A 0/1 Integer Programming Model for the Office Space Allocation Problem

Özgür Ülker [1] and Dario Landa-Silva [2]

Automated Scheduling, Optimisation and Planning (ASAP) Research Group
School of Computer Science, University of Nottingham, UK

Abstract

We propose a 0/1 integer programming model to tackle the office space allocation (OSA) problem which refers to assigning room space to a set of entities (people, machines, roles, etc.), with the goal of optimising the space utilisation while satisfying a set of additional requirements. In the proposed approach, these requirements can be modelled as constraints (hard constraints) or as objectives (soft constraints). Then, we conduct some experiments on benchmark instances and observe that setting certain constraints as hard (actual constraints) or soft (objectives) has a significant impact on the computational difficulty on this combinatorial optimisation problem.

Keywords: Office Space Allocation, Integer Programming

1 Introduction

In this work, we tackle the office space allocation (OSA) problem, commonly encountered in universities, companies, government institutions, etc. The problem is that of having a set of rooms (offices, halls, etc.), a set of entities (people, machines, roles, etc.) and then to allocate each of the entities to a room. Each room has a capacity and each entity has a size. One of the goals is to optimise the space utilisation by means of minimising the space wastage (underusing room capacity) and space overuse (exceeding room capacity). Other goals might include satisfying certain requirements that establish conditions for the way in which entities are allocated to rooms (more details below).

[1] Email: oxu@cs.nott.ac.uk
[2] Email: dario.landasilva@nottingham.ac.uk

One of the earliest works on the optimisation of office space utilisation is that of Ritzman et al. [8], who developed a linear programming model for the distribution of academic offices at the Ohio State University. Benjamin et al. [1] also used linear programming for planning the layout of floor space in a manufacturing laboratory. Giannikos et al. [5] developed a goal programming approach to automate the distribution of offices among staff in an academic institution. More recently, researchers have proposed several heuristic approaches, including population-based meta-heuristics and multi-objective methods, to tackle the office space allocation problem in Universities [3,2,4]. The most recent results on benchmark instances of the office space allocation problem investigated here, are reported in the paper by Landa-Siva and Burke [7] who developed an asynchronous cooperative local search method. In this study, we tackle the office space allocation problem as described in Landa-Silva [6]. We develop a 0/1 integer programming formulation for this problem and then apply CPLEX to solve the model.

Section 2 defines the office space allocation problem studied here and presents the proposed 0/1 integer programming model. Section 3 briefly describes the benchmark instances used in our experiments and presents the results from our initial study to investigate the computational difficulty of the proposed model. Our experiments focus on trying to understand whether setting some constraints as hard or as soft has a significant impact of the difficulty of the OSA problem. The Final Remarks section summarises our observations and describes some future work.

2 The Proposed 0/1 IP Formulation

The set of rooms is denoted by R and the set of entities is denoted by E. Let S_e be the size of entity e and C_r the capacity of room r. There is a matrix δ of $|R| \times |E|$ binary decision variables where each $\delta_{er} = 1$ if entity e is allocated to room r, otherwise $\delta_{er} = 0$. Let A be the adjacency list of $|R|$ adjacency vectors each one denoted by A_r and holding the list of rooms that are adjacent to room r. Similarly, let N be the nearby list of $|R|$ nearby vectors each one denoted by N_r and holding the list of rooms that are near to room r. The adjacency vector A_r for a room r is usually quite smaller compared to the nearby vector N_r, i.e. more rooms are considered to be near to room r than considered adjacent to the same room.

There are seven requirements or constraints that we handle here. Any of these constraints can be set as *hard* (must be satisfied) or *soft* (desirable to satisfy) in our formulation. In other words, when a constraint is set as soft, minimising its violation becomes an objective in the problem formulation. The exception here is the *All allocated* constraint (all entities must be allocated) which is always enforced. The next subsections present these alternative formulations.

2.1 Modelling Hard Constraints

All allocated: each entity $e \in E$ must be allocated to exactly one room $r \in R$.

$$\sum_{r \in R} \delta_{er} = 1 \quad \forall e \in E \tag{1}$$

Allocation: entity e has to be allocated to room r.

$$\delta_{er} = 1 \tag{2}$$

Same room: entities e_1 and e_2 have to be allocated to the same room.

$$\begin{gathered} \delta_{e_1 r} = 1 \leftrightarrow \delta_{e_2 r} = 1 \quad \forall r \in R \quad \text{i.e.} \\ \delta_{e_1 r} - \delta_{e_2 r} = 0 \quad \forall r \in R \end{gathered} \tag{3}$$

Not sharing: entity e should not share a room with any other entity.

$$\begin{gathered} \delta_{er} = 1 \rightarrow \delta_{\beta r} = 0 \quad \forall \beta \in (E - e) \quad \forall r \in R \quad \text{i.e.} \\ \delta_{er} + \delta_{\beta r} \leq 1 \quad \forall \beta \in (E - e) \quad \forall r \in R \end{gathered} \tag{4}$$

$$\begin{gathered} \delta_{er} = 1 \rightarrow \sum_{e \in E} \delta_{er} = 1 \quad \forall r \in R \quad \text{i.e.} \\ \delta_{er} \leq \sum_{e \in E} \delta_{er} \leq |E| - (|E| - 1)\delta_{er} \quad \forall r \in R \end{gathered} \tag{5}$$

Adjacency: entities e_1 and e_2 have to be allocated to adjacent rooms.

$$\begin{gathered} \delta_{e_1 r} = 1 \rightarrow \sum_{s \in A_r} \delta_{e_2 s} = 1 \quad \forall r \in R \quad \text{i.e.} \\ \delta_{e_1 r} \leq \sum_{s \in A_r} \delta_{e_2 s} \leq 1 \quad \forall r \in R \end{gathered} \tag{6}$$

Group by: entities in a group have to be allocated near to the group head.

$$\begin{gathered} \delta_{er} = 1 \rightarrow \sum_{s \in N_r} \delta_{\beta s} = 1 \quad \forall r \in R \quad \text{i.e.} \\ \delta_{er} \leq \sum_{s \in N_r} \delta_{\beta s} \leq 1 \quad \forall r \in R \end{gathered} \tag{7}$$

Away from: entities e_1 and e_2 have to be allocated in rooms away from each other.

$$\begin{gathered} \delta_{e_1 r} = 1 \rightarrow \sum_{s \in N_r} \delta_{e_2 s} = 0 \quad \forall r \in R \quad \text{i.e} \\ 0 \leq \sum_{s \in N_r} \delta_{e_2 s} \leq 1 - \delta_{e_1 r} \quad \forall r \in R \end{gathered} \tag{8}$$

2.2 Modelling Constraints as Objectives

Allocation: indicator variable σ^{c1} is set if soft constraint not satisfied.

$$\sigma^{c1} = 1 - \delta_{\alpha r} \tag{9}$$

Same room: indicator variable σ^{c2} is set if soft constraint not satisfied (ε is an arbitrarily small number which we set to 0.01).

$$2\sigma_r^{c2} - 1 \leq \delta_{e_1 r} - \delta_{e_2 r} \leq 1 - \varepsilon + \varepsilon\sigma_r^{c2} \quad \forall r \in R \tag{10}$$

$$\sigma^{c2} = \sum_{r \in R} \sigma_r^{c2} \tag{11}$$

Not sharing: indicator variable σ^{c3} is set if soft constraint not satisfied.

$$\forall \beta \in (E - e) \quad (1+\varepsilon) - (1+\varepsilon)\sigma_r^{c3} \leq \delta_{er} + \delta_{\beta r} \leq 2 - \sigma_r^{c3} \quad \forall r \in R \tag{12}$$

$$\sigma^{c3} = \sum_{r \in R} 1 - \sigma_r^{c3} \tag{13}$$

Adjacency: indicator variable σ^{c4} is set if soft constraint not satisfied.

$$\sigma_r^{c4} + \delta_{e_1 r} - 1 \leq \sum_{s \in A_r} \delta_{e_2 s} \leq \delta_{e_1 r} - \varepsilon + (1+\varepsilon)\sigma_r^{c4} \tag{14}$$

$$\sigma^{c4} = \sum_{r \in R} 1 - \sigma_r^{c4} \tag{15}$$

Group by: indicator variable σ^{c5} is set if soft constraint not satisfied.

$$\sigma_r^{c5} + \delta_{er} - 1 \leq \sum_{s \in N_r} \delta_{\beta s} \leq \delta_{er} - \varepsilon + (1+\varepsilon)\sigma_r^{c5} \quad \forall r \in R \tag{16}$$

$$\sigma^{c5} = \sum_{r \in R} 1 - \sigma_r^{c5} \tag{17}$$

Away from: indicator variable σ^{c6} is set if soft constraint not satisfied.

$$1 - \delta_{e_1 r} + \varepsilon - (1+\varepsilon)\sigma_r^{c6} \leq \sum_{s \in N_r} \delta_{e_2 s} \leq 2 - \delta_{er} - \sigma_r^{c6} \quad \forall r \in R \tag{18}$$

$$\sigma^{c6} = \sum_{r \in R} 1 - \sigma_r^{c6} \tag{19}$$

Note that we use two different formulations for the *Not sharing* constraint. The first formulation is based on comparing each row vector (rooms) of that specific entity with all the other row vectors of the other entities. The other formulation is based on the notion that the column sum of a room should be equal to one if that specific entity is allocated to that room. We observed that the former formulation

uses too much memory in the linear relaxation stage at the root node but yields better results faster while the latter formulation uses up considerably less memory. Therefore, in our experiments, we used a combination of two formulations for a balance of memory consumption and computation time.

2.3 Objective Function

We consider the minimisation objective to be a weighted aggregating function comprising of two parts:

$$ObjectiveFunction = UsagePenalty + SoftConstraintPenalty$$

The *UsagePenalty* part is the weighted sum of the penalty due to the overuse and underuse of office space. Usually, it is more undesirable for a room to be overused than underused. Then, we penalise overuse more by setting a weight equal to 2. The amount of overused or underused office space is simply calculated by taking the absolute value of the difference between the room capacity and the space used by the entities allocated to the room.

$$UsagePenalty = underuse + 2 \times overuse$$

The *SoftConstraintPenalty* is the weighted sum of the penalty due to the violation of the constraints that are considered soft. Since we want to compare our results with those previously reported in the literature, we use the penalty weights for the violation of soft constraints as in [6]. Recall that S_e is the size of entity e and C_r is the capacity of room r. The penalty weight for the violation of soft constraint c_j is denoted as w^{cj}. Then, the objective function to minimise is given by:

$$Z = \sum_{r \in R} max\left(C_r - \sum_{e \in E} \delta_{er} S_e \, , \, 2 \sum_{e \in E} \delta_{er} S_e - C_r\right) + \sum_{j=1}^{j=6} w^{cj} \sum^{|cj|} \sigma^{cj} \tag{20}$$

3 Experiments and Results

We used a set of six benchmark instances taken from [6]. To solve the 0/1 IP formulation, we used CPLEX 11 (single-threaded mode) on a PC with processor Core 2 Duo E8400 3Ghz and 2GB of RAM. When dealing with the constraints as objectives, we used the penalty weights as in [6]: *Allocation* penalty = 20, *Same room* penalty = 10, *Not sharing* penalty = 50, *Adjacency* penalty = 10, *Grouped by* penalty = 11.18, and *Away from* penalty = 10.

In our initial experiments, we set all constraints as hard and tried to minimise the space usage penalty. However, we were unable to find a feasible solution for any of the benchmark datasets. We then observed that the computation time required

by the solver increases considerably as more terms are included in the objective function. For example, we incorporated all constraints as soft into the objective function. For larger instances, this resulted in unreasonable computation times with poor lower bounds. Therefore, we later decided to set some of the soft constraints as hard removing them from the objective function. By doing this, we were able to obtain better results in considerably less computation time. We now report on experiments in which the two constraints studied are the *Allocation* and *Same room* constraints.

Table 1 shows the results obtained by setting the *Same room* constraints as hard and using 2000 seconds as the limit on the computation time. The 2nd and 3rd columns show the obtained results and percentage gaps from the optimal solution when all the *Allocation* constraints are set as hard. The 4th and 5th columns show the obtained results and percentage gaps when all the *Allocation* constraints are set as soft. Note that except for instance Notta, the solver was able to obtain optimal results for all instances. We can also observe that for several instances, the optimal result remains the same regardless of the *Allocation* constraints being set as hard or soft when *Same room* constraints are set as hard.

	Allocation (hard)		**Allocation (soft)**	
Instance	*Result*	*Gap %*	*Result*	*Gap %*
Notta	379.88	0.0	N/A	N/A
Nottb	410.26	0.0	410.26	0.0
Nottc	414.58	0.0	305.73	0.0
Nottd	202.73	0.0	202.73	0.0
Notte	177.70	0.0	177.70	0.0
Wolver	634.20	0.0	634.20	0.0

Table 1
Results obtained on the satisfaction of the *Allocation* constraints when the *Same room* constraints are set as hard.

We observed from our experiments that the most difficult soft constraint to optimise seems to be the *Same room* constraint. To clarify this issue, we now present additional results on the Notta and Nottb instances from experiments in which we set a percentage of the *Same room* constraints as hard while *Allocation* constraints are set as hard (the other constraints are set as before). The time limit for these experiments was 1 hour and 2 hours for Nottb and Notta respectively. The results are presented in Table 2 where the 1st column shows the percentage of *Same room* con-

	Notta			Nottb		
%	*Result*	*Bound*	*Gap %*	*Result*	*Bound*	*Gap %*
0	410.26	410.26	0.00	379.88	379.88	0.00
20	351.06	350.96	0.03	379.88	353.22	7.02
40	312.28	286.80	8.16	379.88	353.22	7.02
60	309.61	260.52	15.86	385.38	333.89	13.36
80	265.26	180.51	31.95	415.91	307.88	25.97
100	246.18	139.35	43.40	392.88	307.88	21.64

Table 2
Results obtained by setting a percentage of Same Room constraints as soft while *Allocation* constraints are set as hard.

straints set as hard in each instance. The columns *result*, *bound* and *gap* show the results found and the percentage gap from the lower bound obtained after the time limit expires. As it can be seen from these results, while it is possible to 'soften' the *Same room* constraint and obtain better results for Nottb, the lower bound drops drastically for this dataset (from 410.26 to 139.35). However for Notta instance, the drop for the lower bound was not as drastic (from 379.88 to 307.88) yet we were not even able to improve the results when we 'soften' the *Same room* constraint.

Instance	*Best Obtained*	*Best Literature*
Notta	379.88	482.20
Nottb	246.18	417.20
Nottc	305.73	315.40
Nottd	202.70	N/A
Notte	177.70	N/A
Wolver	634.20	634.20

Table 3
Best results obtained compared with the literature.

Finally, in Table 3 we compare the best results from our experiments to the best results reported in the literature for the benchmark instances considered here [7]. We were able to improve on the previous results considerably.

4 Final Remarks

We presented a 0/1 IP formulation to tackle the office space allocation problem. In this model, constraints can be set as hard (actual constraints) or soft (objectives), giving flexibility to model different situations arising in real-world scenarios. We presented some results from our experiments on solving the model with CPLEX. So far, our results indicate that setting all constraint types as hard makes the problem unsolvable. Also, setting the satisfaction of the *Same room* constraint as objective seems to be particularly challenging. Future work contemplates modifying the IP model to improve computational time and memory consumption, developing a general IP model, and developing hybrids between the IP model and heuristics.

References

[1] Benjamin, C., I. Ehie and Y. Omurtag, *Planning facilities at the university of missouri-rolla*, Interfaces **22** (1992), pp. 94–105.

[2] Burke, E. K., P. Cowling and J. D. Landa Silva, *Hybrid population-based metaheuristic approaches for the space allocation problem*, in: *Proceedings of the 2001 Congress on Evolutionary Computation* (2001), pp. 232–239.

[3] Burke, E. K., P. Cowling, J. D. Landa Silva and B. McCollum, *Three methods to automate the space allocation process in uk universities*, in: *The Practice and Theory of Automated Timetabling III, LNCS 2079* (2001), pp. 254–273.

[4] Burke, E. K., J. D. Landa Silva and E. Soubeiga, *Multi-objective hyper-heuristic approaches for space allocation and timetabling*, in: T. Ibaraki, K. Nonobe and M. Yagiura, editors, *Meta-heuristics: Progress as Real Problem Solvers, Selected Papers from the 5th Metaheuristics International Conference* (2005), pp. 129–158.

[5] Giannikos, J., E. El-Darzi and P. Lees, *An integer goal programming model to allocate offices to staff in an academic instituition*, Journal of the Operational Research Society **46** (1995), pp. 713–720.

[6] Landa-Silva, D., "Metaheuristics and Multiobjective Approaches for Space Allocation," Ph.D. thesis, School of Computer Science and Information technology, University of Nottingham (2003).

[7] Landa-Silva, D. and E. Burke, *Asynchronous cooperative local search for the office-space-allocation problem*, INFORMS Journal on Computing **19** (2007), pp. 575–587.

[8] Ritzman, L., J. Bradford and R. Jacobs, *A multiple objective approach to space planning for academic facilities*, Managament Science **25** (1980), pp. 895–906.