Communication Papers of the Federated Conference on Computer Science and Information Systems pp. 53–59 DOI: 10.15439/2018F230 ISSN 2300-5963 ACSIS, Vol. 17

# "Passeport Vacances": an assignment problem with cost balancing

Corentin Beffa, Sacha Varone
University of Applied Sciences Western Switzerland (HES-SO),
HEG Genève, Switzerland
Rue de la Tambourine 17, 1227 Carouge, Switzerland
Email: {corentin.beffa, sacha.varone}@hesge.ch

***Abstract*—Passeport Vacances is an offer for school-aged children to discover a set of activities during holidays. For more than 30 years, it has been an established social function in several countries, including Germany and Switzerland. Proposed activities might occur several times during the Passeport Vacances. The assignment of activities to children is computed in order to maximize the children's preferences, as well as to balance each child's incurred cost, toward an equity goal. There are several sets of constraints associated with the assignment problem: no overlapping activities assigned to the same child, minimal and maximal ages per activity, minimum number of children for opening an activity, maximal size of a group for each activity, no similar activities assigned to the same child, no already assigned 'lifetime'-activity per child, and at most one activity per period and per child. We propose a binary linear programming model that describes the assignment problem, report CPU computation issues regarding the model implementation, and report numerical results based on a state-of-the-art MIP solver. Tests where conducted with real data from the 2016 edition of Passeport Vacances in Morges.**

## I. Introduction

PASSEPORT Vacances is an offer for school-aged children to discover a set of activities during holidays. For more than 30 years, it has been an established social function in several countries, including Germany and Switzerland. Proposed activities might occur several times during the Passeport Vacances. Some weeks before the start of Passeport Vacances, children are asked to choose at most four activities per day and to give a ranking to the selected activities for each day, from 1 for the most attractive one, to 4 for the least attractive one. Each child receives a personal identificator and has to give his birthdate, as well as other useful information such as phone number, address, etc. The assignment of children to activities is computed in order to maximize the preferences specified by the children, as well as to balance the incurred cost by each child, toward an equity goal. There are several sets of constraints associated with the assignment problem: no overlapping activities assigned to the same child, minimal and maximal ages per activity, minimum number of children for opening an activity, maximal size of a group for each activity, no similar activities assigned to the same child, no already assigned 'lifetime'- activity per child, and at most one activity per period and per child. During a given horizon of time, generally between 5 and 14 days, Passeport Vacances offers a set of activities to children during holidays. This holiday time is divided into periods, most often days or half-days. Before PV actually takes place, children are asked to select a few activities per period, often 4 selections ranked by preference. After closing this selection phase, the assignment process can begin.

Many areas are interested in load balancing: for example, to balance the workloads of teaching assistants [1], students [2], or professors [3] or for socioeconomic variation between schools [4]. This balancing is often done by minimizing the deviation to the mean value or, as proposed by Domenech and Lusa [5] and De La Torre, Lusa and Mateo [3], combined to the maximum relative deviation. However, the different measures of deviation are not linear and need to be adapted. Ünal and Uysal [2] proposed a linearisation of the 4 norms $L_0$, $L_1$, $L_{\inf}$, and $L_{max}$ by adding variables. The balancing objective is often one of several goals of the problem. Different ways to combine multiple goals have been proposed in the literature. In the purpose of assigning students to projects, Pan, Chu, Han, Guangyue, and Huang [6] proposed a goal programming model. Another well-known method consists of making a mixed integer linear programming (MILP) model as a weighted sum of the different objectives. The weights can be adapted to privilege one or another goal, but many studies have focused on finding a Pareto optimal solution: a solution where all objectives could not be improved without deteriorating the others. A small literature review and a method to find the Pareto front was proposed by Kim and De Weck [7]. In order to propose a convenient and powerful model, we chose to use a weighted sum of the different goals with fixed weights.

This paper reports our mathematical model, implemented in the Julia language and solved with a Mixed Integer Programming solver. State-of-the-art solvers are able to pre-compute the MIP model before launching the Simplex algorithm, so that redundant constraints are dropped, and also, as a consequence, some variables are fixed. Although this generally saves quite a lot of computation time, it is sometimes recommended to avoid part of this step by changing a straightforward implementation with a more data-focused model. The goal is to reduce the time needed to build the model, which could seriously increase the total CPU computing time. We explain in this report our verification and the way we handled the model construction. We then report experimental results and give some insight on the resulting computing time.

TABLE I
ACTIVITY

| | |
|---|---|
| idactivity | unique identificator |
| nboccurrence | number of occurrences during the considered horizon |
| pricefixed | fixed price |
| pricechild | price per child |
| life | binary indicator about the status of 'lifetime' |
| minchild | minimum number of children per occurrence to open the activity |
| maxchild | maximal number of children per occurrence |
| minage | minimal age to perform the activity |
| maxage | maximal age to perform the activity |
| similarity | identificator of similar activities |

TABLE II
OCCURRENCE

| | |
|---|---|
| idoccurrence | unique identificator |
| idactivity | associated activity |
| occurrencebegin | begin date and time of the occurrence |
| occurrenceend | end date and time of the occurrence |
| inactive | binary indicator variable |
| next | next occurrence in case of multi-occurrence activity |
| previous | previous occurrence in case of multi-occurence activity |

TABLE III
CHILD

| | |
|---|---|
| idchild | unique identificator |
| birthdate | birthdate |

## II. PROBLEM DESCRIPTION

### A. Data

Data was stored in a PostgreSQL database. It is therefore presented as a set of tables with their fields.

*Table I* specifies the activities proposed during the considered PV horizon.

- Field 'fixedprice' is the fixed cost that PV has to pay for each occurrence of the activity that occurs, whereas 'pricechild' is the variable cost per child that PV has to pay. Those costs can be considered in the following way: a balanced cost between children of the same age category is appreciated, so that a fair assignment of children to activities can be done.
- The 'life' field indicates if an activity has to be considered as a lifetime activity, which means that if a child has already been assigned to that activity in some past PV, he can no longer be assigned to this activity. Its value is TRUE for a lifetime activity, and FALSE otherwise.
- Field 'minchild' indicates the minimum number of children to open an occurrence of an activity. In other words, if there are not enough children for a specific occurrence, then this occurrence is cancelled. Field 'maxchild' restricts the size of the group for an occurrence of the activity.
- Fields "minage" and "maxage" are respectively limitations on the minimal age and maximal age for doing an activity.
- Field 'similarity' indicates similar activities. For example, activity A = visit to the zoo Alpha and activity B = visit to the zoo Beta are considered similar and therefore belong to the same similarity group. The goal of this field is to avoid similar activities being assigned to a same child. Modalities of the 'similarity' field are natural numbers.

*Table II* contains the occurrences of the activities. The fields 'next' and 'previous' get the value of 'idoccurrence' for activities that only need one occurrence to be done. For activities requiring consecutive occurrences, like a several days internship within a company, the 'next' field refers to the next occurrence, unless it is the last one; in this case, it contains the same value as 'idoccurrence'. The 'previous' field is similarly defined. Cancelled occurrences are referred to with the 'inactive' field, whose value is TRUE for inactive occurrences and FALSE otherwise.

*Table III* contains the characteristics of children. Note that all personal information, such as first name, family name, phone number, etc., were not provided since they are useless for the optimization process. The day of each birth date has been modified to the first day of the month in order to anonymize the data and their IDs have been changed.

*Table IV* represents k-tuples of children, indicating groups of children willing to participate together, for each of the assigned activities. This is generally useful in case of children belonging to a same family, or friends willing to share activities.

Note that this not only refers to binomes, but also $k$-nomes (i.e. several children with the same set of assigned occurrences) can be constituted.

*Table V* indicates ranked preferences given by the children. Each child ranks at most k (k=4 in this data-set) activities per day of the PV, 1 being the preferred activity, up to k being the kth preferred activity.

*Table VI* expresses already-assigned 'lifetime' activities to children. This is a history of past PV assignments. A child

TABLE IV
KNOME

| | |
|---|---|
| idchild1 | first child |
| idchild2 | second child |

TABLE V
PREFERENCE

| | |
|---|---|
| idpreference | unique identificator |
| idchild | child identificator |
| idoccurrence | occurrence identificator |
| choice | choice rank of the occurrence |

TABLE VI
LIFETIME

| | |
|---|---|
| idchild | child identificator |
| idactivity | activity identificator |
| idlifetime | unique identificator |
| idchild | child identificator |
| idactivity | activity identificator |
| idpreference | preference identificator |

TABLE VII
PERIOD

| | |
|---|---|
| idperiod | unique identificator |
| periodbegin | beginning of the period |
| periodend | end of the period |
| maxassigned | maximum number of occurrences for this period |

who already received a specific lifetime activity in the past PV cannot again obtain this activity.

*Table VII* represents periods on which restrictions might apply. This means that during a specific period of time 'idperiod', beginning at 'periodbegin' and ending at 'periodend', a maximum of 'maxassigned' occurrences can be assigned to the same child.

## B. Data preparation

In order to facilitate the model generation, we modified raw data by adding computed tables, adding some columns to existing tables, or reducing unnecessary information.

Passeport Vacances partitions the time horizon into periods, which are most often days but sometimes half-days. Children are then asked to fill a formula to express their preferences about activities. They must do this for each period.

We computed the incompatibility graph of preferences, which represents overlapping occurrences that appear in the selected activities by a same child.

Based on these inequalities, we created a new table called *incompatible* that contains all incompatible preferences, one for each edge of the incompatibility graph.

TABLE VIII
INCOMPATIBLE

| | |
|---|---|
| idincompatible | unique identificator |
| idpref1 | first preference identificator |
| idpref2 | second preference identificator |

## III. MATHEMATICAL MODEL

We use the notation proposed in Table IX.

There is a set O of occurrences, because an activity can be organised several times during the whole Passeport Vacances period. Therefore, several occurrences might correspond to a same activity. A preference $p \in P$ refers to a child's choice for an occurrence. The sets A, T, C, I, L, $MA_t$, and $C_p$ are self explanatory. The cost for the organizers is composed of a fixed cost that remains the same regardless of the number of children and a price per child. Set $PC_p$ represents the fixed cost, whereas $PF_p$ represents the variable cost. Each activity is part of a group of similar activities with the same reference. Similar activities are given a same label. This set of labels is noted S.

TABLE IX
NOTATIONS

| | |
|---|---|
| $P$ | Set of preferences |
| $A$ | Set of activities |
| $O$ | Set of occurrences |
| $T$ | Set of periods |
| $C$ | Set of child |
| $C_p$ | Choice's value for the preference p |
| $I$ | Set of inactive preferences |
| $L$ | Set of forbidden preferences |
| $PC_p$ | Price per child for each preference |
| $PF_p$ | Fixed price by activity |
| $MA_t$ | Maximum number of assignations during period t |
| $S$ | Set of similarity labels |

We modelled this problem as an integer problem, in which the objective function and the constraints are linearly defined.

### A. Variables

A solution to the considered problem is a set of pairwise associations between a child and an occurrence. We considered each of these pairwise associations to get either a true value if the child is assigned to the occurrence, and false if not. We therefore defined a set of variables that describes valid assignment of an occurrence and a child:

$$x_i = \begin{cases} 1 & \text{if preference } i \text{ is selected} \\ 0 & \text{otherwise} \end{cases}$$

where $i \in P$.
A variable is associated to each element of the table *preference*. Therefore, $x_i$ true means that the associated occurrence of an activity is assigned to the associated child. In other words, this is equivalent to defining the following two-indexes variables, but reduces the number of variables, as a preference exists only if a child chose an occurrence.

$$x_{ij} = \begin{cases} 1 & \text{if child } i \text{ gets occurrence } j \\ 0 & \text{otherwise} \end{cases}$$

where $i \in C, j \in O$. For example, for a child choosing 4 occurrences between 10 possible occurrences, it will need 10 variables for the two-indexes variables, while only 4 variables are needed for the single-indexed model.
We also defined auxiliary variables that indicate if an occurrence is open or not. This was necessary since some occurrences can only be open if there is a minimum number of children to do the activity.

$$y_j = \begin{cases} 1 & \text{if occurrence } j \text{ is open} \\ 0 & \text{otherwise} \end{cases}$$

where $j \in O$.

## B. Constraints

We considered all constraints as hard constraints.

1) Forbid assignment to cancelled occurrences.
This constraint is formalised with the assignment to a null value for each preference set to a cancelled occurrence. For efficiency reasons, it is advised to sum all such cases to zero.

$$\sum_{p \in I} x_p = 0 \tag{1}$$

2) Do not assign two incompatible occurrences to the same child.

$$x_a + x_b \leq 1 \quad \forall (a,b) \in \mathit{Incompatible} \tag{2}$$

3) Maximal number of activities per child and per period.
This constraint avoids assigning too many activities during the same period to a child, even if their occurrences do not overlap.
Let's define, for each child $c$ and each period $t$, the subset $\mathit{CP}_{c,t}$ of preferences for which the occurrences begin during period $t$. This subset $\mathit{CP}_{c,t}$ contains the preferences expressed by child $c$ for activities occuring during period $t$.

$$\sum_{p \in \mathit{CP}_{ct}} x_p \leq \mathit{MA}_t \tag{3}$$

$$\forall (c,t) : c \in C, t \in T$$

4) At most 1 activity from a set of similar activities.
Let's define for each child $c$ and each similar value $s$ the subset $\mathit{SP}_{c,s}$ of preferences with the same similarity label $s$ for child $c$.

$$\sum_{p \in \mathit{SP}_{c,s}} x_p \leq 1 \tag{4}$$

$$\forall (c,s) : c \in C, s \in S$$

5) Forbid reassignement to previously assigned 'lifetime' activities.
The constraint consists in assigning a null value to each such preference.

$$\sum_{p \in L} x_p = 0 \tag{5}$$

6) Minimal age to perform an activity.
Let's define the set $\mathit{BA} \subset P$ of preferences indexes for which the child's age is below the minimum required age. We therefore forbid such assignments.

$$\sum_{p \in \mathit{BA}} x_p = 0 \tag{6}$$

7) Maximal age to perform an activity.
Let's define the set $\mathit{AA} \subset P$ of preferences indexes for which the child's age is above the maximum age. We therefore forbid such assignments.

$$\sum_{p \in \mathit{AA}} x_p = 0 \tag{7}$$

8) Knome: group of children performing the same set of activities.
This constraint is split into two parts. In the first part, if members of a knome have chosen different occurrences, then each of them can not be accepted and therefore corresponding assignments get a value of zero. Let's define $D_p \in \{0,1\}$ with value 1 if the corresponding occurrence is not chosen by the other member(s) of the knome, and value 0 otherwise.

$$\sum_{p \in P} D_p \cdot x_p = 0 \tag{8}$$

In the second part, the chosen occurrences should be either both accepted or both rejected; therefore, the corresponding variables must be equal.

$$x_i = x_j \tag{9}$$

$$\forall (i,j) \in \mathit{Knome}$$

9) Maximum number of children per occurrence.
Let's define for each occurrence $o$ the subset $\mathit{PO}_o$ of preferences that applies on occurrence $o$, and $M_o$ the constant value 'maxchild' that indicates the maximal size of the assigned children's group. The following constraints specify the maximum size of the occurrence, and forbid assignement if the occurrence is not open.

$$\sum_{p \in \mathit{PO}_p} x_p \leq M_o \cdot y_o \tag{10}$$

$$\forall o \in O$$

10) Minimum number of children per occurrence to open it.
Let's consider again for each occurrence $o$ the subset $\mathit{PO}_o$ of preferences that applies on occurrence $o$. Define $m_o$ as the constant value 'minchild' that indicates the minimal size of the assigned children's group. The constraint specifies that either the occurrence is performed by a minimum number of children, or it is not open.

$$\sum_{p \in \mathit{PO}_p} x_p \geq m_o \cdot y_o \tag{11}$$

$$\forall o \in O$$

11) Multi-occurrences activities.
This constraint is split into two parts. In the first part, an assignment gets a value of zero if it belongs to a multi-occurrences activity for which not all necessary occurrences have been chosen. In other words, incomplete activities can not be assigned to children. Let's define a

new set $IV$ , with value 1 if such a case happens, and value 0 otherwise.

$$\sum_{p \in P} IV_p \cdot x_p = 0 \quad (12)$$

In the second part, all occurrences belonging to the same multi-occurrence activity should be either all accepted or all rejected; therefore, the corresponding variables have to be equal. Let's define $PN$ as the set of corresponding preferences of successive occurrences.

$$x_p = x_{PN_p} \quad (13)$$

$$\forall p \in P$$

### C. Objective function:

Several models exist to achieve the goals: maximizing the choices of the kids and minimizing the cost differences between the children. This could be solved via goal programming, via looking for a pareto optimal solution, or via converting the problem into a mono-objective one. In order to maximize the preferences of each child while minimizing the gap between the cost of each child, the function to maximize was defined as a pondered sum of the total deviation from the mean cost per child.

1) Preferences maximisation
Let $nbc$ be the number of choices that each child can express for each period. This means that the preferred occurrence gets the choice value 1, and the least preferred occurrence within the period gets a choice value of at most $nbc$ (a child might wish to select fewer occurrences per period than $nbc$).
To mitigate disparities between preferences, we applied a power function to the preference weights and hence maximized the following $z_{pref}$ function:

$$z_{pref} = \sum_{p \in P} x_p \cdot 2^{(nbc - C_p)} \quad (14)$$

Without this power function, two sets of choices {1,1,1,4} and {2,2,2,1} would have the same assignation score of 9, although with this power function, the first one would receive a score of 25 and the second one a score of 20, favouring the assignation to the first choices.

2) Cost balancing
   a) Cost minimization
   Each activity has its own cost paid by the organizers. Children, however, pay the same price for the whole duration of Passeport Vacances, whether they get a helicopter flight or a museum visit. This means that the cost paid by the children does not depend on the assignment to activities and as there may be significant cost differences between activities. The organizers would like to balance fairly the true cost between the children.

$$z_{price1} = \sum_{p \in P} x_p \cdot PC_p + \sum_{p \in O_p} y_o \cdot PF_p \quad (15)$$

   b) Deviation cost minimization
   The first fairness proposition minimizes the total cost of the organisation. The goal is now to minimize the deviation from the mean cost per child as follows:

$$z_{price2} = \sum_{c \in C} \mid \frac{z_{price1}}{\mid C \mid} \cdot cst - cost_c \mid \quad (16)$$

Where $N_{child}$ is the total number of children, $cst$ is a constant fixed to 1.2 to allow a small margin to the average, and $cost_c$, representing the total assignment cost of each child, id defined as follows:

$$cost_c = \sum_{p \in PR_c} x_p \cdot PC_p + \frac{x_p \cdot PF_p}{\sum_{p_1 \in O_p} x_{p_1}} \quad (17)$$

$$\forall c \in C$$

Where $PR_c$ is the set of all the preferences of child c.
The second part of this sum is not linear. We approximated this sum by dividing by the maximum allowed number of children for the preference ($MC_p$) as follows:

$$cost_c = \sum_{p \in PR_c} x_p \cdot PC_p + \frac{x_p \cdot PF_p}{MC_p} \quad (18)$$

The linearization of the absolute value is a well-known technique (see for example Ünal et Uysal [2]). Adding two real variables, $c^-$ representing the lack and $c^+$ representing the surplus, we can rewrite as follows:

$$\frac{z_{price}}{\mid C \mid} \cdot cst - cost_c = c_c^+ - c_c^- \quad (19)$$

$$\text{for } c \in \text{C}$$

And the following constraints:

$$c_j^+ \geq 0 \text{ for } j \in \text{C} \quad (20)$$

$$c_j^- \geq 0 \text{ for } j \in \text{C} \quad (21)$$

Finally, we chose to minimize only the cost above the average cost. The idea here is not to privilege a child with low cost assignment ($c^-$[c]), but to penalize the costs more highly ($c^+$[c]).

$$z_{price} = \sum_{c \in C} c_c^+ \quad (22)$$

Therefore, the objective function is defined as

$$z = w_{pref} \cdot z_{pref} - w_{price} \cdot z_{price} \quad (23)$$

TABLE X
SUMMARY OF THE OBJECTIVES FUNCTIONS

| | $w_{pref}$ | $w_{pice}$ | $z_{price}$ | $z$ |
|---|---|---|---|---|
| Preference maximization | 1 | 0 | - | (14) |
| Cost minimization | 2 | 1 | (15) | (23) |
| Deviation cost minimization | 2 | 1 | (22) | (23) |

TABLE XI
AMOUNT OF DATA USED TO TEST THE METHODS

| | |
|---|---|
| 634 Child | 16621 preferences |
| 1121 activity | 50 knomes |
| 532 occurrences | 3 periods |

The following weights have been defined in the purpose of fixing the importance of each objective by normalizing each one between 0 and 1 and multiplying by a factor to weight each objective relatively.

$$w_{pref} = 2/\sum_{p\in P} x_p \cdot 2^{(nbc-C_p)} \tag{24}$$

$$w_{cost} = 1/\sum_{p\in P} PC_p + \sum_{o\in O} PF_o \tag{25}$$

We chose to fix the importance of the preference maximization as 2 times that of the cost balancing.

Table X summarizes our three distinct objective functions: preference maximization, which focuses only on the maximization of the preferences; cost minimization that maximizes the preferences and minimizes the costs; and deviation cost minimization that maximizes the preferences and minimizes the deviation from the mean cost.

## IV. Resolution

The implementation of our exact LP model was done via the open source Julia 0.6.2 language [8], [9] and the MIP solver Gurobi 0.3.3 [10]. Tests were carried out on a 3.2 GHz Intel Core i5 CPU computer with 4 GB RAM, running 64-bit Ubuntu 16.04 LTS. Julia is defined as a "high-level, high-performance dynamic programming language for numerical computing" [11]. It allows, among other things, distributed parallel execution and shows a very good performance compared to the C language. Gurobi is a state-of-the-art MIP solver. The amount of data for this real-world problem is presented in Table XI. Compared to the currently used 20-year old software, based on an iterative heuristic, time to solve the problem was drastically reduced from 11 hours to less than 5 minutes on similar single core computers.

## V. Evaluation

To evaluate the balancing, both methods, minimization of the total cost and minimization of the deviation from the mean cost, were tested with real data and compared to the solution obtained from an objective function without any fairness component.

Unsurprisingly, the cost minimization method obtains the smallest mean price, but does not improve the value of the standard deviation or the coefficient of variation compared to the preference maximization. Moreover, cost minimization decreases the total number of assignments, which goes against the desired goal. The second balancing form, deviation cost minimization, has a more equitable repartition of the costs between children. The standard deviation is reduced by 18% for the group of older children and by 25% for the younger group, which has the smallest coefficient of variation. The number of assignments is slightly higher, at the price of a small decrease in children's preferences. We also note that adding new variables for the approximation of the absolute value penalizes the performance, as shown in Table XIV, but it still remains reasonable in this context. Table XII and Figure 1 present a summary of the obtained solutions. Column cv corresponds to the coefficient of variation and column threshold refers to outliers detection (i.e. a threshold value beyond which a cost is considered as an outlier). Finally, Table XIII presents the number of assignments by order of choice's priority: in a fair solution, fewer first choices are assigned to children, replaced by second or third choices.

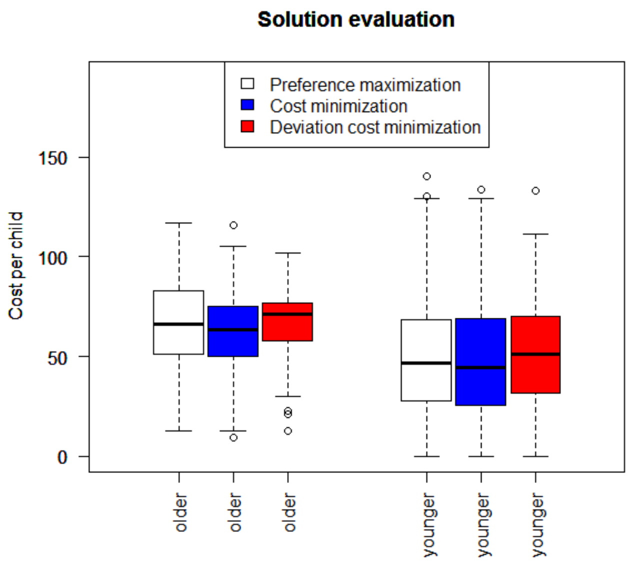

Fig. 1. Comparison of cost's repartition between children

## VI. Conclusion

This article presents the modelization of a real-world assignment problem in which the objective is to assign children to activities they chose respecting as much as possible their preference order without violating the different constraints. The second part of the study focuses on the desire to balance the costs fairly between children, as they all pay the same subscription cost to Passeport Vacances. Two methods are proposed, which take into account the costs and are compared with the solution without cost balancing. The cost's minimization deteriorates the quality of the assignment without decreasing the inequality between each children, while the

TABLE XII
SUMMARY OF THE ASSIGNATION SOLUTION FOR EACH METHOD

| | Age category | Mean cost | Std cost | Nb child | Threshold | cv |
|---|---|---|---|---|---|---|
| Preference maximization | 1 | 49.60 | 29.37 | 570 | 131.83 | 0.59 |
| | 2 | 65.32 | 25.40 | 51 | 136.45 | 0.39 |
| Cost minimization | 1 | 48.31 | 29.39 | 570 | 130.60 | 0.61 |
| | 2 | 62.43 | 25.93 | 51 | 135.05 | 0.42 |
| Deviation cost minimization | 1 | 49.83 | 24.17 | 570 | 117.51 | 0.49 |
| | 2 | 65.31 | 18.87 | 51 | 118.16 | 0.29 |

TABLE XIII
NUMBER OF ASSIGNATION, ORDERED BY CHOICE'S PRIORITY FOR EACH OF THE 3 METHODS

| Choices | Nbr of assignations Preference maximization | Nbr of assignation Cost minimization | Nrb of assignations Deviation cost minimization |
|---|---|---|---|
| 1 | 3372 | 3375 | 3369 |
| 2 | 1014 | 1011 | 1028 |
| 3 | 414 | 392 | 416 |
| 4 | 173 | 156 | 172 |
| total | 4973 | 4934 | 4985 |

TABLE XIV
TIME OF RESOLUTION FOR EACH MODELS

| Time of assignation Preference maximization | Time of assignation Cost minimization | Time of assignations Deviation cost minimization |
|---|---|---|
| 2 minutes and 7 seconds | 2 minutes and 7 seconds | 3 minutes and 27 seconds |

minimization of the deviation from the mean price gets closer to the desired goal, but increases the computational time because of the increase in the number of variables. Improving cost balancing without exploding the number of variables or computational time, by means of valid inequalities, is one possible future research direction.

## REFERENCES

[1] F. Uney-Yuksektepe and İ. Karabulut, "Mathematical programming approach to course-teaching assistant assignment problem," in *Proceedings of the 41st International Conference on Computers & Industrial Engineering*, 2011, pp. 878–883.

[2] Y. Z. Ünal and Ö. Uysal, "A new mixed integer programming model for curriculum balancing: Application to a turkish university," *European Journal of Operational Research*, vol. 238, no. 1, pp. 339–347, 2014. doi: https://doi.org/10.1016/j.ejor.2014.03.015

[3] R. de la Torre, A. Lusa, and M. Mateo, "A MILP model for the long term academic staff size and composition planning in public universities," *Omega*, vol. 63, pp. 1–11, sep 2016. doi: 10.1016/j.omega.2015.09.008

[4] E. L. Bouzarth, R. Forrester, K. R. Hutson, and L. Reddoch, "Assigning students to schools to minimize both transportation costs and socioeconomic variation between schools," *Socio-Economic Planning Sciences*, sep 2017. doi: 10.1016/j.seps.2017.09.001

[5] B. Domenech and A. Lusa, "A MILP model for the teacher assignment problem considering teachers' preferences," *European Journal of Operational Research*, vol. 249, no. 3, pp. 1153–1160, mar 2016. doi: 10.1016/j.ejor.2015.08.057

[6] L. Pan, S. Chu, G. Han, and J. Z. Huang, "Multi-criteria student project allocation: A case study of goal programming formulation with dss implementation," in *The Eighth International Symposium on Operations Research and Its Applications (ISORA'09), Zhangjiajie, China*, 2009, pp. 75–82.

[7] I. Kim and O. de Weck, "Adaptive weighted-sum method for bi-objective optimization: Pareto front generation," *Structural and Multidisciplinary Optimization*, vol. 29, no. 2, pp. 149–158, sep 2004. doi: 10.1007/s00158-004-0465-1

[8] J. Bezanson, S. Karpinski, V. B. Shah, and A. Edelman, "Julia: A fast dynamic language for technical computing," *CoRR*, vol. abs/1209.5145, 2012. [Online]. Available: http://arxiv.org/abs/1209.5145

[9] J. Bezanson, A. Edelman, S. Karpinski, and V. B. Shah, "Julia: A fresh approach to numerical computing," *SIAM Review*, vol. 59, no. 1, pp. 65–98, 2017. doi: 10.1137/141000671. [Online]. Available: http://julialang.org/publications/julia-fresh-approach-BEKS.pdf

[10] [Online]. Available: http://www.gurobi.com/

[11] [Online]. Available: https://julialang.org