SAINT PETERSBURG STATE UNIVERSITY

Khaled Alkhaled

Game theoretic approach to transportation problems on networks

Master thesis

Direction 01.04.02 « Applied mathematics and computer science »

Scientific supervisor:

Professor, Doctor Leon Petrosjan

Saint Petersburg

2022

Contents

Abstract.

In this thesis, we consider a network game in which n-player want to reach the fixed node with minimal time (cost). It is assumed that the trajectories of players should (have no common arcs, have no common vertices) i.e. must not intersect. The last condition complicates the problem, since the sets of strategies turn out to be mutually dependent. A family of Nash equilibrium is constructed and it is also shown that the minimum total time (cost) of players is achieved in a strategy profile that is a Nash equilibrium. A cooperative approach to solving the problem is proposed. Also, another cooperative mini maximal approach to solving the problem is proposed. Then we consider the proportional solution and the Shapley value to allocate total minimal cost between players. Two approaches for constructing the characteristic function have been developed. In both cases, to construct the characteristic function, approaches are used that were previously proposed for constructing the Nash equilibrium. Then we consider players are coalitions and discuss (time consistency problem).

Introduction

Theory games on networks have been growing in recent research. (Mazalov and Chirkova (2019) [2]) provided a comprehensive disquisition on the topic. Given that most practical game situations are more dynamic (intertemporal) rather than static, dynamic network games have become a field that attracts theoretical and technical developments. One special case of network games is transportation game. The was considered in the articles by (Petrosyan 2011.[9]) and by (Seryakov 2012.[3]) about the game theoretic transportation model in the network. In these articles [9] and [3] a game theoretic approach is considered for n-player which want to reach the fixed node of the network with minimal time (cost). It is assumed that the trajectories of the players should (have not common arcs) i.e. must not intersect. The last condition significantly complicates the problem, since the sets of strategies turn out to be mutually dependent. A family of Nash equilibrium is constructed and it is also shown that the minimum total time (cost) of players is achieved in a strategy profile is a Nash equilibrium. A cooperative approach to solving the problem is proposed.

We consider the same game theoretic approach (Petrosyan [9]) and suggest another cooperative mini maximal approach to solving the problem is proposed. Several algorithms from the book (Ferreira 2014 [1]) had been modified to calculate for n-player Nash equilibrium (cooperative, non-cooperative) and cooperative mini maximal under condition the trajectories of the players should have no common arcs.

Then we consider the same game theoretic approach (Petrosyan [9]), but under new condition the trajectories of the players should have no common vertices i.e. must not intersect. The last condition complicates the problem, since as in previous case the sets of strategies turn out to be mutually dependent. A family of Nash equilibrium is constructed and it is also shown that the minimum total time (cost) of players is achieved in a strategy profile that is a Nash equilibrium. A cooperative approach to solving the problem is proposed. And suggest another cooperative mini maximal approach to solving the problem is proposed. Several algorithms from the book (Ferreira 2014 [1]) had been modified to calculate for n-player Nash equilibrium (non-cooperative) and cooperative mini maximal under condition the trajectories of the players should have no common vertices.

Coordinating players in a network to minimize their joint cost and distribute the cooperative gains in a dynamically stable solution is a topic of ongoing research. The Shapley 1953.[16] value is credited to be one of the best solutions in attributing a fair gain to each player in a complex situation like a network. However, the determination of the cost of the subsets of players (characteristic function) in the Shapley value is not indisputably unique.

We consider cooperative game theoretic transportation model in the network . Then consider the proportional solution([17]) and The Shapley value [16] to allocate total minimal the cost between players. In n-player case.Two approaches for constructing the characteristic function have been proposed. In both cases, to construct the characteristic function, approaches are used that were previously proposed for constructing the Nash equilibrium.

The concept of time consistency and its implementation was initially proposed in (Petrosyan, 1977,[4]), (Petrosyan and Danilov, 1979,[5]). Some new results about time consistency can be found in (Petrosyan and Zaccour, 2003,[6]), (Yeung and Petrosyan, 2005,[7]), and (Gao et al., 2014,[8]). It shown on example that the characteristic function is not time consistent in game theoretic transportation model in the network (Petrosyan 2011.[9]).

Then consider new game theoretic transportation model in the network, where the players are coalitions under the condition the trajectories of the players (coalitions) should have no common arcs i.e. must not intersect. The trajectories of the players inside coalition can intersect (have common arcs). A family of Nash equilibrium is constructed and it is also shown that the minimum total cost of players (coalitions) is achieved in a strategy profile that is a Nash equilibrium. A cooperative approach to solving the problem is proposed. Then the proportional solution([17]) to allocate total minimal cost between coalitions are proposed and The Shapley value [16] to allocate the costs inside each coalition. Two approaches for constructing the characteristic function have been developed. In both cases, to construct the characteristic function, approaches are used that were previously proposed for constructing the Nash equilibrium. It shown on example that the characteristic function is not time consistent and the two stage solution concept in game is developed.

1 Minimization of transportation time in the case when paths have no common arcs.

1.1 Model

The game takes place on the network $G = (X, D)$, where X is a finite set, called the vertex set and $D-$ set of pairs of the form (y, z), where $y \in X$, $z \in X$, called arcs. Points $x \in X$ will be called vertices or nodes of the network. On a set of arcs D a non-negative symmetric real valued function is given $\gamma(x, y) = \gamma(y, x) \geq 0$, interpreted for each arc $(x, y) \in D$ as the time associated with the transition from x to y by arc (x, y).

1.2 Description of transportation game

Define n-player transportation game on network G. The transportation game Γ is system $\Gamma_1 = \langle G, N, x(N), a \rangle$, where $G-$ network,$N = \{1, \ldots, n\}-$ is set of players,$a \in X$ - some fixed node of the network $G, x(N) \subset X$ - subset of vertices of network $G, x(N) = \{1(x), 2(x), \ldots, i(x), \ldots, n(x)\}$, indicating the vertices in which players are located in $x(N)$ at the beginning of the game process (the initial position of the players). For example $i(x)$ means the vertex $x \in X$, in which the player i is located at the beginning of the game. The set $x(N)$ may contain coinciding vertices, i.e. at the beginning of the game, several players can be at the same vertex. In some cases, in order not to complicate the notation, so by $i(x)$ we will also mean the vertex in which the player i is located. On a path in the game Γ_1, any finite sequence of arcs of the form $h = \{(x_0, x_1), (x_1, x_2), \ldots, (x_{l-1}, x_l)\}$, under condition that the initial vertex in each arc coincides with the final vertex of previous arc is called a path. Also, we suppose that there is player $i \in N$, $x_0 = i(x_0) \in x(N)$ and $x_l = a$. Thus, a path is a sequence of arcs connecting the initial positions of the players in the network to fixed node a. We will say that the paths h' and h'' do not intersect and write $h' \cap h'' = \emptyset$, if they do not have common arcs .

1.3 Minimization of transportation time ($n-$player in game).

We have $n-$player located in initial positions (vertices) which want to reach the fixed node a in network in minimal time, in such way that the corresponding paths have not contain common arcs. Denote this game as Γ_1.

1.4 Strategies in Γ_1.

Strategies of player i in the game Γ_1 are the paths in which the starting vertex $x_0 = i\,(x_0)$, and the final vertex coincides with $a \in X$. Denote the strategy of player i as:

$h^i = \{(x_0, x_1), (x_1, x_2), \ldots, (x_k, x_{k+1}), \ldots, (x_{l-1}, a)\}$,

A bunch of of all strategies of player i will be denoted by $H^i = \{h^i\}, i = 1, \ldots, n$.

1.5 Admissible strategy profiles in Γ_1.

The admissible strategy profiles in the game in Γ_1(see[9]). Strategy profiles $h = (h^1, \ldots, h^n), h^1 \in H^1, \ldots, h^n \in H^n$ are called admissible if the paths h^j and h^k not intersect (not contain common arcs). $h^j \cap h^k = \emptyset, j \neq k$. The set of all admissible strategy profiles is denoted by H.

1.6 Cost Function in Γ_1

In this section we define for each arc (x_k, x_{k+1}) the values of cost function $\gamma_i(x_k, x_{k+1})$ as the time necessary to reach the node x_{k+1} from node x_k by player i. For each strategy profile $h = (h^1, \ldots, h^n) \in H$. Denote the player i time to reach the fixed node a as $K_i(h)$ (see[9]).

$$K_i(h) = \sum_{k=0}^{l-1} \gamma_i(x_k, x_{k+1}) = k\left(h^i\right) \tag{1}$$

Here $\{(x_0, x_1), (x_1, x_2), \ldots, (x_{l-1}, x_l)\} = h^i$. Thus, we see for player i, the time $K_i(h)$ depends on his strategy h^i and depends on the strategies of other players in that the strategy h^i (path of player i) should not intersect with the strategies of other players. Therefore, in some cases, when this will not lead to misunderstandings, we instead $K_i(h)$ will use the notation $k(h^i)$, meaning the player i time along the path h^i.

1.7 Nash equilibrium in game Γ_1.

In the game Γ_1 the strategy profile $\left(\bar{h} = \bar{h}^1, \ldots, \bar{h}^n\right)$is called a Nash equilibrium , if $K_i\left(\overline{h} \parallel h^i\right) \geq K_i\left(\overline{h}\right)$ holds for all admissible strategy profiles $\left(\overline{h} \parallel h^i\right) \in H$ and $i \in N$.

Let π be some permutation of numbers $1, \ldots, n, \pi = (i_1, \ldots, i_n)$. Consider an auxiliary transportation problem on the network G for player i_1. Find the path in the network G, minimizing the total time of player i_1 to move from vertex $i_1(x) \in x(N)$ to vertex $a \in X$. Denote the path that solves this problem $\bar{h}^{i_1}$.

$$k\left(\overline{h}^{i_1}\right) = \min_{h^{i_1} \in H^{i_1}} k\left(h^{i_1}\right) \tag{2}$$

Denote by $G \backslash \bar{h}^{i_1}$ a subnetwork not containing the path $\bar{h}^{i_1}$. Consider an auxiliary transportation problem for player i_2 on network $G \backslash \bar{h}^{i_1}$. Find the path in subnetwork $G \backslash \bar{h}^{i_1}$, minimize the player i_2 time to reach from vertex $i_2(x) \in x(N)$ to fixed node $a \in X$. Denote the path that solves this problem by $\bar{h}^{i_2}$.

$$k\left(\overline{h}^{i_2}\right) = \min_{h^{i_2} \in H^{i_2}} k\left(h^{i_2}\right). \tag{3}$$

Proceeding further in a similar way, we introduce into consideration the subnetworks of the network G, that do not contain the paths $\overline{h}^{i_1}, \ldots, \overline{h}^{i_{m-1}}$. Consider the auxiliary transportation problem of the player i_m on the network $G \setminus \cup_{l=1}^{m-1} \overline{h}^{i_l}$. Find the subnetwork $G \setminus \cup_{l=1}^{m-1} \overline{h}^{i_l}$, minimize the player i_m time where $i_m(x) \in x(N)$ and

$a \in X$. Denote the path that solves this problem by $\overline{h}^{i_m}$.

$$k\left(\overline{h}^{i_m}\right) = \min_{h^{i_m} \in H^{i_m}} k\left(h^{i_m}\right). \quad (4)$$

As a result, we get a sequence of paths $\overline{h}^{i_1}, \ldots, \overline{h}^{i_n}$, minimizing the total time of players $i_1, i_2, \ldots, i_m, \ldots, i_n$ on subnetworks:

$$G, G \setminus \overline{h}^{i_1}, \ldots, G \setminus \cup_{l=1}^{m-1} \overline{h}^{i_l}, \ldots, G \setminus \cup_{l=1}^{n-1} \overline{h}^{i_l}.$$

The sequence of paths $\overline{h}^{i_1}, \ldots, \overline{h}^{i_m}, \ldots, \overline{h}^{i_n}$ by construction consists of pairwise non-intersecting paths, and each of them $\overline{h}^{i_l} \in H^{i_l}$. Therefore the strategy profile

$$\left(\overline{h}^{i_1}, \ldots, \overline{h}^{i_m}, \ldots, \overline{h}^{i_n}\right) = \overline{h}\left(\pi\right) \in H$$

is admissible in Γ_1(see[9]).

1.8 Equilibrium strategy profile.

Theorem(see[3]): the strategy profile $\bar{h}(\pi) \in H$ is an equilibrium strategy profile in Γ_1 for any permutation π.

Proof : Consider the strategy profile. $\left[\bar{h}(\pi) \| h^{i_m}\right]$, where $h^{i_m} \neq \bar{h}^{i_m}$, $h^{i_m} \in H^{i_m}$, $\left[\bar{h}(\pi) \| h^{i_m}\right] \in H$. By construction $\bar{h}^{i_m}$ is determined from the condition

$$k\left(\bar{h}^{i_m}\right) = \min_{h^i m \in G \setminus U_{l=1}^{m-1} \bar{h}^{i_l}} k\left(h^{i_m}\right),$$

However, the strategy profile $\left[\bar{h}(\pi) \| h^{i_m}\right]$ is admissible (if $h^{i_m} \in G \setminus \bigcup_{l=1}^{m-1} \bar{h}^{i_l}$) and therefore $k\left(\bar{h}^{i_m}\right) \leq k\left(h^{i_m}\right) = K_{i_m}\left[\bar{h}(\pi) \| h^{i_m}\right]$, $k\left(\bar{h}^{i_m}\right) = K_{i_m}(\bar{h}(\pi))$, and $K_{i_m}[\bar{h}(\pi)] \leq K_{i_m}\left[\bar{h}(\pi) \| h^{i_m}\right]$ for all $\left[\bar{h}(\pi) \| h^{i_m}\right] \in H$, which proves the theorem.This theorem indicates a rich family of pure strategy equilibrium profiles in Γ_1 depending on permutation π. Thus, in Γ_1 we have at lest $n!$ equilibrium strategy profiles in pure strategies (if the initial states of players are different).

1.9 Best Nash equilibrium in Γ_1

The strategy profile $\bar{h}(\hat{\pi})$ is called a best equilibrium if (see[9])

$$\sum_{i=1}^{n} K_i(\bar{h}(\hat{\pi})) = \min_{\pi} \sum_{i=1}^{n} K_i(\bar{h}(\pi)) = W \tag{5}$$

1.10 Cooperative solution in game Γ_1

However, there are other Nash equilibrium in Γ_1. Consider the strategy profile $\bar{\bar{h}}$, solving the minimization problem (see[9])

$$\min_{h} \sum_{i=1}^{n} K_i(h) = \sum_{i=1}^{n} K_i(\bar{\bar{h}}) = V \tag{6}$$

We can simply show that $\bar{\bar{h}}$ is also a Nash equilibrium strategy profile. Because if one player change his strategy and other players do not change their strategies his time under this conditions will be more or equal of his time in case has not change his strategy. Consider the strategy profile $\left(\bar{\bar{h}} = \bar{\bar{h}}^1, \ldots, \bar{\bar{h}}^i, \ldots, \bar{\bar{h}}^n\right)$ if player i change his strategy, we get

$$\sum_{i=1}^{n} K_i(\bar{\bar{h}} \parallel h^i) \geq \sum_{i=1}^{n} K_i(\bar{\bar{h}})$$

,

$$K(\bar{\bar{h}}^1) + K(\bar{\bar{h}}^2) + \ldots + K(h^i) + \ldots + K(\bar{\bar{h}}^n) \geq K(\bar{\bar{h}}^1) + K(\bar{\bar{h}}^2) + \ldots + K(\bar{\bar{h}}^i) + \ldots + K(\bar{\bar{h}}^n)$$

so $K(h^i) \geq K(\bar{\bar{h}}^i)$. We call the strategy profile $\bar{\bar{h}}$ a cooperative equilibrium in Γ_1. In some cases $V = W$, (see the example).

1.11 Chart of the minimum time algorithm for one player in Γ_1

We use a modification of Dijkstra's algorithm, Dijkstra's algorithm is an algorithm that solves the problem of finding the minimum transportation time for one player from the initial position to reach the fixed node a (see[1]).

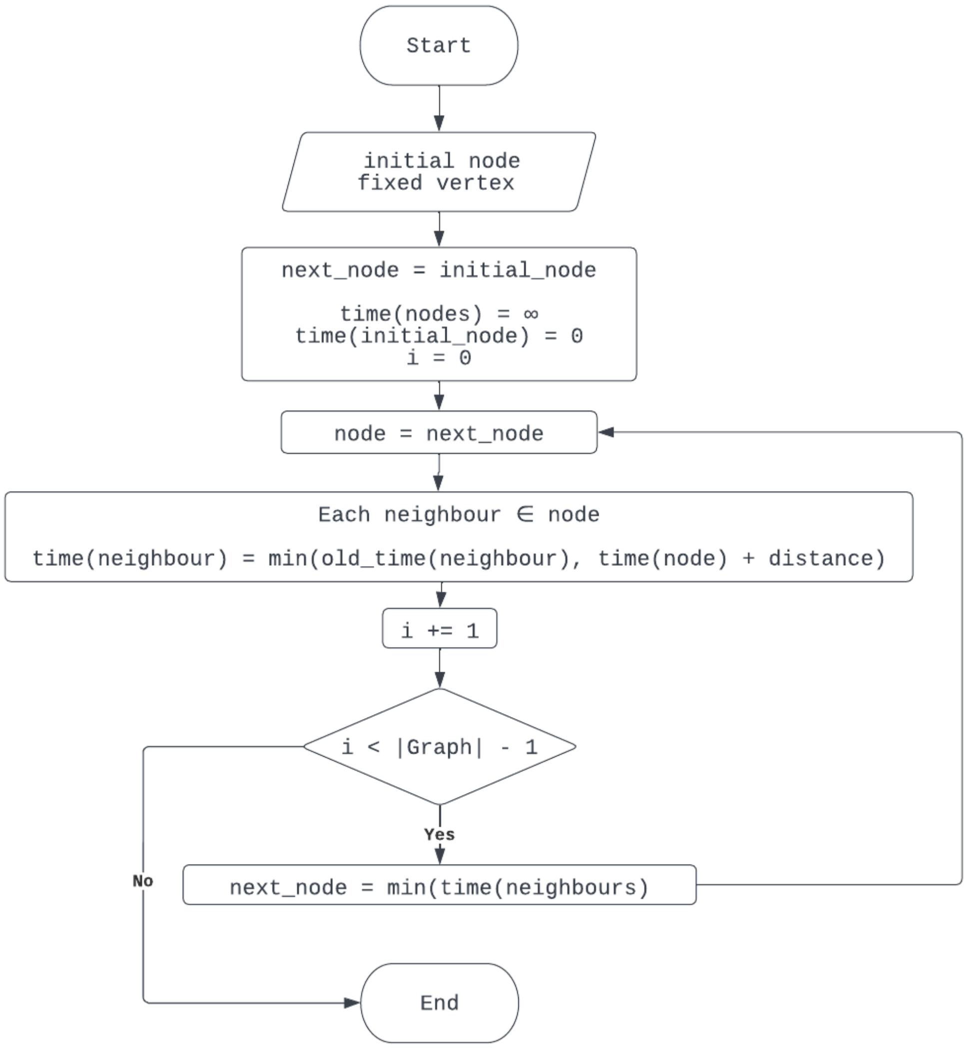


Figure 1:

1.12 Example for one player in Γ_1

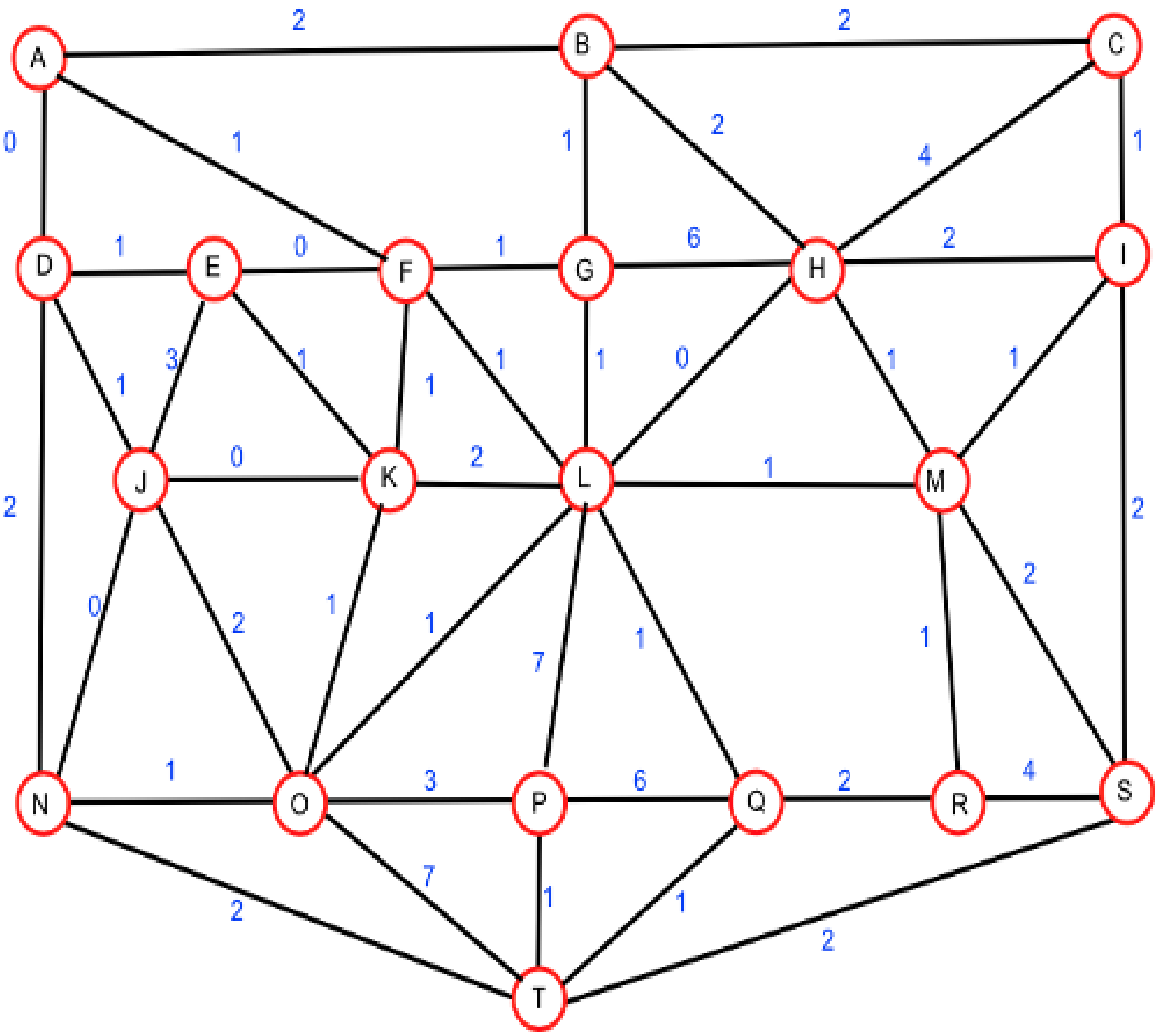


Figure 2: one player in game Γ_1

In this figure we denote nodes by capital Latin letters , $N = \{1\}$ the set $x(N) = \{A\}$.The transportation times are written on the network in this figure over the arcs and are equal, respectively to

$$\gamma(A,B)=2,\gamma(A,F)=1,\gamma(A,D)=0,\gamma(B,G)=1,$$

$$v(B,H)=2,\gamma(B,C)=2,\gamma(C,H)=4,\gamma(C,I)=1,$$

$$\gamma(D,N)=2,\gamma(D,E)=1,\gamma(D,J)=1,\gamma(E,F)=0,$$

$$\gamma(E,J)=3,\gamma(E,K)=1,\gamma(F,G)=1,\gamma(F,K)=1,$$

$$\gamma(F,L)=1,\gamma(G,H)=6,\gamma(G,L)=1,\gamma(H,I)=2,$$

$$\gamma(H,M)=1,\gamma(H,L)=0,\gamma(J,N)=0,\gamma(J,K)=0,$$

$$\gamma(J,O)=2,\gamma(K,L)=2,\gamma(K,O)=1,\gamma(L,M)=1,$$

$$\gamma(L,O)=1,\gamma(L,P)=7,\gamma(L,Q)=1,v(M,R)=1,$$

$$\gamma(M,S)=2,\gamma(M,I)=1,\gamma(I,S)=2,\gamma(N,T)=2,$$

$$\gamma(N,O)=1,\gamma(O,P)=3,\gamma(O,T)=7,\gamma(P,T)=1,$$

$$\gamma(P,Q)=6,\gamma Q,R)=2,\gamma(T,Q)=1,\gamma(T,S)=2,\gamma(S,R)=4.$$

We find the minimum transportation times from vertex A to all vertices. Making necessary computation, we get:

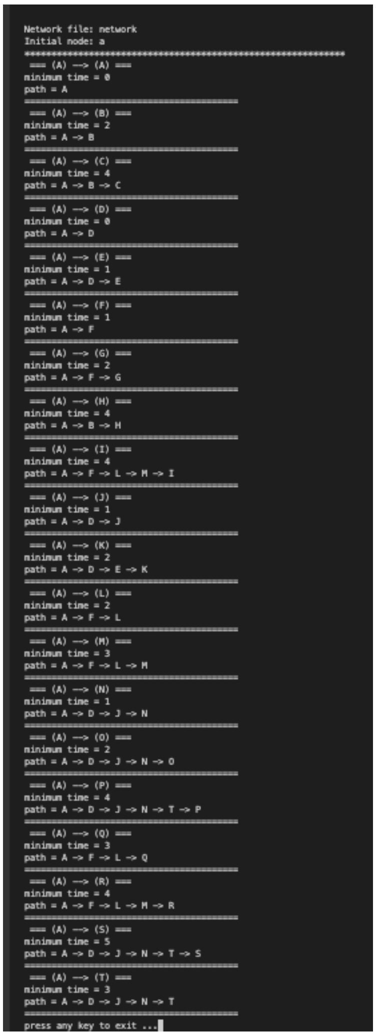


Figure 3:

1.13 Chart of the minimum time algorithm for $n-$ player case in Γ_1.

We developed Dijkstra's algorithm to find best Nash equilibrium for any network in $n-$player game Γ_1 and it is a following chart :

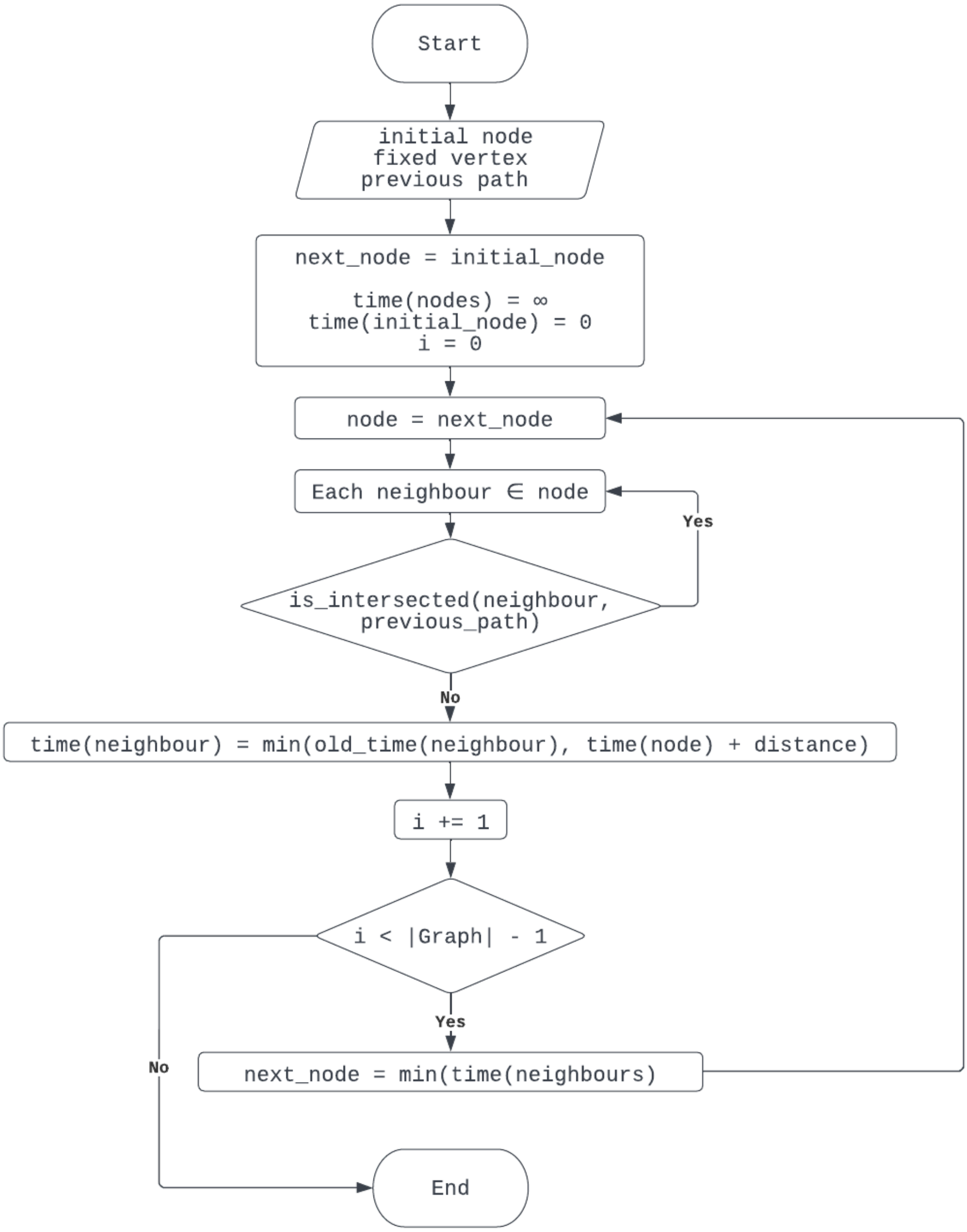


Figure 4: Best Nash equilibrium (arc) function

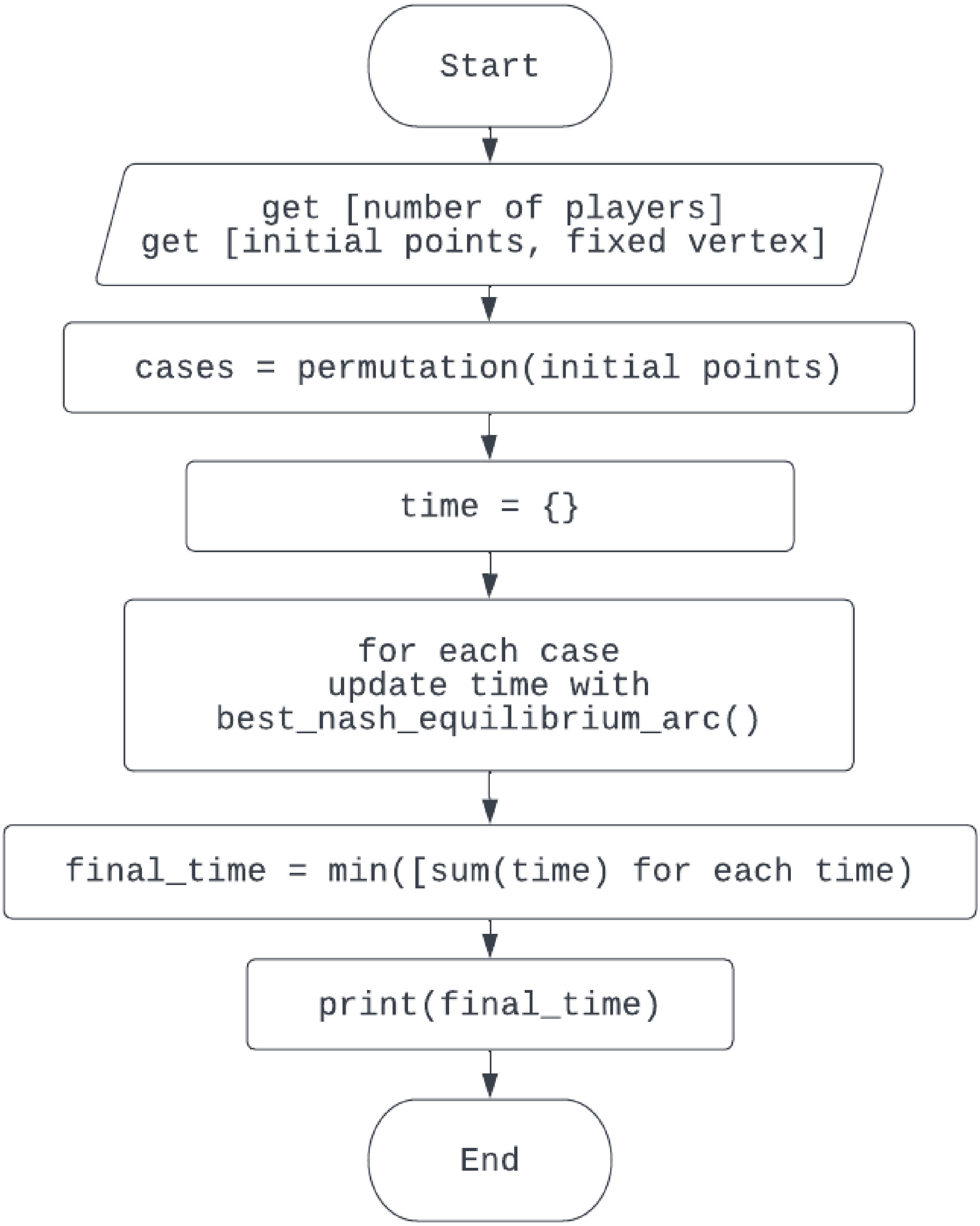


Figure 5: Best Nash equilibrium (arc)

We developed Dijkstra's algorithm to find cooperative solutions for any network in $n-$player game Γ_1 and it is a following chart :

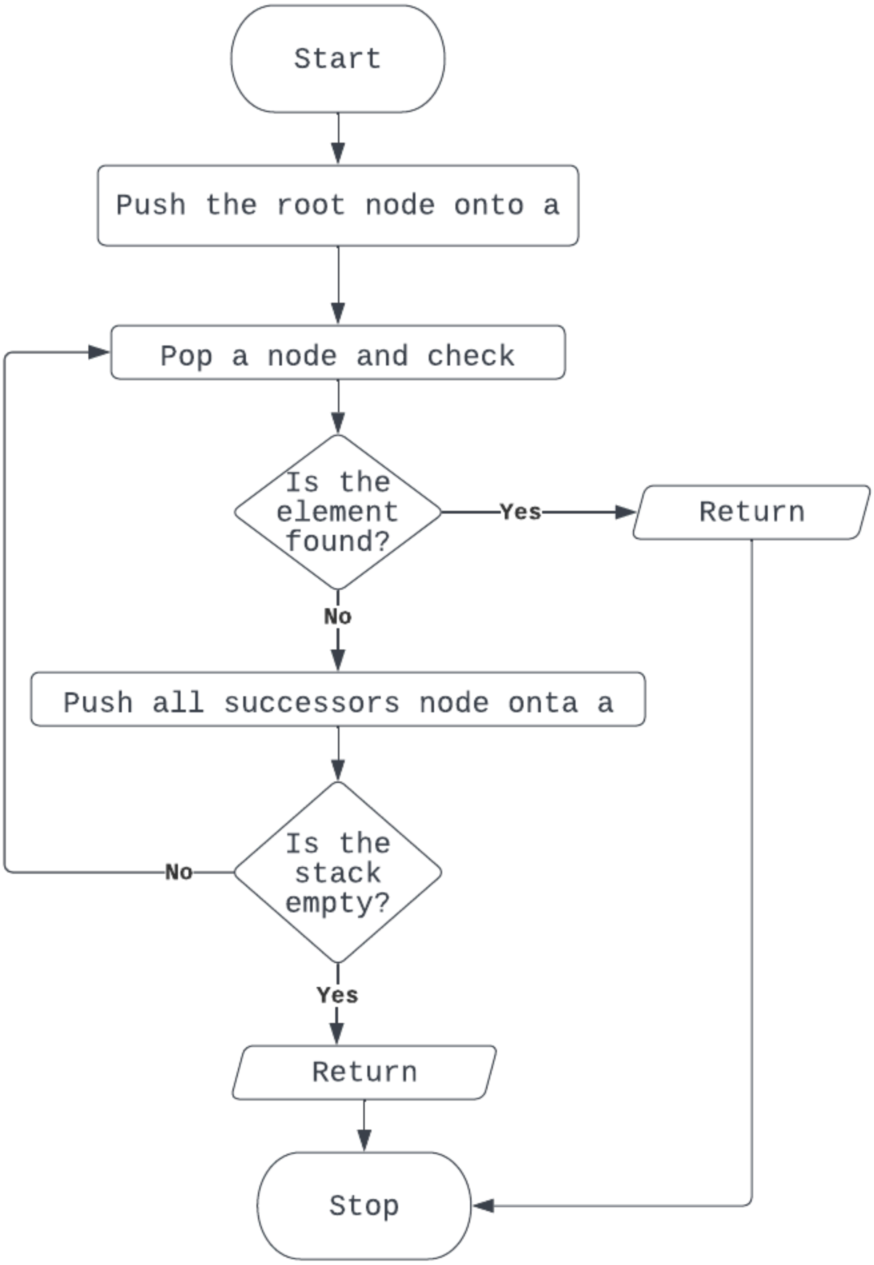


Figure 6: DFS

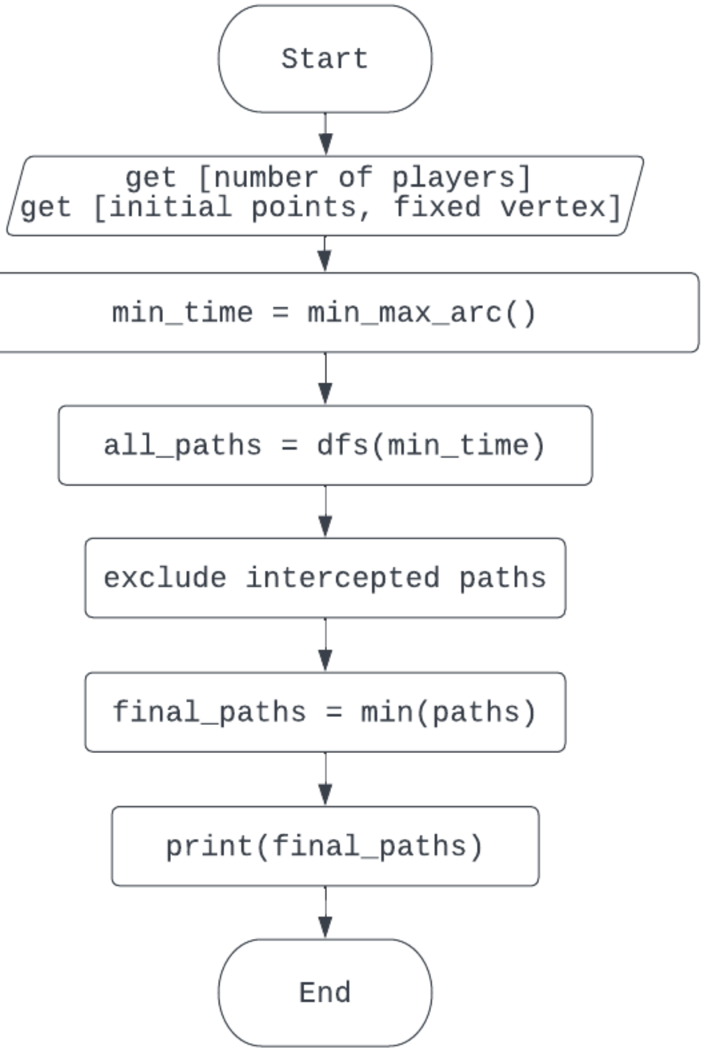


Figure 7: cooperative solution

1.14 Example, two player case in Γ_1

This example shows us best Nash equilibrium and give the same result (time) as cooperative solution

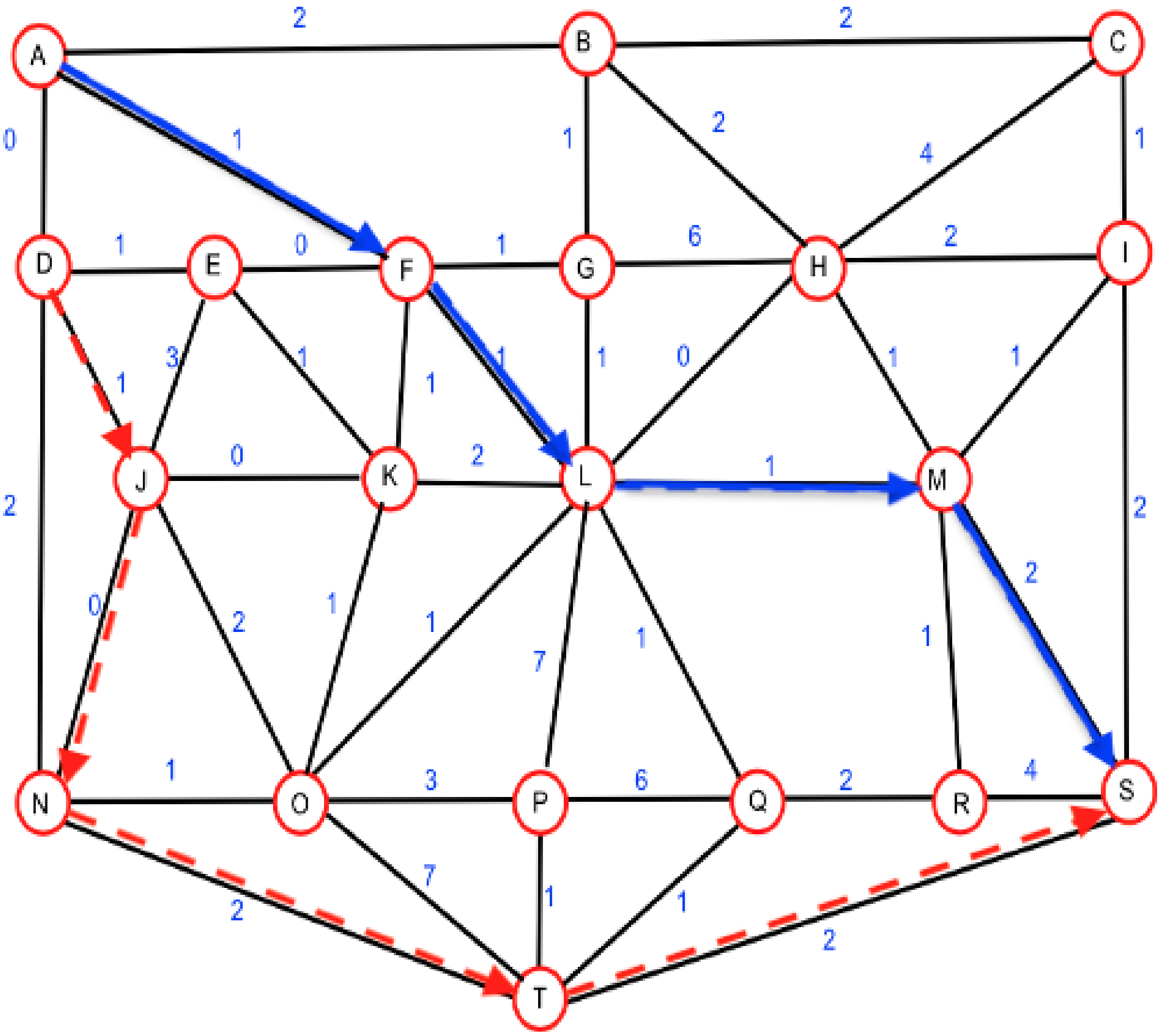


Figure 8: two player in game Γ_1

In this figure we denote nodes by capital Latin letters. $N = \{1, 2\}$ the set $x(N) = \{A, D\}$

The transportation times are written on the network in this figure over the arcs and are equal, respectively to

$$\gamma(A,B)=2,\gamma(A,F)=1,\gamma(A,D)=0,\gamma(B,G)=1,$$
$$\gamma(B,H)=2,\gamma(B,C)=2,\gamma(C,H)=4,\gamma(C,I)=1,$$
$$\gamma(D,N)=2,\gamma(D,E)=1,\gamma(D,J)=1,\gamma(E,F)=0,$$
$$\gamma(E,J)=3,\gamma(E,K)=1,\gamma(F,G)=1,\gamma(F,K)=1,$$
$$\gamma(F,L)=1,\gamma(G,H)=6,\gamma(G,L)=1,\gamma(H,I)=2,$$
$$\gamma(H,M)=1,\gamma(H,L)=0,\gamma(J,N)=0,\gamma(J,K)=0,$$
$$\gamma(J,O)=2,\gamma(K,L)=2,\gamma(K,O)=1,\gamma(L,M)=1,$$
$$\gamma(L,O)=1,\gamma(L,P)=7,\gamma(L,Q)=1,\gamma(M,R)=1,$$
$$\gamma(M,S)=2,\gamma(M,I)=1,\gamma(I,S)=2,\gamma(N,T)=2,$$
$$\gamma(N,O)=1,\gamma(O,P)=3,\gamma(O,T)=7,\gamma(P,T)=1,$$
$$\gamma(P,Q)=6,\gamma Q,R)=2,\gamma(T,Q)=1,\gamma(T,S)=2,\gamma(S,R)=4.$$

We find the minimal transportation time for two player A, D to reach the fixed node S under condition (paths have no common arcs). Making necessary computation, we get the best Nash equilibrium in this game.

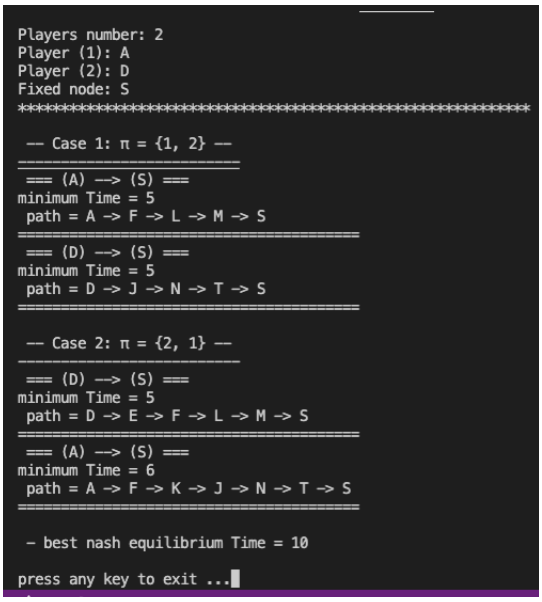


Figure 9: Here we get $V = 10$

Making necessary computation, we get the cooperative solutions in this game.

```
Network: network
Players number: 2
Player (1): A
Player (2): D
Fixed vertex: S
************************************************************
 === (A) ---> (S) ===
 minimum Time = 5
path = A -> D -> E -> F -> L -> H -> M -> S
========================================
 === (D) ---> (S) ===
 minimum Time = 5
path = D -> J -> N -> T -> S
========================================
------------------------- OR -------------------------
 === (A) ---> (S) ===
 minimum Time = 5
path = A -> F -> L -> M -> S
========================================
 === (D) ---> (S) ===
 minimum Time = 5
path = D -> J -> N -> T -> S
========================================
------------------------- OR -------------------------
 === (A) ---> (S) ===
 minimum Time = 5
path = A -> D -> J -> N -> T -> S
========================================
 === (D) ---> (S) ===
 minimum Time = 5
path = D -> E -> F -> L -> H -> M -> S
========================================
------------------------- OR -------------------------
 === (A) ---> (S) ===
 minimum Time = 5
path = A -> D -> E -> F -> L -> M -> S
========================================
 === (D) ---> (S) ===
 minimum Time = 5
path = D -> J -> N -> T -> S
========================================
------------------------- OR -------------------------
 === (A) ---> (S) ===
 minimum Time = 5
path = A -> F -> L -> H -> M -> S
========================================
 === (D) ---> (S) ===
 minimum Time = 5
path = D -> J -> N -> T -> S
========================================
------------------------- OR -------------------------
 === (A) ---> (S) ===
 minimum Time = 5
path = A -> D -> J -> N -> T -> S
========================================
 === (D) ---> (S) ===
 minimum Time = 5
path = D -> E -> F -> L -> M -> S
========================================

 - cooperative solution time = 10
```

Figure 10: $W = 10$ thus in this case $W = V$

1.15 Another example, two player case in Γ_1

This example show that best Nash equilibrium give us different result as cooperative solution and $V < W$.

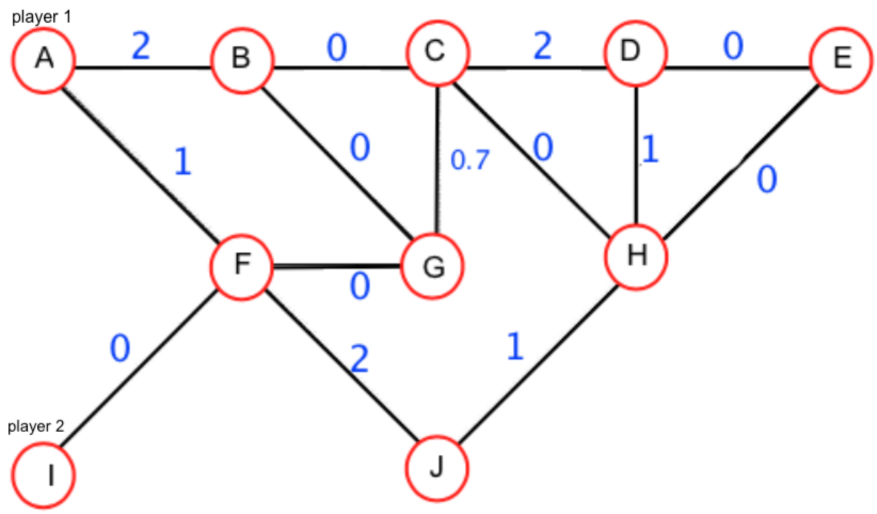


Figure 11: In this case : $V < W$

In this figure we denote nodes by capital Latin letters . $N = \{1, 2\}$ the set $x(N) = \{A, I\}$. Two player want to reach the fixed node E under condition (paths have no common arcs). The transportation times are written in the network in this figure over the arcs and are equal, respectively to

$$\gamma(A, B) = 2, \gamma(A, F) = 1, \gamma(B, C) = 0, \gamma(B, G) = 0,$$
$$\gamma(C, D) = 2, \gamma(C, H) = 0, \gamma(C, G) = 0.7, \gamma(D, E) = 0,$$
$$\gamma(D, H) = 1, \gamma(I, F) = 0, \gamma(F, G) = 0, \gamma(F, J) = 2,$$
$$\gamma(J, H) = 1, \gamma(H, E) = 0,$$

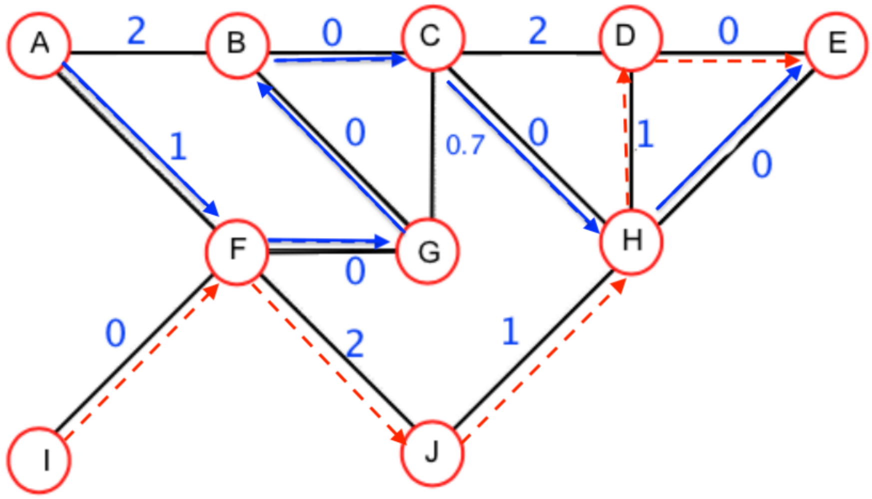


Figure 12: Best Nash equilibrium $\pi(1, 2)$

$K_1(\bar{h}(1, 2)) = 1$,$K_2(\bar{h}(1, 2)) = 4$

$K_1(\bar{h}(1, 2))$+$K_2(\bar{h}(1, 2)) = 5$

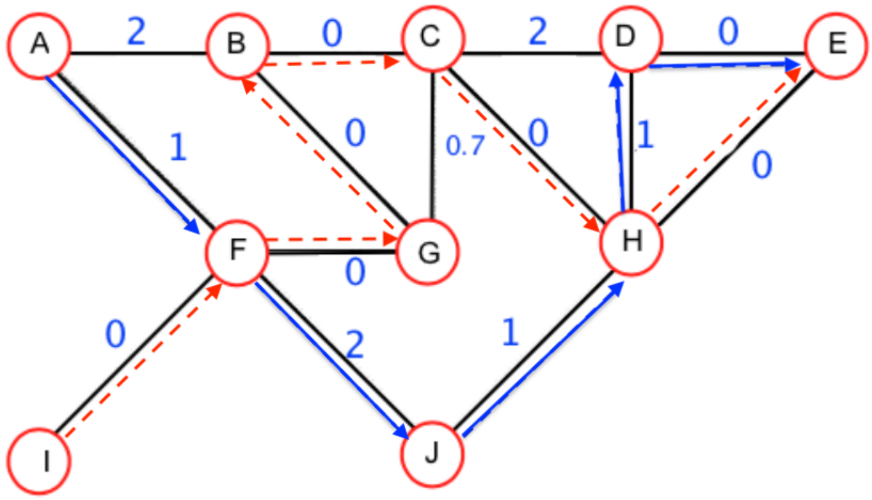


Figure 13: Best Nash equilibrium $\pi = (2, 1)$

$K_1(\bar{h}(2, 1)) = 5$,$K_2(\bar{h}(2, 1))) = 0$

$K_1(\bar{h}(2, 1))$+ $K_2(\bar{h}(2, 1))) = 5$

Making necessary computation, we get best Nash equilibrium in this game.

```
Network file: NETWORK2
Players number: 2
Player (1): A
Player (2): I
Fixed node: E
************************************************************

 -- Case 1: π = {1, 2} --
--------------------------
 === (A) --> (E) ===
minimum Time = 1
 path = A -> F -> G -> B -> C -> H -> E
========================================
 === (I) --> (E) ===
minimum Time = 4
 path = I -> F -> J -> H -> D -> E
========================================

 -- Case 2: π = {2, 1} --
--------------------------
 === (I) --> (E) ===
minimum Time = 0
 path = I -> F -> G -> B -> C -> H -> E
========================================
 === (A) --> (E) ===
minimum Time = 5
 path = A -> F -> J -> H -> D -> E
========================================

 - best nash equilibrium Time = 5

press any key to exit ...
```

Figure 14: W=5

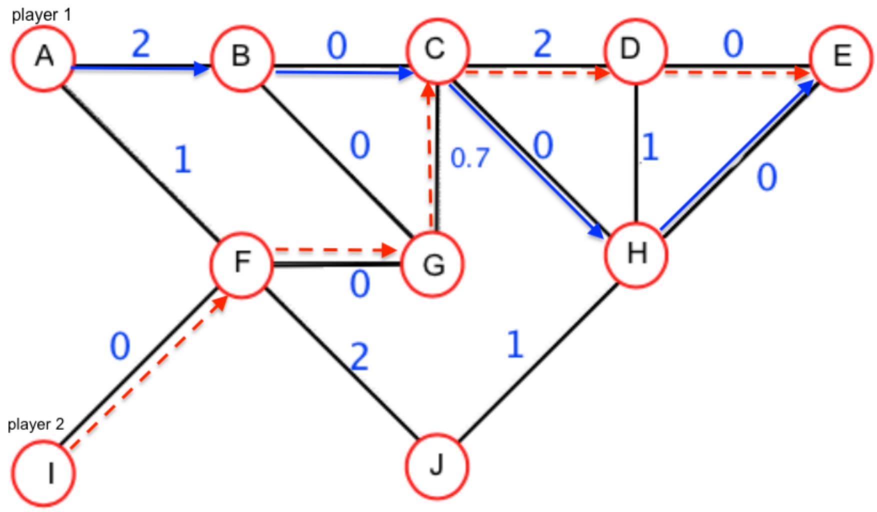


Figure 15: First solution

$K_1(\bar{\bar{h}}) = 2$, $K_2(\bar{\bar{h}}) = 2.7$

$K_1(\bar{\bar{h}})+ K_2(\bar{\bar{h}}) = 4.7$

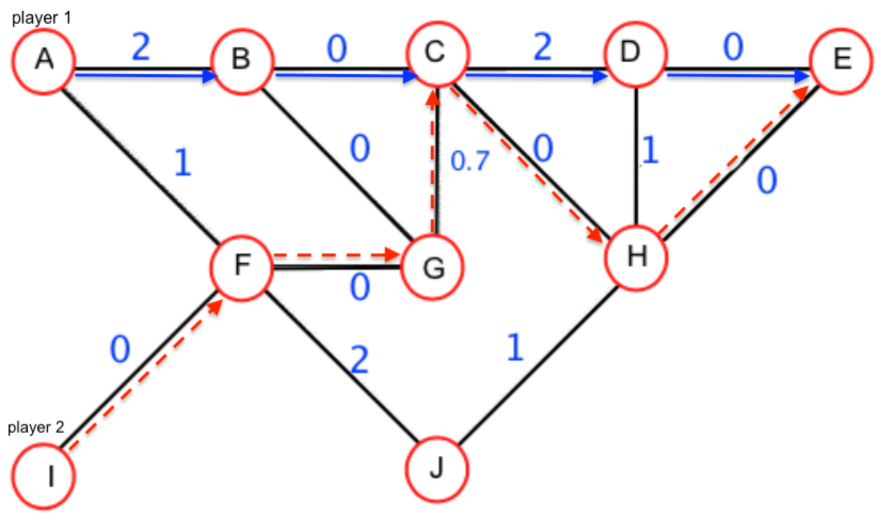
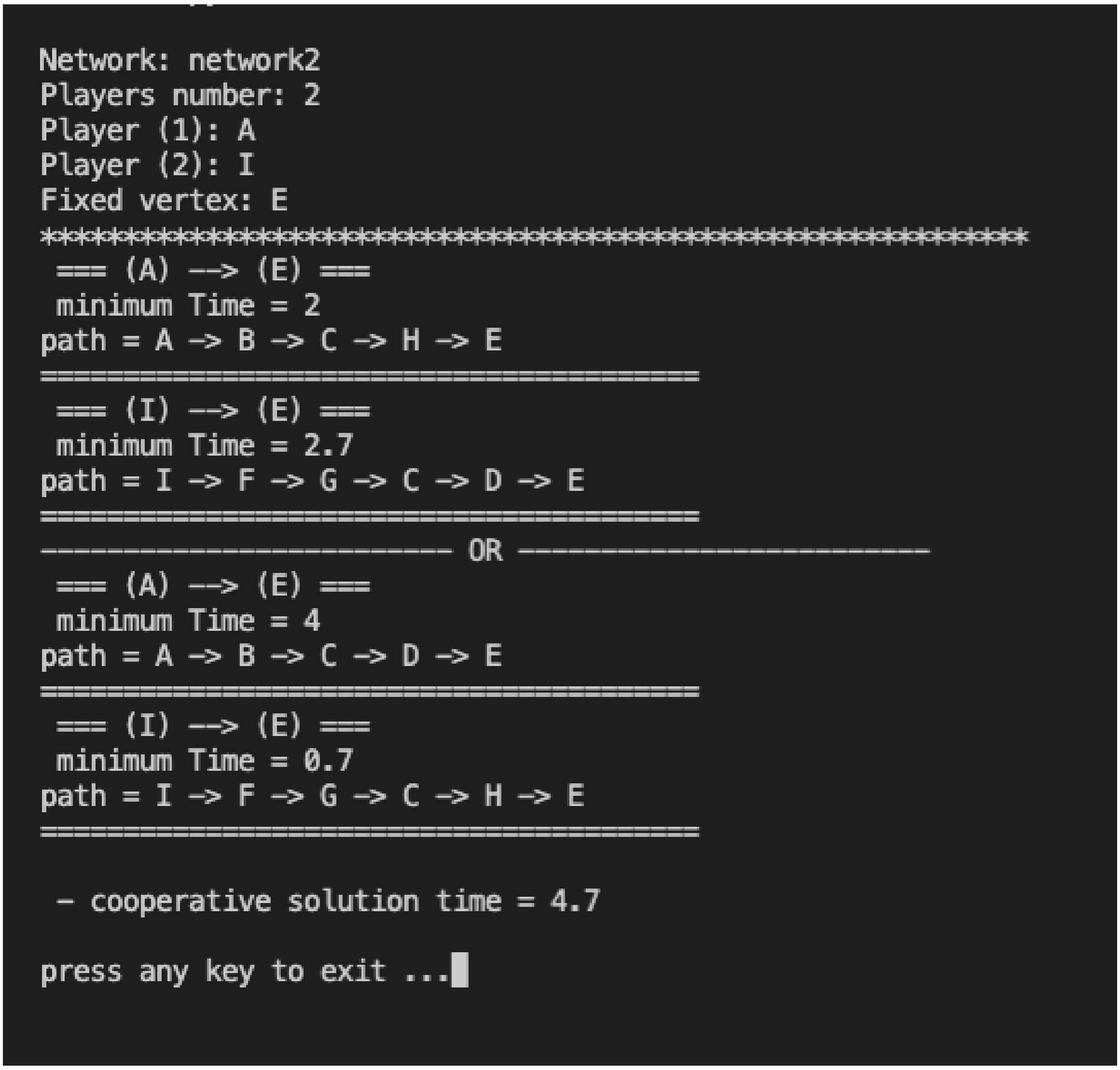


Figure 16: Second solution

$K_1(\bar{\bar{h}}) = 4$, $K_2(\bar{\bar{h}}) = 0.7$

$K_1(\bar{\bar{h}})+ K_2(\bar{\bar{h}}) = 4.7$

Making necessary computation, we get cooperative solutions in this game



```
Network: network2
Players number: 2
Player (1): A
Player (2): I
Fixed vertex: E
**************************************************************
 === (A) --> (E) ===
 minimum Time = 2
path = A -> B -> C -> H -> E
========================================
 === (I) --> (E) ===
 minimum Time = 2.7
path = I -> F -> G -> C -> D -> E
========================================
------------------------- OR -------------------------
 === (A) --> (E) ===
 minimum Time = 4
path = A -> B -> C -> D -> E
========================================
 === (I) --> (E) ===
 minimum Time = 0.7
path = I -> F -> G -> C -> H -> E
========================================

 - cooperative solution time = 4.7

press any key to exit ...
```

Figure 17: v =4.7

1.16 Consider cooperative solution in game Γ_1 as mini maximal time

Consider now another approach to define the cooperative solution. For each strategy profile we define the player i with the maximal time necessary to reach from the $i(x_0)$ to fixed node a,then from all strategies profiles we select such strategy profile for which this maximal time is minimal. This strategy profile will shale call cooperative mini maximal strategy profile $\bar{\bar{\bar{h}}}(\hat{\pi})$.

$$K_i\left(\bar{\bar{\bar{h}}}(\hat{\pi})\right) = \min_{\pi}\left[\max_{i}\left(\bar{h}^i(\pi)\right)\right] = R_1 \tag{7}$$

1.17 Chart of the algorithm for cooperative solution in game Γ_1 as min maximal time

We developed Dijkstra's algorithm to find mini maximal time for any network in $n-$player game Γ_1 and it is a following chart :

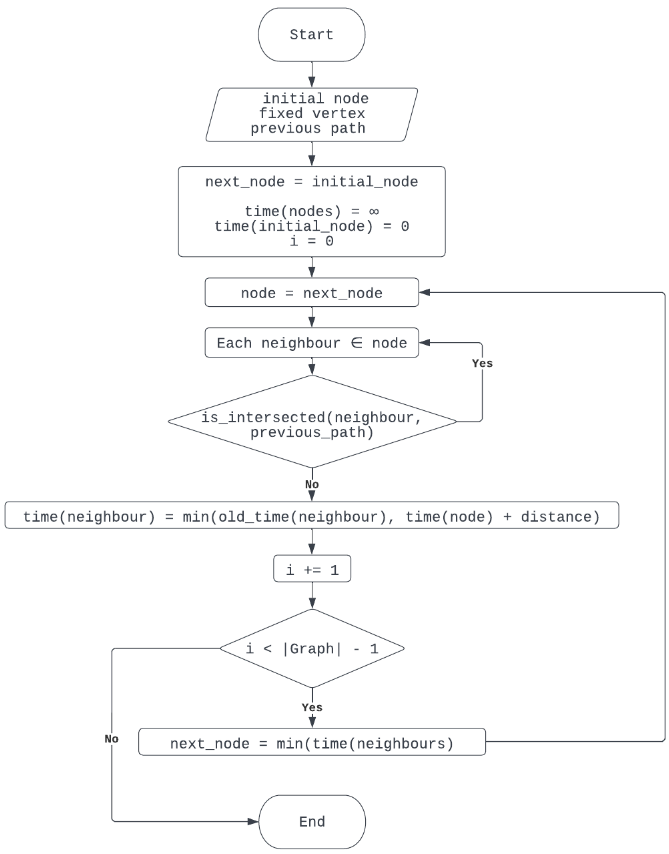


Figure 18: Nash equilibrium (arc)

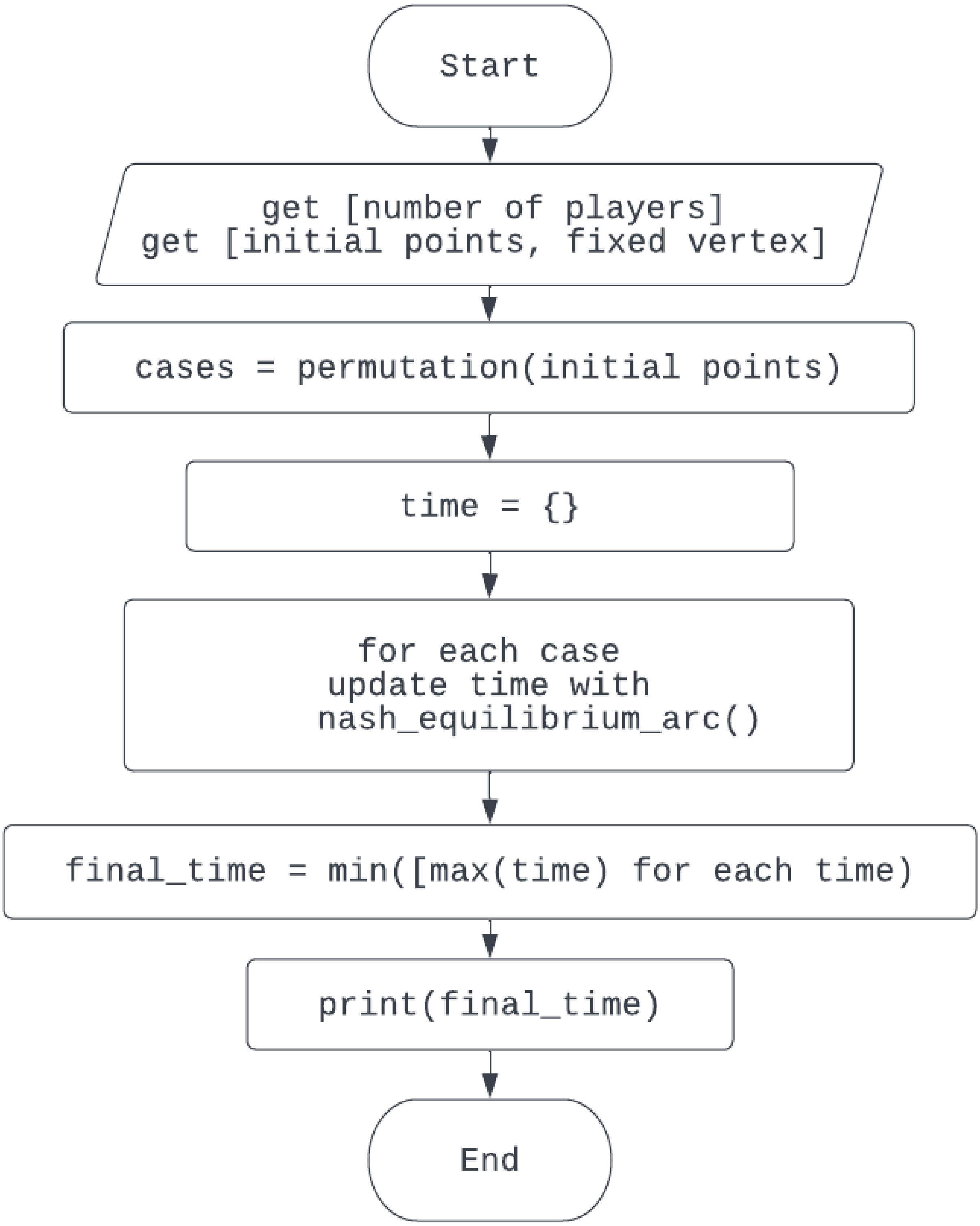

Start
get [number of players]
get [initial points, fixed vertex]
cases = permutation(initial points)
time = {}
for each case
update time with
nash_equilibrium_arc()
final_time = min([max(time) for each time)
print(final_time)
End

Figure 19:

1.18 Example for cooperative solution in game Γ_1 as mini maximal time

Under conditions of example (1.15). Making necessary computation, we get:

```
Players number: 2
Player (1): A
Player (2): I
Fixed vertex: E
****************************************************************

 -- Case 1: π = {1, 2} --
---------------------------
 === (A) --> (E) ===
  minimum Time = 1
 path = A -> F -> G -> B -> C -> H -> E
==========================================
 === (I) --> (E) ===
  minimum Time = 4
 path = I -> F -> J -> H -> D -> E
==========================================

 -- Case 2: π = {2, 1} --
---------------------------
 === (I) --> (E) ===
  minimum Time = 0
 path = I -> F -> G -> B -> C -> H -> E
==========================================
 === (A) --> (E) ===
  minimum Time = 5
 path = A -> F -> J -> H -> D -> E
==========================================

 - minimal of maximal time = 4

press any key to exit ...
```

Figure 20:

This example shows that in some cases $R_1 < V \leq W$. It is interesting to investigate this property in the general case.

1.19 Optimal cooperative trajectory.

Remind the definition of cooperative path (6)

$$\bar{h} = [\{(x_{01}^1, x_{11}^1), (x_{11}^1, x_{21}^1), \ldots, (x_{l_1-1}^1, a)\}, \ldots \{(x_{0i}^i, x_{1i}^i), (x_{1i}^i, x_{2i}^i), \ldots, (x_{l_i-1}^i, a)\}, \ldots$$

$\{(x_{0n}^n, x_{1n}^n), (x_{1n}^n, x_{2n}^n), \ldots, (x_{l_n-1}^n, a)\}]$, where $L = \max_{1 \le i \le n} l_i$.

Denote $\bar{x}(k)$ cooperative trajectories corresponding to cooperative path $\bar{h}$.

$$\bar{x} = (x_{01}^1, x_{11}^1, x_{21}^1, \ldots, x_{l_1-1}^1, a), \ldots (x_{0i}^i, x_{1i}^i, x_{2i}^i, \ldots, x_{l_i-1}^i, a), \ldots (x_{0n}^n, x_{1n}^n, x_{2n}^n, \ldots, x_{l_n-1}^n, a)$$

The subgame starting from state $\bar{x}(k) = (x_{k1}^1, \ldots, x_{ki}^i, \ldots, x_{kn}^n)$,

where $x_{ki}^i = (x_{0i}^i, x_{1i}^i, x_{2i}^i, \ldots, x_{l_i-1}^i, a), i = 1, \ldots, n$, and k stage number for players.

1.20 The proportional Solution in game Γ_1

In the cooperative version of the game we suppose that all players jointly minimal the total costs and this minmize total cost we denote by $V(N)$. The problem in cooperative game theory how to allocate this total minimal cost between players. In our sitting we will use as optimality principle the proportional solution.We have $n-$player in Γ_1 which want to reach the fixed node in network in minimal cost (sum of the costs necessary to reach the fixed node by all players). In such way that the corresponding paths do not contain common arcs. The proportional solution defined as (see [17]):

$$\tilde{\varphi}_i(x_0) = \frac{V(i; x_0)}{\sum_{i=1}^{n} V(i; x_0)} V(N; x_0); \quad i \in N$$

$\tilde{\varphi}_i(x_0)$: is proportional solution for player i in the his initial vertex (x_0).

$V(i; x_0)$: is minimal total cost of player i in the his initial vertex (x_0).

The proportional solution in cooperative game is defined in a classical way:

$$\tilde{\varphi}_i(\bar{x}(k), k) = \frac{V(i; \bar{x}(k), k)}{\sum_{i=1}^{n} V(i; \bar{x}(k), k)} V(N; \bar{x}(k), k); \quad i \in N$$

$\tilde{\varphi}_i(\bar{x}(k), k)$:is the proportional solution for player i along his trajectory $\bar{x}(k)$.

$V(N, \bar{x}(k), k)$: is a minimal total cost for all players jointly (cooperative solution) along cooperative trajectories $\bar{x}(k)$.

$V(i, \bar{x}(k), k)$: is a minimal total cost for player i along cooperative trajectory $\bar{x}(k)$.

It is shown on example $\tilde{\varphi}_i(\bar{x}(0), 0) \neq \tilde{\varphi}_i(\bar{x}(1), 1)+$ (one cost out).

1.21 Example of the Proportional Solution in game Γ_1

Under the same conditions and the same transportation costs in the example (1.15) an

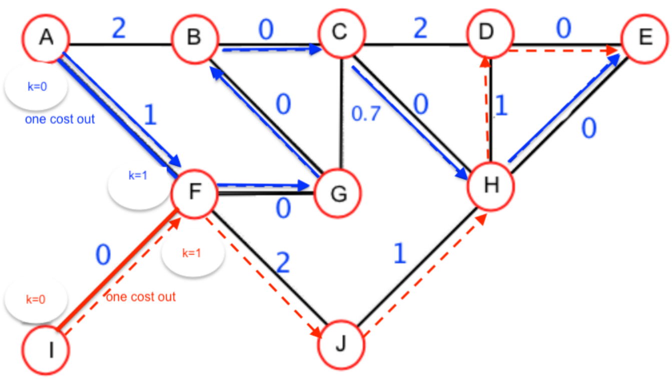


Figure 21: Best Nash equilibrium $\pi = (1, 2)$

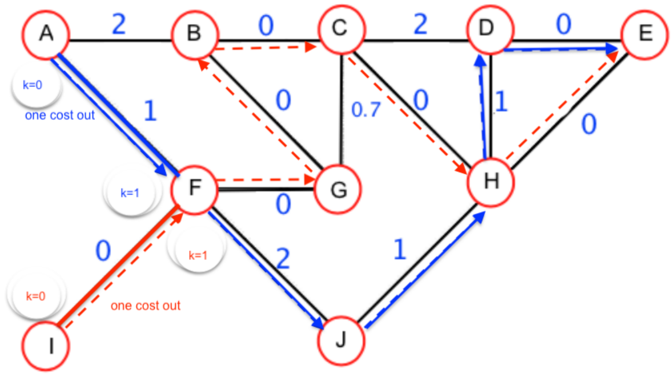


Figure 22: Best Nash equilibrium $\pi = (2, 1)$

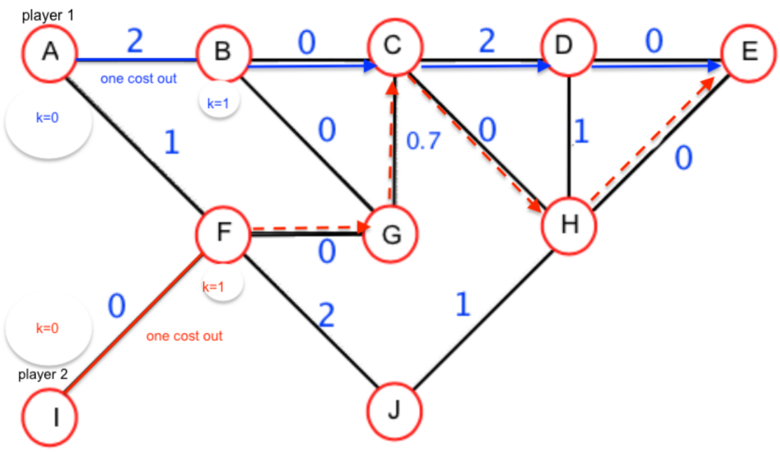


Figure 23: Cooperative solution

At k = 0 in case best Nash equilibrium $\pi = (1, 2)$

$$\begin{aligned}\tilde{\varphi}_1(\bar{x}(0),0) =& \frac{V(1,\bar{x}(0),0)}{V(1,\bar{x}(0),0)+V(2,\bar{x}(0),0)}V((1,2),\bar{x}(0),0) = \frac{1}{5}4.7 = 0.94 \\ \tilde{\varphi}_2(\bar{x}(0),0) =& \frac{V(2,\bar{x}(0),0)}{V(1,\bar{x}(0),0)+V(2,\bar{x}(0),0)}V((1,2),\bar{x}(0),0) = \frac{4}{5}4.7 = 3.76\end{aligned}.$$

At k = 1 in case best Nash equilibrium $\pi = (1, 2)$

$$
\begin{aligned}
\tilde{\varphi}_1(\bar{x}(1),1) =& \frac{V(1,\bar{x}(1),1)}{V(1,\bar{x}(1),1)+V(2,\bar{x}(1),1)} V((1,2),\bar{x}(1),1) = \frac{0}{4}2.7 = 0 \\
\tilde{\varphi}_2(\bar{x}(1),1) =& \frac{V(2,\bar{x}(1),1)}{V(1,\bar{x}(1),1)+V(2,\bar{x}(1),1)} V((1,2),\bar{x}(1),1) = \frac{4}{4}2.7 = 2.7
\end{aligned}
$$

Compare the results

In case non-cooperative game $\pi = (1,2)$

$$\tilde{\varphi}_1(\bar{x}(1),1) + 1 = 1 \neq \tilde{\varphi}_1(\bar{x}(0),0) = 0.94$$

$$\tilde{\varphi}_2(\bar{x}(1),1) + 0 = 2.7 \neq \tilde{\varphi}_2(\bar{x}(0),0) = 3.76$$

So $\tilde{\varphi}_i(\bar{x}(0),0) \neq \tilde{\varphi}_i(\bar{x}(1),1) +$ (one cost out). The characteristic function of the proportional solution is not time consistent in Γ_1 .

1.22 The Shapely value in cooperative game Γ_1

Let $V(S); S \subset N$ and $V(1), V(2)$ where $V(1) + V(2) \geqslant V(N)$ and $V(S \cup T) \leqslant V(S) + V(T)$, And $n = |N|, S = |S|$ where $S \subset N$, And $S \cap T = \emptyset$

The Shapely value $Sh = \{Sh_i\}_{i \in N}$ in the game Γ_1 is a vector, such that(see[16]):

$$Sh_i\left(\bar{x}(k),k\right) = \sum_{i \in S \subset N} \frac{(n-s)!(s-1)!}{n!} \left(V\left(S,\bar{x}(k),k\right) - V\left(S \backslash \{i\}, \bar{x}(k), k\right)\right.$$

$V\left(S,\bar{x}(k),k\right)$: is a minimal total cost for subset of players jointly (cooperative solution) along cooperative trajectories $\bar{x}(k)$.

$V\left(S \backslash \{i\}, \bar{x}(k), k\right)$: is a minimal total cost for subset all players jointly (cooperative solution) without player i along cooperative trajectories $\bar{x}(k)$.

If we have 2 players the formula of the Shapley value will be:

$$Sh_1(\bar{x}(k),k) = V(1,\bar{x}(k),k) -$$
$$-\frac{V(1,\bar{x}(k),k)+V(2,\bar{x}(k),K)-V((1,2),\bar{x}(k),k))}{2}$$
$$Sh_2(\bar{x}(k),k) = V(2,\bar{x}(k),k) -$$
$$-\frac{V(1,\bar{x}(k),k)+V(2,\bar{x}(k),k)-V((1,2),\bar{x}(k),k)}{2}$$

And we will get

$Sh_1(\bar{x}(k),k) + Sh_2(\bar{x}(k),k) = V((1,2),\bar{x}(k),k)$

How we defined the value of $V(S); S \subset N$ in game if $N = \{1,2\}$

Table 1:

Value of $V(S)$; $S \in N$			
FIRST CASE	$(N\|S)$ then S	$V(1)$	The value at $\pi = (2,1)$
		$V(2)$	The value at $\pi = (1,2)$
SECOND CASE	S then $(N\|S)$	$V(1)$	The value at $\pi = (1,2)$
		$V(2)$	The value at $\pi = (2,1)$

It is shown on example the characteristic function of the Shapely value is not time consistent in Γ_1 .

$$Sh_i(\bar{x}(0),0) \neq Sh_i(\bar{x}(1),1) + (\text{ one cost out})$$

1.23 Example of the shapley value in cooperative game Γ_1

Under the same conditions and the same transportation costs in the example (1.15)

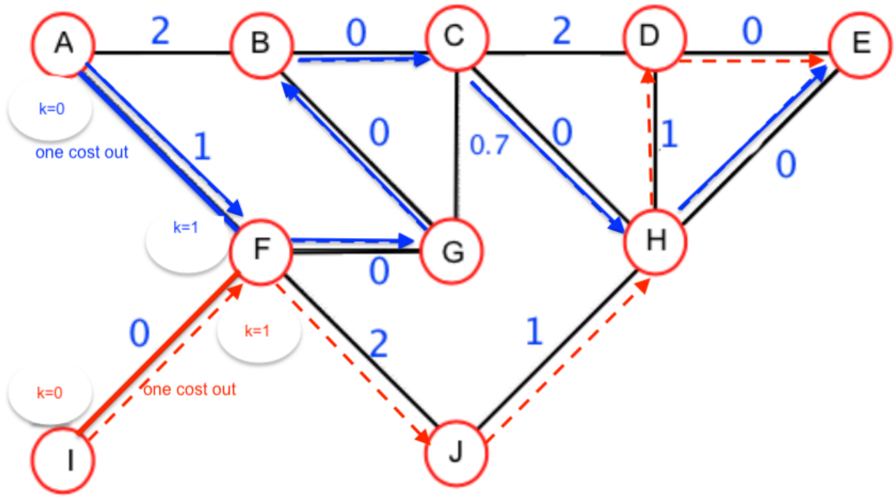


Figure 24: Best Nash equilibrium $\pi = (1, 2)$

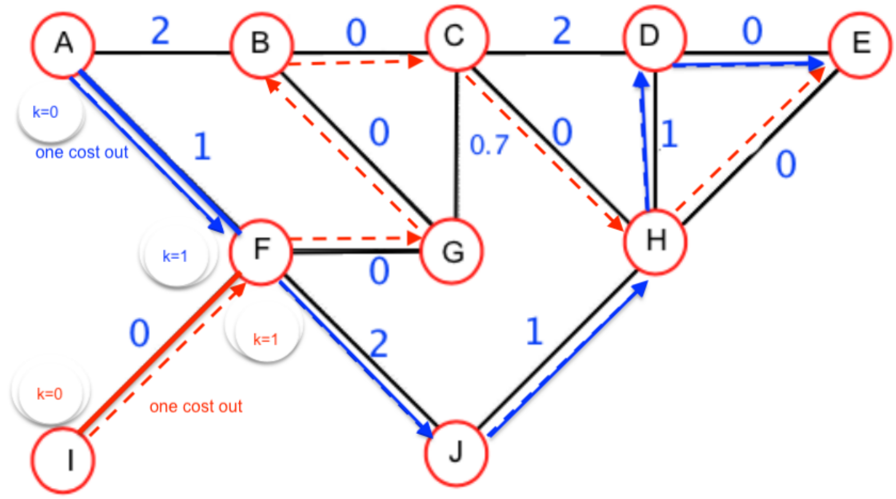


Figure 25: Best Nash equilibrium $\pi = (2, 1)$

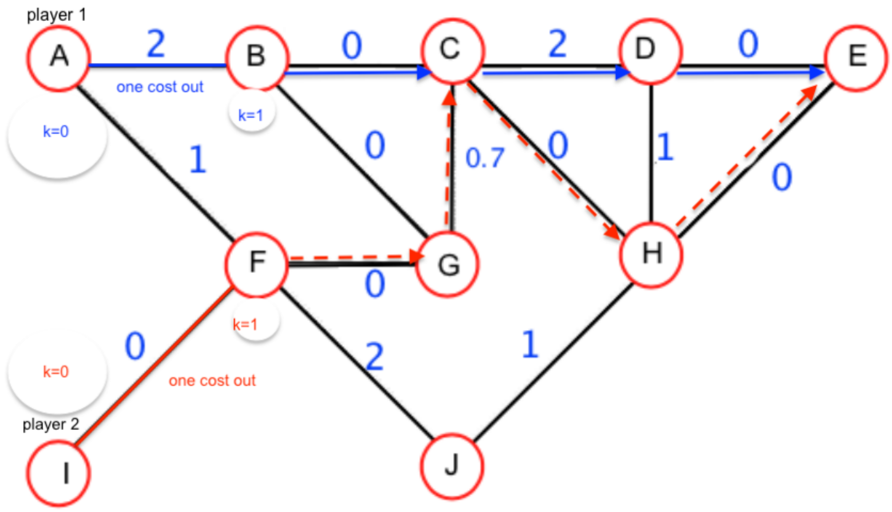


Figure 26: Cooperative solution

At k = 0,$\pi = (1,2)$

$$Sh_1(\bar{x}(0),0) = 5 - \frac{5+4-4.7}{2} = 2.85$$
$$Sh_2(\bar{x}(0),0) = 4 - \frac{5+4-4.7}{2} = 1.85$$

At k = 1,$\pi = (1,2)$

$$Sh_1(\bar{x}(1),1) = 4 - \frac{4+4-2.7}{2} = 1.35$$
$$Sh_2(\bar{x}(1),1) = 4 - \frac{4+4-2.7}{2} = 1.35$$

Compare the results

$$Sh_1(\bar{x}(1),1) + 1 = 1.35 + 1 = 2.35 \neq 2.85 = Sh_1(\bar{x}(0),0)$$
$$Sh_2(\bar{x}(1),1) + 1 = 1.35 + 1 = 2.35 \neq 1.85 = Sh_2(\bar{x}(0),0)$$

The characteristic function of the Shapely value is not time consistent in Γ_1 .

2 Minimization of transportation time in the case when paths have no common vertices

2.1 Model

The game takes place on the network $G = (X, D)$, where X is a finite set, called the vertex set and $D-$ set of pairs of the form (y, z), where $y \in X$, $z \in X$, called arcs. points $x \in X$ will be called vertices or nodes of the network. On a set of arcs D a non-negative symmetric real valued function is given $\gamma(x, y) = \gamma(y, x) \geq 0$, interpreted for each arc $(x, y) \in D$ as the time associated with the transition from x to y by arc (x, y).

2.2 Description of transportation game

Define n-player transportation game on network G. The transportation game Γ is system $\Gamma_2 = \langle G, N, x(N), a \rangle$, where $G-$ network,$N = \{1, \ldots, n\}-$ is set of players,$a \in X$ - some fixed node of the network $G, x(N) \subset X$ - subset of vertices of network $G, x(N) = \{1(x), 2(x), \ldots, i(x), \ldots, n(x)\}$, indicating the vertices in which players are located in $x(N)$ at the beginning of the game process (the initial position of the players). For example $i(x)$ means the vertex $x \in X$, in which the player i is located at the beginning of the game. The set $x(N)$ may contain coinciding vertices, i.e. at the beginning of the game, several players can be at the same vertex. In some cases, in order not to complicate the notation, so by $i(x)$ we will also mean the vertex in which the player i is located. On a path in the game Γ any finite sequence of arcs of the form $e = \{(x_0, x_1), (x_1, x_2), \ldots, (x_{l-1}, x_l)\}$, under condition that the initial vertex in each arc considers with the final vertex of previous arc is calles path. Also we suppose that there is player $i \in N$, such that $x_0 = i(x_0) \in x(N)$ and $x_l = a$. Thus, a path is a sequence of arcs connecting the initial positions of the players in the network to fixed node a. Denote F as sequence of vertices of the form $F = \{x_0, x_1, x_1, x_2, \ldots, x_{l-1}, x_l\}$ in the path e. We will say that corresponding sequences of vertices do not intersect and write $F_{e'} \cap F_{e''} = \phi$ if they do not have common vertices.

2.3 Minimization of transportation time in ($n-$player game)

We have $n-$player located in initial positions (vertices) which want to reach the fixed node a in network in minimal time, in such way that the corresponding paths have not contain common vertices. Denote this game as Γ_2.

2.4 Strategies in Γ_2

Strategies of player i in the game $\Gamma_2$are the paths in which the starting vertex $x_0 = i\,(x_0)$, and the final vertex coincides with $a \in X$. Denote the strategy of player i as:

$e^i = \{(x_0, x_1), (x_1, x_2), \ldots, (x_k, x_{k+1}), \ldots, (x_{l-1}, a)\}$,

A bunch of all strategies of player i will be denoted by $E^i = \{e^i\}, i = 1, \ldots, n$.

2.5 Admissible strategy profiles in Γ_2

The admissible strategy profiles in the game Γ_2. The strategy profiles $e = (e^1, \ldots, e^n), e^1 \in E^1, \ldots, e^n \in E^n$ are called admissible if the corresponding sequences of vertices do not intersect and write $F_{e'} \cap F_{e''} = \phi$ where F is sequence of vertices of the form $F = \{x_0, x_1, x_1, x_2, \ldots, x_{l-1}, x_l\}$ in the path e if they do not have common vertices. The set of all admissible strategy profiles is denoted by E.

2.6 Cost Function in Γ_2

In this suction we define for each arc (x_k, x_{k+1}) the values of cost function $\gamma_i\,(x_k, x_{k+1})$ as time necessary to reach the node x_{k+1} from node x_k by player i. For each strategy profile $e = (e^1, \ldots, e^n) \in E$.

Denote the player i time is $K_i(e)$ to reach the node a.

$$K_i(e) = \sum_{k=0}^{l-1} \gamma_i\left(x_k, x_{k+1}\right) = k\left(e^i\right) \tag{8}$$

Here $\{(x_0, x_1), (x_1, x_2), \ldots, (x_{l-1}, x_l)\} = e^i$. Thus, F sequence of vertices of the form $F_{e^i} = \{x_0, x_1, x_1, x_2, \ldots, x_{l-1}, x_l\}$, we see for the player i time $K_i(e)$ depends on his strategy e^i and on the strategies of other players in that the strategy e^i (path of player i) should not intersect with the strategies of other players. Therefore, in some cases, when this will not lead to misunderstandings, we instead $K_i(e)$ will use the notation $k\left(e^i\right)$, meaning the player i time along the path e^i.

2.7 Nash equilibrium in $n-$player game Γ_2

In the game Γ_2 the strategy profile $\bar{e} = (\bar{e}^1, \ldots, \bar{e}^n)$ is called a Nash equilibrium, if $K_i\left(\bar{e} \| e^i\right) \geq K_i(\bar{e})$ holds for all admissible strategy profile $(\bar{e} \| e^i) \in E$ and $i \in N$.

Let π be some permutation of numbers $1, \ldots, n, \pi = (i_1, \ldots, i_n)$. Consider an auxiliary transportation problem on the network G for player i_1. Find the path in the network G, minimizing the time of player i_1 to reach from vertex $i_1(x) \in x(N)$ to vertex $a \in X$. Denote the path that solves this problem $\bar{e}^{i_1}$

$$k\left(\bar{e}^{i_1}\right) = \min_{e^{i_1} \in E^{i_1}} k\left(e^{i_1}\right). \tag{9}$$

Denote by $G \backslash F_{\bar{e}^{i_1}}$ a subnetwork not containing $F_{\bar{e}^i}$. Consider an auxiliary transportation problem for player i_2 on network $G \backslash F_{\bar{e}^i}$. Find the path in subnetwork $G \backslash F_{\bar{e}^{i}1}$, which minimizing the player i_2 time to reach from vertex $i_2(x) \in x(N)$ to vertex $a \in X$. Denote the path that solves this problem $\bar{e}^{i_2}$.

$$k\left(\bar{e}^{i_2}\right) = \min_{e^{i_2} \in E^{i_2}} k\left(e^{i_2}\right). \tag{10}$$

Proceeding further in a similar way, we introduce into consideration the subnetworks of the network G, that do not containing vertices which belong to $F_{\bar{e}^{i_1}}, \ldots, F_{\bar{e}^{i_{m-1}}}$. Consider the auxiliary transportation problem of the player i_m on the network network $G \backslash \cup_{l=1}^{m-1} F_{\bar{e}^{i_l}}$. Find the subnetwork $G \backslash \cup_{l=1}^{m-1} F_{\bar{e}^{i_l}}$, minimizing the player i_m time where $i_m(x) \in x(N)$ and $a \in X$. Denote the path that solves this problem $\bar{e}^{i_m}$.

$$k\left(\bar{e}^{i_m}\right) = \min_{e^{i_m} \in E^{i_m}} k\left(e^{i_m}\right) \tag{11}$$

As a result, we get a sequence of paths $\bar{e}^{i_1}, \ldots, \bar{e}^{i_n}$, minimizing the total time of players $i_1, i_2, \ldots, i_m, \ldots, i_n$ on subnetworks:

$$G, G \backslash F_{\bar{e}^{i_1}}, \ldots, G \backslash \cup_{l=1}^{m-1} F_{\bar{e}^{i_l},\ldots}, \ldots, G \backslash \cup_{l=1}^{n-1} F_{\bar{e}^{i_l}}.$$

The sequence of paths $\bar{e}^{i_1}, \ldots, \bar{e}^{i_m}, \ldots, \bar{e}^{i_n}$ by construction consist of pairwise non-intersecting vertices, and each of them $\bar{e}^{i_l} \in E^{i_l}$. Therfore the strategy profile $(\bar{e}^{i_1}, \ldots, \bar{e}^{i_m}, \ldots, \bar{e}^{i_n}) = \bar{e}(\pi) \in E$ is admissible in Γ_2.

2.8 Equilibrium strategy profile/

theorem: The strategy profile $\bar{e}(\pi) \in E$ is an equilibrium strategy profile in Γ_1 for any permutation π.

Proof : Consider the strategy profile. $[\bar{e}(\pi) \| e^{i_m}]$, where $e^{i_m} \neq \bar{e}^{i_m}, e^{i_m} \in E^{i_m}, [\bar{e}(\pi) \| e^{i_m}] \in E$. By construction $\bar{e}^{i_m}$ is determined from the condition

$$k\left(\bar{e}^{i_m}\right) = \min_{e^i m \in G \backslash U_{l=1}^{m-1} \bar{e}^{i_l}} k\left(e^{i_m}\right),$$

However, the strategy profile $[\bar{e}(\pi) \| e^{i_m}]$ is admissible (if $e^{i_m} \in G \backslash \bigcup_{l=1}^{m-1} \bar{e}^{i_l}$) and therefore $k\left(\bar{e}^{i_m}\right) \leq k\left(e^{i_m}\right) = K_{i_m}\left[\bar{e}(\pi) \| e^{i_m}\right]$, However $k\left(\bar{e}^{i_m}\right) = K_{i_m}(\bar{e}(\pi))$, and $K_{i_m}[\bar{e}(\pi)] \leq K_{i_m}\left[\bar{e}(\pi) \| e^{i_m}\right]$ for all $[\bar{e}(\pi) \| e^{i_m}] \in E$, which proves the theorem.

This theorem indicates a rich family of pure strategy equilibrium profiles in Γ_1 depending on permutation π. Thus in Γ_2 we have at lest $n!$ equilibrium strategy profiles in pure strategies,(if the initial states of players are different).

2.9 Best Nash equilibrium in game Γ_2

The strategy profile $\bar{e}(\hat{\pi})$ is called best Nash equilibrium if

$$\sum_{i=1}^{n} K_i(\bar{e}(\hat{\pi})) = \min_{\pi} \sum_{i=1}^{n} K_i(\bar{e}(\pi)) = W_2 \tag{12}$$

2.10 Cooperative solution in game Γ_2

However, there are other Nash equilibrium in Γ_2 is also of Nash equilibrium. Consider the strategy profile $\bar{\bar{e}}$, solving the minimization problem

$$\min_{e} \sum_{i=1}^{n} K_i(e) = \sum_{i=1}^{n} K_i(\bar{\bar{e}}) = V_2 \tag{13}$$

We can simply show that $\bar{\bar{e}}$ is also a Nash equilibrium strategy profile. Because if one player change his strategy and other players do not change their strategies his time under this conditions will be more or equal of his time in case has not change his strategy. Consider the strategy profile $\left(\bar{\bar{e}} = \bar{\bar{e^1}}, \ldots, \bar{\bar{e^i}}, \ldots, \bar{\bar{e^n}}\right)$ if player i change his strategy, we get

$$\sum_{i=1}^{n} K_i(\bar{\bar{e}} \parallel e^i) \geq \sum_{i=1}^{n} K_i(\bar{\bar{e}})$$

,

$$K(\bar{\bar{e^1}}) + K(\bar{\bar{e^2}}) + ... + K(e^i) + ... + K(\bar{\bar{e^n}}) \geq K(\bar{\bar{e^1}}) + K(\bar{\bar{e^2}}) + ... + K(\bar{\bar{e^i}}) + ... + K(\bar{\bar{e^n}})$$

so $K(e^i) \geq K(\bar{\bar{e^i}})$. We call the strategy profile $\bar{\bar{e}}$ cooperative equilibrium in Γ_2. In some cases $V_2 = W_2$, (see the example).

2.11 Chart of the minimum time algorithm for n- player case in Γ_2

We developed Dijkstra's algorithm to find best Nash equilibrium for any network in $n-$player game Γ_2 and it is a following chart :

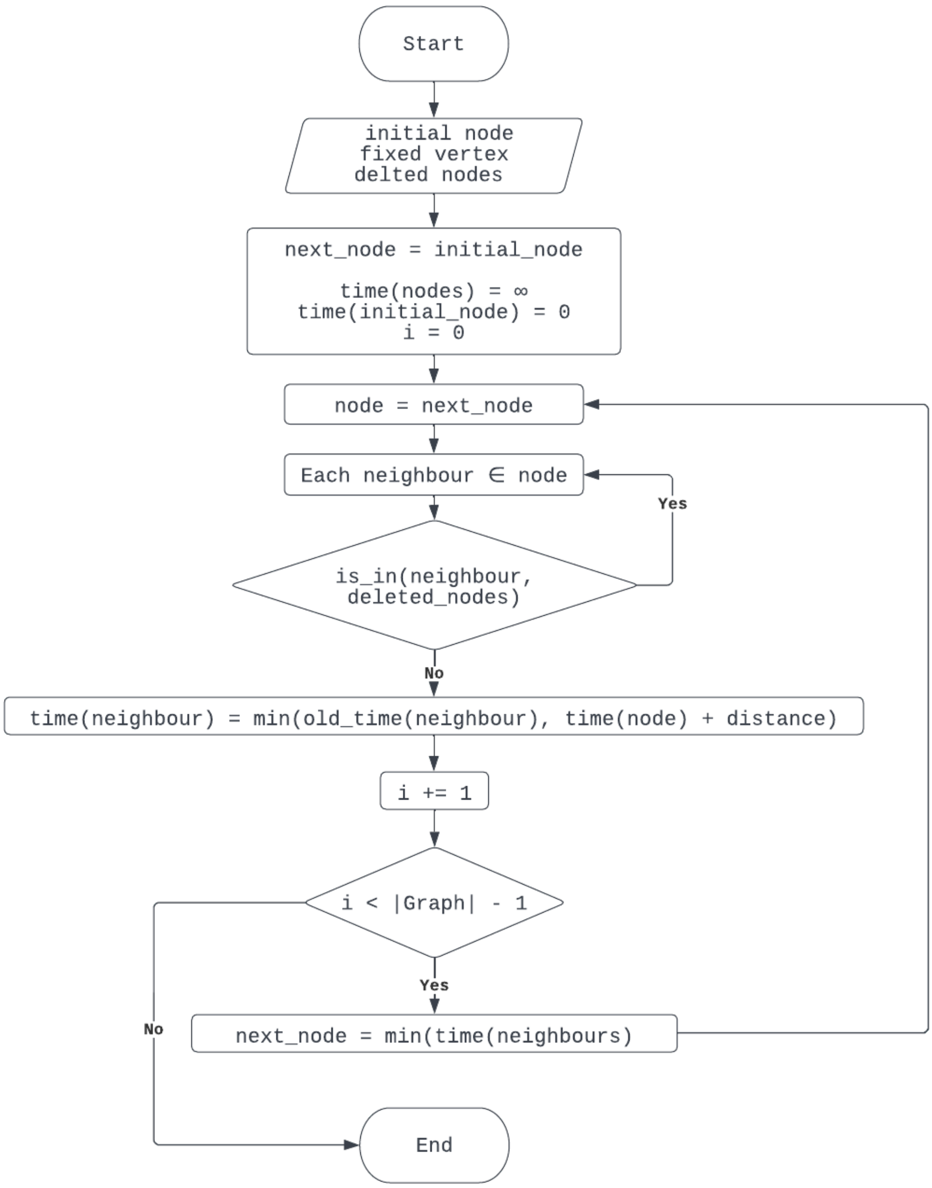


Figure 27: Best Nash equilibrium (vertex) function

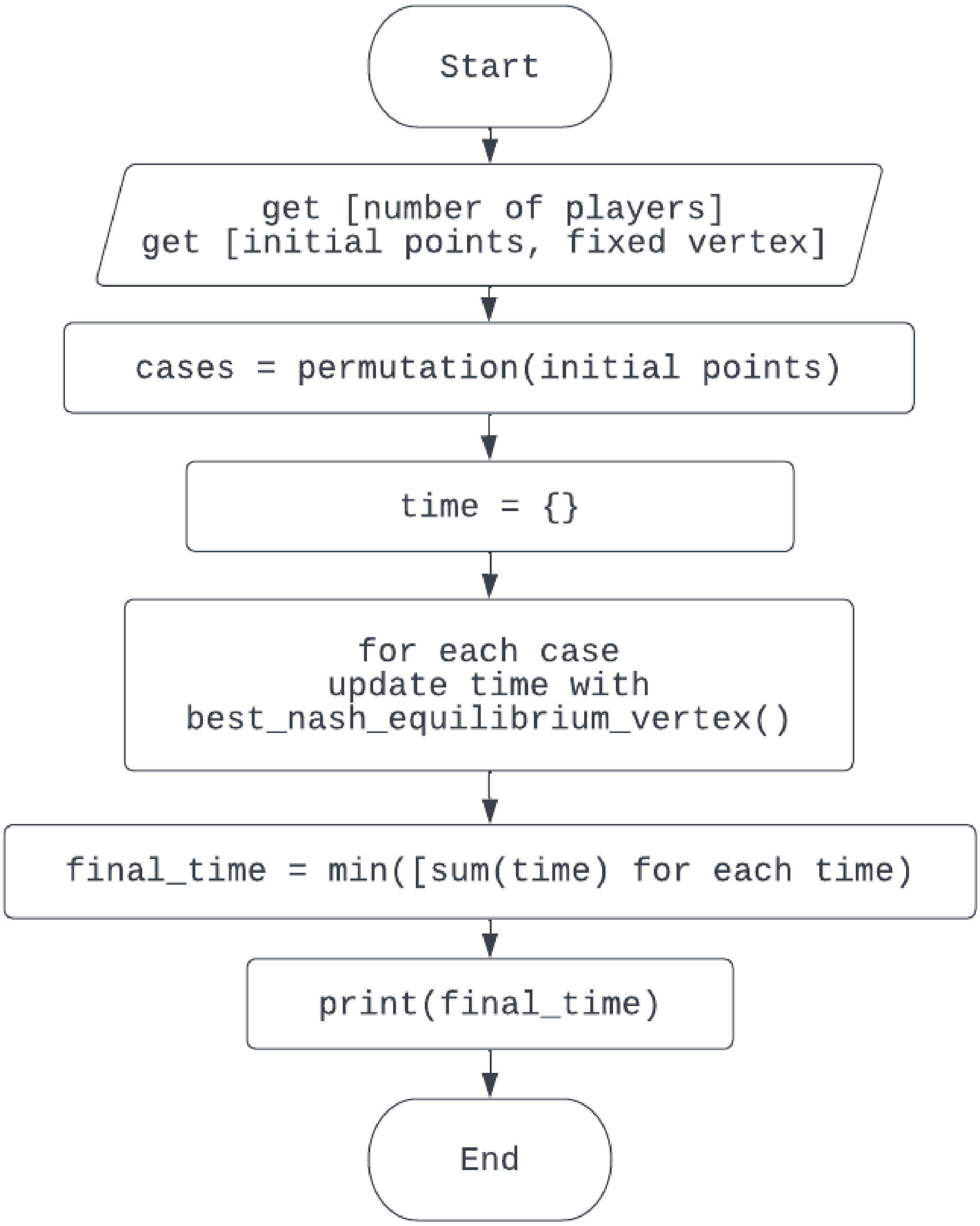


Figure 28: Best Nash equilibrium (vertex)

2.12 Example for two player case in Γ_2

This example show us best Nash equilibrium and give the same result (time) as cooperative solution

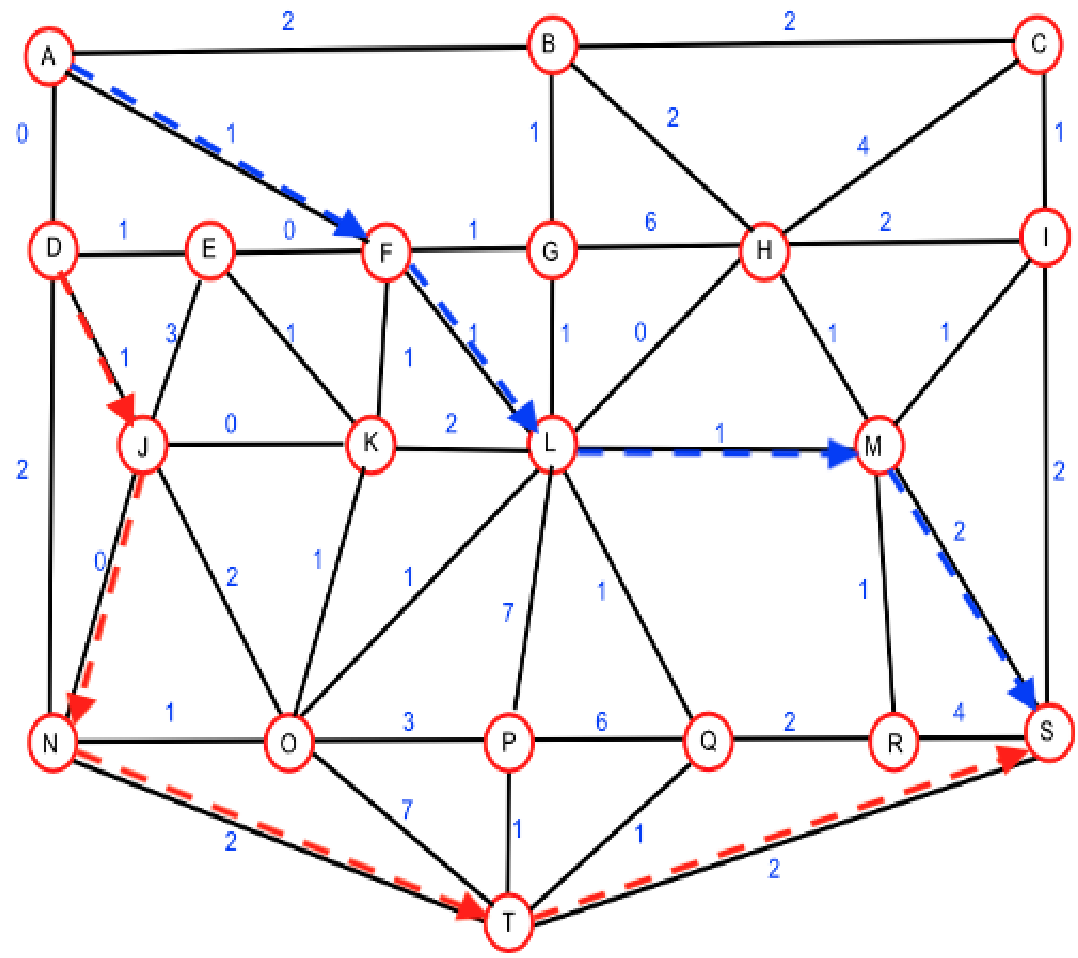


Figure 29: two player in game Γ_2

In this figure we denote nodes by capital Latin letters. $N = \{1, 2\}$ the set $x(N) = \{A, D\}$.The transportation times are written in the network in this figure over the arcs and are equal, respectively to

$$\gamma(A,B)=2,\gamma(A,F)=1,\gamma(A,D)=0,\gamma(B,G)=1,$$

$$\gamma(B,H)=2,\gamma(B,C)=2,\gamma(C,H)=4,\gamma(C,I)=1,$$

$$\gamma(D,N)=2,\gamma(D,E)=1,\gamma(D,J)=1,\gamma(E,F)=0,$$

$$\gamma(E,J)=3,\gamma(E,K)=1,\gamma(F,G)=1,\gamma(F,K)=1,$$

$$\gamma(F,L)=1,\gamma(G,H)=6,\gamma(G,L)=1,\gamma(H,I)=2,$$

$$\gamma(H,M)=1,\gamma(H,L)=0,\gamma(J,N)=0,\gamma(J,K)=0,$$

$$\gamma(J,O)=2,\gamma(K,L)=2,\gamma(K,O)=1,\gamma(L,M)=1,$$

$$\gamma(L,O)=1,\gamma(L,P)=7,\gamma(L,Q)=1,\gamma(M,R)=1,$$

$$\gamma(M,S)=2,\gamma(M,I)=1,\gamma(I,S)=2,\gamma(N,T)=2,$$

$$\gamma(N,O)=1,\gamma(O,P)=3,\gamma(O,T)=7,\gamma(P,T)=1,$$

$$\gamma(P,Q)=6,\gamma Q,R)=2,\gamma(T,Q)=1,\gamma(T,S)=2,\gamma(S,R)=4.$$

We find the minimal transportation time for two player A, D to reach the fixed node S under condition (paths have no common vertices).

Making necessary computation, we get best Nash equilibrium in this Γ_2:

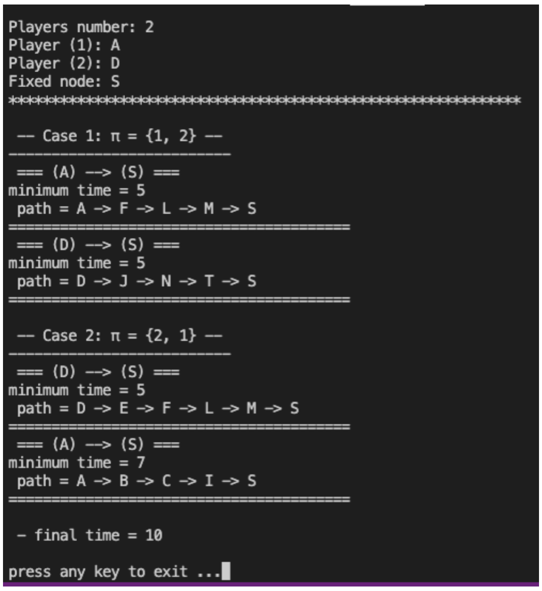


Figure 30: $W_2 = 10$

Making necessary computation, we get Cooperative solution in this Γ_2:

$\bar{\bar{e}}_1 = [(A, F), (F, L)(L, M), (M, S))]$

$\bar{\bar{e}}_2 = [(D, J), (J, N)(N, T), (T, S)]$

$K_1(\bar{e}) = 5$, $K_2(\bar{e}) = 5$

$K_1(\bar{e})+ K_2(\bar{e}) = 10 = V_2$,$W_2 = V_2$

2.13 Another example for two player in Γ_2

This example show that best Nash equilibrium give us different result as cooperative solution and $V_2 < W_2$.

In figure (11), we denote nodes by capital Latin letters.

We have an undirected network and non-negative symmetric real valued functions $N = \{1, 2\}$ the set $x(N) = \{A, I\}$.

Two player want to reach the fixed node E under condition (paths have no common vertices).

The transportation times are written in the network in figure (11), over the arcs and are equal, respectively to

$$\gamma(A, B) = 2, \gamma(A, F) = 1, \gamma(B, C) = 0, \gamma(B, G) = 0,$$
$$\gamma(C, D) = 2, \gamma(C, H) = 0, \gamma(C, G) = 0.7, \gamma(D, E) = 0,$$
$$\gamma(D, H) = 1, \gamma(I, F) = 0, \gamma(F, G) = 0, \gamma(F, J) = 2,$$
$$\gamma(J, H) = 1, \gamma(H, E) = 0,$$

For permuation : $\pi = \{1, 2\}$

$\bar{e_1} = [(A, F), (F, G)(G, B), (B, C)(C, H), (H, E)]$

$K_1(\bar{e}(1, 2)) = 1$,$K_2(\bar{e}(1, 2)) = \infty$

$K_1(\bar{e}(1, 2))$+$K_2(\bar{e}(1, 2)) = \infty$

For permuation : $\pi = \{2, 1\}$

$\bar{e_2} = [(I, F), (F,)(G, B), (B, C)(C, H), (H, E)]$

$K_1(\bar{e}(2, 1)) = \infty$,$K_2(\bar{e}(2, 1))) = 0$ $K_1(\bar{e}(2, 1))+ K_2(\bar{e}(2, 1))) = \infty$

Thus , both equilibrium $\bar{e}(2, 1)$ and $\bar{e}(1, 2)$ are conditionally cooperative equilibrium

(best Nash equilibrium) in Γ_2 and get $W_2 = \infty$

Cooperative solution

$\bar{\bar{e_1}} = [(A,B),(B,C)(C,D),(D,E))]$

$\bar{\bar{e_2}} = [(I,F),(F,J)(J,H),(H,E)]$

$K_1(\bar{\bar{e}}) = 4$, $K_2(\bar{\bar{e}}) = 3$

$K_1(\bar{\bar{e}})+ K_2(\bar{\bar{e}}) = 7 = V_2$

We get the result $V_2 < W_2$

2.14 Consider cooperative solution in game Γ_2 as min maximal time

Consider now another approach to define the cooperative solution.For each strategy profile we define the player i with maximal time necessary to reach from the $i(x_0)$ to fixed node a,then from all strategies profiles we select such strategy profile for which this maximal time is minimal .This strategy profile will shale call cooperative mini maximal strategy profile $\bar{\bar{\bar{e}}}(\hat{\pi})$.

$$K_i\left(\bar{\bar{\bar{e}}}(\hat{\pi})\right) = \min_{\pi}\left[\max_{i}\left(\bar{e}^i(\pi)\right)\right] = R_2 \tag{14}$$

2.15 Chart of the algorithm for cooperative solution in game Γ_2 as mini maximal time

We developed Dijkstra's algorithm to find mini maximal time for any network in $n-$player game Γ_2 and it is a following chart :

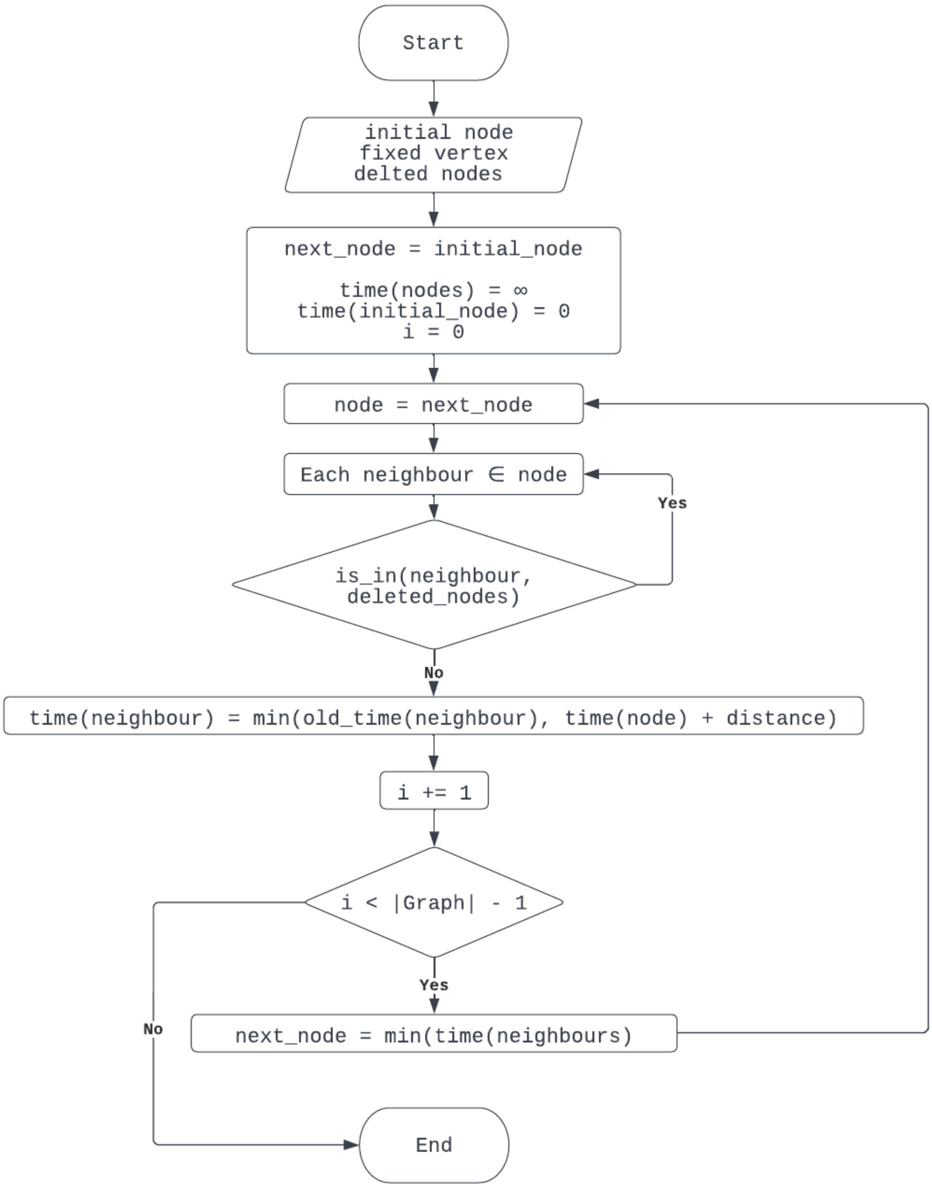

Start
initial node
fixed vertex
delted nodes
next_node = initial_node
time(nodes) = ∞
time(initial_node) = 0
i = 0
node = next_node
Each neighbour ∈ node
is_in(neighbour, deleted_nodes)
Yes
No
time(neighbour) = min(old_time(neighbour), time(node) + distance)
i += 1
i < |Graph| - 1
Yes
No
next_node = min(time(neighbours))
End

Figure 31:

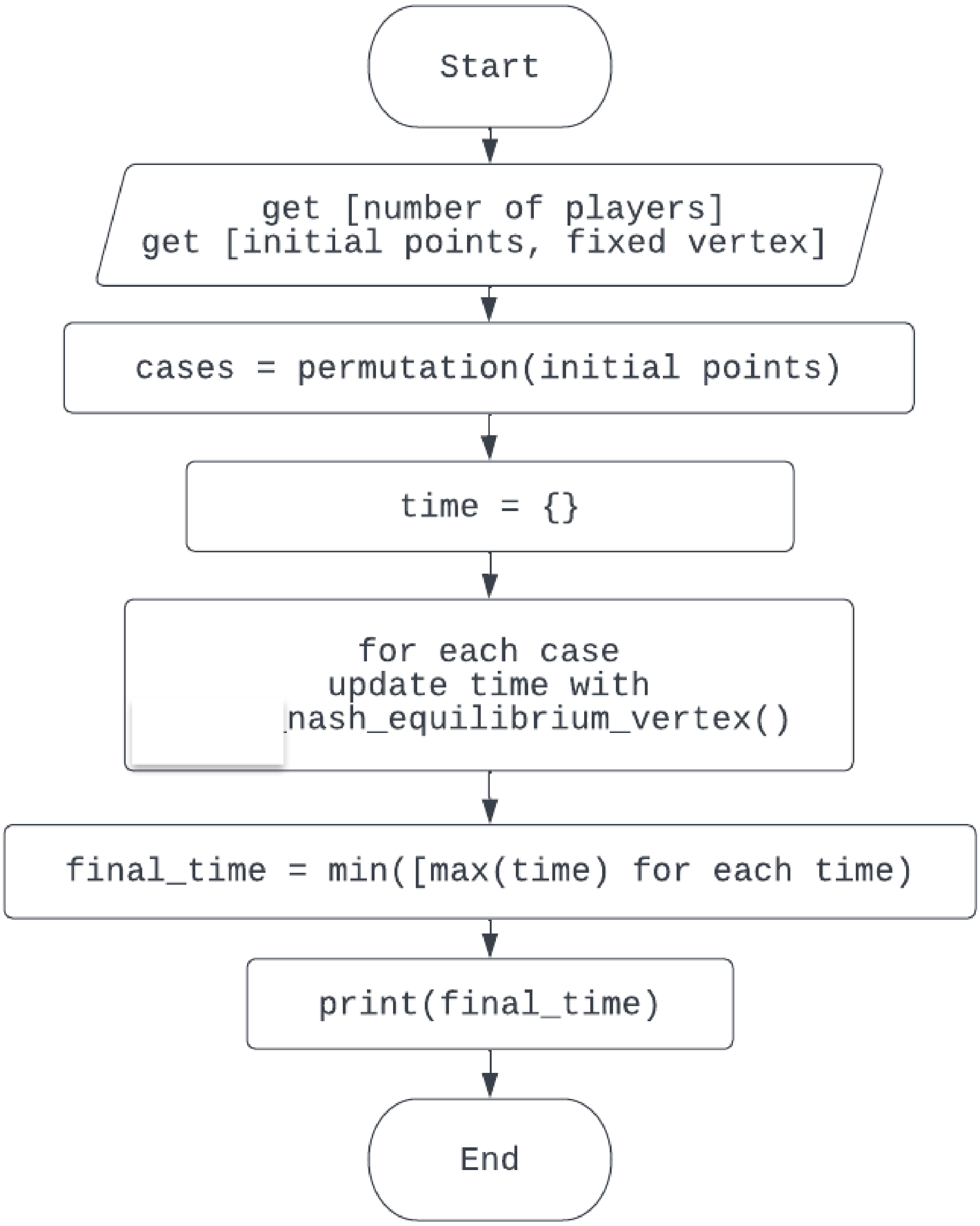

Start
get [number of players]
get [initial points, fixed vertex]
cases = permutation(initial points)
time = {}
for each case
update time with
nash_equilibrium_vertex()
final_time = min([max(time) for each time)
print(final_time)
End

Figure 32:

2.16 Example for cooperative solution in game Γ_2 as mini maximal time

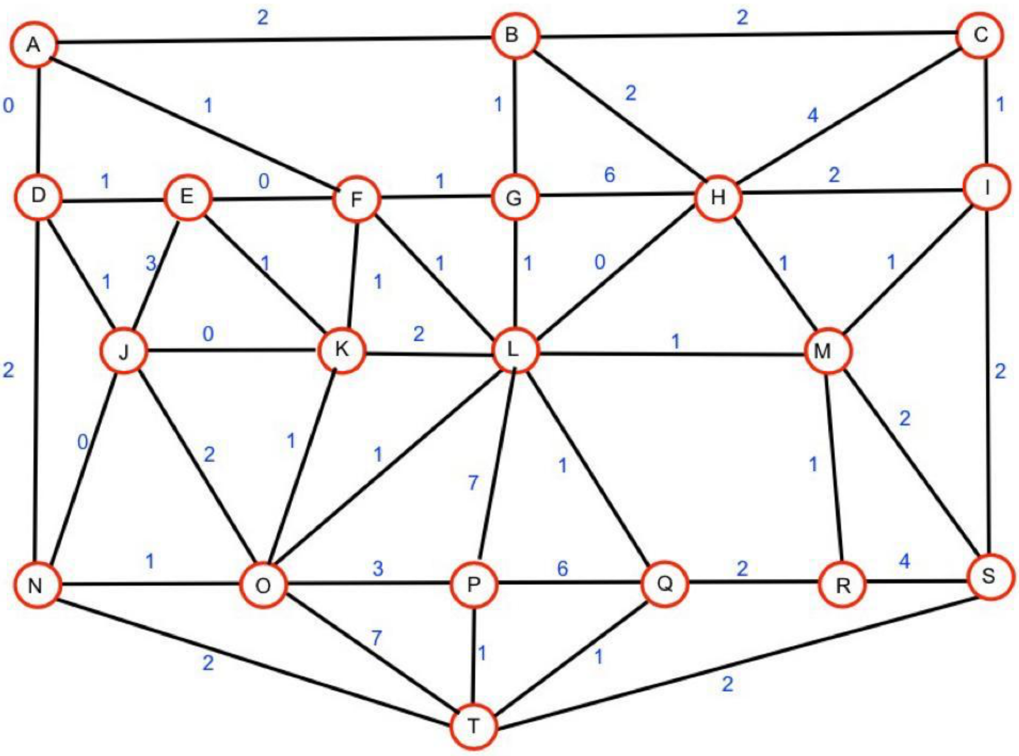


Figure 33: two player in game Γ_2

In this figure we denote nodes by capital Latin letters. $N = \{1, 2\}$ the set $x(N) = \{A, C\}$, The transportation times are written in the network on this figure over the arcs and are equal, respectively to

$$\gamma(A,B)=2,\gamma(A,F)=1,\gamma(A,D)=0,\gamma(B,G)=1,$$

$$v(B,H)=2,\gamma(B,C)=2,\gamma(C,H)=4,\gamma(C,I)=1,$$

$$\gamma(D,N)=2,\gamma(D,E)=1,\gamma(D,J)=1,\gamma(E,F)=0,$$

$$\gamma(E,J)=3,\gamma(E,K)=1,\gamma(F,G)=1,\gamma(F,K)=1,$$

$$\gamma(F,L)=1,\gamma(G,H)=6,\gamma(G,L)=1,\gamma(H,I)=2,$$

$$\gamma(H,M)=1,\gamma(H,L)=0,\gamma(J,N)=0,\gamma(J,K)=0,$$

$$\gamma(J,O)=2,\gamma(K,L)=2,\gamma(K,O)=1,\gamma(L,M)=1,$$

$$\gamma(L,O)=1,\gamma(L,P)=7,\gamma(L,Q)=1,v(M,R)=1,$$

$$\gamma(M,S)=2,\gamma(M,I)=1,\gamma(I,S)=2,\gamma(N,T)=2,$$

$$\gamma(N,O)=1,\gamma(O,P)=3,\gamma(O,T)=7,\gamma(P,T)=1,$$

$$\gamma(P,Q)=6,\gamma Q,R)=2,\gamma(T,Q)=1,\gamma(T,S)=2,\gamma(S,R)=4.$$

We find the minimal transportation time for two player A, C to reach the fixed node T under condition (paths have no common vertices). Making necessary computation, we get cooperative solution in game Γ_2 as mini maximal time as

Figure 34:

This example shows that in some cases $R_2 = V_2 = W_2$. It is interesting to investigate this property in the general case.

3 Time consistency problem

3.1 Model

The game takes place on the network $G = (X, D)$, where X is a finite set, called the vertex set and $D-$ set of pairs of the form (y, z), where $y \in X$, $z \in X$, called arcs. Points $x \in X$ will be called vertices or nodes of the network. On a set of arcs D a non-negative symmetric real valued function is given $\gamma(x, y) = \gamma(y, x) \geq 0$, interpreted for each arc $(x, y) \in D$ as the time associated with the transition from x to y by arc (x, y). In this section we consider the case when players from coalitions.Suppose we have $p-$ coalition M_1,. . .,M_k,. . . ,M_p this coalitions do not intersect and contains same vertices from network G.

3.2 Description of transportation game

Define p-player transportation game on network G. The transportation game Γ_3 is system $\Gamma_3 = \langle G, P, M(P), a \rangle$, where $G-$ network,$P = \{1, \ldots, p\}-$ is set of players (coalitions), $a \in X$ - some fixed node of the network G. $M(P)$ - subset of coalitions of network G, $M(P) = \{1(M), 2(M), \ldots, k(M), \ldots, p(M)\}$, indicating the coalitions in which players are located in $M(P)$ at the beginning of the game process (the initial position of the players(coalitions)). We will say that the paths of players(coalitions) $h^{M'}$ and $h^{M''}$ do not intersect and write $h^{M'} \cap h^{M''} = \emptyset$, if they do not have common arcs. Denoted this game as Γ_3.

3.3 Strategies in Γ_3.

The set $M_k = \left\{i_1^k, \ldots, i^k{}_r, \ldots, i^k{}_{r_k}\right\}$ in network G, we call coalition. The Strategies of coalition are defined as $M_k = \left\{i_1^k, \ldots, i^k{}_r, \ldots, i^k{}_{r_k}\right\}$ as any path connecting his initial position(initial position of players from M_k) with a fixed node a. The paths of players inside coalition may intersect.

Denote as $h^{M_k} = \left\{h^{i^k{}_1}, \ldots, h^{i^k{}_r}, \ldots, h^{i^k{}_{r_k}}\right\}$. ,where $\left\{h^{i^k{}_1}, \ldots h^{i^k{}_r}, \ldots, h^{i^k{}_{r_k}}\right\}$ are strategies of players $\left\{ {i^k}_1, \ldots, {i^k}_r, \ldots, {i^k}_{r_k} \right\}$ in coalition M_k.

$h^{i_r^k} = \left\{ \left(x_{0r}^k, x_{1r}^k\right), \left(x_{1r}^k, x_{2r}^k\right), \ldots, \left(x_{l_r-1}^k, a\right) \right\}$, are the strategies of player i_r^k (inside coalition M_k) and x_{0r}^k is intial postion of player i_r^k inside coalition M_k.

l_r is a number of arc of $h^{i_r^k}$ for player i_r^k inside coalition M_k . The strategies of coalition M_k have the form:

$$h^{M_k} = [\ \left\{ \left(x_{01}^k, x_{11}^k\right), \left(x_{11}^k, x_{21}^k\right), \ldots, \left(x_{l_1-1}^k, a\right) \right\}, \ldots\ldots\ldots\ldots\ldots\ldots$$
$$\left\{ \left(x_{0r}^k, x_{1r}^k\right), \left(x_{1r}^k, x_{2r}^k\right), \ldots, \left(x_{l_r-1}^k, a\right) \right\}, \ldots\ldots,$$
$$\left\{ \left(x_{0r_k}^k, x_{1r_k}^k\right), \left(x_{1r_k}^k, x_{2r_k}^k\right), \ldots, \left(x_{l_{r_k}-1}^k, a\right) \right\}\ ].$$

A bunch of all strategies of M_k we denote by H^{M_k}

3.4 Admissible strategy profiles in Γ_3

The strategy profiles $h^M = \left(h^{M_1}, \ldots, h^{M_p}\right), h^{M_1} \in H^{M_1}, \ldots, h^{M_p} \in H^{M_p}$ are called admissible if the paths $h^{M_{k_i}}$ and $h^{M_{k_j}}$ not intersect (not contain common arcs). $h^{M_{k_i}} \cap h^{M_{k_j}} = \emptyset, k_i \neq k_j$. The set of all admissible strategy profiles is denoted by H^M.

3.5 Cost function in Γ_3

In this suction we define for each arc (x_{fm}^k, x_{f+1m}^k) the values of function $\gamma_i\left({x^k}_{fm}, {x^k}_{f+1m}\right)$ are equal to the cost which necessary to reach the node x_{f+1m}^k from node x_{fm}^k by player M_k (coalitionM_k) is equal to

$$C_{M_k}(h^M) = \sum_{r=1}^{r_k} \sum_{f=0}^{l_m-1} \gamma_i\left({x^k}_{fm}, {x^k}_{f+1m}\right) = C\left(h^{\bar{M}_k}\right) \tag{15}$$

3.6 Nash equilibrium between coalitions $M_1, \ldots, M_k, \ldots, M_p$ in Γ_3(paths of two different coalitions have no common arcs)

In the game Γ_3 the strategy profile $\left(h^{\bar{M}} = \bar{h}^{M_1}, \ldots, \bar{h}^{M_p}\right)$ is called a Nash equilibrium, if $C_{M_k}\left(\overline{h^M} \parallel h^{M_k}\right) \geq C_{M_k}\left(\overline{h^M}\right)$ holds for all admissible strategy profiles $\left(\overline{h^M} \parallel h^{M_k}\right)$

$\in H^M$ and $k \in P$.

Let π be some permutation of numbers $1, \ldots, p, \pi = \left(M_{k_1}, \ldots, M_{k_p}\right)$. Consider an auxiliary transportation problem on the network G for player(coalition) M_{k_1}. Find the path in the network G, minimizing the player (coalition) M_{k_1} cost to each from initial postion to fixed node $a \in X$. Denote the path that solves this problem by $h^{\bar{M}_{k_1}}$

$$C\left(h^{\bar{M}_{k_1}}\right) = \min_{h^{M_{k1}} \in H^{m_{k1}}} C\left(h^{M_{k1}}\right). \tag{16}$$

Remind that the players inside the coalition may use paths with common arcs. Denote by $G \backslash h^{\bar{M}_{k_1}}$ a subnetwork not containing arcs $h^{\bar{M}_{k_1}}$. Consider an auxiliary transportation problem for player(coalition) M_{k_2}on network $G \backslash h^{\bar{M}_{k_1}}$. Find the path in subnetwork $G \backslash h^{\bar{M}_{k_1}}$, which minimizing the player (coalition) M_{k_2} cost to reach from his intial postion to fixed node $a \in X$. Denote the path that solves this problem by $h^{\bar{M}_{k_2}}$

$$C\left(h^{\bar{M}_{k_2}}\right) = \min_{h^{M_{k2}} \in H^{M_{k2}}} C\left(h^{M_{k2}}\right). \tag{17}$$

Proceeding further in a similar way, we introduce into consideration the subnetworks of the network G, that do not containing arcs which belong to strategy path $h^{\bar{M}_{k_1}}$, $\ldots$, $h^{M_{\bar{k}_{m-1}}}$. Consider the auxiliary transportation problem of the player M_{k_m} on the network network $G \backslash \cup_{l=1}^{m-1} h^{\bar{M}_{k_l}}$. Find the subnetwork $G \backslash \cup_{l=1}^{m-1} h^{\bar{M}_{k_l}}$, minimizing the player (coalition) M_{k_m} cost to reach the node $a \in X$.Denote the path that solves this problem by $h^{\bar{M}_{k_m}}$

$$C\left(h^{\bar{M}_{k_m}}\right) = \min_{h^{M_{km}} \in H^{M_{km}}} C\left(h^{M_{km}}\right). \tag{18}$$

As a result, we get a sequence of paths $h^{\bar{M}_{k_1}}, \ldots, h^{\bar{M}_{k_p}}$, minimizing the players (coalitions) $M_{k_1}, M_{k_2}, \ldots, M_{k_m}, \ldots, M_{k_p}$ cost on subnetworks:

$$G, G \backslash h^{\bar{M}_{k_1}}, \ldots, G \backslash \cup_{l=1}^{m-1} h^{\bar{M}_{k_m}}, \ldots, G \backslash \cup_{l=1}^{m-1} h^{\bar{M}_{k_l}}.$$

The sequence of bonages of paths $h^{\bar{M}_{k_1}}, \ldots, h^{\bar{M}_{k_m}}, \ldots, h^{\bar{M}_{k_p}}$ by construction consist of pairwise non-intersecting arcs, and each of them $h^{\bar{M}_{k_l}} \in H^{\bar{M}_{k_l}}$. Therfore the strategy profile $\left(h^{\bar{M}_{k_1}}, \ldots, h^{\bar{M}_{k_m}}, \ldots, h^{\bar{M}_{k_p}}\right) = h^{\bar{M}}(\pi) \in H^M$ is admissible in Γ_3.

3.7 Equilibrium strategy profile

Theorem : The strategy profile $h^{\bar{M}}(\pi) \in H^M$ is an equilibrium strategy profile in Γ_3 for any permutation π.

Proof : Consider the strategy profile. $\left[h^{\bar{M}}(\pi) \| h^{M_{k_m}}\right]$, where $h^{M_{k_m}} \neq \bar{h}^{M_{k_m}}$, $h^{M_{K_m}} \in H^{M_{k_m}}$, $\left[h^{\bar{M}}(\pi) \| h^{M_{k_m}}\right] \in H^M$. By construction $\bar{h}^{M_{k_m}}$ is determined from the condition

$$C\left(\bar{h}^{M_{k_m}}\right) = \min_{h^{M_k} m \in G \setminus U_{l=1}^{m-1} \bar{h}^{M_{k_l}}} C\left(h^{M_{k_m}}\right),$$

However, the strategy profile $\left[h^{\bar{M}}(\pi) \| h^{M_{k_m}}\right]$ is admissible (if $h^{M_{K_m}} \in G \setminus \bigcup_{l=1}^{m-1} \bar{h}^{M_{k_l}}$) and therefore $C\left(\bar{h}^{M_{k_m}}\right) \leq C\left(h^{M_{k_m}}\right) = C_{M_{k_m}}\left[h^{\bar{M}}(\pi) \| h^{M_{k_m}}\right]$, $C\left(\bar{h}^{M_{k_m}}\right) = C_{M_{k_m}}(h^{\bar{M}}(\pi))$, and $C_{M_{k_m}}[h^{\bar{M}}(\pi)] \leq C_{M_{k_m}}\left[h^{\bar{m}}(\pi) \| h^{M_{k_m}}\right]$ for all $\left[h^{\bar{M}}(\pi) \| h^{M_{k_m}}\right] \in H^M$, which proves the theorem.

This theorem indicates a rich family of pure strategy equilibrium profiles in Γ_3 depending on permutation π. Thus, in Γ_3 we have at lest $n!$ equilibrium strategy profiles in pure strategies. If the initial state of players(coalitions) are different.

3.8 Best Nash equilibrium in Γ_3

The strategy profile $h^{\bar{M}}(\hat{\pi})$ is called a best equilibrium if

$$\sum_{k=1}^{p} C_{M_k}(h^{\bar{M}}(\hat{\pi})) = \min_{\pi} \sum_{k=1}^{P} C_{M_k}(h^{\bar{M}}(\pi)) = W_3 \tag{19}$$

3.9 Cooperative solution in game Γ_3

However, there are other Nash equilibrium profiles in Γ_3. Consider the strategy profile $h^{\bar{\bar{M}}}$, solving the minimization problem

$$\min_{h^M} \sum_{k=1}^{P} C_{M_k}(h^M) = \sum_{k=1}^{P} C_{M_k}(h^{\bar{\bar{M}}}) = V_3 \tag{20}$$

We can simply show that $h^{\bar{\bar{M}}}$ is also a Nash equilibrium strategy profile. Because if one player changes his strategy and other players do not change their strategies his time under this condition will be more than equal of his time in case has not changed his strategy. Consider the strategy profile $\left(h^{\bar{\bar{M}}} = h^{\bar{\bar{M}}_1}, \ldots, h^{\bar{\bar{M}}_K}, \ldots, h^{\bar{\bar{M}}_p}\right)$ if player i change his strategy, we get $\sum_{k=1}^{p} C_M(h^{\bar{\bar{M}}} \parallel h^{M_k}) \geq \sum_{k=1}^{p} C_{M_k}(h^{\bar{\bar{M}}})$

$$C(h^{\bar{\bar{M}}_1})+C(h^{\bar{\bar{M}}_2})+...+C(h^{M_k})+...+C(h^{\bar{\bar{M}}_p}) \geq C(h^{\bar{\bar{M}}_1})+C(h^{\bar{\bar{M}}_2})+...+C(h^{\bar{\bar{M}}_k})+...+C(h^{\bar{\bar{M}}_p})$$

so $C(h^{M_k}) \geq C(h^{\bar{\bar{M}}_k})$. We call the strategy profile $h^{\bar{\bar{M}}}$ a cooperative equilibrium in Γ_3. In some cases $V_3 = W_3$, (see the example)

3.10 Optimal cooperative trajectory.

Remind the definition of cooperative path (coalition) (3.9)

$$\bar{\bar{h}}^M = [\{(\bar{\bar{x}}_{01}^{M_1}, \bar{\bar{x}}_{11}^{M_1}), (\bar{\bar{x}}_{11}^{M_1}, \bar{\bar{x}}_{21}^{M_1}), \ldots, (\bar{\bar{x}}_{l_1-1}^{M_1}, a)\}, \ldots \left\{\left(\bar{\bar{x}}_{0i}^{M_k}, \bar{\bar{x}}_{1}^{M_k}\right), \left(\bar{\bar{x}}_{1k}^{M_k}, \bar{\bar{x}}_{2k}^{M_k}\right), \ldots, (\bar{\bar{x}}_{l_k-1}^{M_1}, a)\right\}, \ldots$$

$\left\{\left(\bar{\bar{x}}_{0p}^{M_p}, \bar{\bar{x}}_{1p}^{M_1}\right), \left(\bar{\bar{x}}_{1p}^{M_p}, \bar{\bar{x}}_{2p}^{M_p}\right), \ldots, \left(\bar{\bar{x}}_{l_p-1}^{M_p}, a\right)\right\}]$, where $L = \max_{1 \leq k \leq p} l_k$.

Denote $\bar{\bar{\bar{x}}}(r)$ cooperative trajectories corresponding to cooperative path $\bar{\bar{h}}^M$.

$$\bar{\bar{\bar{x}}} = (\bar{\bar{x}}_{01}^{M_1}, \bar{\bar{x}}_{11}^{M_1}, \bar{\bar{x}}_{21}^{M_1}, \ldots, \bar{\bar{x}}_{l_1-1}^{M_1}, a), \ldots (\bar{\bar{x}}_{0k}^{M_k}, \bar{\bar{x}}_{1k}^{M_k}, \bar{\bar{x}}_{2k}^{M_k}, \ldots, \bar{\bar{x}}_{l_k-1}^{M_k}, a), \ldots (\bar{\bar{x}}_{0p}^{M_p}, \bar{\bar{x}}_{1p}^{M_p}, \bar{\bar{x}}_{2p}^{M_p}, \ldots, \bar{\bar{x}}_{l_p-1}^{M_p}, a)$$

The subgame starting from state $\bar{\bar{\bar{x}}}(r) = (\bar{\bar{x}}_{r1}^{M_1}, \ldots, \bar{\bar{x}}_{rk}^{M_k}, \ldots, \bar{\bar{x}}_{rp}^{M_p})$, where $\bar{\bar{x}}_{rk}^{M_k} = (\bar{\bar{x}}_{0k}^{M_k}, \bar{\bar{x}}_{1k}^{M_k}, \bar{\bar{x}}_{2k}^{M_k}, \dots, \bar{\bar{x}}_{l_k-1}^{M_k}, a), k = 1, \dots, P$, and r stage number for players(coalitions).

3.11 The proportional solution for coalition in game Γ_3

In the cooperative version of the game between coalitions we suppose that all players (coalitions) jointly minmize the total costs and this minimal total cost we denote by $V(P)$. As previous section the problem how to allocate this total minimal cost between players (coalitions). In our sitting we will use as optimality principle the proportional solution[17].We have $p-$player (coalitions) in Γ_3 want to reach the fixed node in network in minimal cost (sum of the costs necessary to reach the fixed node by all players(coalitions)). In such way that the corresponding paths of coalitions do not contain common arcs. The proportional solution defined as (see [17]): The proportional solution in cooperative gameγ_3 is defined in a classical way:

$$\tilde{\varphi}_{M_k}(\bar{\bar{\bar{x}}}(r), r) = \frac{V(M_k; \bar{\bar{\bar{x}}}(r), r)}{\sum_{k=1}^{p} V(M_k; \bar{\bar{\bar{x}}}(r), r)} V(P; \bar{\bar{\bar{x}}}(r), r); \quad K \in P$$

$\tilde{\varphi}_{M_k}(\bar{\bar{\bar{x}}}(r), r)$:is the proportional solution for player M_k along his trajectories $\bar{\bar{\bar{x}}}(r)$.
$V(P; \bar{\bar{\bar{x}}}(r), r)$: is a minimal total cost for all players jointly (cooperative solution) along cooperative trajectories $\bar{\bar{\bar{x}}}(r)$.
$V(M_k; \bar{\bar{\bar{x}}}(r), r)$: is a minimal total cost for player M_k along cooperative trajectories $\bar{\bar{\bar{x}}}(r)$.
It is shown on example $\tilde{\varphi}_{M_k}(\bar{\bar{\bar{x}}}(0), 0) \neq \tilde{\varphi}_{M_k}(\bar{\bar{\bar{x}}}(1), 1)) +$ (one cost out).

3.12 The Shapley value in game Γ_3

Let we have $V(S); S \subset P$ and $V(M_1), V(M_2)$ where $V(M_1) + V(M_2) \geqslant V(P)$ and $V(S \cup T) \leqslant V(S) + V(T)$, And $p = |P|, S = |S|$ where $S \subset P$, And $S \cap T = \emptyset$

The Shapley value $Sh = \{Sh_{M_k}\}_{k \in N}$ in cooperative game Γ_3 is a vector, such that(see[16]):

$$Sh_{M_k}(\bar{\bar{\bar{x}}}(r),r) = \sum_{M_k \in S \subset P} \frac{(p-s)!(s-1)!}{p!} \left(V\left(S,\bar{\bar{\bar{x}}}(r),r\right) - V\left(S\backslash\{M_k\},\bar{\bar{\bar{x}}}(r),r\right)\right) \tag{21}$$

$Sh_{M_k}(\bar{\bar{\bar{x}}}(r),r)$:is the Shapley value for player M_k along his trajectories $\bar{\bar{\bar{x}}}(r)$.

$V(S;\bar{\bar{\bar{x}}}(r),r)$: is a minimal total cost for subset of players jointly (cooperative solution) along cooperative trajectories $\bar{\bar{\bar{x}}}(r)$.

$V\left(S\backslash\{M_k\},\bar{\bar{\bar{x}}}(r),r\right)$: is minimal total cost for all subset of players(coalitions) jointly (cooperative solution) without player M_k along his trajectories $\bar{\bar{\bar{x}}}(r)$.

If we have 2 players(coalitions) the formula of the Shapley value will be:

$$Sh_{M_1}(\bar{\bar{\bar{x}}}(r),r) = V(M_1,\bar{\bar{\bar{x}}}(r),r) - \frac{V(M_1,\bar{\bar{\bar{x}}}(r),r) + V(M_2,\bar{\bar{\bar{x}}}(r),r) - V((M_1,M_2),\bar{\bar{\bar{x}}}(r),r)}{2}$$
$$Sh_{M_2}(\bar{\bar{\bar{x}}}(r),r) = V(M_2,\bar{\bar{\bar{x}}}(r),r) - \frac{V(M_2,\bar{\bar{\bar{x}}}(r),r) + V(M_1,\bar{\bar{\bar{x}}}(r),r) - V((M_1,M_2),\bar{\bar{\bar{x}}}(r),r)}{2}$$

And we will get $Sh_{M_1}(\bar{\bar{\bar{x}}}(r),r) + Sh_{M_2}(\bar{\bar{\bar{x}}}(r),r) = V((M_1,M_2),\bar{\bar{\bar{x}}}(r),r)$

How we defined the value of $V(S); S \subset P$ in game if $P = \{1,2\}$

Table 2:

Value of $V(S)$; $S \in P$			
FIRST CASE	$(P\|S)$ then S	$V^T(M_1)$	The value at $\pi = (2,1)$
		$V^T(M_2)$	The value at $\pi = (1,2)$
SECOND CASE	S then $(P\|S)$	$V^T(M_1)$	The value at $\pi = (1,2)$
		$V^T(M_2)$	The value at $\pi = (2,1)$

It is shown on example the characteristic function of the Shapely value is not time consistent in Γ_3 .

$Sh_{M_k}(\bar{\bar{\bar{x}}}(0),0)) \neq Sh_{M_k}(\bar{\bar{\bar{x}}}(1),1)+$ one cost out

3.13 Two stage solution concept in Γ_3

We consider the solution by the following :

First approach:cooperative game between players (coalitions), then find Proportional solution $\tilde{\varphi}_{M_k}$ in Γ_3.This solution consider as loses of every given coalition , then the problem how to distribute this loses between members of coalition. For this reason we compute the Shapley value and it is necessary to define the characteristic function for players inside the coalition.The characteristic function is defined in following way : suppose $S \subset P$ then $V(S)$ can be taken as the loses of S in some fixed Nash equilibrium (under fixed permutation) in the game played by (coalitions) S with other players as individual players(we may suppose that the strategies of players do not have common arcs).Denote the Shapley value inside coalition is $sh_i(M_k)$;$M_k \subset S$. We decide to allocate the loses as

$$\psi_i(M_k) = \frac{sh_i(M_k)}{\sum_{i=1}^{p_k} sh_i(M_k)} \tilde{\varphi}_{M_k}; \quad k \in \{1, ..., p\}. \tag{22}$$

Second approach: cooperative game between players (coalition), then find the Shapley value sh_{M_k} in Γ_3.This solution consider as loses every given coalition , then the problem how to distribute this loses between members of coalition. For this reason we compute the proportional solution it is necessary to define the characteristic function for players inside the coalition.The characteristic function is defined in following way : suppose $S \subset P$ then $V(S)$ can be taken as the loses of coalition S in some fixed Nash equilibrium (under fixed permutation) in the game played by (coalitions) S with other players as individual players(we may suppose that the strategies of players do not have common arcs).Denote the Proportional solution inside coalition is $\tilde{\varphi}_i(M_k)$;$M_k \subset S$ .We decide to allocate the loses as

$$\theta_i(M_k) = \frac{\tilde{\varphi}_i(M_k)}{\sum_{i=1}^{p_k} \tilde{\varphi}_i(M_k)} sh_{M_k}; \quad k \in \{1, ..., p\}. \tag{23}$$

3.14 Example (time consistency problem game Γ_3):

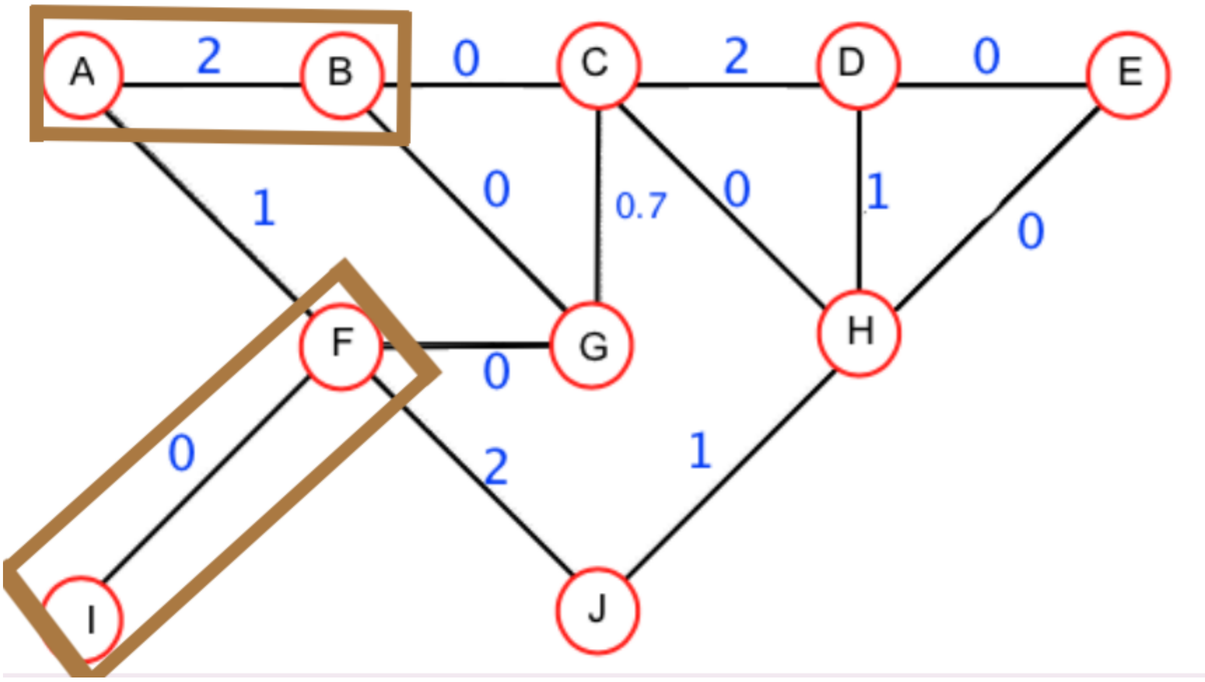


Figure 35: two player in game Γ_3

In this figure we denote nodes by capital Latin letters.$P = \{1, 2\}$ the coalitions $M = \{M_1, M_2\}$; $M_1 = A, B, M_2 = I, F$

Two player (coalitions)want to reach the fixed node E under condition (paths have no common arcs).

The transportation times are written in the network in this figure over the arcs and are equal, respectively to

$$\gamma(A,B)=2,\gamma(A,F)=1,\gamma(B,C)=0,\gamma(B,G)=0,$$
$$\gamma(C,D)=2,\gamma(C,H)=0,\gamma(C,G)=0.7,\gamma(D,E)=0,$$
$$\gamma(D,H)=1,\gamma(I,F)=0,\gamma(F,G)=0,\gamma(F,J)=2,$$
$$\gamma(J,H)=1,\gamma(H,E)=0,$$

. Non- cooperative solution

For permuation : $\pi=\{1,2\}$

$$h^{\bar{M}_1}=[(A,F),(F,G)(G,B),(B,C)(C,H),(H,E)],[(B,C),(C,H)(H,E)]$$
$$C_{M_1}(h^{\bar{M}})=1+0=1$$
$$h^{\bar{M}_2}=[(I,F),(F,J)(J,H),(H,D)(D,E)],[(F,J)(J,H),(H,D)(D,E)]$$
$$C_{M_2}(h^{\bar{M}})=4+4=8$$

For permuation : $\pi=\{2,1\}$

$$h^{\bar{M}_2}=[(I,F),(F,G)(G,B),(B,C)(C,H),(H,E)],[(F,G)(G,B),(B,C)(C,H),(H,E)]$$
$$C_{M_2}(h^{\bar{M}})=0+0=0$$
$$h^{\bar{M}_1}=[(A,F),(F,J)(J,H),(H,D)(D,E),(H,E)],$$
$$[(B,A),(A,F),(F,J)(J,H),(H,D)(D,E),(H,E)]$$
$$C_{M_1}(h^{\bar{M}})=5+7=12$$

Thus , both equilibrium $h^{\bar{M}}(2,1)$ and $h^{\bar{M}}(1,2)$ are conditionally cooperative equilibrium (best Nash equilibrium) in Γ_3 and get $W_3=9$

Cooperative solution

$$h^{\bar{\bar{M}}_1}=[(A,B),(B,C)(C,D),(D,E)],[(B,C)(C,D),(D,E)]$$
$$C_{M_1}(h^{\bar{M}})=4+2=6$$
$$h^{\bar{\bar{M}}_2}=[(I,F),(F,G)(G,C),(C,H)(H,E)],[(F,G)(G,C),(C,H)(H,E)]$$
$$C_{M_2}(h^{\bar{M}})=0.7+0.7=1.4$$
$$C_{M_1}(h^{\bar{M}})+C_{M_2}(h^{\bar{M}})=6+1.4=7.4=V_3$$

We get the result $V_3 < W_3$

The proportional solution in game Γ_3

At r $= 0$,$\pi = (1, 2)$

$$\tilde{\varphi}_{M_1}(\bar{\bar{\bar{x}}}(0), 0) = (1/9)7.4 = 0.822, \quad \tilde{\varphi}_{M_2}(\bar{\bar{\bar{x}}}(0), 0) \quad = (8/9)7.4 = 6.578$$

At r $= 0$,$\pi = (2, 1)$

$$\tilde{\varphi}_{M_1}(\bar{\bar{\bar{x}}}(0), 0) = (12/12)7.4 = 7.4, \quad \tilde{\varphi}_{M_2}(\bar{\bar{\bar{x}}}(0), 0) \quad = (0/12)7.4 = 0$$

At r $= 1$,$\pi = (1, 2)$

$$\tilde{\varphi}_{M_1}(\bar{\bar{\bar{x}}}(1), 1) = (0/6)5.4 = 0, \quad \tilde{\varphi}_{M_2}(\bar{\bar{\bar{x}}}(1), 1) \quad = (6/6)5.4 = 5.4$$

At r $= 1$,$\pi = (2, 1)$

$$\tilde{\varphi}_{M_1}(\bar{\bar{\bar{x}}}(1), 1) = (9/9)5.4 = 5.4, \quad \tilde{\varphi}_{M_2}(\bar{\bar{\bar{x}}}(0), 0) \quad = (0/12)5.4 = 0$$

Compare the results

$$\tilde{\varphi}_{M_1}(\bar{\bar{\bar{x}}}(1), 1) + 1 = 1 \neq \tilde{\varphi}_{M_1}(\bar{\bar{\bar{x}}}(0), 0) = 0.822$$
$$\tilde{\varphi}_{M_2}(\bar{\bar{\bar{x}}}(1), 1) + 2 = 7.4 \neq \tilde{\varphi}_{M_2}(\bar{\bar{\bar{x}}}(0), 0) = 6.578$$

$$\tilde{\varphi}_{M_1}(\bar{\bar{\bar{x}}}(1), 1) + 3 = 8.4 \neq \tilde{\varphi}_{M_1}(\bar{\bar{\bar{x}}}(0), 0) = 7.4$$
$$\tilde{\varphi}_{M_2}(\bar{\bar{\bar{x}}}(1), 1) + 0 = 0 = \tilde{\varphi}_{M_2}(\bar{\bar{\bar{x}}}(0), 0)$$

The characteristic function of the proportional solution is not time consistent in Γ_3

.

The Shapley value in game Γ_3

At r = 0 ,$\pi = (1,2)$

$$Sh_{M_1}(\bar{\bar{\bar{x}}}(0),0) = 12 - \frac{12+8-7.4}{2} = 5.7$$
$$Sh_{M_2}(\bar{\bar{\bar{x}}}(0),0) = 8 - \frac{8+12-7.4}{2} = 1.7$$

At r = 0 ,$\pi = (2,1)$

$$Sh_{M_1}(\bar{\bar{\bar{x}}}(0),0) = 1 - \frac{1+0-7.4}{2} = 4.2$$
$$Sh_{M_2}(\bar{\bar{\bar{x}}}(0),0) = 0 - \frac{0+1-7.4}{2} = 3.2$$

At r = 1 ,$\pi = (1,2)$

$$Sh_{M_1}(\bar{\bar{\bar{x}}}(1),1) = 9 - \frac{9+6-5.7}{2} = 2.85$$
$$Sh_{M_2}(\bar{\bar{\bar{x}}}(1),1) = 6 - \frac{6+9-5.7}{2} = 1.35$$

At r = 1 ,$\pi = (2,1)$

$$Sh_{M_1}(\bar{\bar{\bar{x}}}(1),1) = 1 - \frac{1+0-5.7.4}{2} = 3.35$$
$$Sh_{M_2}(\bar{\bar{\bar{x}}}(1),1) = 0 - \frac{0+1-5.7}{2} = 2.35$$

Compare the results

$$Sh_{M_1}(\bar{\bar{\bar{x}}}(1),1) + 1 = 2.85 + 1 = 3.85 \neq 5.7 = Sh_{M_1}(\bar{\bar{\bar{x}}}(0),0)$$
$$Sh_{M_2}(\bar{\bar{\bar{x}}}(1),1) + 2 = 1.35 + 2 = 3.35 \neq 1.7 = \mathrm{Sh}_{\mathrm{M}_2}(\bar{\bar{\bar{x}}}(0),0)$$

$$Sh_{M_1}(\bar{\bar{\bar{x}}}(1),1) + 3 = 3.35 + 3 = 6.35 \neq 4.2 = Sh_{M_1}(\bar{\bar{\bar{x}}}(0),0)$$
$$Sh_{M_2}(\bar{\bar{\bar{x}}}(1),1) + 0 = 2.35 + 0 = 2.35 \neq 3.2 = \mathrm{S}\,h_{M_1}(\bar{\bar{\bar{x}}}(0),0)$$

The characteristic function of the Shapley value is not time consistent in Γ_3 .

Two stage solution concept in Γ_3

In the case best Nash equilibrium $\pi = (1,2)$ we get :

$sh_1(M_1) = 1, sh_2(M_1) = 0, sh_1(M_2) = 4, sh_2(M_2) = 4,$

$$\tilde{\varphi}_{M_1} = 0.822,\ \tilde{\varphi}_{M_2} = 6.578$$

So $\psi_1(M_1) = (0.822)(1) = 0.822, \quad \psi_2(M_1) = (0.82)(0) = 0$

$\psi_1(M_2) = (6.578)(4/8) = 3.289, \quad \psi_2(M_2) = (6.578)(4/8) = 3.289$

$$\tilde{\varphi}_1(M_1) = 1, \tilde{\varphi}_2(M_1) \quad = 0, \quad \tilde{\varphi}_3(M_2) = 4, \tilde{\varphi}_2(M_2) \quad = 4,$$

$$h_{M_1} = 5.7,\ sh_{M_2} = 1.7$$

So $\theta_1(M_1) = (5.7)(1/1) = 5.7, \quad \theta_2(M_1) = (5.7)(0) = 0$

$\theta_1(M_2) = (1.7)(4/8) = 0.85, \quad \theta_2(M_2) = (1.7)(4/8) = 0.85$

Reference

[1] Ferreira F. P. DISJOINT PATH PAIR CALCULATION CONSIDERING BANDWIDTHS, da Cruz, 2014.

https://estudogeral.uc.pt/bitstream/10316/40425/1/Disjoint%20path%20pair%20calculation%20considering%20bandwidths.pdf

[2] V. Mazalov and J.V. Chirkova, Networking Games: Network Forming Games and Games on Networks, Academic Press, 2019.

https://www.elsevier.com/books/networking-games/mazalov/978-0-12-816551-5

[3] Seryakov I. A. Game-theoretical transportation model with limited tracffic capacities, , at. Teor. Igr Pril., 2012, Volume 4, Issue 3, Pages 101116.

http://www.mathnet.ru/php/archive.phtml?wshow=paper&jrnid=mgta&paperid=91&option_lang=eng

[4] Petrosyan, L. (1977). Stable Solutions of Differential Games with Many Participants. Viestnik of Leningrad University, 19, 46–52.

[5] Petrosyan, L. and Danilov, N. N. (1979). Stability of Solutions in Nonzero Sum Differential Games with Transferable Payoffs. Journal of Leningrad University N1, 52–59 (in Russian).

[6] Petrosyan, L. and Zaccour, G. (2003). Time-Consistent Shapley Value of Pollution Cost Reduction. Journal of Economic Dynamics and Control, 27, 381–398.

[7] Yeung, D.W. K., Petrosyan, L. (2005). Subgame Consistent Economic Optimization Springer, New York, NY.

[8] Gao, H., Petrosyan, L., Sedakov, A. (2014). Strongly Time-Consistent Solutions for TwoStage Network Games. Procedia Computer Science, 31, 255–264.

[9] Петросян Л. А. Одна транспортная теоретико-игровая модель на сети - 2011, 3:4, 89–98.

http://www.mathnet.ru/php/archive.phtml?wshow=paper&jrnid=mgta&paperid=70&option_lang=rus

[10] Карпов.М. И. , Петросян.Л. А. Кооперативные решения в коммуникационных сетях.

http://www.mathnet.ru/php/archive.phtml?jrnid=vspui&paperid=92&wshow=paper&option_lang=rus

[11] Серяков.Илья.А. Теоретико-игровая транспортная задача на сети с заданными пропускными способностями.

http://www.mathnet.ru/php/archive.phtml?wshow=paper&jrnid=mgta&paperid=91&option_lang=rus

[12] Серяков.Илья .А Кооперативные решения в коммуникационных сетях .

http://www.mathnet.ru/php/archive.phtml?wshow=paper&jrnid=mgta&paperid=91&option_lang=rus

[13] Mazalov.Vladimir, Chirkova.Julia Networking Games1st Edition.

https://www.elsevier.com/books/networking-games/mazalov/978-0-12-816551-5

[14] G.BergantiñosaM. Gómez-RúaaN. LlorcabM.PulidocJ. Sánchez-Sorianob. A new rule for source connection problems.

https://www.sciencedirect.com/science/article/abs/pii/S0377221713008060

[15] Epsteina. Amir, Feldmanb. Michal, Mansoura.Yishay Mansoura.Strong equilibrium in cost sharing connection games.

https://www.sciencedirect.com/science/article/abs/pii/S0899825608001383

[16] Shapley . L. S., A value for n-person games, in Contributions to the Theory of Games, vol. 2, Annals of Mathematics Studies, 28, Princeton University Press, Princeton, NJ, 1953, 307–317.

https://www.degruyter.com/document/doi/10.1515/9781400881970-018/html

[17] Feldman Barry the Proportional Value of a Cooperative Game ,December 10, 1999 .

http://fmwww.bc.edu/RePEc/es2000/1140.pdf

[18] Kazutoshi Ando, Koichi Takase MONTE CARLO ALGORITHM FOR CALCULATING THE SHAPLEY VALUES OF MINIMUM COST SPANNING TREE GAMES.

https://www.jstage.jst.go.jp/article/jorsj/63/1/63_31/_article

Appendix A

The minimum time algorithm for one player in Γ_1

```
# --- Modules --- #
from sys import maxsize
from heapq import heapify, heappush
from json import loads
from os import path

# -------------------------------------------------- #
FILE_PATH = path.dirname(path.abspath(__file__))
# -------------------------------------------------- #

# Debugging function
def debug(current_node, nodes, visited, min_heap):
print(f"=====> Node {current_node} <=====")
for node, data in nodes.items():
print(f"---- Node {node} ----")
for key, value in data.items():
print(f"{key} = {value}")
print(f"visited -> {visited}")
print(f"min_heap -> {min_heap}")
print('------------------------ Next ------------------------')
```

```
# Print results
def print_results(result, source, destination, N=40):

sp_cost, sp = result

print(f" === ({source}) --> ({destination}) ===")

if (sp_cost == maxsize):
print("Cost = infinty")
print("Can't find path")
print("="*N)
else:
shortest_path = " -> ".join(sp)
print(f"Cost = {sp_cost}")
print(f"Shortest path = {shortest_path}")
print("="*N)
# ------------------------------------------------------------ #

# MAIN ALGORITHM
def dijkstra(graph, src, dest, Debug=False):

# Make node data
all_nodes = set()
nodes = {}
for node in graph:
nodes[node] = {"cost": maxsize, "pred": []}
all_nodes.add(node)

# Assign cost for source point to 0
nodes[src]["cost"] = 0

# visited nodes
visited = set()
```

```

# Assign node to src
node = src
for _ in range(len(nodes) - 1):

if (node not in visited):
visited.add(node)

# Get current node cost
current_cost = nodes[node]["cost"]

# Create "Min␣Heap"
min_heap = []

# Check all neighbors
neighbors = graph[node]
for neighbor, distance in neighbors.items():

if (neighbor not in visited):
old_cost = nodes[neighbor]["cost"]
cost = current_cost + distance

# Change node cost
if (cost < old_cost):
nodes[neighbor]["cost"] = cost
nodes[neighbor]["pred"] = nodes[node]["pred"] + [node]

heappush(min_heap, (nodes[neighbor]["cost"], neighbor))

# Check if heap is empty to push unvisited nodes to it
if (len(min_heap) == 0):
not_visited = list(all_nodes - visited)
for node in not_visited:
heappush(min_heap, (nodes[node]["cost"], node))
```

```
heapify(min_heap)

# Debug
if Debug:
debug(node, nodes, visited, min_heap)

# Reassign source node
node = min_heap[0][1]

# ------------ Return Results ------------ #
shortest_path_cost = nodes[dest]["cost"]
shortest_path = nodes[dest]["pred"] + [dest]

return shortest_path_cost, shortest_path


if (__name__ == "__main__"):

network_file = input("Network␣file:␣").strip()
source = input("Source:␣").strip().title()

# Input
with open(path.join(FILE_PATH, f"{network_file}.json"), "r") as f:
graph = loads(f.read())

destinations = [d for d in graph]

print("*"*60)

# Dijkstra
for destination in destinations:

result = dijkstra(graph, source, destination)
```

```
        # print results
        print_results(result, source, destination)


        input("press any key to exit ...")
```

Appendix B

The minimum time algorithm for $n-$player case in Γ_1(best Nash equilibrium(arcs))

```
# —— Modules —— #
from sys import maxsize
from heapq import heapify, heappush
from json import loads
from os import path

# ---------------------------------------------------- #
FILE_PATH = path.dirname(path.abspath(__file__))
# ---------------------------------------------------- #

# Debugging function
def debug(current_node, nodes, visited, min_heap):
print(f"====> Node {current_node} <====")
for node, data in nodes.items():
print(f"---- Node {node} ----")
for key, value in data.items():
print(f"{key} = {value}")
```

```
print(f"visited -> {visited}")
print(f"min_heap -> {min_heap}")
print('------------------------- Next -------------------------')

# Print results
def print_results(result, source, destination, N=40):

sp_cost, sp = result

print(f" === ({source}) ---> ({destination}) ===")

if (sp_cost == maxsize):
print("Cost = infinity")
print("Can't find path")
print("="*N)
else:
shortest_path = " -> ".join(sp)
print(f"Cost = {sp_cost}")
print(f"Shortest path = {shortest_path}")
print("="*N)
# ---------------------------------------------------------- #

# MAIN ALGORITHM
def dijkstra(graph, src, dest, Debug=False):

# Make node data
all_nodes = set()
nodes = {}
for node in graph:
nodes[node] = {"cost": maxsize, "pred": []}
all_nodes.add(node)

# Assign cost for source point to 0
nodes[src]["cost"] = 0
```

```
# visited nodes
visited = set()

# Assign node to src
node = src
for _ in range(len(nodes) - 1):

if (node not in visited):
visited.add(node)

# Get current node cost
current_cost = nodes[node]["cost"]

# Create "Min␣Heap"
min_heap = []

# Check all neighbors
neighbors = graph[node]
for neighbor, distance in neighbors.items():

if (neighbor not in visited):
old_cost = nodes[neighbor]["cost"]
cost = current_cost + distance

# Change node cost
if (cost < old_cost):
nodes[neighbor]["cost"] = cost
nodes[neighbor]["pred"] = nodes[node]["pred"] + [node]

heappush(min_heap, (nodes[neighbor]["cost"], neighbor))

# Check if heap is empty to push unvisited nodes to it
if (len(min_heap) == 0):
```

```
not_visited = list(all_nodes - visited)
for node in not_visited:
heappush(min_heap, (nodes[node]["cost"], node))

heapify(min_heap)

# Debug
if Debug:
debug(node, nodes, visited, min_heap)

# Reassign source node
node = min_heap[0][1]

# ---------- Return Results ---------- #
shortest_path_cost = nodes[dest]["cost"]
shortest_path = nodes[dest]["pred"] + [dest]

return shortest_path_cost, shortest_path


if (__name__ == "__main__"):

network_file = input("Network file: ").strip()
source = input("Source: ").strip().title()

# Input
with open(path.join(FILE_PATH, f"{network_file}.json"), "r") as f:
graph = loads(f.read())

destinations = [d for d in graph]

print("*"*60)

# Dijkstra
```

```
        for destination in destinations:

        result = dijkstra(graph, source, destination)

        # print results
        print_results(result, source, destination)


        input("press any key to exit ...")
```

Appendix C

The minimum time algorithm for $n-$ player case in Γ_1(cooperative solution (arcs))

```
# — Modules — #
from heapq import heappush, heapify, nsmallest

from os import path
from json import loads

from dijkstra import dijkstra

from itertools import permutations
# ===========================================================================
FILE_PATH = path.dirname(__file__)
# ===========================================================================

# — FUNCTIONS — #
def intercept(path1, path2):
for p1 in path1:
for p2 in path2:
```

```
if (sorted(p1) == sorted(p2)):
return True
return False

def get_valid_paths(path1, path2):
result = []
for i, p1 in enumerate(path1):
for j, p2 in enumerate(path2):
if not intercept(p1[1], p2[1]):
res = (p1[0] + p2[0], (i, j))
heappush(result, res)
heapify(result)

return result

def pathify(path, reverse=False):
result = []
if not reverse:
for i in range(len(path) - 1):
result.append((path[i], path[i+1]))

else:
for p in path:
result.append(p[0])
result.append(path[-1][-1])

return result

def result_next(paths, source1, source2, results):
r_next = []
for result in results:
cost = result[0]
i1 = result[1][0]
i2 = result[1][1]
```

```
p = (cost, paths[source1][i1][1] + paths[source2][i2][1])

r_next.append(p)

return r_next

def print_result(path, source, destination, N=40):
cost = path[0]
path = pathify(path[1], reverse=True)

print(f" === ({source}) --> ({destination}) ===")

shortest_path = " -> ".join(path)
print(f"Cost = {cost}")
print(f"Shortest path = {shortest_path}")
print("="*N)

def dijkstra_help(graph, sources, destination):

cases = list(permutations(sources, source_points))

# Fill empty cost dictionary
costs = dict()
for num in range(1, len(cases) + 1):
costs[f"Case {num}"] = list()

for num, case in enumerate(cases, start=1):

prev_path = []
for source in case:

deleted_nodes = [n for n in case if n != source]

```

```
result = dijkstra(graph, source, destination, deleted_nodes, prev_path
costs[f"Case {num}"].append(result[0])

# Next step
prev_path += [None] + result[1]

costs[f"Case {num}"] = sum(costs[f"Case {num}"])

final_costs = []
for _, cost in costs.items():
final_costs.append(cost)

return min(final_costs)

# DFS Algorithm
def all_possible_paths(graph, src, dest, min_cost):

result = []
def dfs(path, cost, src):

# check if reached distance
if src == dest:
final_path = pathify(path + [src])
heappush(result, (cost, final_path))

else:
for neighbour, n_cost in graph[src].items():
current_cost = n_cost + cost
if (neighbour not in path) and (current_cost <= min_cost):
dfs(path + [src], current_cost,  neighbour)

dfs([], 0, src)

heapify(result)
```

```
return result


if __name__ == '__main__':

# — input — #
network_file = input("Network:␣").strip()

with open(path.join(FILE_PATH, f'{network_file}.json'), "r") as f:
graph = loads(f.read())


source_points = int(input("Source␣points␣number:␣").strip())
sources = []
for s in range(source_points):
source = input(f"Source␣({s+1}):␣").strip().title()
sources.append(source)

destination = input("Destination:␣").strip().title()

print("*"*60)

# Get Disjkstra minimum cost
min_cost = dijkstra_help(graph, sources, destination)

# Get all possible paths
paths = dict()
for source in sources:
paths[f"{source}"] = all_possible_paths(graph, source, destination, m

# Get all non intercepted paths
result = paths[sources[0]]
result_next_paths = result
for i in range(source_points - 1):
```

```
result = get_valid_paths(result_next_paths, paths[sources[i+1]])

# if not in final loop, check path with next source
if (i != source_points - 2):
result_next_paths = result_next(paths, sources[i], sources[i+1], resu


# Get minimum costs
if (len(result) == 0):
print(" couldn't find a cooperative path :(")
input("\npress any key to exit ...")
exit()

min_cost = nsmallest(1, result)[0]
indexes = [min_cost[1]]
for r in result:
if (r != min_cost) and (r[0] == min_cost[0]):
indexes.append(tuple(r[1]))

# Print results
for index in indexes:
for num, source in enumerate(sources):
print_result(paths[source][index[num]], source, destination)
if (index != indexes[-1]):
print("-"*25, "OR", "-"*25)


final_cost = min_cost[0]
print(f"\n - Final Cost = {final_cost}")


input("\npress any key to exit ...")
```

Appendix D

The algorithm for $n-$ player case in Γ_1(cooperative solution as mini maximal time (arcs))

```
# —— Modules —— #
from sys import maxsize
from heapq import heapify, heappush
from json import loads
from os import path

from itertools import permutations
# ———————————————————————————————————————————— #
FILE_PATH = path.dirname(path.abspath(__file__))
# ———————————————————————————————————————————— #

def get_conn(node, prev_path):

if node not in prev_path:
return []

max_index = len(prev_path) - 1
```

```
min_index = 0
node_index = prev_path.index(node)

if (node_index == min_index):
return [prev_path[node_index + 1]]
elif (node_index == max_index):
return [prev_path[node_index - 1]]
else:
return [prev_path[node_index - 1], prev_path[node_index + 1]]

# Debugging function
def debug(current_node, nodes, visited, min_heap):
print(f"====> Node {current_node} <====")
for node, data in nodes.items():
print(f"--- Node {node} ---")
for key, value in data.items():
print(f"{key} = {value}")
print(f"visited -> {visited}")
print(f"min_heap -> {min_heap}")
print('---------------------------------------- Next ----------------------------------------')

# Print results
def print_results(result, source, destination, N=40):

sp_cost, sp = result

print(f" === ({source}) --> ({destination}) ===")

if (sp_cost == maxsize):
print("Time = infinty")
print("Can't find path")
print("="*N)
else:
shortest_path = " -> ".join(sp)
```

```
print(f"Time␣=␣{sp_cost}")
print(f"Shortest␣path␣=␣{shortest_path}")
print("="*N)

def get_case(case):
case = map(str, case)
case = ",␣".join(case)
return f"{{{case}}}"

# ------------------------------------------------------------ #

# MAIN ALGORITHM
def dijkstra(graph, src, dest, deleted_nodes=[], prev_path=[], Debug=False):

# Make node data
all_nodes = set()
nodes = {}
for node in graph:
nodes[node] = {"cost": maxsize, "pred": []}
all_nodes.add(node)

# Assign cost for source point to 0
nodes[src]["cost"] = 0

# visited nodes
if deleted_nodes:
visited = set(deleted_nodes)
else:
visited = set()

# Assign node to src
node = src
for _ in range(len(nodes) - len(deleted_nodes) - 1):

```

```
if (node not in visited):
visited.add(node)

# Get current node cost
current_cost = nodes[node]["cost"]

# Create "Min␣Heap"
min_heap = []

# check in previous path
paths = get_conn(node, prev_path)

# Check all neighbors
neighbors = graph[node]
for neighbor, distance in neighbors.items():

if (neighbor not in visited) and (neighbor not in paths):
old_cost = nodes[neighbor]["cost"]
cost = current_cost + distance

# Change node cost
if (cost < old_cost):
nodes[neighbor]["cost"] = cost
nodes[neighbor]["pred"] = nodes[node]["pred"] + [node]

heappush(min_heap, (nodes[neighbor]["cost"], neighbor))

# Check if heap is empty to push unvisited nodes to it
if (len(min_heap) == 0):
not_visited = list(all_nodes - visited)
for node in not_visited:
heappush(min_heap, (nodes[node]["cost"], node))

heapify(min_heap)
```

```
# Debug
if Debug:
debug(node, nodes, visited, min_heap)

# Reassign source node
node = min_heap[0][1]

# ———— Return Results ———— #
shortest_path_cost = nodes[dest]["cost"]
shortest_path = nodes[dest]["pred"] + [dest]

return shortest_path_cost, shortest_path


if (__name__ == "__main__"):

# — input — #

network_file = input("Network file: ").strip()

source_points = int(input('Players number: ').strip())
sources = []
for s in range(source_points):
source = input(f"Player ({s+1}): ").strip().title()
sources.append(source)

# source = input(f"Source: ").strip().title()
destination = input("Fixed vertex: ").strip().title()

# open network file
with open(path.join(FILE_PATH, f"{network_file}.json"), "r") as f:
graph = loads(f.read())
```

```
print("*"*60)

# -- Dijkstra -- #

cases = list(permutations(sources, source_points))
cases_num = list(permutations(range(1, source_points + 1), source_points))

# Fill empty cost dictionary
costs = dict()
for num in range(1, len(cases) + 1):
costs[f"Case {num}"] = list()

for num, case in enumerate(cases, start=1):

print('-'*26)

prev_path = []
for source in case:

deleted_nodes = [n for n in case if n != source]

result = dijkstra(graph, source, destination, deleted_nodes, prev_path)
costs[f"Case {num}"].append(result[0])

# print results
print_results(result, source, destination)

# Next step
prev_path += [None] + result[1]



# Get final cost
final_cost = min([max(cost) for _, cost in costs.items()])
if (final_cost == maxsize):
```

```
        final_cost = "infinty"
        print(f"\n - Final Time = {final_cost}")

        input('\npress any key to exit ...')
```

Appendix E

The minimum time algorithm for $n-$ player case in Γ_1(best Nash equilibrium (vertices))

```
# —— Modules —— #
from sys import maxsize
from heapq import heapify, heappush
from json import loads
from os import path

from itertools import permutations
# ———————————————————————————————————————————————— #
FILE_PATH = path.dirname(path.abspath(__file__))
# ———————————————————————————————————————————————— #

def get_conn(node, prev_path):

if node not in prev_path:
return []

max_index = len(prev_path) - 1
```

```
min_index = 0
node_index = prev_path.index(node)

if (node_index == min_index):
return [prev_path[node_index + 1]]
elif (node_index == max_index):
return [prev_path[node_index - 1]]
else:
return [prev_path[node_index - 1], prev_path[node_index + 1]]

# Debugging function
def debug(current_node, nodes, visited, min_heap):
print(f"====> Node {current_node} <====")
for node, data in nodes.items():
print(f"---- Node {node} ----")
for key, value in data.items():
print(f"{key} = {value}")
print(f"visited -> {visited}")
print(f"min_heap -> {min_heap}")
print('------------------------- Next -------------------------')

# Print results
def print_results(result, source, destination, N=40):

sp_cost, sp = result

print(f" === ({source}) --> ({destination}) ===")

if (sp_cost == maxsize):
print("Time = infinity")
print("Can't find path")
print("="*N)
else:
shortest_path = " -> ".join(sp)
```

```
print(f"minimum␣time␣=␣{sp_cost}")
print(f"␣path␣=␣{shortest_path}")
print("="*N)

def get_case(case):
case = map(str, case)
case = ",␣".join(case)
return f"{{{case}}}"
# ---------------------------------------------------------------- #

# MAIN ALGORITHM
def dijkstra(graph, src, dest, deleted_nodes=[], Debug=False):

# Make node data
all_nodes = set()
nodes = {}
for node in graph:
nodes[node] = {"cost": maxsize, "pred": []}
all_nodes.add(node)

# Assign cost for source point to 0
nodes[src]["cost"] = 0

# visited nodes
if deleted_nodes:
visited = set(deleted_nodes)
else:
visited = set()

# Assign node to src
node = src
for _ in range(len(nodes) - len(deleted_nodes) - 1):

if (node not in visited):
```

```
visited.add(node)

# Get current node cost
current_cost = nodes[node]["cost"]

# Create "Min␣Heap"
min_heap = []

# Check all neighbors
neighbors = graph[node]
for neighbor, distance in neighbors.items():

if (neighbor not in visited):
old_cost = nodes[neighbor]["cost"]
cost = current_cost + distance

# Change node cost
if (cost < old_cost):
nodes[neighbor]["cost"] = cost
nodes[neighbor]["pred"] = nodes[node]["pred"] + [node]

heappush(min_heap, (nodes[neighbor]["cost"], neighbor))

# Check if heap is empty to push unvisited nodes to it
if (len(min_heap) == 0):
not_visited = list(all_nodes - visited)
for node in not_visited:
heappush(min_heap, (nodes[node]["cost"], node))

heapify(min_heap)

# Debug
if Debug:
debug(node, nodes, visited, min_heap)
```

```python

# Reassign source node
node = min_heap[0][1]

# ————————— Return Results ————————— #
shortest_path_cost = nodes[dest]["cost"]
shortest_path = nodes[dest]["pred"] + [dest]

return shortest_path_cost, shortest_path


if (__name__ == "__main__"):

# — input — #

network_file = input("Network file: ").strip()

source_points = int(input('Players number: ').strip())
sources = []
for s in range(source_points):
source = input(f"Player ({s+1}): ").strip().title()
sources.append(source)

destination = input("Fixed node: ").strip().title()

# open network file
with open(path.join(FILE_PATH, f"{network_file}.json"), "r") as f:
graph = loads(f.read())

print("*"*60)

# — Dijkstra — #

cases = list(permutations(sources, source_points))
```

```
cases_num = list(permutations(range(1, source_points + 1), source_points))

# Fill empty cost dictionary
costs = dict()
for num in range(1, len(cases) + 1):
costs[f"Case {num}"] = list()

for num, case in enumerate(cases, start=1):

print('-'*26)

deleted_nodes = []
for source in case:

deleted_sources = [n for n in case if n != source]

result = dijkstra(graph, source, destination, deleted_nodes + deleted_sources)
costs[f"Case {num}"].append(result[0])

# print results
print_results(result, source, destination)

# Next step
deleted_nodes += result[1][1:-1]


# Get final cost
final_cost = min([sum(cost) for _, cost in costs.items()])
if (final_cost == maxsize):
final_cost = "infinity"
print(f"\n - Minimal of maximal time = {final_cost}")

input('\npress any key to exit ...')
```

Appendix F

The algorithm for $n-$ player case in Γ_1(cooperative as mini maximal time (vertices))

```
# —— Modules —— #
from sys import maxsize
from heapq import heapify, heappush
from json import loads
from os import path

from itertools import permutations
# ———————————————————————————————————————————————— #
FILE_PATH = path.dirname(path.abspath(__file__))
# ———————————————————————————————————————————————— #

def get_conn(node, prev_path):

if node not in prev_path:
return []

max_index = len(prev_path) - 1
```

```
min_index = 0
node_index = prev_path.index(node)

if (node_index == min_index):
return [prev_path[node_index + 1]]
elif (node_index == max_index):
return [prev_path[node_index - 1]]
else:
return [prev_path[node_index - 1], prev_path[node_inde

# Debugging function
def debug(current_node, nodes, visited, min_heap):
print(f"====> Node {current_node} <====")
for node, data in nodes.items():
print(f"---- Node {node} ----")
for key, value in data.items():
print(f"{key} = {value}")
print(f"visited -> {visited}")
print(f"min_heap -> {min_heap}")
print('------------------------- Next ------------------------

# Print results
def print_results(result, source, destination, N=40):

sp_cost, sp = result

print(f" === ({source}) --> ({destination}) ===")

if (sp_cost == maxsize):
print("Time = infinty")
print("Can't find path")
print("="*N)
else:
shortest_path = " -> ".join(sp)
```

```
print(f"Time = {sp_cost}")
print(f"Shortest path = {shortest_path}")
print("="*N)

def get_case(case):
case = map(str, case)
case = ", ".join(case)
return f"{{{case}}}"
# ------------------------------------------------------------------ #

# MAIN ALGORITHM
def dijkstra(graph, src, dest, deleted_nodes=[], Debug

# Make node data
all_nodes = set()
nodes = {}
for node in graph:
nodes[node] = {"cost": maxsize, "pred": []}
all_nodes.add(node)

# Assign cost for source point to 0
nodes[src]["cost"] = 0

# visited nodes
if deleted_nodes:
visited = set(deleted_nodes)
else:
visited = set()

# Assign node to src
node = src
for _ in range(len(nodes) - len(deleted_nodes) - 1):

if (node not in visited):
```

```
visited.add(node)

# Get current node cost
current_cost = nodes[node]["cost"]

# Create "Min␣Heap"
min_heap = []

# Check all neighbors
neighbors = graph[node]
for neighbor, distance in neighbors.items():

if (neighbor not in visited):
old_cost = nodes[neighbor]["cost"]
cost = current_cost + distance

# Change node cost
if (cost < old_cost):
nodes[neighbor]["cost"] = cost
nodes[neighbor]["pred"] = nodes[node]["pred"] + [node

heappush(min_heap, (nodes[neighbor]["cost"], neighbor

# Check if heap is empty to push unvisited nodes to i
if (len(min_heap) == 0):
not_visited = list(all_nodes - visited)
for node in not_visited:
heappush(min_heap, (nodes[node]["cost"], node))

heapify(min_heap)

# Debug
if Debug:
debug(node, nodes, visited, min_heap)
```

```
# Reassign source node
node = min_heap[0][1]

# ------------ Return Results ------------ #
shortest_path_cost = nodes[dest]["cost"]
shortest_path = nodes[dest]["pred"] + [dest]

return shortest_path_cost, shortest_path


if (__name__ == "__main__"):

# -- input -- #

network_file = input("Network file: ").strip()

source_points = int(input('Players number: ').strip())
sources = []
for s in range(source_points):
source = input(f"Player ({s+1}): ").strip().title()
sources.append(source)

destination = input("Fixed vertex: ").strip().title()

# open network file
with open(path.join(FILE_PATH, f"{network_file}.json")
graph = loads(f.read())

print("*"*60)

# -- Dijkstra -- #

cases = list(permutations(sources, source_points))
```

```
cases_num = list(permutations(range(1, source_points -

# Fill empty cost dictionary
costs = dict()
for num in range(1, len(cases) + 1):
costs[f"Case {num}"] = list()

for num, case in enumerate(cases, start=1):


print('-'*26)

deleted_nodes = []
for source in case:

deleted_sources = [n for n in case if n != source]

result = dijkstra(graph, source, destination, deleted_
costs[f"Case {num}"].append(result[0])

# print results
print_results(result, source, destination)

# Next step
deleted_nodes += result[1][1:-1]


# Get final cost
final_cost = min([max(cost) for _, cost in costs.items
if (final_cost == maxsize):
final_cost = "infinity"
print(f"\n - Final Time = {final_cost}")

input('\npress any key to exit ... ')
```