**AI – Assignment 1**

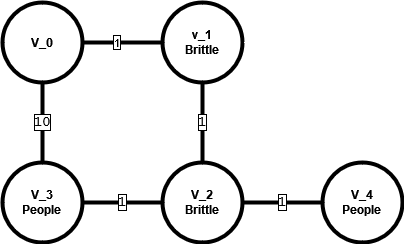
Pan Eyal – 208722058

**Question 1**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Poker Game | Home Repair | Covid-19 Response | Cryptarithmetic Puzzles | Sokoban Puzzles |
| Observable | Partially | Yes | Partially | Yes | Yes |
| Deterministic | No | Yes | No | Yes | Yes |
| Episodic | No | No | No | No | No |
| Static | Yes | No | No | Yes | Yes |
| Discrete | Yes | No | No | Yes | Yes |
| Single-Agent | No | Yes | No | Yes | Yes |
| Agent-Type | Utility-based | Goal-based | Utility-based | Goal-based | Utility-based |

**Question 2**

Question 6 settings:



Let us represent a state as:  
where , and

The heuristic is: edge weight sum of the minimum spanning tree that containing only vertices that need saving (connecting neighbors of nonessential vertices as necessary, as shown in class).

The initial state is: and

For all agents, is retrieved, and it is not a goal state, so it is expanded with:

Now, for the agents:

1. **Greedy search**

Greedy search takes as it has the lowest and return with . In next search it expands with:

Greedy search takes as it has the lowest (the state that would have returned to would cause an entry of duplicate state and therefore ignored) and return with . In next search it expands with:

Greedy search takes as it has the lowest and return with . Now it sees that it reached a goal state and terminates.

1. **A\* search**

A\* search takes as it has the lowest and expands with:

A\* search takes as it has the lowest (the state that would have returned to would cause an entry of duplicate state and therefore ignored) and now expands with:

A\* takes as it breaks ties with lower number vertex, and expands with:

A\* search takes since it is the lowest , and can’t expand since broke. is not a goal but it is empty, so A\* takes , and expands with:

A\* search takes as it has the lowest (the state that would have returned to would cause an entry of duplicate state and therefore ignored) and now expands with:

A\* search takes as it has the lowest , sees that it is a goal state,  
returns the sequence: and terminates.

1. **Real-Time A\***

RTA\* search choose as it has the lowest and expands with:

RTA\* takes as it has the lowest (the state that would have returned to would cause an entry of duplicate state and therefore ignored) and runs out of allowed expansions. It returns with the sequence: . In next search it expands with:

RTA\* takes as it breaks ties with lower number vertex, and expands with:

RTA\* search takes since it is the lowest , and runs out of allowed expansions. It returns with the sequence: . In next search it can’t expand since already broke.   
RTA\* does not reach the goal!

1. **Now with**

It makes no difference for greedy search in this example and in this domain, that is because won’t cause any node queue ordering ().

and

For all agents, is retrieved, and it is not a goal state, so it is expanded with:

It makes no difference for greedy search in this example and in this domain, that is because won’t cause any change in the node queue ordering.

For A\*: A\* search takes as it has the lowest and expands with:

A\* search takes as it has the lowest (the state that would have returned to would cause an entry of duplicate state and therefore ignored) and now expands with:

A\* takes as it breaks ties with lower number vertex, and expands with:

A\* search takes since it is the lowest , and can’t expand since broke. is not a goal but it is empty, so A\* takes , and expands with:

A\* search takes as it has the lowest (the state that would have returned to would cause an entry of duplicate state and therefore ignored) and now expands with:

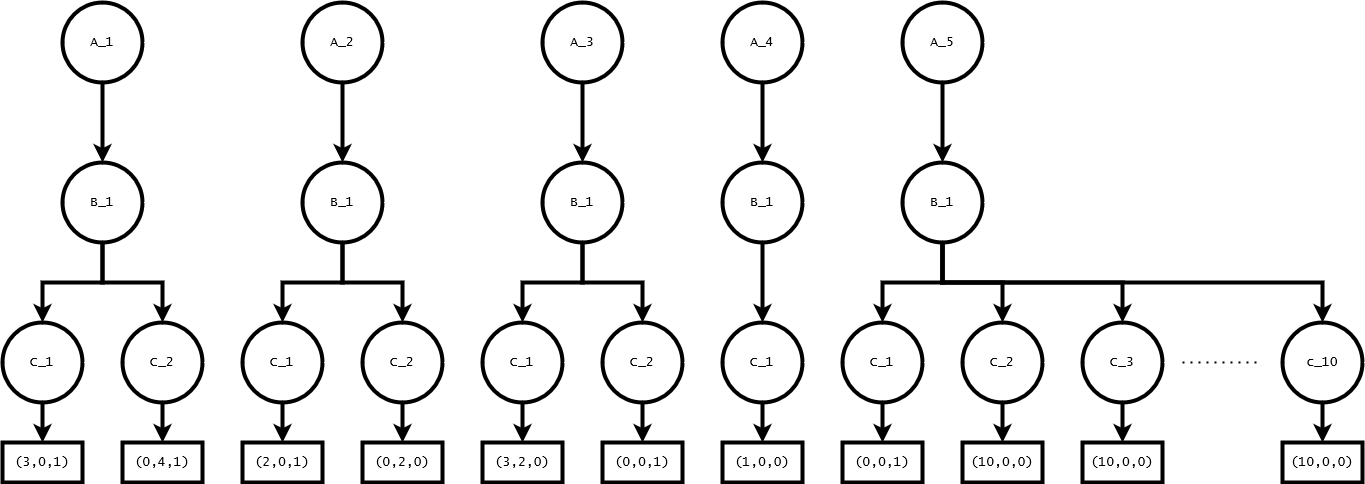
A\* search takes as it has the lowest , sees that it is a goal state,  
returns the sequence: and terminates.

RTA\* will behave similarly as A\* did.

The multiplication variable wasn’t enough to surpass the part in and cause the algorithm to choose different path.

heuristic is not admissible! We can see in state for example that where .

**Question 3**

Consider the following tree. The scores represented at the end as:

In the case where:

1. **Each agent out for itself, and they cannot communicate:**

In , agent C will choose as ties broken adversarially (he will prefer one player with score 3, instead of one player with score 4).

That means agent A will receive a score of 3.

In all other choices:

because in max score of agent A is 2

because in agent C will maximize his own score and choose .

because in max score of agent A is 1

because in agent C will maximize his own score and choose .

1. **As in a, except B and C are semi-cooperative:**

In , agent C will choose as it is prioritizing itself.

That means agent A will receive a score of 2.

In all other choices:

because in agent C will break ties with coop and choose .

because in agent C will maximize his own score and choose .

because in max score of agent A is 1

because in agent C will maximize his own score and choose .

1. **As in a, except B and C are partners aiming to maximize the sum of their scores:**

In , agent C will choose as it is prioritizing the sum of itself and agent B.

That means agent A will receive a score of 3.

In all other choices:

because in agent C will maximize his and agent B scores and choose .

because in agent C will maximize his and agent B scores and choose .

because in max score of agent A is 1

because in agent C will maximize his and agent B scores and choose .

1. **Paranoid assumption: B and C are against A, no matter what their score is:**

In , agent A will receive a score of 1.

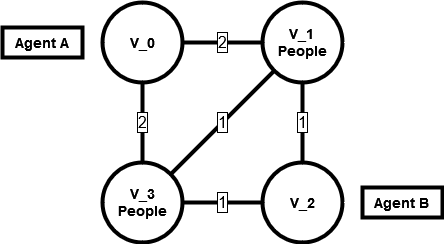
In all other choices agent A can receive a score of 0, and agent C will choose that option.

1. **As in a, except C plays randomly with uniform distribution:**In , agent A will receive an expected score of

In all other choices for agent , his score will be lower than 9.

**Question 4**

1. Example settings: Our graph contains 4 vertices with edges as follows:



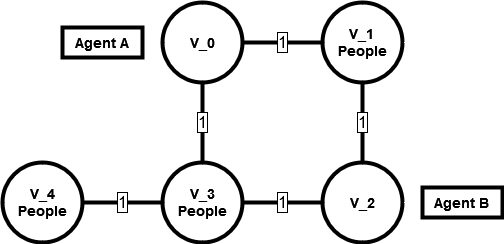
Also, we have agents A that starts in and agent B that starts in , with A moving first.

Agent A starts in vertex and agent B starts in vertex .

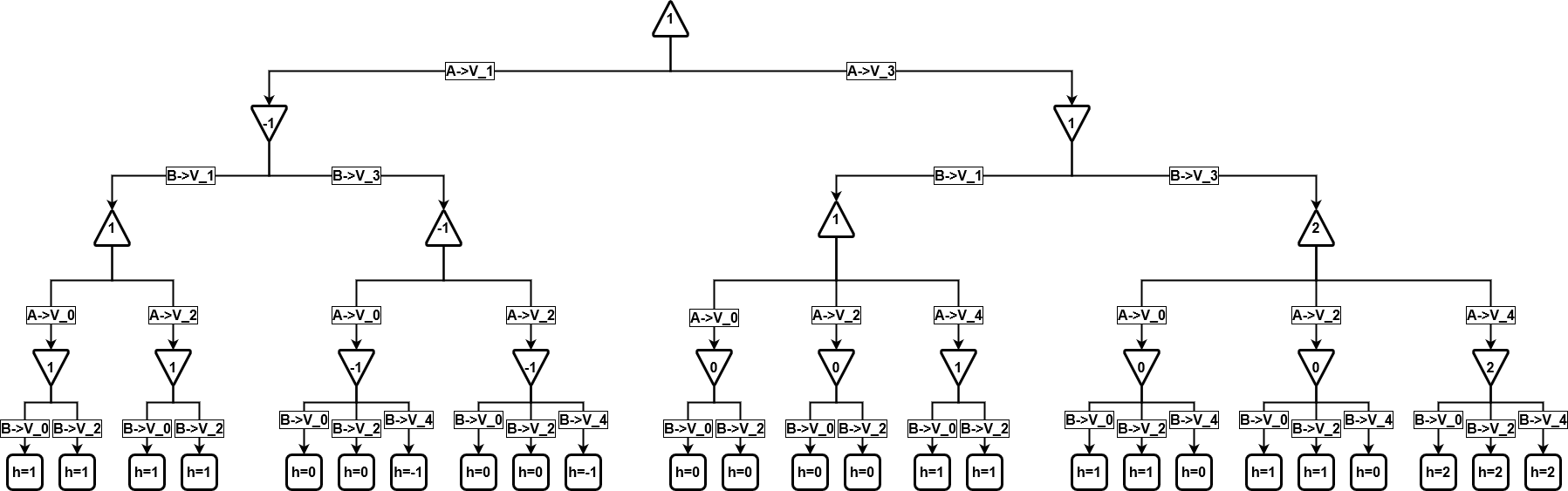
We can easily see that if agent A starts moving towards or , agent B will be able to move to the same location, reach there faster and save the people before agent A. and then also reach the people on the other vertex as well. In this scenario agent A remains with a score of 0.

If agent A starts with *no\_op* action, Agent B will move towards one of the vertices, and then agent A will be able to move to the other location, race agent B, and get a score as well. In this scenario agent A gains a score of 1 and ties with agent B.

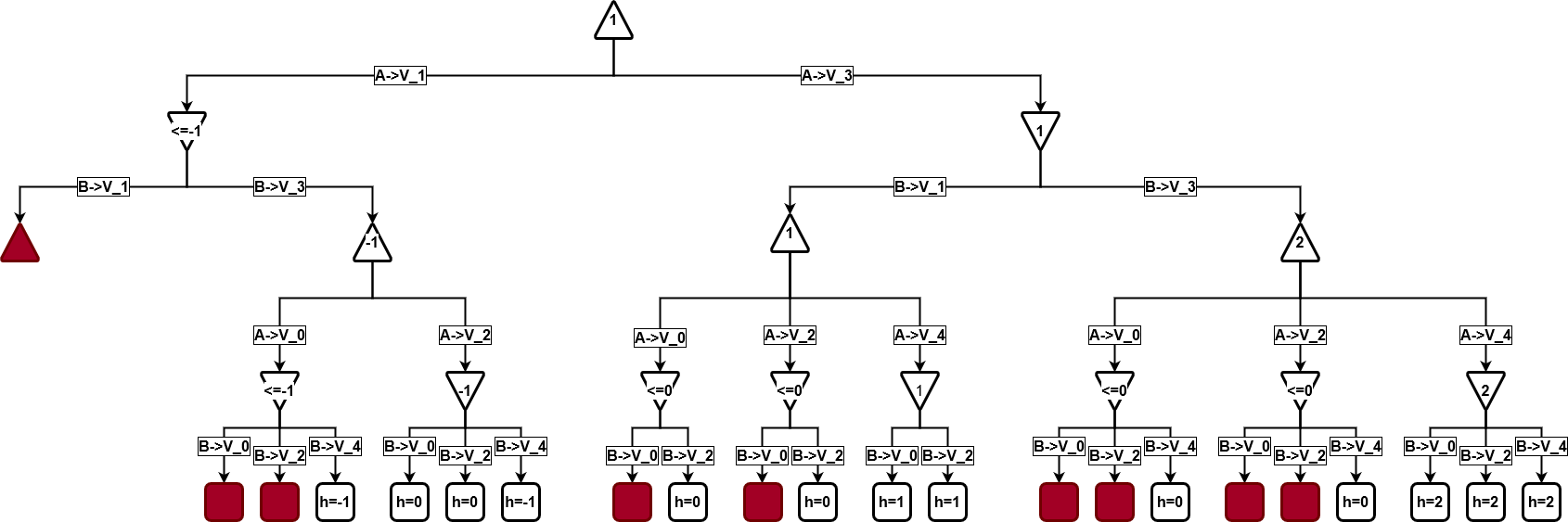
1. Example settings: Our graph contains 5 vertices with edges as follows:



Also, we have agents A that starts in and agent B that starts in , with A moving first.

The static heuristic function will be the score achieved until current state:

agent A want to maximize it and agent B wants to minimize it.  
We will perform minimax on this graph and receive the following tree for agent A:

1. Now for alpha-beta pruning, red slots in graph mean pruning had occurred. We use the assumption that we evaluate from right to the left (just because more pruning will happen):

**Question 5**

Satisfiable 3 models:

* through are true and is true.
* through are true and is false.
* through are false.

Satisfiable models:

* All assignments except either all propositions true or all propositions false.

Unsatisfiable 0 models:

* Unsatisfiable, because we have a conjunction with the unsatisfiable (A and not A).

Satisfiable models:

* B is entailed by and C is entailed by and entailed by.

Unsatisfiable, because this is a negation of a valid sentence.

Satisfiable models:

* is false, is true.
* is false, is false.
* is true, is true.

**Question 6**

1. Predicates:

We will also define edges and their weights for no-op operations:

And define trivial inequalities:

In this world, we are dealing with integers as constants, eg. . To represent calculations based on these constants, we’ll define predicates to incorporate all the options for “integer” constants in our KB. For example, .

Initial fluents:

Loc(V0, S0)

U(V1, S0), U(V2, S0) and auxiliary: U(V0, S0), U(V3, S0) U(V4, S0)

PeopleAt(V3, S0), PeopleAt(V4, S0)

Time(0, S0)

Frame axioms:

1. 1) Equal to themselves

2) Not equal with each other

Create for every and rule:

1. Undirected graph
2. Can’t be in two places at once
3. People saved only when visiting their vertex
4. Vertices break only when visiting their vertex
5. Non-brittle vertices do not break
6. Broken vertices stay broken

Effect axioms:

1. Legal traversal
2. Traversal generate a legal location
3. Saving people:
4. Breaking vertices:
5. Part A – conversion to CNF
6. 1) Equal to themselves

2) Not equal with each other

Create for every and rule:

1. Undirected graph
2. Can’t be in two places at once
3. People saved only when visiting their vertex
4. Vertices break only when visiting their vertex
5. Non-brittle vertices do not break
6. Broken vertices stay broken
7. Legal traversal
8. Traversal generate a legal location
9. Saving people:
10. Breaking vertices:

Part B – Formally defining the query:

Query:

Let split this query in two and try to prove each in it’s turn:

Negation:

Part C – Using resolutions to reach a contradiction:

**Proving**

Res:

Res:



Res:

Res:

Res:

Res:



**Proving**

Res:

Res:

Res:

Res:

Res:

Res:

Res:

Res:

Res:

Res:

Res:

Res:

Res:

Res:

Res:

Res:

Res:

Res:

Res:

Res:

Res:

Res:

Res:

Res:

Res:

Res:

Res:

Res:

Res:

1. We saw that we just couldn’t complete the proof without frame axioms during the resolution.
2. To be able to prove in forward chaining, we must use only Horn Form. We use negations in the knowledge base so forward chaining is not possible.