Solving Employee Timetabling Problems Using Boolean Satisfiability

Fadi Aloul, Bashar Al-Rawi*, Anas Al-Farra, Basel Al-Roh
Department of Computer Engineering, American University of Sharjah (AUS), UAE
*Department of Computer Engineering, American University in Dubai (AUD), UAE

Abstract — **The Employee Timetabling Problem (ETP) is concerned with assigning a number of employees into a given set of shifts over a fixed period of time, e.g. a week, while meeting the employee's preferences and organizational work regulations. The problem also attempts to optimize the performance criteria and distribute the shifts equally among the employees. The problem is considered a classical NP-complete optimization problem. It has received intensive research during the past few years given its use in industries and organizations. Several formulations and algorithms based on local search have been proposed to solve ETPs [12, 15, 17, 18]. In this paper, we propose a complete approach using integer linear programming (ILP) to solve these problems. The ILP model of interest is developed and solved using the generic ILP solver CPLEX and the Boolean Satisfiability ILP solver PBS. Experimental results indicate that the proposed model is tractable for reasonable-sized ETP problems.**

Index Terms — **Employee Timetabling, Optimization, Scheduling, ILP, Boolean Satisfiability.**

1. INTRODUCTION

Employee Timetabling Problem (ETP) represent an important class of optimization problems in operational research. The problem was originally associated with timetabling of classes in schools and universities [22], but has recently been extended to schedule employees in large organizations, such as hospitals [12], factories, etc. Given a number of employees and shifts, the goal is to assign employees to shifts while satisfying all of the employees and organizational constraints. Many formulations and algorithms have been proposed to solve ETP. Most of these algorithms are based on *local* search techniques, namely hill climbing, simulated annealing, and tabu search [15, 17, 21, 18, 7]. Such algorithms cannot prove unsatisfiability or guarantee that a solution is optimal. In other words, if a solution is found, it cannot guarantee that this solution has the best possible optimization cost.

In this paper, we propose an integer linear programming (ILP) approach to solve the ETP. The approach is *complete* and hence examines the entire search space defined by the problem to prove that either the problem has no solution, i.e. the problem is unsatisfiable, or that a solution does exist, i.e. the problem is satisfiable. If the problem is satisfiable, our approach will search all possible solutions to find the optimal solution.

Recently, advanced Boolean Satisfiability (SAT) solvers have been extended to solve ILP problems. The SAT problem is a central problem in artificial intelligence and computer science and has received considerable attention from researchers. Many complex Engineering problems have been successfully solved using SAT. Such problems include routing [20], power optimization [2], verification [4], and graph coloring [9], etc. Today, several powerful SAT solvers exist and are able of handling problems consisting of thousands of variables and millions of constraints.

In this paper, we will also show how to formulate the ETP as an ILP problem and explore the possibility of using advanced SAT techniques to solve the ETP problem. We also use the commercial generic-based ILP solver, CPLEX, to test our instances. We report results for a variety of organization sizes. Initial results indicate the effectivity of the proposed approach. The proposed approach is *complete* and is guaranteed to identify the optimal schedule.

This paper is organized as follows. Section 2 provides a general overview of Boolean Satisfiability. Section 3 shows how to formulate the ETP as an ILP instance. A detailed example is shown in Section 5. Experimental results are presented and discussed in Section 5. The paper is concluded in Section 6.

2. BOOLEAN SATISFIABILITY

The Boolean satisfiability (SAT) problem involves finding an assignment to a set of binary variables that satisfies a given set of constraints. In general, these constraints are expressed in *products-of-sum* form, also known as *conjunctive normal form* (CNF). A CNF formula φ on n binary variables $x_1, \ldots, x_n$ consists of the conjunction (AND) of m clauses $\omega_1, \ldots, \omega_m$ each of which consists of the disjunction (OR) of k literals. A literal l is an occurrence of a Boolean variable or its complement.

Most current SAT solvers [13, 19, 16] are based on the original Davis-Putnam backtrack search algorithm [10]. The algorithm performs a depth first search process that traverses the space of 2^n variable assignments until a satisfying assignment is found (the formula is satisfiable), or all combinations have been exhausted (the formula is unsatisfiable). Originally, all variables are unassigned. The algorithm begins by choosing a decision assignment to an unassigned variable. A decision tree is maintained to keep track of variable assignments. After each decision, the algorithm determines the implications of the assignment on other variables. This is obtained by forcing the assignment of the variable representing an unassigned literal in an unresolved clause, whose all other literals are assigned to 0, to satisfy the clause. This is referred to as the *unit clause* rule. If no conflict is detected, the algorithm makes a new decision on a new unassigned variable. Otherwise, the backtracking process unassigns one or more recently assigned variables and the search continues in another area of the search space.

As an example, a CNF instance $f = (a+b)(\bar{b}+c)$ consists of 3 variables, 2 clauses, and 4 literals. The assignment $\{a = 0, b = 1, c = 0\}$ leads to a conflict, whereas the assignment $\{a = 0, b = 1, c = 1\}$ satisfies f. Note that a problem with n variables will have 2^n possible assignments to test. The above example with 3 variables has 8 possible assignments. A instance with 100 variables will have 1.27e+30 assignments. Assuming a processor that can verify an assignment every 1 nanosecond, the processor will complete testing all 2^{100} assignments in 4e+12 years.

Despite the SAT problem being NP-Complete [5], several powerful methods have been proposed to expedite the backtrack search algorithm. One of the best methods is known as the conflict analysis procedure [16] and has been implemented in almost all SAT solvers. Whenever a conflict is detected, the procedure identifies the causes of the conflict and augments the clause database with additional clauses, known as *conflict-induced clauses*, to avoid regenerating the same conflict in future parts of the search process. In essence, the procedure performs a form of learning from the encountered conflicts. Significant speedups have been achieved with the addition of conflict-induced clauses, as they tend to effectively prune the search space.

Intelligent decision heuristics and random restarts [19], also played an important role in enhancing the SAT solvers performance. Chaff [19] proposed an effective decision heuristic, known as VSIDS, and implemented several other enhancements, including random restarts, which lead to dramatic performance gains on many CNF instances.

Another recent extension to SAT solvers deals with its input format. Restricting the input of SAT solvers to CNF formulas can restrict their usage in various domains. Therefore, researchers have focused on extending SAT solvers to handle stronger input representations. Specifically, SAT solvers [1, 6, 23] have recently been extended to handle pseudo-Boolean (PB) constraints which are linear inequalities with integer coefficients that can be expressed in the normalized form [1] of:

$$a_1x_1 + a_2x_2 + \ldots + a_nx_n \geq b \tag{1}$$

where $a_i, b \in Z^+$ and x_i are literals of Boolean variables. Note that any CNF clause can be viewed as a PB constraint, e.g. clause $(a \vee b \vee c)$ is equivalent to $(a + b + c \geq 1)$.

PB constraints can, in some cases, replace an exponential number of CNF constraints. They have been found to be very efficient in expressing "counting constraints" [1]. Furthermore, PB extends SAT solvers to handle *optimization* problems as opposed to only *decision* problems. Subject to a given set of CNF and PB constraints, one can request the minimization (or maximization) of an objective function which consists of a linear combination of the problem's variables.

$$\sum_{i=1}^{n} a_i x_i \tag{2}$$

This feature has introduced many new applications to the SAT domain. Specifically, all 0-1 ILP problems (i.e. ILP problems whose variables are Boolean) can be easily solved now by SAT solvers. Recent studies has shown that SAT-based optimization solvers can in fact compete with the best generic-based ILP solvers [1, 6].

3. Problem formulation

This section develops the 0-1 ILP model for the ETP and illustrates its applicability using small examples. Consider an organization with m employees. Assume the organization runs for d days/week, has s shifts per day, and needs w employees to be working in each shift. We first introduce the following decision variables per employee:

$$E_i = \begin{cases} 1 : \text{employee i is working} \\ 0 : \text{otherwise} \end{cases} \tag{3}$$

$$D_{ij} = \begin{cases} 1 : \text{employee i is working at day j} \\ 0 : \text{otherwise} \end{cases} \tag{4}$$

$$S_{ijk} = \begin{cases} 1 : \text{employee i is assigned at day j to shift k} \\ 0 : \text{otherwise} \end{cases} \tag{5}$$

Hence the total number of variables will be $m + md + mds$. Two sets of constraints are added to represent the organizational regulations and employee preferences, respectively. These are described below.

A. Organizational Constraints

The organization needs to make sure that w employees are assigned to each possible shift. This is expressed in the following constraint:

$$\left(\sum_{i=1}^{m} S_{ijk}\right) = w \qquad \forall j, \forall k \tag{6}$$

The constraint can be customized per shift. For example the morning shift can have more employees than the afternoon or evening shift.

B. Employee Constraints

The first two constraints define the relationship between the variables E_i, D_{ij}, and S_{ijk}. These constraints are added for all employees. Constraint (7) ensures that if an employee is assigned to any shift, then the employee is working:

$$\left(\bigcup_{k=1}^{s} S_{ijk}\right) \leftrightarrow D_{ij} \qquad \forall j, \forall i \tag{7}$$

Constraint (8) ensures that if an employee is working during any day, then the employee is working, i.e. not on vacation, etc.

$$\left(\bigcup_{j=1}^{d} D_{ij}\right) \leftrightarrow E_i \qquad \forall i \tag{8}$$

The following set of constraints deal with the employee preferences. They can be specifically assigned to each employee depending on his/her preferences.

- **Alternating days:** An employee cannot work for two consecutive days.

$$D_{ij} \rightarrow \overline{D_{i(j+1)}} \qquad \forall i, \forall j, j \neq d \tag{9}$$

- **Number of shifts per employee**: An employee can work upto k_{max} shifts per day.

$$\left(\sum_{k=1}^{s} S_{ijk}\right) \leq k_{max} \qquad \forall j, \forall i \tag{10}$$

- **Number of working days per employee**: An employee must work not less than h_{min} days and no more than h_{max} days per week.

$$\left(\sum_{j=1}^{d} D_{ij}\right) \geq h_{min} \qquad \forall i \tag{11}$$

$$\left(\sum_{j=1}^{d} D_{ij}\right) \leq h_{max} \qquad \forall i \tag{12}$$

- **Off-days**: An employee may be sick or on vacation and unable to work. Assuming employee *i* cannot work on day *j*, this can be expressed using:

$$D_{ij} = 0 \tag{13}$$

- **Off-shifts**: An employee may be working part-time or unable of working during specific shifts. Assuming employee *i* cannot work in the morning shifts (indicated by $k = 1$), due to another job. This can be expressed using:

$$\sum_{j=1}^{d} S_{ij1} = 0 \tag{14}$$

Note that constraints (9)-(14) are optional and a valid schedule can be generated without them. However, the schedule is likely to be meaningless, since the employees preferences have not been taken into consideration.

The last goal is to distribute the work evenly among all employees and maximize their usage. This is expressed in the following linear cost function which must be maximized:

$$Max\left(\sum_{i=1}^{m} E_i\right) \tag{15}$$

This objective function will minimize the number of idle workers. Other optimization functions can be similarly expressed for other organizational purposes such as reducing the labor cost, minimizing the number of employees, etc.

4. Illustrative Example

In this example, we examine a company consisting of two employees. Assume the company is open 3 days a week and includes two shifts per day (e.g. morning and evening shift). Each shift must be assigned to a single employee. As shown in Section 3, the number of variables is $2 + 2(3) + 2(3)(2) = 20$. Figure 1 shows the distribution of variables. We use the variable naming convention used in the previous section. Each node represents a Boolean variable. For example, E_1 and E_2 represent employees 1 and 2, respectively. D_{12} represents employee 1 working in day 2. S_{231} represents employee 2 working in the morning shift of day 3.

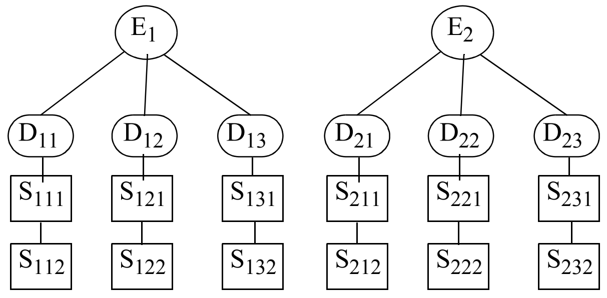

Fig. 1. Employee Timetabling Example

The company's constraint indicating that each shift must be assigned to a single employee is expressed as:

$S_{111} + S_{211} = 1$	$S_{112} + S_{212} = 1$
$S_{121} + S_{221} = 1$	$S_{122} + S_{222} = 1$
$S_{131} + S_{231} = 1$	$S_{132} + S_{232} = 1$

The above constraint are represented in PB form. The following set of CNF constraints enforce the relationship between the variables.

Logic Expression	CNF Constraint
$S_{111} \vee S_{112} \rightarrow D_{11}$	$(\overline{S_{111}} \vee D_{11})(\overline{S_{112}} \vee D_{11})$
$S_{121} \vee S_{122} \rightarrow D_{12}$	$(\overline{S_{121}} \vee D_{12})(\overline{S_{122}} \vee D_{12})$
$S_{131} \vee S_{132} \rightarrow D_{13}$	$(\overline{S_{131}} \vee D_{13})(\overline{S_{132}} \vee D_{13})$

$S_{211} \vee S_{212} \rightarrow D_{21}$	$(\overline{S_{211}} \vee D_{21})(\overline{S_{212}} \vee D_{21})$
$S_{221} \vee S_{222} \rightarrow D_{22}$	$(\overline{S_{221}} \vee D_{22})(\overline{S_{222}} \vee D_{22})$
$S_{231} \vee S_{232} \rightarrow D_{23}$	$(\overline{S_{231}} \vee D_{23})(\overline{S_{232}} \vee D_{23})$

Logic Expression	CNF Constraint
$D_{13} \rightarrow S_{131} \vee S_{132}$	$(\overline{D_{13}} \vee S_{131} \vee S_{132})$
$D_{11} \rightarrow S_{111} \vee S_{112}$	$(\overline{D_{11}} \vee S_{111} \vee S_{112})$
$D_{12} \rightarrow S_{121} \vee S_{122}$	$(\overline{D_{12}} \vee S_{121} \vee S_{122})$
$D_{21} \rightarrow S_{211} \vee S_{212}$	$(\overline{D_{21}} \vee S_{211} \vee S_{212})$
$D_{22} \rightarrow S_{221} \vee S_{222}$	$(\overline{D_{22}} \vee S_{221} \vee S_{222})$
$D_{23} \rightarrow S_{231} \vee S_{232}$	$(\overline{D_{23}} \vee S_{231} \vee S_{232})$

Logic Expression	CNF Constraint
$E_1 \rightarrow D_{11} \vee D_{12} \vee D_{13}$	$(\overline{E_1} \vee D_{11} \vee D_{12} \vee D_{13})$
$E_2 \rightarrow D_{21} \vee D_{22} \vee D_{23}$	$(\overline{E_2} \vee D_{21} \vee D_{22} \vee D_{23})$
$D_{11} \vee D_{12} \vee D_{13} \rightarrow E_1$	$(\overline{D_{11}} \vee E_1)(\overline{D_{12}} \vee E_1)$ $(\overline{D_{13}} \vee E_1)$
$D_{21} \vee D_{22} \vee D_{23} \rightarrow E_2$	$(\overline{D_{21}} \vee E_2)(\overline{D_{22}} \vee E_2)$ $(\overline{D_{23}} \vee E_2)$

The final set of constraints represents the employees preferencs. Assume the following working preferences:

- Employee 1 works on alternative days

Logic Expression	CNF Constraint
$D_{11} \rightarrow \overline{D_{12}}$	$(\overline{D_{11}} \vee \overline{D_{12}})$
$D_{12} \rightarrow \overline{D_{13}}$	$(\overline{D_{12}} \vee \overline{D_{13}})$

- Employee 2 works for a maximum of 4 shifts per week

  PB: $S_{211} + S_{212} + S_{221} + S_{222} + S_{231} + S_{232} \leq 4$
- Employee 1 cannot work during morning shifts

  PB: $S_{111} + S_{121} + S_{131} = 0$
- Employee 2 cannot work on day 3

  PB: $D_{23} = 0$
- Employee 1 must work at least 2 days per week

  PB: $D_{11} + D_{12} + D_{13} \geq 2$

In summary, the instance consists of 20 variables, 10 PB constraints, and 24 CNF constraints.

5. Experimental Results

In this section, we evaluate the use of ILP solvers to solve the ETP problem. A tool was developed using Visual Basic with an easy-to-use GUI interface. The user inputs the employee preferences and organization constraints. The tool translates the user input into a 0-1 ILP instance as described in Section 3. The instance is then passed into an ILP solver and once the search is completed, the solution, if satisfiable, is translated into an easy-to-read schedule. The tool can handle an unlimited number of employees, has the ability to add unique preferences for each employee, and generates a schedule for a period of month. We used two of the best 0-1 ILP solvers: (1) The SAT-based 0-1 ILP solver, PBS [1, 3], and (2) the leading generic commercial ILP solver, CPLEX [14]. PBS is an advanced SAT solver that can handle both CNF and PB constraints. It employs the latest advances in the SAT technology. The experiments were conducted on a Pentium Xeon 3.2 GHz machine, equipped with 4 GBytes of RAM, and running Linux. The runtime limit was set to 1000 seconds.

We present runtime results for 12 examples of organizations. Table 1 shows the organization and employee preferences, the runtime in seconds of PBS and CPLEX, and the result of the search. A SAT result indicates that the instance is satisfiable, and hence a schedule is generated. A UNS result indicates that the instance is unsatisfiable, and hence a schedule cannot be generated given these constraints. In this case, relaxing the constraints, such as adding more employees or reducing the number of shifts or working days, can help make the problem satisfiable. Note that if the search is completed, both solvers must yield the same optimal result.

The first 4 columns in Table 1 show the organization regulations and the last 3 columns show the employee preferences. A brief description of the preferences is given below:

Column	**Constraint Description**
2	Number of employees
3	Number of working days per week
4	Number of shifts per day
5	Number of employees per shift. If 3 numbers are listed, these correspond to the 3 shifts, respectively
6	Alternating Days policy enforced
7	Maximum number of shifts per day for each employee
8	Maximum number of working days per week for each employee

The fairness constraint was enforced in all examples. Several observations are in order:

- PBS was able to solve most instances, but timed-out in 3 cases. In these cases, the instances were unsatisfiable, and hence didnt have a solution. CPLEX, on the other hand, was able to solve all instances (both satisfiable and unsatisfiable) in less than a second.

- The larger the organization, hence the 0-1 ILP instance, the longer is the search runtime.
- The approach is complete and is guaranteed to find a fair schedule given enough time and memory resources.

6. Conclusion

In this paper, we present an ILP-based approach to generating employees timetables. The proposed approach utilizes advanced generic-based and Boolean satisfiability-based ILP solvers to find a schedule that satisfies the organization's rules and its employees preferences. The proposed approach can be used to optimize a given objective function such as minimizing labor costs or maximizing fairness among employees. We show how to formulate the employee timetabling as an ILP problem. The approach was tested on organizations with various sizes and showed promising results. The approach is *complete* and will find the required timetable, or will indicate that no timetable exists that meets the current organization's conditions.

Table 1. Experimental results using the ILP solvers PBS and CPLEX.

	2	3	4	5	6	7	8			
	Organization				Employee					
Example	# of employees	# working days	# shifts/day	# employees/shift	Alternating days	Max shifts/day	Max days/week	Result	PBS4 Time	CPLEX Time
1	6	5	3	1	Y	1	-	SAT	0	0.02
2	12	5	3	2	Y	1	-	SAT	0	0.03
3	18	5	3	3	Y	1	-	SAT	0	0.16
4	15	5	3	3	Y	1	-	UNS	>1000	0.04
5	25	5	3	5	Y	1	-	UNS	>1000	0.06
6	27	5	3	4	Y	1	-	SAT	0	0.07
7	25	5	3	3	Y	1	-	SAT	0	0.06
8	18	5	3	3, 2, 1	Y	1	-	SAT	0	0.04
9	15	5	3	3, 2, 1	Y	1	-	SAT	0.01	0.04
10	30	5	3	6, 4, 2	Y	1	-	SAT	0.01	0.06
11	45	5	3	9, 6, 3	Y	1	-	SAT	0.01	0.11
12	18	5	3	2	Y	1	2	UNS	>1000	0.05

7. References

[1] F. Aloul, A. Ramani, I. Markov, and K. Sakallah, "Generic ILP Versus Specialized 0-1 ILP: An Update," in *Proc. of the Int'l Conference on Computer Aided Design*, 450-457, 2002.

[2] F. Aloul, S. Hassoun, K. Sakallah, and D. Blaauw, "Robust SAT-Based Search Algorithm for Leakage Power Reduction," in *Proc. of the Int'l Workshop on Power and Timing Modeling, Optimization, and Simulation*, 167-177, 2002.

[3] B. Al-Rawi and F. Aloul, PBS4 SAT Solver, 2005. Available at: *http://www.eecs.umich.edu/~faloul/Tools/pbs4*

[4] A. Biere, A. Cimatti, E. Clarke, M. Fujita, and Y. Zhu, "Symbolic Model Checking Using SAT Procedures Instead of BDDs," in *Proc. of the Design Automation Conference* (DAC), 317-320, 1999.

[5] S. Cook, "The Complexity of Theorem Proving Procedures," in *Proc. of the ACM Symposium on the Theory of Computing*, 151-158, 2004.

[6] D. Chai and A. Kuehlmann, "A Fast Pseudo-Boolean Constraint Solver", in *Proc. of* DAC, 830-835, 2003.

[7] M. Chiarandini, A. Schaerf, and F. Tiozzo, "Solving Employee Timetabling Problems with Flexible Workload Using Tabu Search," in *Proc. of the Int'l Conf. on the Practice and Theory of Automated Timetabling*, 298-302, 2000.

[8] T. Cooper and J. Kingston, "The Complexity of Timetable Construction Problems," in *Proc. of the International Conference on Practice and Theory of Automated Timetabling, LNCS 1153, Springer-Verlag,* 283-295, 1996.

[9] N. Creignou, S. Kanna, and M. Sudan, "Complexity Classifications of Boolean Constraint Satisfaction Problems", *SIAM Press*, 2001.

[10] M. Davis, G. Longman, and D. Loveland "A Machine Program for Theorem Proving," in *Journal of the ACM*. 5(7), 394-397, 1962.

[11] D. de Werra, "The Combinatorics of Timetabling," *in European Journal of Operational Research*, 504-513, 1997.

[12] K. Dowsland, "Nurse scheduling with tabu search and strategic oscillation," in *European Journal of Operational Research*, v. 106, 393-407, 1998.

[13] E. Goldberg and Y. Novikov, "BerkMin: A Fast and Robust SAT-solver," in *Proc. of the Design Automation and Test Conference in Europe* (DATE), 142-149, 2002.

[14] ILOG CPLEX, *http://www.ilog.com/products/cplex*

[15] L. Kragelund, "Solving a timetabling problem using hybrid genetic algorithms," in *Software - Practice and Experience*, v. 27, 1121-1134, 1997.

[16] J. Marques-Silva and K. Sakallah "GRASP: A Search Algorithm for Propositional Satisfiability," in *IEEE Trans. on Computers*, 48(5), 506-521, 1999.

[17] A. Meisels, E. Gudes, and G. Solotorevsky, "Combining Rules and Constraints for Employee Timetabling," in *Int'l Journal of Intelligent Systems*, v. 12, 419-439, 1997.

[18] A. Meisels and A. Schaerf, "Modelling and Solving Employee Timetabling Problems," in *Annals of Mathematics and Artificial Intelligence*, 39(1-2), 41-59, 2003.

[19] K. Moskewicz, C. Madigan, Y. Zho, and S. malik "Chaff: Engineering an efficient SAT solver," in *Proc. of the Design Automation Conference,* 503-535, 2001.

[20] G. Nam, F. A. Aloul, K. A. Sakallah, and R. Rutenbar, "A Comparative Study of Two Boolean Formulations of FPGA Detailed Routing Constraints," in ISPD, 222-227, 2001.

[21] A. Schaerf and A. Meisels, "Solving Employee Timetabling Problems by Generalized Local Search," in *Proc. of the Italian Conference on Artificial Intelligence*, 493-502, 1999.

[22] A. Schaerf and L. Gaspero, "Local Search Techniques for Educational Timetabling Problems," in *Proc. of the Int'l Symp. on Operations Research in Slovenia*, 13-23, 2001.

[23] H. Sheini and K. Sakallah, "Pueblo: A Modern Pseudo-Boolean SAT Solver," in *Proc. of the Design, Automation, and Test Conference in Europe*, vol. 2, 684-685, 2005.