



27008537 ALGORITHMIC GAME THEORY

Exam Project

231847 Guarasci Paola

231848 Murano Francesca



The assignment

The problem assigned to us asks for the construction of a tour between predefined locations (or a subset of them) and a selection of users, among all those participating in the game, who will make this tour.

The **constraints** given concern the maximum length in terms of km of the chosen route and the maximum number of participants in the tour.

The mechanism must also establish the payments for each participant, which in any case may not exceed their respective budgets.



Outline

From the theory

We considered Bayesian games, a game in which players have incomplete information on the other players' strategies and payoffs, but, they have beliefs with known probabilities.

Application

Python implementation of the mechanism and simulation output

Bayesian Game Settings

Quasilinear mechanism

VCG Mechanisms

Problem Resolution

From the theory


The setting is quasilinear because agents have quasilinear preferences with transferable utility

From the theory

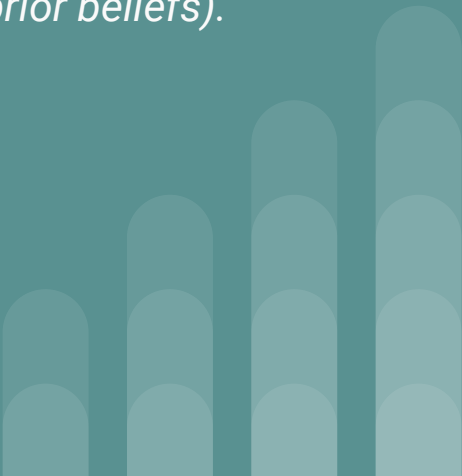
The VCG mechanism is a general way for self-interested agents to choose a social-welfare maximizing outcome. It works in quasilinear utility settings.



Bayesian Game Settings



Bayesian Game (Incomplete Information Game) - a game in which players have incomplete information on the other players' strategies and payoffs, but, they have beliefs with known probabilities. It can be modeled as a normal form game with the difference that each player has multiple types with known probabilities (called a common prior beliefs).





Quasilinear mechanism



A mechanism in the quasilinear setting is a triple (A, χ, P) , where:

- $A = A_1 \times \cdots \times A_n$, where A_i is the set of actions available to agent $i \in N$;
- $x : A \rightarrow \Pi(X)$ maps each action profile to a distribution over choices;
- $\mathcal{P} : A \rightarrow R^n$ maps each action profile to a payment for each agent.

The use of a **quasilinear** mechanism came about because its **strengths**:

- the mechanism can choose to charge or reward agents with an **arbitrary monetary amount**, so our **users can choose their own budget**;
- an agent's preference for its own choice is **independent of the amount**.

So our users will choose the destination of the tour just for **pleasure** without thinking about the amount needed. This feature is really important as it leads users to focus only on their own interests without seeking strategies based on others' choices.

They have no need to lie and always proclaim the **truth**.

A **direct quasilinear mechanism** is a pair (χ, θ) . It defines a mechanism in the quasilinear setting, where for each i , $A_i = \Theta_i$.

The **set of actions** available to each player **is just the set of possible preferences of the player**.



Vickrey-Clarke-Groves (VCG) mechanism



$$\chi(\hat{v}) = \arg \max_x \sum_i \hat{v}_i(x),$$

$$\wp_i(\hat{v}) = \sum_{j \neq i} \hat{v}_j(\chi(\hat{v}_{-i})) - \sum_{j \neq i} \hat{v}_j(\chi(\hat{v})).$$

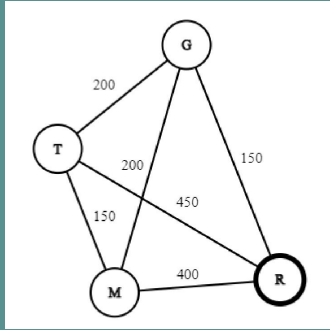
For the problem resolution, the choice fell on a mechanism belonging to the Groves mechanisms. Groves mechanisms belong to the direct quasilinear mechanisms (χ_{θ}). The dominant strategy of the mechanism is **truth-telling**, this mechanism allows us to make efficient choices.

The mechanisms is incentive compatible (truthful or strategy-proof) because is a direct mechanism where declaring true type for every agent is a weakly-dominant strategy **Nash equilibrium**. Every agent fare best or at least not worse by being truthful, regardless of what the others do.

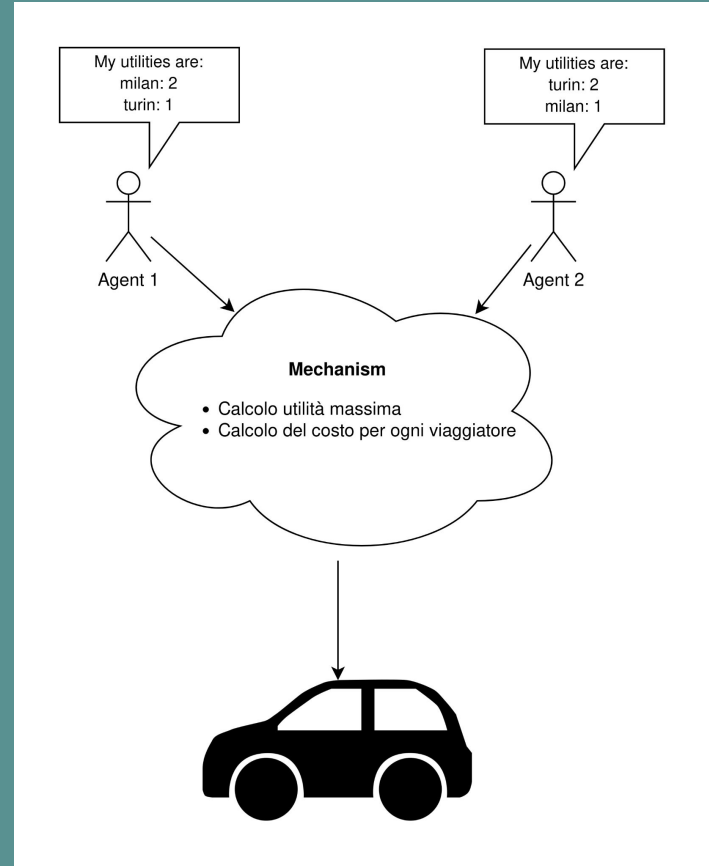


Problem resolution





How we solved the problem



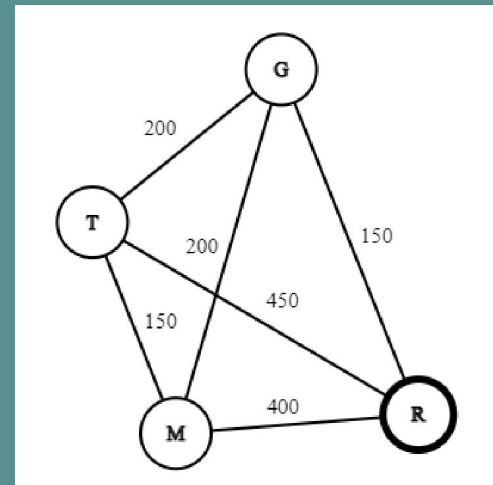


Numerical Example



Data

- $P = \{\text{Paola, Francesca}\}$
- $k = 2$
- $B_{\text{Paola}} = 500 \text{ €}$
- $B_{\text{Francesca}} = 500 \text{ €}$
- $L = \{\text{Rome, Genoa, Milan, Turin}\}$
- $DL = \{$
 - Rome-Turin: 670km,
 - Rome-Milan: 346km,
 - Rome-Genoa: 500km,
 - Milan-Turin: 143km,
 - Milan-Genoa: 142km,
 - Turin-Genoa: 172km $\}$
- $LS = \text{Rome}$
- $PU = \{$
 - Paola = $[M=0, G=1, T=2]$,
 - Francesca = $[M=0, G=2, T=1]$, $\}$
- $MAX_L = 1600 \text{ km}$



1. Generation of all possible combination without repetition (who started from Rome).

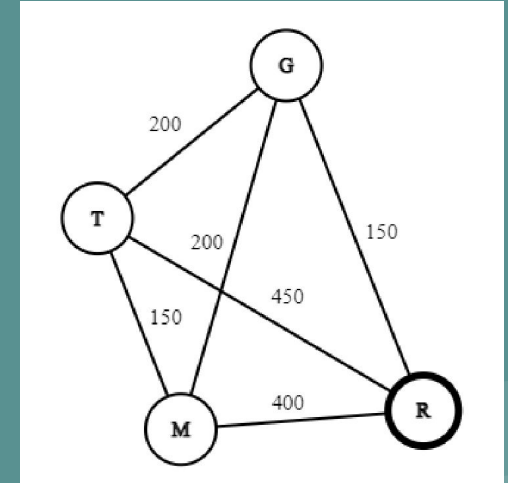
TP = [RGMT, RG, RM, RT, RMG, RGT, RMT]

2. Result of Salesman problem:

TP = [RGMT, RG, RM, RT, RGM, RGT, RMT]

3. Calculate path's distance and eliminate that do not respect the constraint;

- $RT = 670 \text{ km} * 2 = 1340 \text{ km}$,
- $RM = 346 \text{ km} * 2 = 692 \text{ km}$,
- $RG = 500 \text{ km} * 2 = 1000 \text{ km}$,
- $RGT = [(RG) 500 \text{ km} + (GT) 172 \text{ km}] * 2 = 672 \text{ km} * 2 = 1344 \text{ km}$
- $RGM = [(RG) 500 \text{ km} + (GM) 142 \text{ km}] * 2 = 642 \text{ km} * 2 = 1284 \text{ km}$
- $RMT = [(RG) 500 \text{ km} + (MT) 143 \text{ km}] * 2 = 643 \text{ km} * 2 = 1286 \text{ km}$
- $RGMT = [(RG) 500 \text{ km} + (GM) 142 \text{ km} + (MT) 143 \text{ km}] * 2 = 785 \text{ km} * 2 = 1570 \text{ km}$



4. Calculate utilities in descending order

- RGT = 6
- RT = 3
- RG = 3
- RGM = 3
- RMT = 3
- RM = 0

5. We select the first with maximum utilities: RGT.

6.
$$\chi(\hat{v}) = \arg \max_x \sum_i^n v_i(x);$$

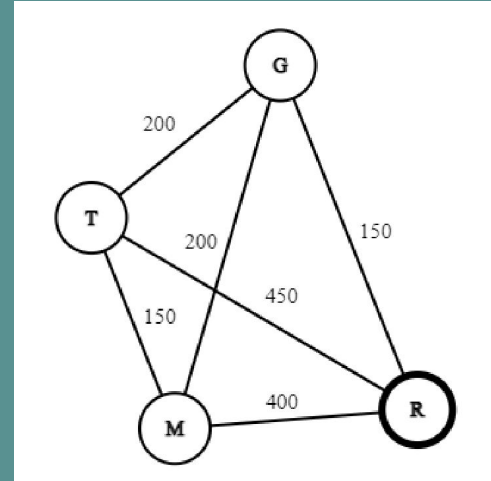
Travel considered = [R G T],

Paola competes for **GT** and Francesca for **RG**.

Each agent represents the **arc** of the major preference in the trip.

G T cities must be crossed, and I am obliged in doing so.

If an agent (arc) is missed, an alternative route (if possible) is sought that crosses the two cities and minimizes costs [Costs are the weight on the arches].



$$7. \quad \wp_i(\hat{v}) = \sum_{i \neq j}^n \hat{v}_j (\chi(\hat{v}_{-i})) - \sum_{i \neq j}^n \hat{v}_j (\chi(\hat{v}));$$

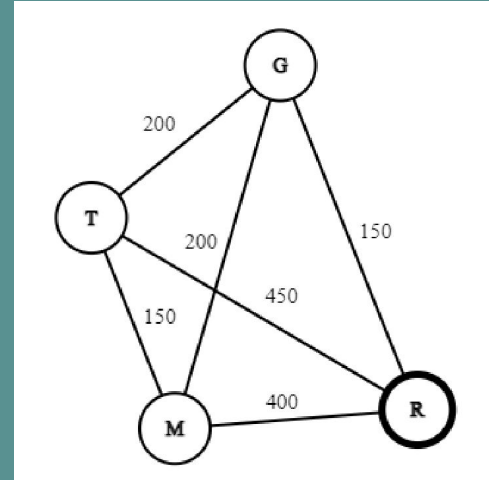
Brief Formula explication = Cost of the alternative tour without player's participation (and consequently without his arch) - Cost of the tour with player's participation but without your financial contribution.

- $CP = (RGMT) - (RGT - GT) =$
 $= 500 \text{ €} - (350 \text{ €} - 200 \text{ €}) = 500 \text{ €} - 150 \text{ €} = 350 \text{ €} < BPaola.$
- $CF = RMGT - (RGT - RG) =$
 $600 \text{ €} - (350 \text{ €} - 150 \text{ €}) = 600 \text{ €} - 200 \text{ €} = 400 \text{ €} < BFrancesca.$

★ $CT = CP + CF = 350 \text{ €} + 400 \text{ €} = 750 \text{ €}.$

8. The budgets were respected however, there is a surplus that we are going to divide according to the kilometers of the destinations that were competed for.

- $\text{Surplus} = (\text{Total Cost}) 750 \text{ €} - (RGT) 350 \text{ €} = 400 \text{ €}$
- $C \text{ per Km} = \text{Surplus} / DLi = 400 \text{ €} / 672 \text{ km} (RG + GT) = 0.59 \text{ €} / \text{km}$
- $SF = C \text{ per Km} * RG \text{ km} = 0.59 \text{ €} * 500 \text{ km} = 296 \text{ €}$
- $SP = C \text{ per Km} * GT = 0.59 \text{ €} * 172 \text{ km} = 104 \text{ €}$
- $CP - \text{Surplus} = 350 \text{ €} - 296 \text{ €} = 54 \text{ €}$
- $CF - \text{Surplus} = 400 \text{ €} - 104 \text{ €} = 296 \text{ €}$





Code Example



```

69 def calculateAgentPayment(selectedTour, agent):
70     agentInterest = agent.getCityMaxUtility(selectedTour)
71     cityAlternative = []
72
73     for city in selectedTour:
74         if city not in agentInterest and city != const.STARTPLACE:
75             cityAlternative.append(city)
76
77     agentInterest.append(const.STARTPLACE)
78     cityAlternative.insert(0, const.STARTPLACE)
79     cityAlternative.append(const.STARTPLACE)
80     if len(cityAlternative) > 1 and len(agentInterest) > 1:
81         _, costoPercorsoAlternativo = cityNetMoney.findAlternative(cityAlternative)
82         _, costoTourScelto = cityNetMoney.findShortestPathBetweenAllSelectedLocations(selectedTour)
83         _, costoTrattaInteresse = cityNetMoney.findAlternativeAB(selectedTour, agentInterest[0][0], agentInterest[0][1])
84         _, distanzaKmTrattaInteresse = cityNetKm.findAlternativeAB(selectedTour, agentInterest[0][0], agentInterest[0][1])
85         agent.kmTrattaInteresse = distanzaKmTrattaInteresse
86         costo = costoPercorsoAlternativo - (costoTourScelto - costoTrattaInteresse)
87         return (costo, agent.budget < costo)
88     return (0, True)

```

Computation of costs per agent