate a schedule and the results are shown in figure 4 and 5. The classifier got wrong only 3 timeslots out of 44. This represented 93.18%. This against the constraint optimization approach which achieved 92.7%. However implementing hard constraints in constraint programming is more complex than classification.

Although the classifier got it correct on a few assignments, in figure 5, we illustrate one of the cases that was not satisfied from the classification. From the figure, the highlighted faculty (F6), according to their extracted schedule requests indicated unavailable at that specific time slot, however, the modeling made an incorrect assignment as seen on the right of the figure 5.

Using a standard Core 2 Duo E8400 at 2.0 GHz, 8GB RAM machine we ran timings for the computation of the two approaches. The timings include the pre-processing, post-processing as well the modeling. The whole process took about 83 seconds for the classification and 107 seconds for the constraint programming to generate a schedule.
6. Conclusion

In this research, we proposed a machine learning approach that can be used in employee scheduling problems. Results show that classification can achieve high accuracy in terms of satisfying both hard and soft constraints and that its performance is comparable to that of normally used constraint programming satisfaction used for automatic scheduling. The classification achieved an accuracy of 93.1% compared to constraint programming that achieved 92.7%. The performance of the classification is expected to improve with more training with more data.

In future, we explore and study the use of machine learning and data mining in constraint programming. As seen in this research, we had to formulate explicitly the constraints that underly the constraint programming application in our case. This has often been a difficult task. Even with the right constraints known, it was a challenge to formalize them in our flexible scheduling. Its possible to have some form of automation such that constraints are learned or formulation is learned from the data using machine learning.

References

[1] Boutell, M.R., Luo, J., Shen, X., Brown, C.M., 2004. Learning multi-label scene classification. Pattern Recognition 37, 1757 – 1771.
[2] Burke, E.K., Petrovic, S., 2002. Recent research directions in automated timetabling. European Journal of Operational Research 140, 266–280.
[3] Bürgy, R., Michon-Lacaze, H., Desaulniers, G., 2019. Employee scheduling with short demand perturbations and extensible shifts. Omega 89, 177 – 192.
[4] Deng, L., Yu, D., 2014. Deep Learning: Methods and Applications. volume 7. Foundations and Trends in Signal Processing.
[5] Glover, F., McMillan, C., 1986. The general employee scheduling problem. an integration of ms and ai. Computers and Operations Research 13, 563 – 573. Applications of Integer Programming.

[6] Hegewisch, O.T.C.B.L.M.A., 2010. Flexible working in europe: The evidence and the implications. European Journal of Work and Organizational Psychology 7, 61–78.

[7] Kletzander, L., Musliu, N., 2020. Solving the general employee scheduling problem. Computers and Operations Research 113, 104794.

[8] Maimon, O., Rokach, L., 2005. Data mining and knowledge discovery handbook. IFAC-PapersOnLine 48, 800 – 805.

[9] Morton, D.P., Popova, E., 2004. A bayesian stochastic programming approach to an employee scheduling problem. IIE Transactions 36.

[10] Nieuwenhuis, R., Oliveras, A., Tinelli, C., 2006. Solving sat and sat modulo theories: From an abstract davis–putnam–logemann–loveland procedure to dpll(t). J. ACM 53, 937–977.

[11] Schapire, R., Singer, Y., 2000. Boostexter: a boosting-based system for text categorization. Machine Learning 39, 135–168.

[12] Soria-Alcaraz, J.A., Ochoa, G., Swan, J., Valadez, J.M.C., Soberanes, H.J.P., Burke, E.K., 2014. Effective learning hyper-heuristics for the course timetabling problem. Eur. J. Oper. Res. 238, 77–86.

[13] Thompson, G.M., 1995. Improved Implicit Optimal Modeling of the Labor Shift Scheduling Problem. Management Science 41, 595–607.

[14] Tsoumakas, G., Katakis, I., 2007. Multi label classification: An overview. International Journal of Data Warehouse and Mining 3, 1–13.

[15] Wang, H., 2005. Flexible flow shop scheduling: Optimum, heuristics and artificial intelligence solutions. Expert Systems 22, 78–85.

[16] Wheatley, D., 2016. Employee satisfaction and use of flexible working arrangements. Work Employment and Society 31, 567–585.

[17] Zhang, L., Madigan, C.F., Moskewicz, M.H., Malik, S., 2001. Efficient conflict driven learning in a boolean satisfiability solver, in: Proceedings of the 2001 IEEE/ACM International Conference on Computer-aided Design, IEEE Press, Piscataway, NJ, USA. pp. 279–285.