



UNIVERSITÀ
DELLA CALABRIA

ALGORITHMIC GAME THEORY

UNIVERSITÀ DELLA CALABRIA

DEPARTMENT OF ARTIFICIAL INTELLIGENCE AND COMPUTER SCIENCE

SEMESTER 1 2021/2022

Travelr - A guide to plan your trip

February 7, 2022

Authors:

Franziska Müller, 238887

Kerstin Greifensteiner, 238886

Ruth Nikol, 234904

Supervisors:

Prof. Gianluigi Greco

Carlo Adornetto

Travelr - A guide to plan your trip: February 7, 2022, Cosenza

Contents

List of figures	III
List of tables	IV
1 Introduction	1
1.1 Scenario	1
1.2 Formalisation	2
2 Mechanism	4
2.1 Expectations	4
2.2 Payments	5
2.3 Final route and participants	8
2.4 Example	11
3 Implementation	15
3.1 Technology	15
3.2 Concept	16
3.3 Mechanism implementation	19
3.4 Known issues	20
3.5 Example	21
4 Conclusion	24
5 Bibliography	25

List of Figures

1.1	Visualisation of the input and output variables of the mechanism	2
3.1	First screen when opening the app	16
3.2	The page to modify the tour parameters. Sample data for easy testing is already given.	17
3.3	The loading screen while the best tour is being computed	17
3.4	The final result will be shown with a list of participants and their payments and a map, displaying the shortest path of the tour and their waypoints.	18
3.5	The button on the upper right triggers the debug log, which shows relevant information of each step in the calculation.	19
3.6	Modification page with all parameters set	22
3.7	The result calculated by the application. Alice and Charlie would go on a trip to Cirella and Tropea.	23
3.8	Debug log from web application's calculation with different Groove's values	23

List of Tables

2.1	Distances in <i>km</i>	11
2.2	Agents, budgets and utilities	12
2.3	Route lengths and costs	12
2.4	Groups of size one	12
2.5	Options with utility for the groups	12
2.6	Groups of size two	14
3.1	Agents, budgets and utilities - Replication of the parameters used for the example of the mechanism	21

1 Introduction

This project aims to apply and consolidate the skills learnt in the course “Algorithmic Game Theory”. The project assignment requires the design of a mechanism that calculates a shared road trip among friends in a fair way. This means that it includes balancing the budget and enforcing the agents to declare their wishes truthfully. In the following, the scenario of the project will be explained in more detail.

A group of friends would like to go on a trip together through different cities. From experience, it can be quite difficult to plan such a trip. Each of the participants has diverse ideas about what the trip should look like. Some of the conflict areas are, for example, that each participant wants to see different cities. Agreeing on a number of cities will not be easy, especially as the cost of travelling to each city has to be taken into account. All people have different possibilities, some will be able to spend a lot of money on the trip, while others simply cannot or do not want to pay that much.

The mechanism to be designed should ensure that a number of people and cities are selected from all possibilities, given by the players and their preferences. An important point is, that all of them should be able to afford the costs that will arise for this trip.

This paper goes into more detail in the following section 1 on the scenario of the task and the assumptions made by the team. In chapter 2, the mechanism is explained. It deals with the theoretical implementation of the truthful expression of preferences and the subsequent cost distribution among the participants in detail. To support the understanding of the mechanism, an execution in pseudo code is included. This chapter also elaborates on problems and unsolvable conflicts in the design of this mechanism. In order to clarify the work in this paper, the mechanism is subsequently explained with an illustrative example in section 2.4. Since there are no implementations of this kind on the market so far, this mechanism has been implemented in a small and simple web application. Applied tools, the structure of the code as well as the implementation of the mechanism are explained in chapter 3. A review and outlook on future work and development is given in the summary in chapter 4.

1.1 Scenario

The scenario consists of a group of friends (also called players or agents) that want to do a round trip together. However, each of the friends has different preferences about the cities they want to visit

and the budget that is available to them. Consequently, a reasonable selection of cities must be made, based on the players' preferences. Additionally, a limit is set on the overall distance they can travel. The means of transport chosen for the tour has a maximum capacity, which limits the number of friends who can join the trip. Therefore, an additional choice must be made as to which players can participate in the trip and which are excluded.

1.2 Formalisation

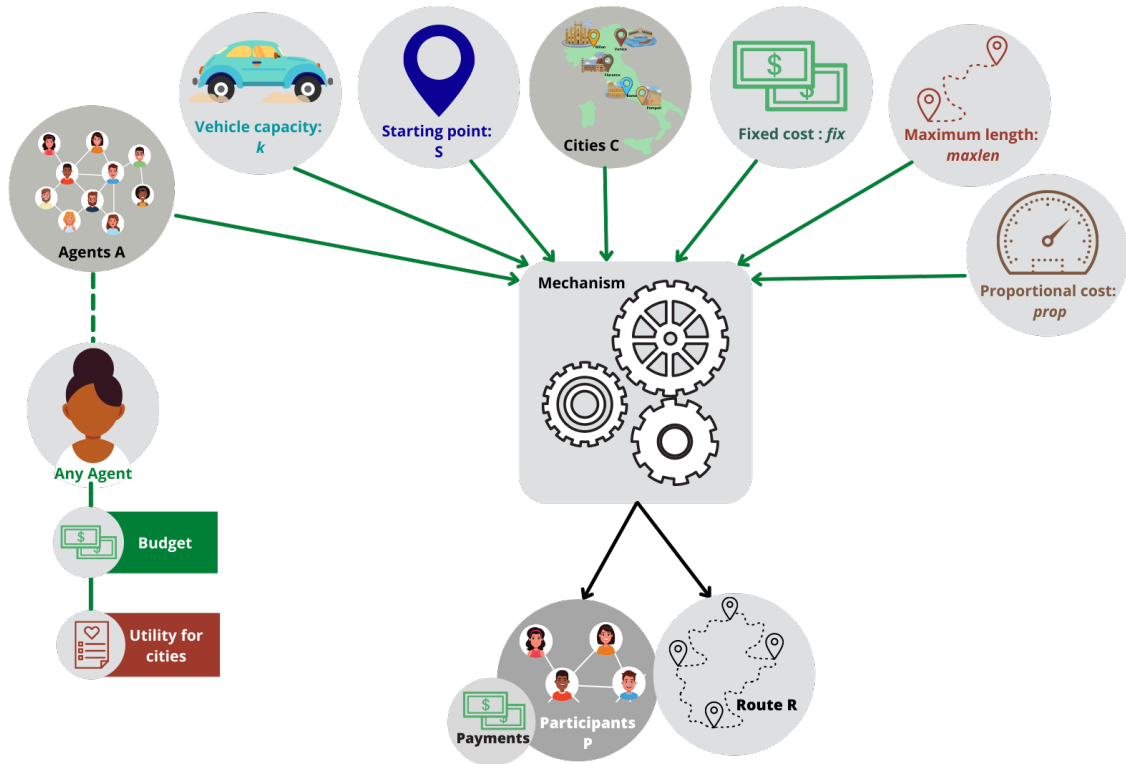


Figure 1.1: Visualisation of the input and output variables of the mechanism

Formalised, our mechanism is given an **input** $I = (F, C, b, u, s, k, fix, prop, maxlen)$:

- **Agents:** a set $A = \{a_1, \dots, a_n\}$ of friends
- **Cities:** a set $C = \{c_1, \dots, c_m\}$ of cities (with coordinates so the distance between two cities can be determined)
- **Budget:** a function $b : A \mapsto \mathbb{R}^+$
- **Utility:** a function $\hat{v} : A \times C \mapsto [-10, 10]$ expressing the preferences the player *claims to have* about cities (assumed 0 if not explicitly given for a city)

In contrast, the function $u : A \times C \mapsto [-10, 10]$ expresses the *true* utilities of the agents, however it is not known to the mechanism

- **Starting point:** a city s with $s \notin C$ (again with coordinates)
- **Vehicle capacity:** a capacity $k \in \mathbb{N}$ indicating the maximum number of players who can participate
- **Fixed cost:** a cost $fix \in \mathbb{R}^+$ that arises for every tour
- **Proportional cost:** a cost per distance $prop \in \mathbb{R}^+$ specifying how much additional costs arise for a certain travel distance
- **Maximum length:** a distance $maxlen \in \mathbb{R}^+$ restricting the length of the tour

The mechanism returns an **outcome** $O = (P, R, p)$:

- **Participants:** the set $P \subseteq A$, $|P| \leq k$ of players to take part in the trip
- **Route:** the set $R \subseteq C$ of cities to be visited
- **Payments:** the distribution $p : P \mapsto \mathbb{R}$ of the overall costs among the participants with $p(a_i) \leq b(a_i)$ and $\sum_{a_i \in P} p(a_i) = TSP(R \cup s) \times prop + fix$, where $TSP(R)$ gives length of the shortest round trip visiting all cities in R

2 Mechanism

In order to meet the requirements of the project, a mechanism was designed that enforces both honesty among the players and the greatest possible social welfare. For this purpose, the following section explains the expectations of the mechanism in more detail. Subsequently, a modification of the well-known VCG mechanism is formulated and applied to the scenario. The whole mechanism is then demonstrated by means of an example.

2.1 Expectations

This section deals with the expectations of the developed mechanism. These are truthfulness, fairness and maximising the utility.

Truthfulness

The distribution of the costs should enforce the players to be *truthful* about their utilities, so that $\hat{v}(a, c) = u(a, c)$. Hence, telling the truth must be a dominant strategy for the players.

Fairness

The costs should be distributed in a *fair* way among the participants. This means, that no player or group of players should have a reason to object the selected outcome.

Definition 2.1.1 (Reason to object). It is reasonable for a group of players to object to participate in an outcome, if the outcome o_c that would be chosen for this group alone either

- gives them a higher utility, implying that all players have the same or a higher utility for o_c and at least one player has a truly higher utility for o_c , *or*
- gives them the same utility but charges them less, implying that the utility is the same for each player but at least one of them has to pay less in o_c while the others pay the same

Maximum utility

The route and set of players chosen by the mechanism should be a possible option and should have a maximum outcome utility (social welfare).

Definition 2.1.2 (Possible Outcome). A possible outcome is an outcome that complies with the following requirements:

- The number of participants does not exceed the vehicle's capacity: $|P| \leq k$
- No participant is charged more than they can afford: $p(a_i) \leq b(a_i)$
- The length of the tour does not exceed $maxlen$
- No subgroup of players has reason to object to the tour cf. *Fairness*

Definition 2.1.3 (Utility of an outcome). The outcome utility U of an outcome $O = (P, R, p)$ is defined as follows:

$$U(O) = \sum_{a \in P} \sum_{c \in R} u(a, c)$$

2.2 Payments

If $|P| = 1$, only one agent is participating in the tour the payment distribution is trivial. The participating agent is assigned the total cost of the tour. In all other cases, the costs are distributed according to a payment function, which will be discussed in the following.

The VCG mechanism

The truthfulness of the players is to be enforced through the distribution of the costs. Therefore truth-telling has to be a dominant strategy. The Vickrey-Clarke-Groves (VCG) mechanism fulfils this requirement. [1, Chapter 10.4.2]

Definition 2.2.1 (Vickrey-Clarke-Groves mechanisms). [1, Definition 10.4.5] The VCG mechanism is a direct quasilinear mechanism (χ, ρ) where

$$\begin{aligned} \chi(\hat{v}) &= \operatorname{argmax}_x \sum_i \hat{v}_i(x) \\ \rho_i(\hat{v}) &= \sum_{j \neq i} \hat{v}_j(\chi(\hat{v}_{-i})) - \sum_{j \neq i} \hat{v}_j(\chi(\hat{v})) \end{aligned}$$

Mapping to total costs

Due to how this scenario is set up, the VCG mechanism cannot be applied as it is to distribute the costs. It is necessary to map $\rho_i(\hat{v})$ to the overall costs for the selected tour. This can be achieved with the following formula:

$$p_i(\hat{v}) = \frac{\rho_i(\hat{v})}{\sum_{a_j \in P} \rho_j(\hat{v})} \times \operatorname{costs}(R)$$

The ratio of a players VCG payment to the overall distributed VCG payments is used and applied to the total costs of the tour.

Engaging all participants in the payment

A problem with this approach is that $\rho_i(\hat{v}) = 0$ will occur also for participants who's utility is greater than zero. In particular, this happens for a participant f_i if the other participants would choose the same tour without them, thus their utility would be unchanged. However, the outcome with a_i added has a higher utility than without them, since a_i gains utility by participating. In this scenario it is not desirable that a_i is not engaged in the payment. Moreover it could lead to no player being charged, which contradicts the requirement that the costs must be covered by the participants and also results in a division by zero in the payment distribution function.

Example

$$b(\text{Alice}) = b(\text{Bob}) = 30$$

$$\text{cost}(\{\text{Rome}\}) = 30$$

$$k = 2$$

It is supposed that the route $\{\text{Rome}\}$ does not exceed *maxlen*.

Utilities	Rome
Alice	1
Bob	2

Both Alice and Bob on their own would go to Rome. Also, together they would choose $R = \{\text{Rome}\}$. This would result in:

$$\rho_{\text{Alice}} = 2 - 2 = 0 \quad \rho_{\text{Bob}} = 1 - 1 = 0$$

$$p_{\text{Alice}} = \frac{30}{0+0} \times 0 \quad p_{\text{Bob}} = \frac{30}{0+0} \times 0$$

To circumvent this problem $p_i(\hat{v})$ must be defined differently. A better solution can be achieved by substituting $\rho_i(\hat{v})$ with $\rho'_i(\hat{v})$ where

$$\rho'_i(\hat{v}) = \sum_{j \neq i} \hat{v}_j(\chi(\hat{v}_{-1})) \times \lambda - \sum_{j \neq i} \hat{v}_j(\chi(\hat{v})) \quad \text{with } \lambda > 1$$

λ can be any value greater than 1. Since it is constant for all players, the ratio between the $\rho'_i(\hat{v})$ s for the participants does not change. Therefore the finally assigned payments stay the same for any constant $\lambda > 1$.

This is an improvement, since a player, who gains utility by participating and does not increase other player's utilities, is always charged:

$$\sum_{j \neq i} \hat{v}_j(\chi(\hat{v}_{-1})) \geq \sum_{j \neq i} \hat{v}_j(\chi(\hat{v})) \implies \rho'_i(\hat{v}) > 0$$

In case the other player's utility increases by adding a_i to the participants, no general statement can be made about $\rho'_i(\hat{v})$. There are three possible cases:

1. If $\hat{v}_i < 0$, the player will object to participate in any case. The outcome is impossible and does

not have to be further considered.

2. If $\hat{v}_i = 0$, the player will object to participate if charged. So, the costs have to be split among the other participants. They cannot afford it, otherwise they would have selected the tour also without a_i and their utility would not have increased. In conclusion, the outcome is also impossible and does not have to be further considered.
3. If $\hat{v}_i > 0$, the player might be charged for participating. This depends on how much the utility of the other players is increased. If a_i is charged, the tour might be possible and it has to be checked if it can be afforded. If not, the outcome is impossible, cf. case 2.

This implies, that in any meaningful outcome each participant is charged. Hence, it can be assumed that $\rho_i(\hat{v})$ is always positive for all participants. The division by zero, that was an issue in the original payment distribution function, can not occur anymore with this adaption.

Fairness

This solution is an improvement to the first approach. However, it exhibits a strange behaviour regarding the ratio of the payments to be made by different players. It becomes apparent when examining the example specified on the previous page.

$P = \{\text{Alice}, \text{Bob}\}$ and $R = \{\text{Rome}\}$ result in the highest outcome utility. Intuitively, Bob would be expected to pay more, because he has a higher utility for the route than Alice. However, the mechanism will charge him less.

$$\begin{aligned} \rho'_{\text{Alice}} &= 2 \times 2 - 2 = 2 & \rho'_{\text{Bob}} &= 1 \times 2 - 1 = 1 \\ p_{\text{Alice}} &= \frac{30}{2+1} \times 2 = 20 & p_{\text{Bob}} &= \frac{30}{2+1} \times 2 = 10 \end{aligned}$$

This issue arises because the $\rho_i(\hat{v})$ of a participant a_i does not depend on their own utility but rather on the utility of the others. Yet, the outcome in which Alice is charged 20 and Bob 10 can still be considered *fair* in the sense, that both gain from travelling together; would they go on the trip alone, they would have the same utility but pay 30. Thus, none of them has a reason to object the outcome.

Truthfulness

It is known, that the payment distribution of the VCG mechanism enforces the agents to be truthful. However, the mapping of $\rho_i(\hat{v})$ to the total costs conflicts this truthfulness. In VCG, truth-telling is a dominant strategy, because $\rho_i(\hat{v})$ does not depend on agent a_i 's own utility. Therefore, a player cannot reduce what they are charged by lying about their utilities. However, in the formula for payment distribution introduced above, the own utility of an agent influences the payment they are charged. To calculate the ratio of player a_i 's payment to the other players' payments, the $\rho_j(\hat{v})$ s of the other

participants have to be taken into account (in the denominator) and these depend on a_i 's preferences. Since the total costs are fixed, a lower ratio of a_i 's payments to the other payments directly results in a lower charge for a_i . Hence a player a_i can reduce their share by raising the VCG payments for the others; since these depend on \hat{v}_i , they can be increased by increasing a_i 's declared utilities.

The problem will appear in any payment distribution function that in some way relies on utilities. This is because an agent's payment always depends on the other agents' payments, since the sum of all charges must always exactly result in the total costs. As soon as a player a_i 's payment depends on any other player a_j 's utility, also a_j 's charge depends on it because it depends on a_i 's payment. Hence, the problem can only be evaded by either making the payments independent of the utilities or by concealing the possibility of cheating from the agents.

The payment distribution function can be rewritten as follows:

$$p_i(\hat{v}) = \frac{\rho_i(\hat{v})}{\sum_{a_j \in P} \rho_j(\hat{v})} \times \text{costs}(R) = \frac{\text{costs}(R)}{\sum_{a_j \in P} \rho_j(\hat{v})} \times \rho_i(\hat{v})$$

Since the total costs as well as the sum over all $\rho_j(\hat{v})$ are the same for each player, the fraction can be treated as a constant. The agents know that their payment is to be calculated by $\text{constant} \times \rho_i(\hat{v})$. Since VCG does not give them any reason to lie about their preferences, truthfulness will appear as a dominant strategy for them.

2.3 Final route and participants

The participants and cities included in the tour have to be selected to optimise the outcome utility. Therefore, the obtained utilities of possible outcomes have to be compared. In order to select the best option, the mechanism has to iterate through all city combinations for each group of players that could take part in the trip. In the process, impossible outcomes can already be removed. Finally, the combination with the highest social welfare is selected. In general, there can be more than one solution, if multiple combinations have the same outcome utility.

The algorithms 1 - 3 show the procedure in more detail. The division in three parts is merely to improve readability, algorithm 2 and 3 are both parts of the mechanism (algorithm 1, line 23/24).

In the mechanism, the first step is sorting out all the routes that exceed the maximum tour length and therefore do not need to be considered (line 7-10). Next, for all possible combinations of participants, beginning with the smallest groups (line 7-10), the best outcome has to be calculated. Therefore, the possible routes are sorted descending by the outcome utility they provide to the group (line 14). So then it can be iterated over these city combinations (line 16), while any route that exceeds the

group's common budget can be skipped. For all other city combinations, the payment distribution is calculated according to the function elaborated in section 2.2. This is where it becomes important, that the results for smaller groups have already been calculated, since the outcomes of subgroups have to be accessed in order to calculate the payments. In the following, it has to be checked, that the agents can afford the payment they have been assigned (line 23 / algorithm 2) and that no agent objects to the outcome (line 24 / algorithm 3). If this is the case, the current route is the best option for the group. Since the city combinations had been ordered by utility, none of the following routes could have a higher utility, hence the calculation does not have to be pursued for the current group. The outcome can be stored (line 25-27). Otherwise, if after iterating through all possible routes, none of them was found to be suitable, it is not reasonable for the group to go on the trip together. Instead, the best option is that only a part of them participates, so the subgroup with the highest utility (line 29-32). Finally, the outcome with the highest utility, so the route and the group group of participants as well as their payments are returned.

Algorithm 2 iterates through all participants in the group and checks if the payment, they are assigned by the mechanism, is in their budget. If one of the participants cannot afford their charge, the outcome is not possible and it can be skipped. Therefore, `found_tour` is set FALSE.

Algorithm 3 checks if players would object to the currently selected tour. It is iterated through every subgroup of the current group (line 3) to check if they would object. In line 4-12 it is investigated whether the subgroup's outcome gives them a better utility. This is possible if the group's overall utility is higher for their outcome than for the current one (line 4). If this is the case, it has to be checked if there is a player who gains utility from the new outcome (line 7-10), otherwise the group objects.

If the subgroup's overall utility does not change, they might still object, if their overall payment is smaller (line 14). Once again each player has to be reviewed. If there is a player whose utility changes (line 17-19) or that pays less in the new outcome (line 20-22), the group does not object, otherwise it does (line 23-24).

(It is sufficient to find a player whose utility changes, irrelevant if for the better or the worse, to know that the group does not object. If it changes for the worse, obviously for this player it is not reasonable to object. If it changes for the better, there must be different participant whose utility decreases, for otherwise the overall utility of the group would have increased. Since this other player exists and they would not object to the new outcome, the group cannot object.)

If the group objects to the outcome, `found_tour` is set FALSE so the mechanism will move on to the next route.

Algorithm 1 The mechanism

```

1: function TRAVELR(agents, cities, budgets, utilities, fix, prop, maxlen, k)
2:   for cc in city_combinations do
3:     if length(cc) > maxlen then
4:       remove cc from city_combinations
5:       ▷ Routes that exceed the maximal length do not have to be considered
6:
7:   for group_size in 1 ... k do
8:     ▷ so the results for subgroups (subsets of participants) are always calculated first
9:
10:    for ac in agent_combinations with size group_size do
11:      group_budget =  $\Sigma$  budget[participant in ac]
12:      outcome[ac] = NULL
13:
14:      sort(city_combinations) ▷ Descending by utility, tours with utility  $\leq 0$  are removed
15:
16:      for cc in city_combinations do
17:        if costs(cc) > group_budget then
18:          continue ▷ Participants cannot afford it anyway
19:
20:      payments  $\leftarrow$  calculate_payments(sc, cc, outcomes)
21:
22:      found_tour  $\leftarrow$  TRUE
23:      Affordable payments
24:      No participant objects
25:      if found_tour then
26:        outcome[ac] = cc
27:        break
28:
29:      if outcome[ac] == NULL then
30:        ▷ If no outcome was found, assign the one off the best subgroup
31:        for sg  $\subset$  ac do
32:          outcome[ac]  $\leftarrow$  max_utility(outcome[ac], outcome[sg])
33:
34:  return max_utility(outcome) ▷ Return the outcome with the highest utility

```

Algorithm 2 Affordable payments

```

1: found_tour  $\leftarrow$  TRUE
2: for participant in ac do
3:   if payments[participant] > budget[participant] then
4:     found_tour  $\leftarrow$  FALSE
5:   break

```

Algorithm 3 No participant objects

```

1: object  $\leftarrow$  FALSE
2: if found_tour then
3:   for sg  $\subset$  ac do
4:     if utility(outcome[sg]) > utility(cc, for sg) then
5:        $\triangleright$  If the subgroup on its own has a higher utility
6:       object  $\leftarrow$  TRUE
7:       for par in sg do
8:         if utility(outcome[sg], for par) < utility(cc, for par) then
9:           object  $\leftarrow$  FALSE
10:          break
11:       if object then
12:         break
13:
14:     if utility(outcome[sg]) == utility(cc, for sg) and payments(outcome[sg]) < payments(cc,
    for sg) then  $\triangleright$  If the subgroup on its own has the same utility but less costs
15:       object  $\leftarrow$  TRUE
16:       for par in sg do
17:         if utility(outcome[sg], for par) != utility(cc, for par) then
18:           object  $\leftarrow$  FALSE
19:           break
20:         if payments(outcome[sg], for par) > payments(cc, for par) then
21:           object  $\leftarrow$  FALSE
22:           break
23:         if object then
24:           break
25:       if object then
26:         found_tour  $\leftarrow$  FALSE

```

2.4 Example

C = {Crotone, Diamante,
Siderno, Tropea}

s = Rende

k = 2

fix = 10€

prop = 0.1€/km

maxlen = 420km

<i>distances</i>	Crotone	Diamante	Rende	Siderno	Tropea
Crotone	-	184	118	152	157
Diamante	184	-	69	206	156
Rende	118	69	-	174	124
Siderno	152	206	174	-	83
Tropea	157	156	124	83	-

Table 2.1: Distances in *km*

Agents A	Budgets b	$u(\text{Crotone})$	$u(\text{Diamante})$	$u(\text{Siderno})$	$u(\text{Tropea})$
Alice	24€	-2	5	4	1
Bob	34€	6	4	-9	2
Charlie	24€	-1	3	0	3

Table 2.2: Agents, budgets and utilities

For a start, all routes that exceed the *maxlen* are removed, for all others the costs are calculated. The city names in the following table are abbreviated by their first character.

Route	Length	Cost	Route	Length	Cost	Route	Length	Cost
C	236	33,6	C,S	444		C,S,D	545	
D	138	23,8	C,T	399	49,9	C,T,D	500	
S	348	44,8	D,S	449		C,S,T	477	
T	248	34,8	D,T	349	44,9	D,S,T	482	
C,D	371	47,1	S,T	381	48,1	C,S,T,D	578	

Table 2.3: Route lengths and costs

Then the best outcome for each group is calculated, beginning with groups of size 1.

Alice can only afford to go to go to Diamante, since she has a positive utility for this city she would choose to go on the trip. Since she is alone she pays the total costs. The same applies for Charlie. Bob has the choice between Crotone and Diamante, since his utility for Crotone is higher, he will choose to go there.

Participants	Cities	Payments	Utility
Alice	Diamante	23,8	5
Bob	Crotone	33,6	6
Charlie	Diamante	23,8	3

Table 2.4: Groups of size one

Next come the groups with two participants. First all routes that exceed the group's budget or have utility ≤ 0 can be sorted out.

	C	D	S	T	C,D	C,T	D,T	S,T
Alice, Bob	4	9	x	3	13	7	12	x
Alice, Charlie	x	8	4	4	5	x	12	x
Bob, Charlie	5	7	x	5	12	10	12	x

Table 2.5: Options with utility for the groups
(x: utility ≤ 0 , x: not affordable)

Alice, Bob

The option with the highest utility for them is {Crotone, Diamante}. The payment distribution has to

be calculated.

$$\begin{aligned}\rho_{Alice}(\hat{v}) &= 6 \times 2 - 10 = 2 & \rho_{Bob}(\hat{v}) &= 5 \times 2 - 3 = 7 \\ p_{Alice} &= \frac{47,1}{2+7} \times 2 = 10,4\bar{6} & p_{Bob} &= \frac{47,1}{2+7} \times 7 = 36,6\bar{3}\end{aligned}$$

Bob cannot afford his share of the costs, so the mechanism moves on to the next best route which is {Diamante, Tropea}.

$$\begin{aligned}\rho_{Alice}(\hat{v}) &= 6 \times 2 - 6 = 6 & \rho_{Bob}(\hat{v}) &= 5 \times 2 - 6 = 4 \\ p_{Alice} &= \frac{44,9}{6+4} \times 6 = 26,94 & p_{Bob} &= \frac{44,9}{6+4} \times 4 = 17,96\end{aligned}$$

Here, Alice cannot afford her share of the cost. The next best tour is {Diamante}. Since Bob's utility for this is only 4, while on his own he would get utility 6, he objects to this tour. After this comes {Crotone, Tropea}, Alice's utility for this is negative, therefore she objects. The same happens for {Crotone}. The route {Tropea} is a degradation for both of the participants so they object.

Finally no tour was found that Alice and Bob would be willing to go on together. Therefore the best outcome for the group {Alice, Bob} is the outcome for the subgroup with the highest utility. Bob on his own gets utility 6, Alice utility 5, hence the outcome ({Bob}, {Crotone}, (33,6)) is the optimum for the group.

Alice, Charlie

Alice and Charlie have their highest utility with the tour {Diamante, Tropea}. The payments would be distributed as follows.

$$\begin{aligned}\rho_{Alice}(\hat{v}) &= 3 \times 2 - 6 = 0 & \rho_{Charlie}(\hat{v}) &= 5 \times 2 - 6 = 4 \\ p_{Alice} &= \frac{44,9}{0+4} \times 0 = 0 & p_{Bob} &= \frac{44,9}{0+4} \times 4 = 44,9\end{aligned}$$

Since Bob cannot afford this the next best tour has to be considered. This is going only to Diamante.

$$\begin{aligned}\rho_{Alice}(\hat{v}) &= 3 \times 2 - 3 = 3 & \rho_{Charlie}(\hat{v}) &= 5 \times 2 - 5 = 5 \\ p_{Alice} &= \frac{23,8}{3+5} \times 3 = 8,925 & p_{Bob} &= \frac{23,8}{3+5} \times 5 = 14,875\end{aligned}$$

With this the utility stays the same for both but each of them has to pay less. Hence the best route for Alice and Charlie is {Diamante}.

Bob, Charlie

The utilities for {Crotone, Diamante} and {Diamante, Tropea} are the same for this group. Starting

with {Diamante, Tropea} the following payment distributions can be calculated.

$$\rho_{Bob}(\hat{v}) = 3 \times 2 - 6 = 0 \quad \rho_{Charlie}(\hat{v}) = 6 \times 2 - 6 = 6$$

$$p_{Bob} = \frac{44,90 + 6 \times 0}{2} = 22,45 \quad p_{Charlie} = \frac{44,90 + 6 \times 6}{2} = 44,9$$

Charlie cannot afford this, so {Crotone, Diamante} is considered next. Charlies utility for this is 2, which is smaller than the 3 he would receive if he went to Diamante on his own, so he objects to the tour. The next smaller utility is 10 for {Crotone, Tropea}, as with the previous tour, Charlies utility decreases in respect to going to Diamante alone, hence he objects again. Going to Diamante together is a degradation for Bob, as well as going to Tropea, he objects to both options. For going to Crotone, Charlie even receives a negative utility so he objects. In conclusion, the best tour for Bob and Charlie is Bob going to Crotone alone.

The results for the three groups are summarised in table 2.6.

Group	Participants	Cities	Payments	Utility
Alice, Bob	Bob	Crotone	33,6	6
Alice, Charlie	Alice, Charlie	Diamante	8,925 14,875	8
Bob, Charlie	Bob	Crotone	33,6	6

Table 2.6: Groups of size two

The capacity of the vehicle limits the participants to 2, so the mechanism can stop at this point. The final outcome is the outcome with the highest utility. From table 2.6 it is obvious, that means that Alice and Charlie will go to Diamante together. Alice pays 8,925€ for the trip and Charlie 14,875€.

3 Implementation

The mechanism introduced in this paper combines mechanism design, VCG and the travelling salesman problem (TSP) to determine cities, participants and the shortest path which are still affordable to perform a trip. Since there does not exist such an application, an implementation has been developed. In this chapter, the “Travelr”, a web application for unique travelling, will be presented, as well as the execution of the example illustrated in section 2.4.

“Travelr” allows to insert important parameters, like starting city and fixed cost (more in section 3.2). Another form lets the user insert players (prior named as friends or agents). It takes the name, the maximum budget and a list of preferences for cities the user can choose. Then, the app generates a tour of given cities, considering the given restrictions, like maximum distance in kilometres, and selects the participants that can join the trip. A map will be displayed, showing the shortest route the party can take to travel to said cities. There is also a debug log output for development purposes.

Relevant technologies applied for this project will be presented in section 3.1. The overall workflow and functionality will be explained in more detail in sections 3.2 and 3.3. Some issues that were encountered during development will be listed in section 3.4. Lastly, the example from the theoretical viewpoint of the mechanism will be compared to the same data used in the implementation (3.5).

3.1 Technology

The decision of technologies depended on existing experiences. Since not a new framework or programming language wanted to be learned, the well-known language *JavaScript* was used. It is straightforward and allows for easy assembling of objects (*JSON*). Based on this first decision, the web application was developed using following frameworks:

- **NodeJS** (<https://nodejs.org/en/>) - backend server
- **ExpressJS** (<https://expressjs.com/>) - web framework for NodeJS
- **PugJS** (<https://github.com/pugjs/pug>) - template engine

A key feature is the integrated and interactive map that will be displayed once a suitable route has been found. It is part of the **HERE Maps API** (<https://developer.here.com/>) and usable with a free

account until a specific (quite high) amount of requests per month. For following features, this API has been used:

- Finding longitude and latitude of a city with postal code
- Retrieving the distance between cities
- Shortest path calculation
- Interactive map
- Route display
- City markings

3.2 Concept

The app starts with a welcoming screen (see figure 3.1). It shows the logo and a button to continue.

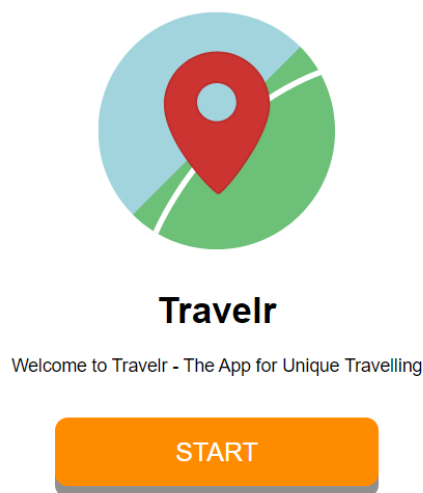



Figure 3.1: First screen when opening the app

Through this displayed button, a page to modify the tour calculation parameters can be accessed (see figure 3.2). It takes following variables:

- Starting and ending city with postal code
- Maximum length of the tour in kilometres



Here you can modify your routing request!

Start/End:	Max. KM:	Max. Seats:	Fixed Cost:	KM Cost:
<input type="text" value="Cosenza"/> <input type="text" value="87100"/>	<input type="text" value="400"/>	<input type="text" value="4"/>	<input type="text" value="100"/>	<input type="text" value="2"/>

Insert a participant:
Name:
Budget:
Preferences:

Active Participants:

Franzi (340€) - {Tropea (89861): 8, Scilla (89058): 3, Reggio (89135): -2}

Kerstin (420€) - {Scilla (89058): 7, Catanzaro (88100): 2, Tropea (89861): 1}

Ruth (370€) - {Tropea (89861): 7, Reggio (89135): 5, Scilla (89058): -3}

Figure 3.2: The page to modify the tour parameters. Sample data for easy testing is already given.

- Maximum amount of seats for participants
- Fixed cost
- Variable cost per kilometre
- List of all participants stating their budget and city preferences

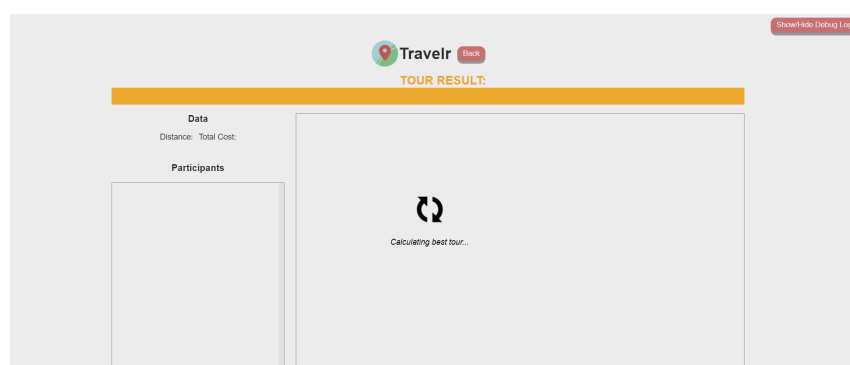


Figure 3.3: The loading screen while the best tour is being computed

Once the user clicks on the “Compute” button, the next page will be loaded (see figure 3.3). It includes a loading screen to indicate, that the request is being processed.

At this point, some errors can occur: 1) the HERE Maps API has rejected all requests to obtain city distances (technical issues) or 2) there has been an error during the calculation. The first error happens if some requests are sent too fast after one another. CORS (Cross-Origin Resource Sharing) might block the request. This is tried to counteract through a small waiting time between sending requests and a maximum number of retries of the same request. The second error can happen when user data was not inserted correctly or when the calculation did not deliver a result (= undefined).

If the route and participants could be determined successfully, following data can be found on the final page (after the loading screen disappears) (see. figure 3.4).

- Cities part of the tour
- Total distance
- Total costs
- Participants of the tour
- The participants' charged payments
- Interactive map (HERE Maps API) displaying all cities an the shortest path (route) to navigate
- Button to show/hide the debug log

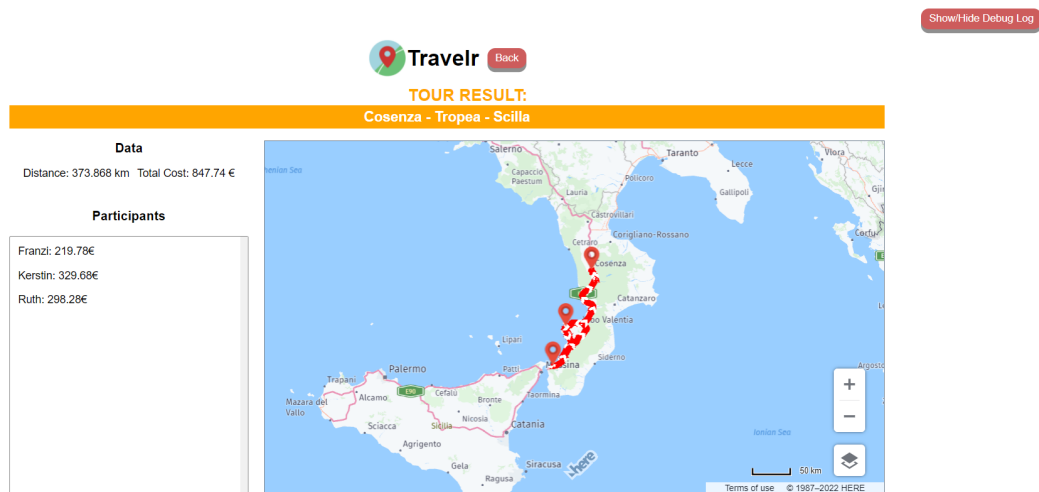


Figure 3.4: The final result will be shown with a list of participants and their payments and a map, displaying the shortest path of the tour and their waypoints.

On the upper left of the screen, a specific button can be found. It shows or hides the debug output that has been built during the calculation. This debug log is supposed to show every important step of the algorithm/mechanism. Only relevant data will be displayed there (see figure 3.5).

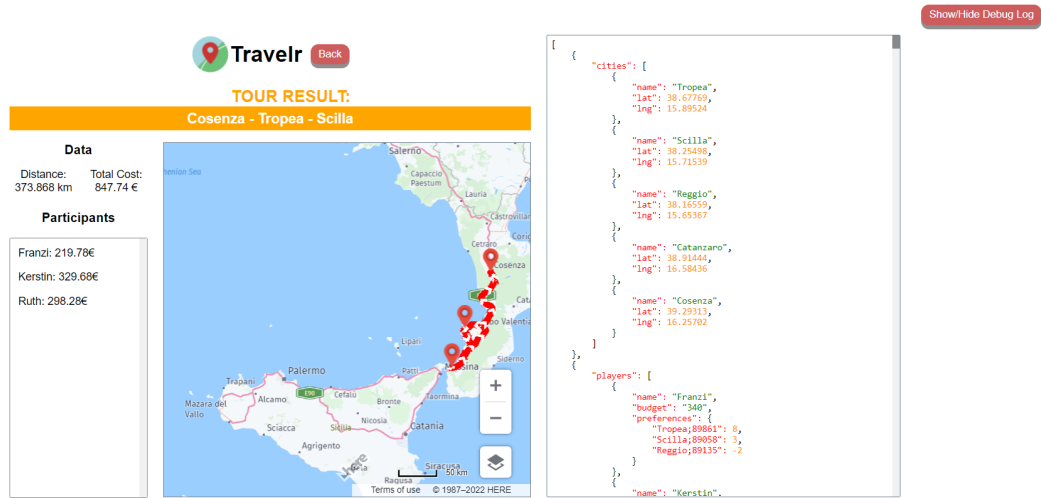


Figure 3.5: The button on the upper right triggers the debug log, which shows relevant information of each step in the calculation.

3.3 Mechanism implementation

In this section the implementation will be described in more detail. However, only relevant parts will be discussed. The complete application can be found on GitHub (<https://github.com/Franziska279/agt>).

It is important to note, that this implementation can result in different outcomes than the theoretical mechanism. This is due to computational limits and complexity. The difference can be found inside the payment calculation derived from VCG: if the route that the other players would choose without the current one is not affordable (and will be detected by the mechanism), this will not be considered by the application's algorithm.

To preprocess, manage and apply the data for the mechanism, the implementation includes several JavaScript files. The following list describes concisely these files' usage.

- *manageParticipants.js* is the file that deals with the insert form on the modify page and adds buttons to delete each entry.
- *sendData.js* gathers relevant entered data of each input element on the modify page and creates a json that will be sent to the next page (compute).
- *processData.js* preprocesses some data and forwards it to calculate the result. Once the result returns, it will be displayed on the compute page and the map will be called. It also has some error handling (explained in section 3.2).
- *calculateResult.js* is the file where the work is done to get a suitable tour result. It reflects the

prior explained mechanism and builds debug objects during the workflow:

- Get all possible player (maximum k) and city combinations
 - Filter city combinations by given maximum length (km)
 - Calculate costs of each city combination
 - Get combinations of all player and city combinations and their utilities
 - Calculate player payments
 - Filter combinations by budget (if one player cannot afford a specific combination, it will be dropped)
 - Gather results and sort by maximum utility
 - Return best tour (first entry of sorted list)
- *handleDebugLog.js* just sets the debug log to show or hide.
 - *map.js* offers functions to assemble a map for the display of the best tour. One of the functions gets the shortest distance between given cities. Another one retrieves the coordinates of a given city with postal code. One of the most important ones adds the shortest path as a route with markings for the cities to the map.

The last mentioned JavaScript file implements the HERE Maps API. Although it seems to be a widely applied API, its documentation lacks in some points. For instance, to obtain an interactive map, the section in the documentation about restrictions of the map movement showed how to implement this feature. Another issue arises when there are automatic requests made to the servers. As already explained, when sending too many requests without some time inbetween, they will be rejected, due to CORS policy (even with a CORS header set for the request). For all interactions with the API, a key is needed, which can be obtained through a free account. This is also a major reason, why HERE Maps API and not Google Maps API was chosen.

3.4 Known issues

“Travelr” is a simple web application to show that our mechanism can be implemented to fill the gap on the market. However, to be able to do so, there are several important and security relevant features missing.

Until now, the user input has to follow strict syntactic guidelines. There is no validation of wrong or harmful input. The application works once started. If there is some downtime, the data cannot be persisted, since it has no database. In order to be used by several people at the same time, some other software design issues have to be considered, like distribution and security (of more than just the user

input).

As already mentioned in section 3.2, another error is highly possible. The error from HERE Maps API, regarding CORS, could not be solved during development of this project.

Lastly, the different implementation of the payment distribution has to be mentioned. As already explained, the calculation of the first variable in the from VCG adapted formula is not the same as in the theoretical mechanism. This is due to computational complexity and runtime issues.

Altogether, for the purpose of this project, these known issues can be neglected or have a temporary workaround. This project serves as a first working implementation of the mechanism. Refinements and dealing with these issues can be done in future development.

3.5 Example

This example correlates to the one showed in section 2.4 for the theoretical implementation of the mechanism. Using these parameters, the web application's computation was executed. This resulted in the insertions of parameters, as seen in figure 3.6.


Agents A	Budgets b		$u(\text{Crotone})$	$u(\text{Cirella})$	$u(\text{Siderno})$	$u(\text{Tropea})$
Alice	24€		-2	5	4	1
Bob	34€		6	4	-9	2
Charlie	24€		-1	3	0	3

Table 3.1: Agents, budgets and utilities - Replication of the parameters used for the example of the mechanism

With these parameters, an issue arose. Since HERE Maps API's geocoding feature does recognise if a postal code is wrong, it incorrectly changes the postal code for Diamante. Therefore, the longitude and latitude returned from the API do not point to the city Diamante in Italy, but to Diamante in Uruguay, South America. Therefore, no route can be found that can be used by the car. To counteract this issue, Cirella was taken instead of Diamante, which is very close.

Then the calculation was started. It returned that Alice and Charlie would take part in the tour, consisting of Cirella and Tropea (see 3.7). In theory, however, Alice and Charlie would only visit Diamante. This is due to the little change for the implementation's algorithm, as explained in chapter 3.

Therefore, they have different ρ values in the computation of the payment. To compare the values, the computation procedure of the value is taken from the debug log, which is included in the application. In the example, for the ρ_{Alice} , the implementation computed $6 \times 2 - 6$, whereas the theoretical approach provided $3 \times 2 - 6$. Again, this is because the first part of the adapted VCG is calculated without



Here you can modify your routing request!

Start/End:	Max. KM:	Max. Seats:	Fixed Cost:	KM Cost:
Rende	420	2	10	0.1
87036				

Insert a participant:

Name:

Budget:

Preferences:

Crotone (88900): -1,
Cirella (87032): 3,
Siderno (89048): 0,
Tropea (89861): 3

Active Participants:

Alice (24€) - {Crotone (88900): -2, Cirella (87032): 5, Siderno (89048): 4, Tropea (89861): 1} X

Bob (34€) - {Crotone (88900): 6, Cirella (87032): 4, Siderno (89048): -9, Tropea (89861): 2} X

Charlie (24€) - {Crotone (88900): -1, Cirella (87032): 3, Siderno (89048): 0, Tropea (89861): 3} X

Insert
Clear

COMPUTE

Figure 3.6: Modification page with all parameters set

considering if the alternative route, that the other players would choose without the current player, can be afforded by the other players. Hence, also the payments are calculated differently. Alice and Charlie have to pay the same (23.09) for this example, whereas for the example by the mechanism, they have diverse payments.

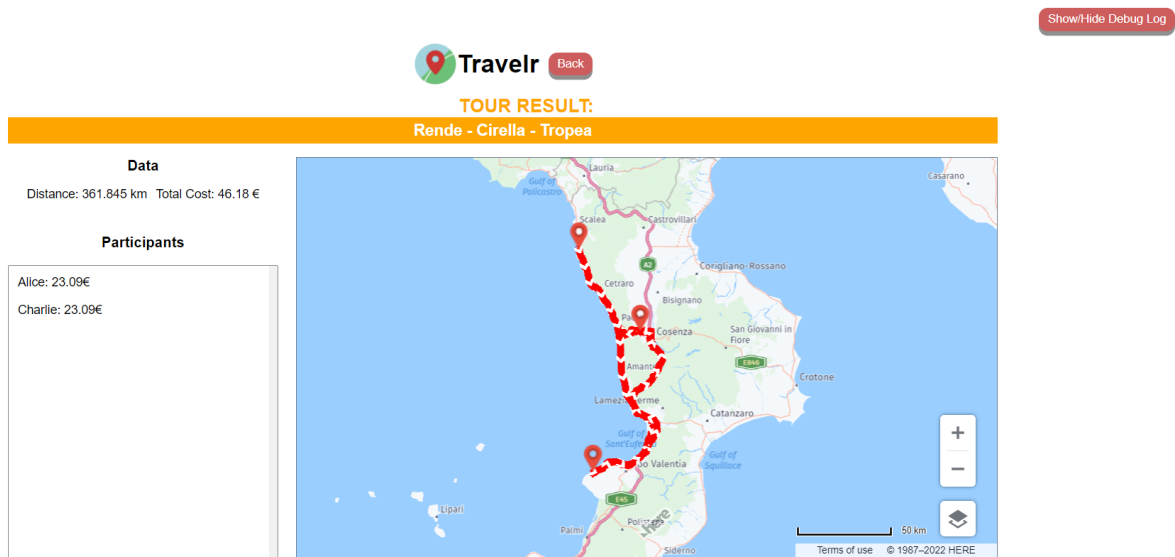


Figure 3.7: The result calculated by the application. Alice and Charlie would go on a trip to Cirella and Tropea.

```
{
  "id": "CirellaTropea-AliceCharlie",
  "Alice": {
    "grooves": {
      "value": 6,
      "calculation": [ "6_ *_ _2_ - _6" ]
    },
    "payment": {
      "budget": 24,
      "calculation": "(6_/_12)_ *_ _46.18",
      "value": 23.09
    }
  },
  "Charlie": {
    "grooves": {
      "value": 6,
      "calculation": [ "6_ *_ _2_ - _6" ]
    },
    "payment": {
      "budget": 24,
      "calculation": "(6_/_12)_ *_ _46.18",
      "value": 23.09
    }
  },
  "maxGrooves": 12,
  "tourCost": "46.18"
}
```

Figure 3.8: Debug log from web application's calculation with different Groove's values

4 Conclusion

The “Travelr” project, which has now been explained in detail, has required a deeper look into many aspects of algorithmic game theory. It needed an integral understanding and application of game theory concepts. To recap, this paper has tried to solve the scenario in which many friends want to visit a number of cities together, according to the definitions of fairness, truthfulness and achieving the greatest social welfare. These requirements demanded that a mechanism had to be found that distributes the costs of the tour among the participants in a fair way, so that no one has a reason to object. Furthermore, the costs are distributed in such a way that players are forced to tell the truth about their utilities for the cities. Therefore, for the players, telling the truth must always be the dominant strategy. In the end, the mechanism selects the tour and the players who, compared to all other options, obtain the highest benefit, i.e. the highest total utility. Finally, this mechanism was implemented as a web application. Although some issues occurred, it is an educational approach to figure out, if the mechanism could be realised as an application.

In summary, the project can be described as very challenging, as the development of the mechanism involved some difficulties that were not easy to solve. Nevertheless, the results of the “Travelr” project can be considered a success, as both, an effective mechanism and a novel implementation for problems of this type of scenario, have been created.

5 Bibliography

- [1] In: SHOHAM, Y. ; LEYTON-BROWN, K. : <http://www.masfoundations.org/index.html>