Finding an optimal seating arrangement for employees traveling to an event

Ninoslav Čerkez[1], Rebeka Čorić[2], Mateja Đumić[2,*] and Domagoj Matijević[2]

[1] *IN2 d.o.o.*
Marohnićeva 1/1, 10000 Zagreb, Croatia
E-mail: ⟨ninoslav.cerkez@in2.hr⟩

[2] *Department of Mathematics, Josip Juraj Strossmayer University of Osijek*
Trg Lj. Gaja 6, 31000 Osijek, Croatia
E-mail: ⟨{rcoric, mdjumic, domagoj}@mathos.hr⟩

Abstract. The paper deals with modelling a specific problem called the Optimal Seating Arrangement (OSA) as an Integer Linear Program and demonstrated that the problem can be efficiently solved by combining branch-and-bound and cutting plane methods. OSA refers to a specific scenario that could possibly happen in a corporative environment, i.e. when a company endeavors to minimize travel costs when employees travel to an organized event. Each employee is free to choose the time to travel to and from an event and it depends on personal reasons. The paper differentiates between using different travel possibilities in the OSA problem, such as using company assigned or a company owned vehicles, private vehicles or using public transport, if needed. Also, a user-friendly web application was made and is available to the public for testing purposes.

Key words: Integer Linear Program, branch and bound, cutting plane method

Received: October 4, 2014; accepted: October 9, 2015; available online: December 31, 2015

DOI: 10.17535/crorr.2015.0032

1. Introduction

The following scenario is considered: A company with a large number of employees wants to organize a visit by their employees to a certain event that lasts for a predefined number of days. The company is assumed to own a certain number of vehicles that can be used for this specific purpose. Company vehicles differ in that they could already have been assigned to particular employees (assigned vehicles), or they are available for all employees (non-assigned vehicles). We assume that each vehicle has a certain capacity. Each employee is free to choose the time for traveling to and from an event depending on personal reasons. Employees may use their personal vehicles, if no company vehicles are available at the desired time, or they may use the public transport such as trains, buses or an airplane, if no

*Corresponding author.

vehicle options are available. We will also assume that employees will 'prefer' to use either assigned or non-assigned company vehicles as opposed to private vehicles, and 'prefer' private vehicles as opposed to public transport. In addition to this, an assigned vehicle can only be used by the employee to whom the vehicle is assigned. The very same condition will be enforced for personal vehicles.

The task is to assign each employee a certain vehicle to and from an event (this need not be the same vehicle), while satisfying the above set of constraints, such that the overall cost of assigning a vehicle is minimal (cheapest) in terms of the overall number of vehicles needed. We refer to this as the Optimal Seating Arrangement problem.

1.1. Previous work

Note that there is a certain similarity between this problem and the classical combinatorial NP-hard problem such as the Set Cover problem (see [13], [2], [8] for more details) and its variants. In the Set Cover problem, a person is given a set of elements called the universe, and a set of sets where the union is equivalent to the universe. In our scenario, the elements can be interpreted as employees, whereas vehicles and public transport as sets. Though the Set Cover problem involves finding a set cover of a universe that uses the fewest number of sets, in OSA we try to'cover'all employees with the fewest number of vehicles. The set cover problem with hard capacities (see [1])assumes that a set can only cover a limited number of its elements and the number of copies of each set is bounded. This generalization of the classical Set Cover enables better modelling of vehicles as sets, as it would interpreting vehicle capacity as a set capacity. However, additional constraints of the OSA problem, such as each employee selecting the time due to personal reasons, makes this problem significantly different from the Set Cover problem, even in the presence of hard capacities.

1.2. Our contribution

Based on our knowledge, IN2, a Croatian software company, was the first to emphasize the importance of this practical problem. We will formulate the problem as a non-trivial integer linear program ([10], [9]), while managing to avoid the use of vehicle-employee assignment variables, which dramatically reduced the dimension of the problem. As a result, we will demonstrate that the problem can be efficiently solved in practice on different testbeds with the use of standard methods, such as the cutting plane [12] and branch and bound methods [11], despite the fact that, theoretically, it is unclear whether the problem in general is polynomially solvable. For the empirical test, we used a state-of-the-art mathematical programming solver GUROBI [7] and output the CPU times. We also tested our algorithm on real-world instances provided to us by IN2.

To make our algorithm widely usable, we developed a user-friendly web application[‡], where it was possible to test the behavior of our model based on different inputs. After logging on, the application allowed the entry of new companies and

[‡]Application is available on vlab.mathos.hr/∼sjelic/projekt2

events, including employees and vehicles into a database, and the ability to assign them to a specified company. After linking a company to an event, the number of vehicles necessary to transport all employees to an event could be calculated. PHP script invokes an external C++ file used to implement our model and subsequently Gurobi was used to solve the problem.

Our paper consists of two main sections. In Section 2, referring to the integer linear programming model, we provide a formal definition of the Optimal Seating Arrangement problem. We also explain what the objective function and the set of linear constraints look like. In Section 3, referring to empirical evaluations, we conducted different numerical experiments on our algorithm. We completed our work in Section 4, referring to our conclusions, where we discuss unresolved issues relating to further research.

2. Integer linear programming model

Let m denote the number of employees, of which the first l are employees to whom company vehicles have been assigned, and the next q are employees willing to use their own vehicles, if necessary. We will assume that the employees are given T different time slots they can choose for travelling to and from an event. Furthermore, we will assume that there are n vehicles, of which the first l vehicles are assigned, the next q vehicles are personal, and $h = n - l - q$ denotes the number of non-assigned company vehicles. We assume that any one employee has at most one company vehicle assigned to them, and that the jth vehicle, $j \leq l$, is assigned to the jth employee. On the other hand, we also assume that the jth personal vehicle, $l + 1 \leq j \leq l + q$, corresponds to the jth employee. The capacity of each vehicle is denoted by $c_j > 0$, for all $j \in \{1, \ldots, n\}$. We can also assume that T vehicles, each with an infinite capacity, are assigned to each time-slot, representing the possibility that public transport was used, in case no other alternatives were available. As part of the input, we define z_{it}, for $i \in \{1, \ldots, m\}, t \in \{1, \ldots, T\}$, to be 1 or 0, depending whether the ith employee goes to an event at time t or not. Similarly, we define z'_{it}, depending on whether the ith employee comes back from an event at time t or not. Note that the non-zero values of z_{it} and z'_{it}, for $1 \leq i \leq l + q$, imply a time constraint as to when the assigned or personal vehicle could actually be used to and from an event, conforming to the requirement that the assigned or personal vehicle is used only by the employee that is assigned or owns the vehicle. For example, $z_{1,3}$ implies that Person 1 chooses to travel at Time 3. If a company vehicle has been assigned to this person, the vehicle may only be used at Time 3. Our model will adhere to this requirement, as will later described.

The set of 0/1 variables that we will use for our model are defined as follows:

- variables v_{jt}, for $j \in \{1, \ldots, n\}$, $t \in \{1, \ldots, T\}$, indicate whether the jth vehicle will be used at time t to an event;
- variables v'_{jt}, for $j \in \{1, \ldots, n\}$, $t \in \{1, \ldots, T\}$ indicate whether the jth vehicle will be used at time t from an event;
- variables a_i, $i \in \{1, \ldots n\}$, denote whether the ith employee will go to an event using public transport;

- variables a'_i, $i \in \{1, \dots m\}$, denote whether the ith employee will return back from an event using public transport;

We model the problem as the following Integer Linear Program:

$$\alpha \cdot \sum_{j=1}^{l} \sum_{t=1}^{T} v_{jt} + \beta \cdot \sum_{j=l+1}^{l+q} \sum_{t=1}^{T} v_{jt} + \gamma \cdot \sum_{j=l+q+1}^{n} \sum_{t=1}^{T} v_{jt} + \delta \cdot \sum_{i=1}^{m} (a_i + a'_i) \to \min \quad (1)$$

subject to:

$$\sum_{t=1}^{T} v_{jt} \leq 1, \qquad j \in \{1, \dots, n\} \quad (2)$$

$$\sum_{t=1}^{T} v'_{jt} \leq 1, \qquad j \in \{1, \dots, n\} \quad (3)$$

$$v_{jt} - \sum_{k=t+1}^{T} v'_{jk} \leq 0, \qquad j \in \{1, \dots, n\}, t \in \{1, \dots, T-1\} \quad (4)$$

$$\sum_{t=1}^{T} (1 - z_{jt}) \cdot v_{jt} = 0, \qquad j \in \{1, \dots, l+q\} \quad (5)$$

$$\sum_{t=1}^{T} z_{jt} \cdot v_{jt} \leq \sum_{t=1}^{T} z'_{jt} \cdot v'_{jt}, \qquad j \in \{1, \dots, l+q\} \quad (6)$$

$$\sum_{j=1}^{n} v_{jt} \cdot c_j + \sum_{i=1}^{m} z_{it} \cdot a_i \geq \sum_{i=1}^{m} z_{it}, \qquad \sum_{i=1}^{m} z_{it} \neq 0, \;\; t \in \{1, \dots, T\} \quad (7)$$

$$\sum_{j=1}^{n} v_{jk} \cdot c_j + \sum_{i=1}^{m} z_{it} \cdot a_i = 0, \qquad \sum_{i=1}^{m} z_{it} = 0, \;\; t \in \{1, \dots, T\} \quad (8)$$

$$\sum_{j=1}^{n} v'_{jk} \cdot c_j + \sum_{i=1}^{m} z'_{it} \cdot a'_i \geq \sum_{i=1}^{m} z'_{it}, \qquad \sum_{i=1}^{m} z'_{it} \neq 0, \;\; t \in \{1, \dots, T\} \quad (9)$$

$$\sum_{j=1}^{n} v'_{jk} \cdot c_j + \sum_{i=1}^{m} z'_{it} \cdot a'_i = 0, \qquad \sum_{i=1}^{m} z'_{it} = 0, \;\; k \in \{1, \dots, T\} \quad (10)$$

$$v_{j,t} \in \{0,1\} \qquad j \in \{1, \dots, n\}, t \in \{1, \dots, T\}$$

$$v'_{j,t} \in \{0,1\} \qquad j \in \{1, \dots, n\}, t \in \{1, \dots, T\}$$

$$a_i \in \{0,1\} \qquad i \in \{1, \dots, m\}$$

$$a'_i \in \{0,1\} \qquad i \in \{1, \dots, m\}$$

The objective function (1) is composed of four different sums, and each is multiplied by a certain positive value that we denoted as α, β, γ and δ. The first sum iterates over the assigned vehicles, the second sums over the personal vehicles while the third sums over the non-assigned ones. The last sum quantifies the number of employees that will travel using public transport. Note that by setting different values for parameters α, β, γ and δ, the objective function can be modelled depending on how much the person actually "pays" for the use of an assigned, personal or non-assigned vehicle, as well as for public transport. We will typically consider $\alpha = \gamma = 1$, i.e. the price for an assigned or non-assigned vehicle does not differ. Parameters β and δ will be assigned sufficiently large numbers to maintain the property condition that no personal vehicle or public transport is used if assigned or non-assigned vehicles are available. In our experiments (as we will later describe in more detail), we set $\beta = \delta = n$. Note that for equal values of β and δ, a person will never use public transport, if a personal vehicle can be used, unless the capacity of the personal vehicle is equal to one.

Constraints (2) and (3) ensure that each vehicle can be used no more than once, while constraint (4) ensures that the vehicle, if used for travelling to an event, has to also come back in one of following time-slots. For example, if $v_{jt_1} = 1$, i.e. the jth vehicle is used at time t_1, then equation (4) demands that v'_{jt} be set to one, but only for time $t > t_1$, since there is no sense in the vehicle returning before it actually headed off to an event. Constraints (5) and (6) ensure that the assigned or personal vehicle can only be used in specific time-slots, as selected by the employee that owns the vehicle or to whom the vehicle is assigned. For example, if $z_{i3} = 1$ and $z'_{i6} = 1$, i.e. the ith employee selects Time 3 for going to and Time 6 for returning from an event, equality (5) ensures that variable v_{i3} is the only one that can be set to 1. On the other hand, inequality (6) demands that for $v_{j3} = 1$, it must hold true that $v'_{j6} = 1$ (since $z'_{j6} = 1$). Constraints (7), (8), (9), (10) deal with capacities. Specifically, constraint (7) ensures that for each time-slot t, the sum of capacities of all vehicles that the ILP will decide to select in addition to the number of employees who end-up using public transport is at least equivalent to the overall number of employees traveling at time t. Constraint (8) prevents the vehicle from being assigned to time-slot t if no employee is actually traveling at time t. Constraints (9) and (10) ensure the same conditions are true when traveling from an event.

The above ILP will for each time-slot output the set of vehicles assigned to it or it will state what employees will be using public transport (variables a_i and a'_i). However, for some time-slot t, all employees that are not using public transport still have to get distributed among vehicles that got assigned to time t. In order to resolve that final issue we use the following straighforward approach for each time-slot t: Employees with assigned or personal vehicles, whose vehicles are assigned to t, will be distributed to their vehicles. The rest of employees will get packed to the rest of to the vehicles by the arbitrary choice.

3. Empirical evaluations

We conducted extensive experiments on different test data and used different parameters for our algorithms. All running times were measured on a quad-core Intel 2.80 MHz i5 CPU with 8GB of RAM.

3.1. Algorithm

There are two practical approaches for solving general integer linear programs: cutting plane methods and variants of the branch and bound method. The most notable state-of-the-art ILP solvers such as Gurobi [7] or CPLEX [6] implement and combine both approaches.

The Gurobi was used to solve the problem observed in this paper. The solvers in the Gurobi Optimizer use the most advanced implementations of the latest algorithms and exploit modern architectures and multi-core processors. Gurobi uses parallelism in solving problems and multiple CPU cores. Hence, we allowed Gurobi to use all four CPU cores of our test machine to solve our problem. We can divide the process in which Gurobi solves ILP problem into two phases: presolving and solving the ILP problem, which we obtain after presolving. Presolving (also known as preprocessing techniques) consists of a wide class of methods used for simplifying a given instance. Simplifying a given instance can significantly reduce the time needed for solving the problem. Some methods that are used for reducing the size of a given problem include removal of empty rows and columns (rows and columns with only zeros), detection and removal of rows dominated by a linear combination of other rows, and the like. Presolving results in a new reduced problem, and which is solved by Gurobi with a combination of branch-and-bound and cutting plane methods. To solve linear programming relaxations, which appear in these two methods, Gurobi uses the primal-dual simplex.

For test purposes, we used two types of inputs, real-world instances and generated instances. Our real-world instances were relatively small in size. Hence, to understand the behavior of our algorithm on larger instances, we generated large inputs randomly. Company IN2 d.o.o. provided the data for 2007, 2009 and 2011, which contained information on a number of IN2 employees attending the HROUG[§] event, the number of vehicles and distribution of people in the vehicles. The data also contained information on assigned vehicles, unassigned vehicles and personal vehicles. We entered the given data into our web application and compared the obtained results with the initial data. For HROUG 2007, the initial data stated that 17 employees attended the event and that they used 6 separate vehicles. Our algorithm computed that they could have organized the trip using only 5 vehicles. The data for HROUG 2009 stated that 23 employees attended the event and 8 vehicles were used. Our model returned the results that they could have organized the trip using only 6 vehicles whereas the data for HROUG 2011 stated that 33 employees attended the event and 11 vehicles were used. Our model returned the results that 8 vehicles were enough, i.e. they could have organized the trip using 8 separate vehicles.

[§]For more details about HROUG see www.hroug.hr

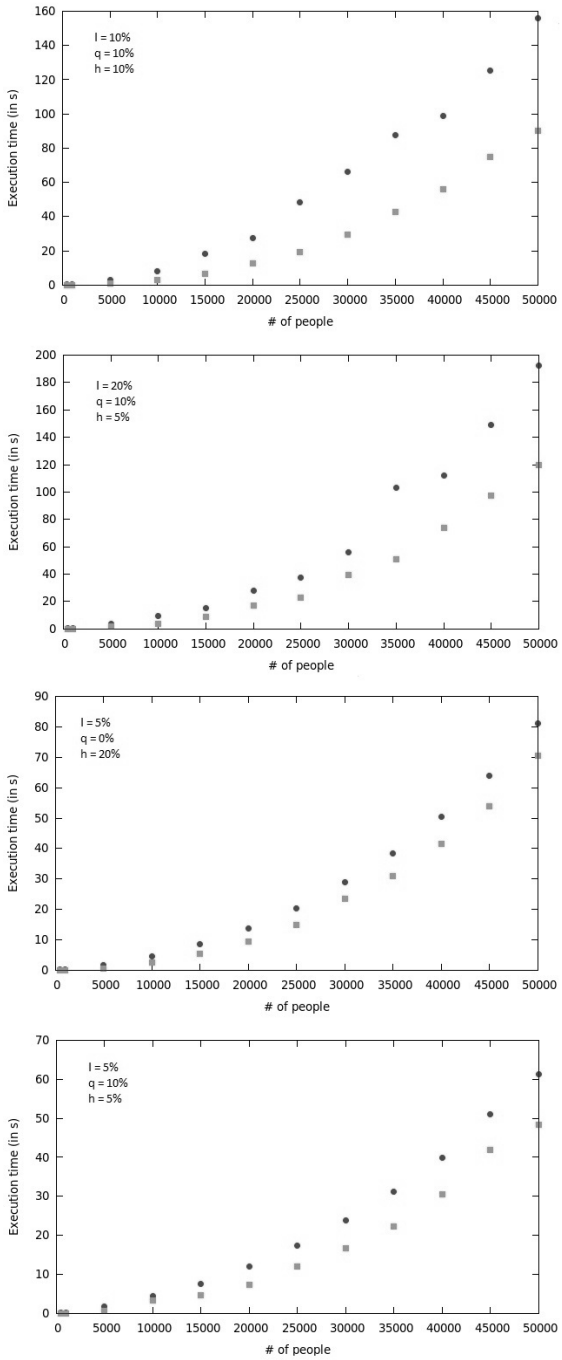

Figure 1: *Test results: the light gray squares denote the time needed for a presolve phase, while the dark gray dots denote the total execution time of our algorithm.*

	# of people											
	500	1000	5000	10000	15000	20000	25000	30000	35000	40000	45000	50000
Init # (rows)	2020	4020	20020	40020	60020	80020	100020	120020	140020	160020	180020	200020
Init # (cols)	4000	8000	40000	80000	120000	160000	200000	240000	280000	320000	360000	400000
Init # ($\neq 0$)	19350	38700	193500	387000	580500	774000	967500	1161000	1354500	1548000	1741500	1935000
# after presolve (rows)	468	918	4518	9018	13518	18018	22518	27018	31518	36018	40518	45018
# after presolve (cols)	1001	1962	9331	18363	27378	36378	45378	54378	63378	72378	81378	90378
# after presolve ($\neq 0$)	4534	9006	44144	87708	131238	174738	218238	261738	305238	348738	392238	435738
Rows removed	76,83%	77,16%	77,43%	77,47%	77,48%	77,48%	77,49%	77,49%	77,49%	77,49%	77,49%	77,49%
Columns removed	74,98%	75,48%	76,67%	77,05%	77,19%	77,26%	77,31%	77,34%	77,37%	77,38%	77,40%	77,41%
Nonzeros removed	76,57%	76,73%	77,19%	77,34%	77,39%	77,42%	77,44%	77,46%	77,46%	77,47%	77,48%	77,48%

Table 1: *The data in the table are given for l=10%, q = 10% and h=10%*

As we have already stated, large inputs were randomly generated. For that purpose we defined three new parameters, l, q and h, each one denoting the percentage of employees with assigned, personal, or non-assigned vehicles, respectively, with respect to the total number of employees, and generated values z_{it}, z'_{it} and c_j randomly. We took the precaution that z_{it_1} and z'_{it_2}, for some i, are generated such that $t_2 > t_1$, i.e. the ith employee cannot return before actually heading off to an event. Figure 1 shows the total execution time and presolve time needed for different number of employees (x-coordinate) for an event that lasts 5 days. Parameters l, q and h are specified in graphs, and the lower and the upper bound for capacities are selected as 2 and 5, respectively.

As one can observe from Figure 1, the computation times are relatively fast and Gurobi solves our largest instance in less than 200 seconds. Table 1 shows the number of variables as well as the number of constraints generated for a given number of people. We also reported the number of non-zero values in the system matrix in order to show the sparsity of the input matrix. For example, in the first column of Table 1, there are 2020 rows and 4000 columns, while the number of non-zero entries is 19350, i.e. only 0.2%. Similar behavior for other input sizes is noticeable. Hence, not surprisingly Gurobi's presolve phase reduces the initial input on average from 70% to 80%. This is most likely the main reason why the computation even on large instances is completed so quickly.

4. Conclusion

This paper suggests modelling the optimal seating arrangement problem as an Integer Linear Program. The authors in this paper based on their previous work in linear optimization (see [3], [4], [5]) have developed an efficient mathematical model and conducted the necessary experiments. The approach in this paper has been to combine branch-and-bound and cutting planes methods in order to solve a given ILP. The empirical evaluations that were conducted showed that it is possible to computationally handle even large instances of the problem using a combination of standard branch-and-bound and cutting planes methods. Furthermore, a user-friendly web application that uses the Integer Programming solver, and run C++ as a subroutine, was implemented.

It still remains unknown whether the problem is NP-hard, which would then justify ILP formulation of the problem. The authors have left this as an open problem and something that they would like to work on in the future.

References

[1] Chuzhoy, J. and Naor, J. (2006.) Covering problems with hard capacities. SIAM Journal on Computing, 36(2), 498–515. doi:10.1137/s0097539703422479.

[2] Chvatal, V. (1979). A greedy heuristic for the set-covering problem. Mathematics of Operations Research, 4(3), 233–235. doi:10.1287/moor.4.3.233.

[3] Eisenbrand, F., Funke, S., Karrenbauer, A. and Matijević, D. (2008). Energy-aware stage illumination. International Journal of Computational Geometry & Applications, 18(1-2), 107–129. doi:10.1142/s0218195908002556.

[4] Elbassioni, K., Matijević, D. and Ševerdija, D. (2012). Guarding 1.5D terrains with demands. International Journal of Computer Mathematics, 89(16), 2143–2151.

[5] Elbassioni, K., Krohn, E., Matijević, D., Mestre, J. and Ševerdija, D. (2011). Improved approximations for guarding 1.5-dimensional terrains. Algorithmica, 60(2), 451–463. doi:10.1007/s00453-009-9358-4.

[6] IBM ILOG CPLEX (2011). CPLEX Users Manual. Version 12, Release 4.

[7] Gurobi Optimization, Inc. (2013). Gurobi Optimizer Reference Manual.

[8] Johnson, D. S. (1974). Approximation algorithms for combinatorial problems. Journal of Computer and System Sciences, 9(3), 256–278.

[9] Schrijver, A. (1998). Theory of Linear and Integer Programming. John Wiley and Sons.

[10] Nemhauser, G. L. and Wolsey, L. A. (1988). Integer and Combinatorial Optimization. Wiley.

[11] Land, A. H. and Doig, A. G. (1960). An automatic method of solving discrete programming problems. Econometrica, 28(3), 497–520.

[12] Marchanda, H., Martinb, A., Weismantelc, R. and Wolseyd, L. (2002). Cutting planes in integer and mixed integer programming, 123(1-3), 397–446.

[13] Vazirani, V. V. (2001). Approximation Algorithms. Springer-Verlag.