

Finding an optimal seating arrangement for employees traveling to an event

Ninoslav Čerkez¹, Rebeka Čorić, Mateja Đumić and Domagoj Matijević²

¹ IN2 d.o.o.

Marohnićeva 1/1, 10000 Zagreb, Croatia

E-mail: {ninoslav.cerkez@in2.hr}

² Department of Mathematics, University of Osijek

Trg Ljudevita Gaja 6, 31000 Osijek, Croatia

E-mail: {rcoric/mdjumić/domagoj@mathos.hr}

Abstract. The paper deals with modelling a specific problem called the Optimal Seating Arrangement (OSA) as an Integer Linear Program and it was demonstrated that this problem can be efficiently solved by combining branch-and-bound and cutting plane methods. OSA is capturing a specific scenario that could happen in a corporative environment when company is trying to minimize travel costs for organizing the traveling of its employees to an event. Each employee is free to choose the time for traveling to and back from an event, depending on her or his selfish reasons. The paper differentiates between using different traveling possibilities in OSA problem, such as the possibility of using a company assigned or a company owned vehicles, private vehicles as well as the possibility of using the public transportation if needed. Also, there was user - friendly web - application implemented that can be publicly used for testing purposes.

Key words: Integer linear program, branch and bound, cutting plane method

Received: xx xx, 201x; accepted: yy yy, 201x; available online: zz zz, 201x

1. Introduction

The following scenario is considered: A company with large number of employees wants to organize a visit of their employees to a certain event that lasts for predefined number of days. Company is assumed to own certain number of vehicles that can be used for that purpose. Company vehicles differ in a sense that they could already be assigned to certain employees (assigned vehicles), or they are available for all employees (non-assigned vehicles). We assume that each vehicle could have its own capacity. Each employee is free to choose the time for traveling to, and time for traveling back from an event depending on her/his selfish 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 train, bus or an airplane, if no vehicle options are available. We will also assume that employees will 'prefer' to use either assigned or non-assigned company vehicles over the private ones, and employees will 'prefer' private vehicles over the public transport. In addition to that, the assigned

vehicle can only be used with the employee this vehicle is assigned to. Very same condition will be enforced to personal vehicles.

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

1.1. Previous work

Note that there is certain similarity of our problem with the classical combinatorial NP-hard problems such as the Set Cover problem (see [13], [2], [8] for more details) and its variants. In the Set Cover problem one is given a set of elements called universe, and a set of sets whose union equals the universe. In our scenario elements could be interpreted as employees and vehicles or public transport as sets. While in Set Cover the problem is to find a set cover of a universe that uses the fewest number of sets, in OSA we are trying to ‘cover’ all the employees with 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 that the number of copies of each set is bounded. This generalization of classical Set Cover would allow for better modelling of vehicles as sets since it would allow to interpret vehicle capacity as a set capacity. However, an additional constraints of the OSA problem, such as that each employee can selfishly select the time, make this problem significantly different from the Set Cover problem even in the presence of hard capacities.

1.2. Our contribution

By our knowledge, Croatian software company IN2 was the first one to raise importance of this practical problem. We formulate the problem as a non-trivial integer linear program ([10], [9]), managing to avoid the use of vehicle-employee assignment variables, which dramatically reduced the dimension of the problem. As a result, we demonstrate that the problem can be efficiently solved in practice on different testbeds with the use of standard methods, such as cutting plane [12] and branch and bound methods [11], despite the fact that theoretically it is not clear that the problem in general is polynomially solvable. For empirical test, we used the state-of-the-art mathematical programming solver GUROBI [7] and output the CPU times. We also tested our algorithm on a real-world instances provided to us by IN2 d.o.o.

In order to make our algorithm widely usable we made user-friendly web application*, where one can test how our model behaves on different inputs. After login, application allows you to insert new companies and events, as well as employees and vehicles into the database and to assign them to specified company. After connecting a company to an event you can calculate how many vehicles it is necessary to transport all employees to an event. PHP script calls external C++ file in which our model is implemented and Gurobi is called.

*Application is available on vlab.mathos.hr/~sjelic/projekt2

Our paper consists of two main sections. In Section 2, Integer linear programming model, we give a formal definition of the Optimal Seating Arrangement problem. We also explain how the objective function as well as the set of linear constraints look like. In Section 3, Empirical evaluations, we conducted different numerical experiments of our algorithm.

We close our work with Section 4, Conclusion, where we discuss open problem for our further research.

2. Integer linear programming model

Let m denote the number of employees, out of which first l are employees with assigned company vehicles and next q are employees that are willing to use their personal vehicles if necessary. We will assume that employees are given T different time slots they can choose from for departure to and arrival from an event. Furthermore, we will assume that there are n vehicles, out of which first l vehicles are assigned ones, next q vehicles are personal, and $h = n - l - q$ denotes the number of non-assigned company vehicles. We assume that single employee has at most one company vehicle assigned, and that the j th vehicle, $j \leq l$, is assigned to j th employee. On the other hand, we also assume that j th personal vehicle, $l+1 \leq j \leq l+q$, corresponds to j th employee. The capacity of each vehicle is denoted by $c_j > 0$, for all $j \in \{1, \dots, n\}$. We can also assume that T vehicles, each with infinite capacity, are assigned to each time-slot, representing the possibility that the public transport was used in case no other alternatives were left. As part of input, we define z_{it} , for $i \in \{1, \dots, m\}, t \in \{1, \dots, T\}$, to be 1 or 0, depending whether i th employee goes to an event in time t or not. Similarly, we define z'_{it} , depending whether i th employee comes back from an event in time t or not. Note that the non-zero values of z_{it} and z'_{it} , for $1 \leq i \leq l+q$, immediately imply the restriction on the time when the assigned or personal vehicle could be actually used on the way to and back, in order to maintain the requirement that the assigned or personal vehicle will be used only by the employee this vehicle is assigned to or owned by, respectively. For example, $z_{1,3}$ will imply that the person 1 chooses to travel in time 3. If he has company vehicle assigned, his vehicle may only be used in time 3. Our model will take care of that requirement, as we will later on describe.

The set of 0/1 variables that we will use for our model are defined in the following:

- variables v_{jt} , for $j \in \{1, \dots, n\}, t \in \{1, \dots, T\}$, indicate whether j th vehicle will be used in time t on the way to an event;
- variables v'_{jt} , for $j \in \{1, \dots, n\}, t \in \{1, \dots, T\}$ indicate whether j th vehicle will be used in time t on the way back from an event;
- variables a_i , $i \in \{1, \dots, n\}$, denote whether the i th employee will travel to an event with the public transport;
- variables a'_i , $i \in \{1, \dots, m\}$, denote whether the i th employee will travel back from an event with the public transport;

We model the problem as the following Integer Linear Program:

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

subject to:

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

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

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

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

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

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

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

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

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

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

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

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

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

The objective function (1) is composed out of four different sums, and each one is multiplied with 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 sum sums over the non-assigned ones. The last sum is quantifying the number of employees that will travel with the public transport. Note that by setting different values for parameters α, β, γ and δ , one can model in the objective function how much one actually "pays" for the use of assigned, personal or non-assigned

vehicle, as well as for public transport. We will typically consider $\alpha = \gamma = 1$, i.e. the price for assigned or non-assigned vehicle does not differ. For parameters β and δ we will use some sufficiently large numbers in order to maintain the property that no personal vehicle or public transport is used if there are assigned or non-assigned vehicles available. In our experiments (as we will later on describe in more details) we set $\beta = \delta = n$. Note that equal values for β and δ , one will never use the public transport, if personal vehicle can be used, unless personal vehicle has capacity one.

Constraints (2) and (3) ensure that each vehicle can only be used at most once, while the constraint (4) ensures that the vehicle, if used for transporting to an event, has to also return back in one of following time-slots. For example, if $v_{jt_1} = 1$, i.e. some j th vehicle is used in time t_1 , then the equation (4) demands from v'_{jt} to be set to one, but only for some $t > t_1$, since it doesn't make sense that the vehicle returns before it actually went to an event. The constraints (5) and (6) ensure that the assigned or personal vehicle can only be used in specific time-slots selected by the employee that owns the vehicle or that the vehicle is assigned to. For example, if $z_{i3} = 1$ and $z'_{i6} = 1$, i.e. i th employee selects time 3 for going to and time 6 for returning back, equality (5) ensures that the variable v_{i3} is the only one that could be set to 1. On the other hand, inequality (6) demands that for $v_{i3} = 1$ it must be that $v'_{i6} = 1$ (since $z'_{i6} = 1$). Constraints (7), (8), (9), (10) deal with the capacities. Namely, constraint (7) ensures that for each time-slot t , the sum of capacities of all vehicles that the ILP will decide to select plus the number of employees who will end-up using the public transport is at least the overall number of employees traveling in time t . The constraint (8) prevents the vehicle to be assigned to time-slot t if no employee is actually traveling in time t . Constraints (9) and (10) ensure the same conditions hold when traveling back from an event.

The above ILP will for each time-slot output the set of vehiclees 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 vehiclees that got assigned to time t . In order to resolve that final issue we use the following straightforward approach for each time-slot t : Employees with assigned or personal vehiclees, whose vehiclees are assigned to t , will be distributed to their vehiclees. The rest of employees will get packed to the rest of the vehiclees by the arbitrary choice.

3. Empirical evaluations

We conducted extensive experiments on different test data and using 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. 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 uses the most advanced implementations of the latest algorithms and exploit modern architectures and multi-core processors. Gurobi uses parallelism while solving problems and can use multiple CPU cores. Hence, we allowed Gurobi to use all four CPU cores of our test machine in solving our problem.

We can divide the process in which Gurobi solves ILP problem in two phases: presolving and solving the ILP problem which we get after presolving. Presolving (also known as preprocessing techniques) consists of 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 which are used for reducing the size of a given problem are: removal of empty rows and columns (rows and columns with only zeros in it), detection and removal of rows dominated by linear combination of other rows, and similar. Presolving results with a new reduced problem, which Gurobi is solving by combining branch-and-bound and cutting plane methods. In order to solve linear programming relaxations which appear in this two methods Gurobi uses the primal-dual simplex.

For the test purposes we used two types of inputs, real-world instances and generated instances. Our real-world instances are relatively small in size. Hence, in order to understand the behaviour 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 of number of IN2 employees that were going to HROUG[†] event, number of vehicles and distribution of people in vehicles. The data also contained information about assigned vehicles, unassigned vehicles and personal vehicles. We entered the given data to our web application and got results which we then compared to the initial data. For HROUG 2007 initial data stated that 17 employees were going to an event and for that they used 6 different vehicles. Our algorithm computed that they could have organized the trip with only 5 vehicles. The data for HROUG 2009 stated that 23 employees were going to an event and 8 vehicles was used. Our model returned that they could have organized the trip with only 6 vehicles and the data for HROUG 2011 stated that 33 employees were going to an event and 11 vehicles was used. Our model returned that 8 vehicles are already enough, i.e. they could have organized the trip with no more than 8 different vehicles.

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

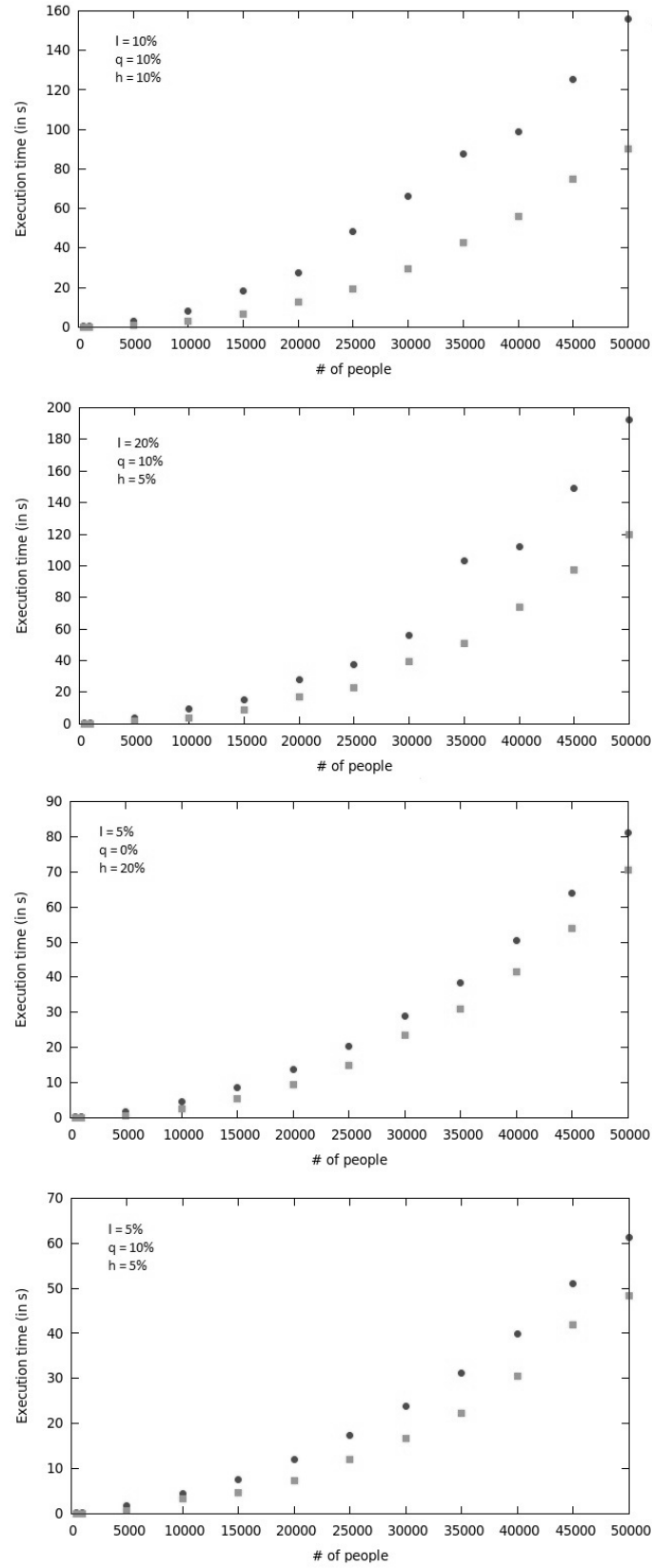


Figure 1: Test results: the light gray squares denote the time needed for a presolve phase, while the dark gray dots denote the total execution time of our algorithm.

	# 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 already stated, we generated large inputs randomly. 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 total number of employees, and generated values z_{it} , z'_{it} and c_j randomly. We took precaution that z_{it_1} and z'_{it_2} , for some i , are generated such that $t_2 > t_1$, i.e. i th employee cannot return before he actually went to an event. The 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 to be 2 and 5, respectively.

As one can observe from the Figure 1, the computation times are relatively fast and Gurobi solves our largest instance in less than 200 seconds. In Table 1 one can see the number of variables as well as the number of constraints that gets generated for a given number of people. We also report the number of non-zero values in the system matrix in order to show how sparse the input matrix is. For example, in first column of the Table 1 the number of rows is 2020 and columns is 4000, while the number of non-zero entries is 19350, i.e. only 0.2%. One can see very similar behaviour for other input sizes. Hence, it is not that of a surprise that already the Gurobi's presolve phase reduces the initial input on average between 70% and 80%. This is most likely the major reason why the computation even on large instances finishes 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]) developed the efficient mathematical model and conducted experiments. Approach used in this paper is combining branch-and-bound and cutting planes methods in order to solve given ILP. The empirical evaluations that were conducted showed that one can computationally deal even with large instances of the problem with the combination of standard branch-and-bound and cutting planes methods. Also, user-friendly web-application was implemented that uses the Integer Programming solver, implemented in C++ as a subroutine.

It is still unknown whether the problem is NP-hard, which would then justify the ILP formulation of the problem. The authors leave 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 J. Comput.*, 36(2), 498-515.
- [2] Chvatal, V. (1974). A Greedy Heuristic for the Set-Covering Problem, *Mathematics of Operations Research*, 4, 233-235.
- [3] Eisenbrand, F., Funke, S., Karrenbauer, A., Matijević, D. (2008). Energy-Aware Stage Illumination, *International Journal of Computational Geometry & Applications*, 18(1-2), 107-129.
- [4] Elbassioni, K., Matijević, D., 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., Ševerdija, D. (2011) Improved Approximations for Guarding 1.5-Dimensional Terrains, *Algorithmica*, 60(2), 451-463.
- [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). pp. 497-520.
- [12] Marchanda, H., Martinb, A., Weismantelc, R. and Wolseyd, L. (2002). Cutting planes in integer and mixed integer programming, 123(1-3), pp 397-446.
- [13] Vazirani, V. V. (2001). *Approximation Algorithms*, Springer-Verlag.