STARCRAFT II

Artificial Intelligence project

Cinvestav Guadalajara

Carlos Cardenas
Emilio Tonix
Julia Abud

Part 4

Bayesian Network

TASKS

27 February 2021 - 15 March 2021

- Implement
  - Bayesian network

Map

Map: Simple64

Description

This map is consist in a map medium map size with four camps that contains minerals and vespene gas.

This map is for two players that try to destroy each other.

Initial State

- 1 command center
- 12 SCV

Rewards

- Win and keep your live

End Conditions

- Destroy the enemy

Time Limit

- No

Additional Notes

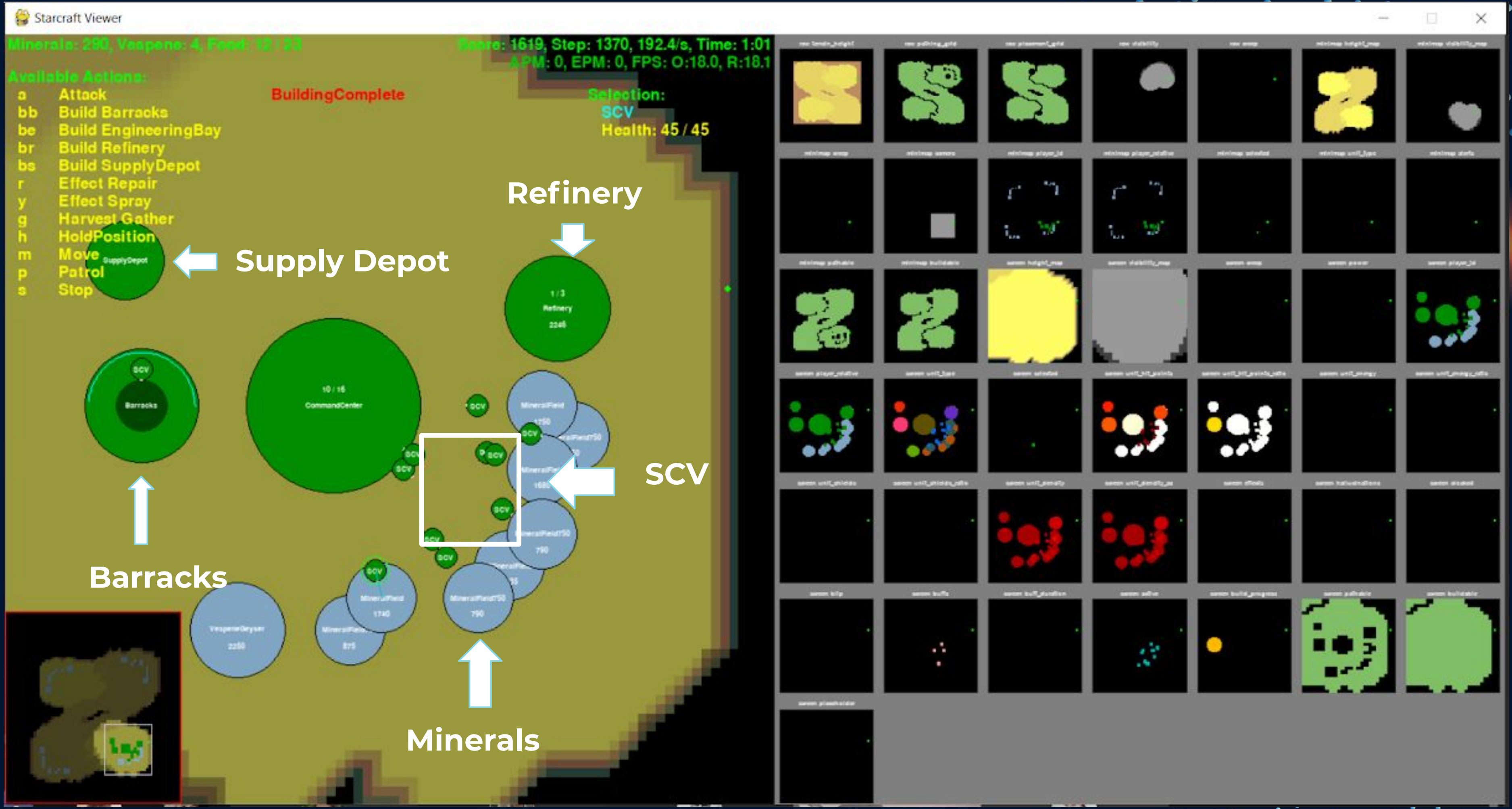
- We manually activated fog of war
- And visualize features

```
def main(unused_argv):
  agent1 = BayesAgent()
  agent2 = PacifistaAgent()
  try:
    with sc2_env.SC2Env(
        map_name="Simple64",
        players=[sc2_env.Agent(sc2_env.Race.terran),
                 sc2_env.Agent(sc2_env.Race.terran)],
        agent_interface_format=features.AgentInterfaceFormat(
            action_space=actions.ActionSpace.RAW,
            use_raw_units=True,
            raw_resolution=64,
            feature_dimensions=features.Dimensions(screen=84, minimap=64),
            use_feature_units=True
        ),
        step_mul=10,
        disable_fog=False,
        visualize=True #visualize: Whether to pop up a window showing the came
    ) as env:

      #agent1.setup(env.observation_spec(), env.action_spec())
      run_loop.run_loop([agent1, agent2], env, max_episodes=1000)

  except KeyboardInterrupt:
    pass
```

Agent 1
Basic Rush Strategy + Bayesian network for scouting

Agent 2
Pacifist

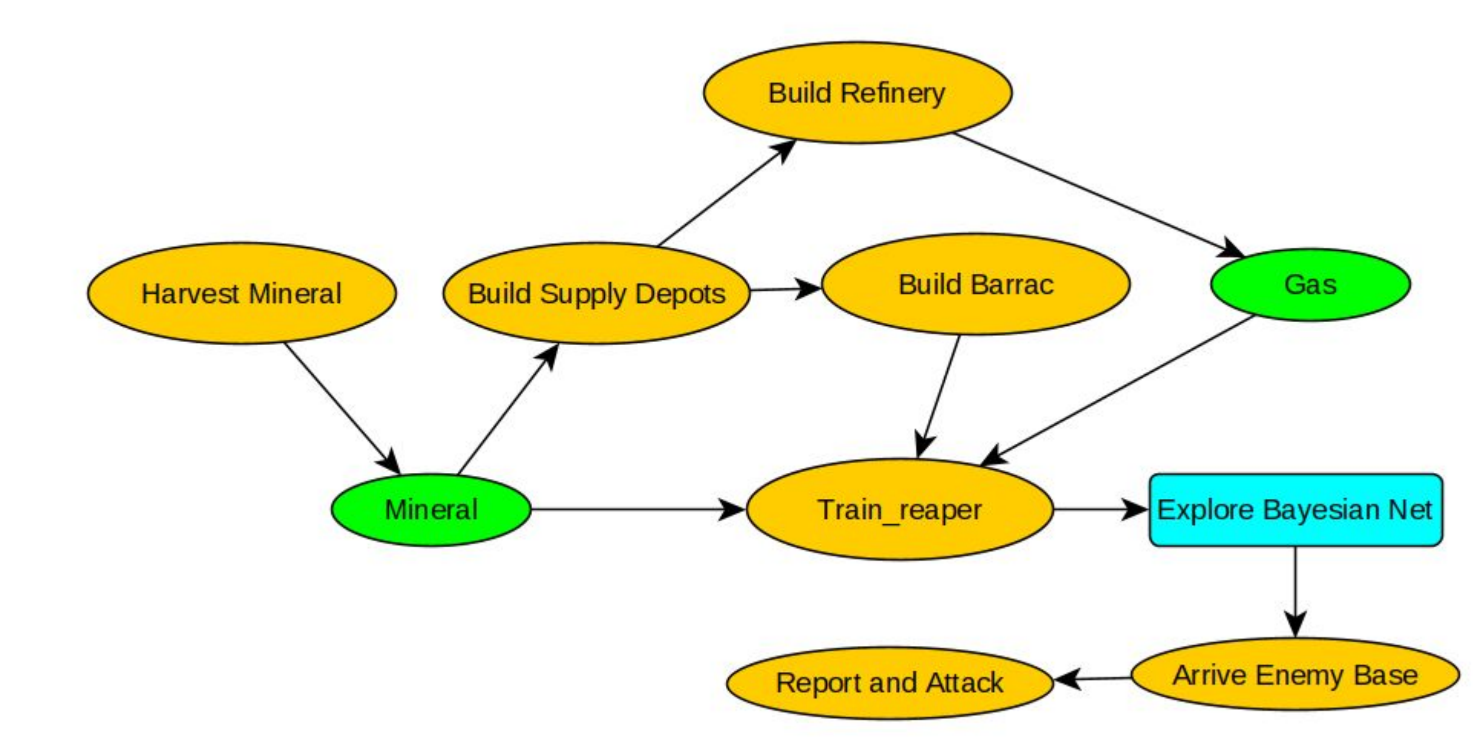
Camp Positions
TopLeft and BottomRight
(any agent can have any of this positions)



Starcraft Viewer
Score: 1619, Step: 1370, 192.4/s, Time: 1:01
APM: 0, EPM: 0, FPS: O:18.0, R:18.1
Available Actions:
a Attack
bb Build Barracks
be Build EngineeringBay
br Build Refinery
bs Build SupplyDepot
r Effect Repair
y Effect Spray
g Harvest Gather
h HoldPosition
m Move
p Patrol
s Stop
BuildingComplete
Selection:
SCV
Health: 45 / 45
Refinery
Supply Depot
SCV
Barracks
Minerals

Rush strategy (basic)

Agent 1 : Rush strategy (part1)

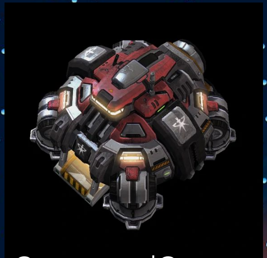


Units used

SCV

Marine

Reaper



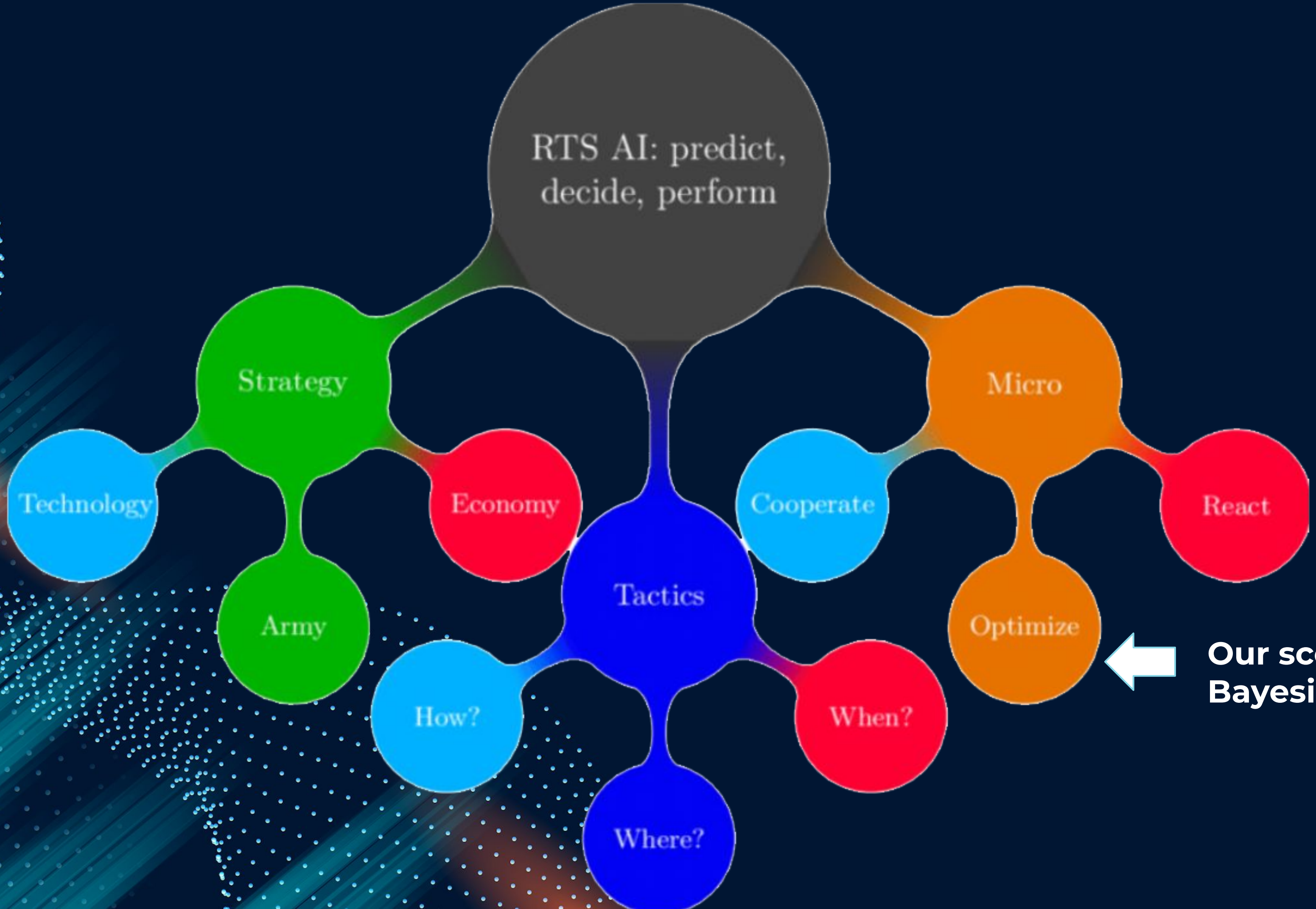
CommandCenter

Supply

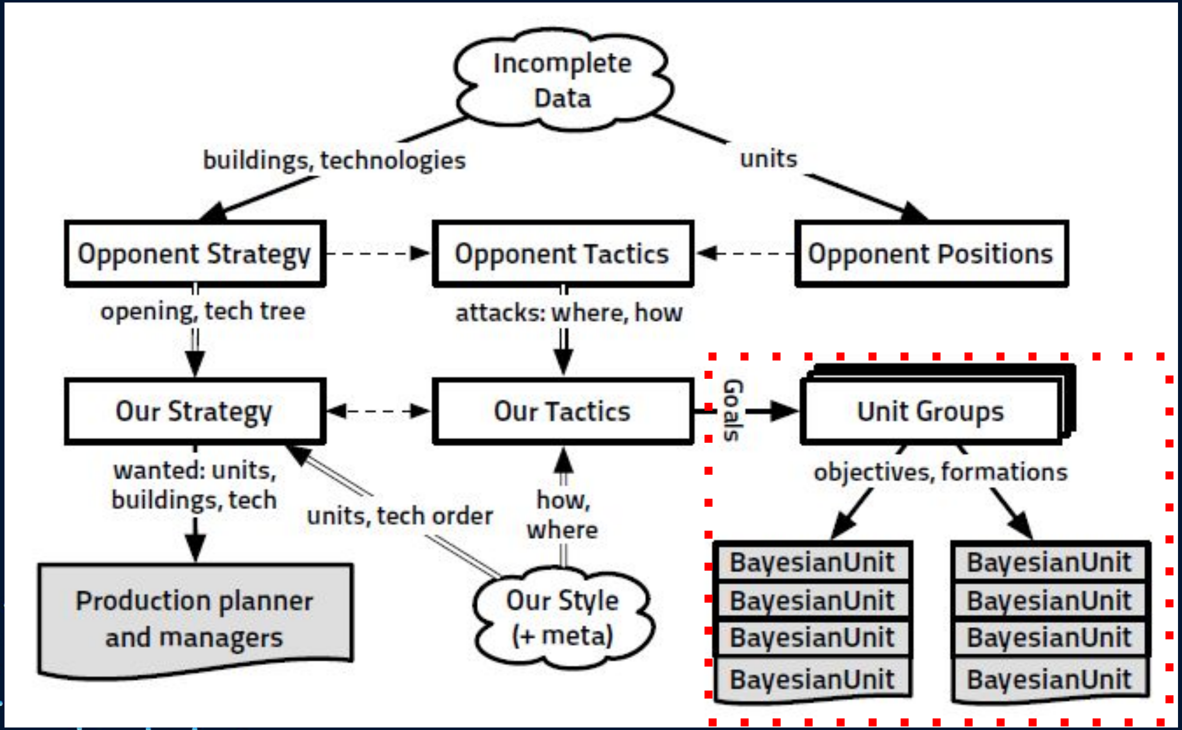
Barrack

Refinery

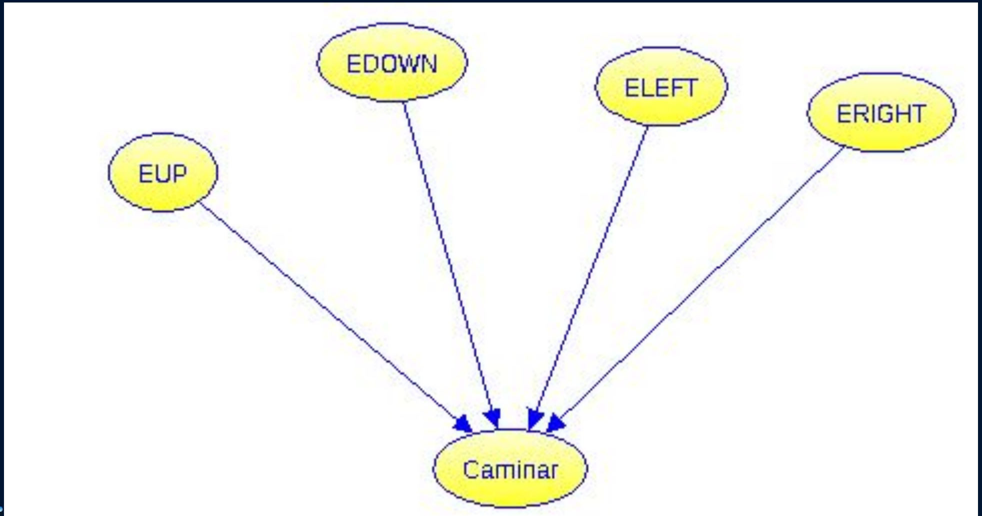
Bayesian Network: Scouting



Our scouting Bayesian Network

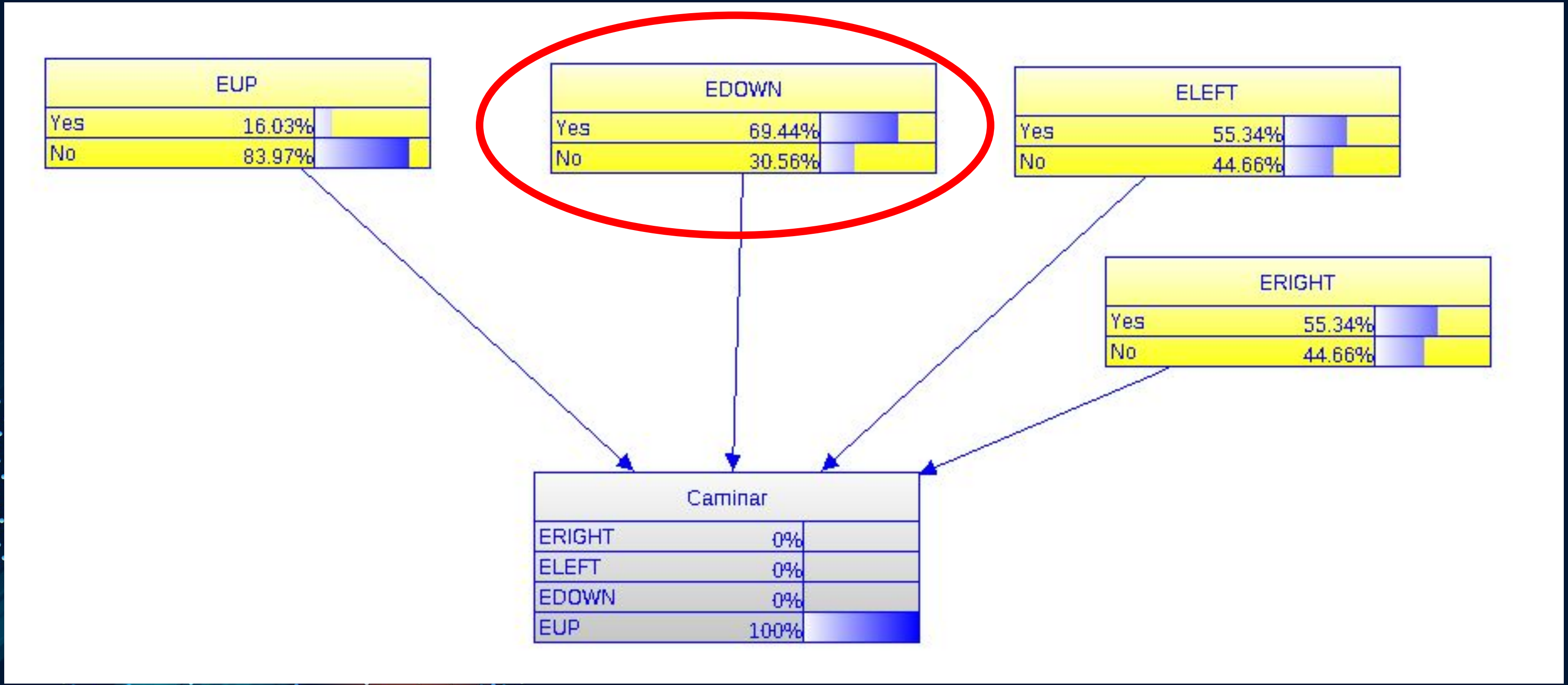

Incomplete Data
buildings, technologies
units
Opponent Strategy
Opponent Tactics
Opponent Positions
opening, tech tree
attacks: where, how
Our Strategy
Our Tactics
Goals
Unit Groups
wanted: units, buildings, tech
units, tech order
how, where
objectives, formations
Production planner and managers
Our Style (+ meta)
BayesianUnit
BayesianUnit
BayesianUnit
BayesianUnit
BayesianUnit
BayesianUnit
BayesianUnit
BayesianUnit

Scouting - Bayesian network DAG

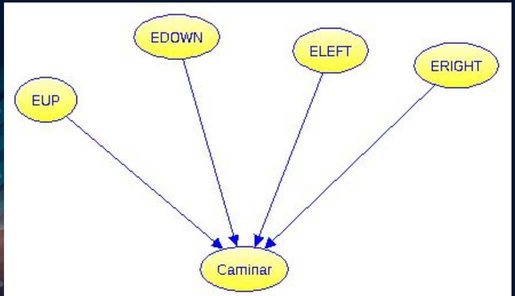


ERIGHT	Yes								No							
ELEFT	Yes				No				Yes				No			
EDOWN	Yes		No		Yes		No		Yes		No		Yes		No	
EUP	Yes	No	Yes	No	Yes	No	Yes	No	Yes	No	Yes	No	Yes	No	Yes	No
ERIGHT	0.25	0	0	0	0	0	1	0	1	0.5	0	0.33	0	0	0.33	0.25
ELEFT	0.25	0	0	0	1	0.5	0	0.33	0	0	0.5	0	0	0.5	0.34	0.25
EDOWN	0.25	0	1	0.5	0	0	0	0.33	0	0	0.5	0.33	0.5	0	0.33	0.25
EUP	0.25	1	0	0.5	0	0.5	0	0.34	0	0.5	0	0.34	0.5	0.5	0	0.25

Scouting - Belief Propagation



Scouting - Bayesian network




```python
class Explorer():
    def __init__(self):
        self.G = BayesianModel([('exploredUp', 'walk'),
                                ('exploredDown', 'walk'),
                                ('exploredLeft', 'walk'),
                                ('exploredRight', 'walk')])



        self.walk_cpd   = TabularCPD(variable = 'walk',
                                     variable_card = 4,
        #explored Right     #                   y                 |                 n
        #explored Left    #         y                   n         |         y                 n
        #explored Down    #     y         n       |    y          n    |    y          n    |    y         n
        #explored Up  #     y    n     y    n     y    n     y    n    y    n     y    n   y    n     y   n
               values = [[0.25, 0.33, 0.33, 0.5, 0.33, 0.5, 0,    1,   0,    0,   0.5,  0, 0.5,  0.5,  0, 0.25], #walk Right
                         [0.25, 0.33, 0.33, 0.5, 0,    0,   0.33, 0,   0.33, 0.5, 0,    1, 0.5,  0,    0, 0.25], #walk Left
                         [0.25, 0.34, 0,    0,   0.33, 0.5, 0.33, 0,   0.33, 0.5, 0,    0, 0,    0.5,  0, 0.25], #walk Down
                         [0.25, 0,    0.34, 0,   0.34, 0,   0.34, 0,   0.34, 0,   0.5,  0, 0,    0,    1, 0.25]],#walk Up
                      evidence=['exploredRight', 'exploredLeft', 'exploredDown', 'exploredUp'],
                      evidence_card=[2, 2, 2, 2])
        self.eUp_cpd    = TabularCPD(variable = 'exploredUp',
                                     variable_card = 2,
                                     values = [[0.5], [0.5]])
        self.eDown_cpd  = TabularCPD(variable = 'exploredDown',
                                     variable_card = 2,
                                     values = [[0.5], [0.5]])
        self.eLeft_cpd  = TabularCPD(variable = 'exploredLeft',
                                     variable_card = 2,
                                     values = [[0.5], [0.5]])
        self.eRight_cpd = TabularCPD(variable = 'exploredRight',
                                     variable_card = 2,
                                     values = [[0.5], [0.5]])

        self.G.add_cpds(self.eUp_cpd,
                        self.eDown_cpd,
                        self.eLeft_cpd,
                        self.eRight_cpd,
                        self.walk_cpd)
```

Pgmpy library

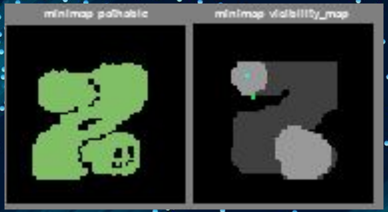
Result: Successful exploration Minimap

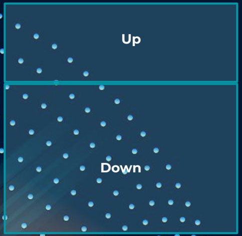
Calculating ‘Explored Percentage’ v00

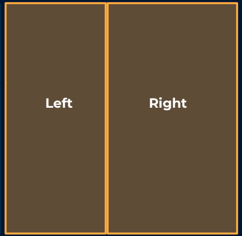
Pixels explored : pathable and explored

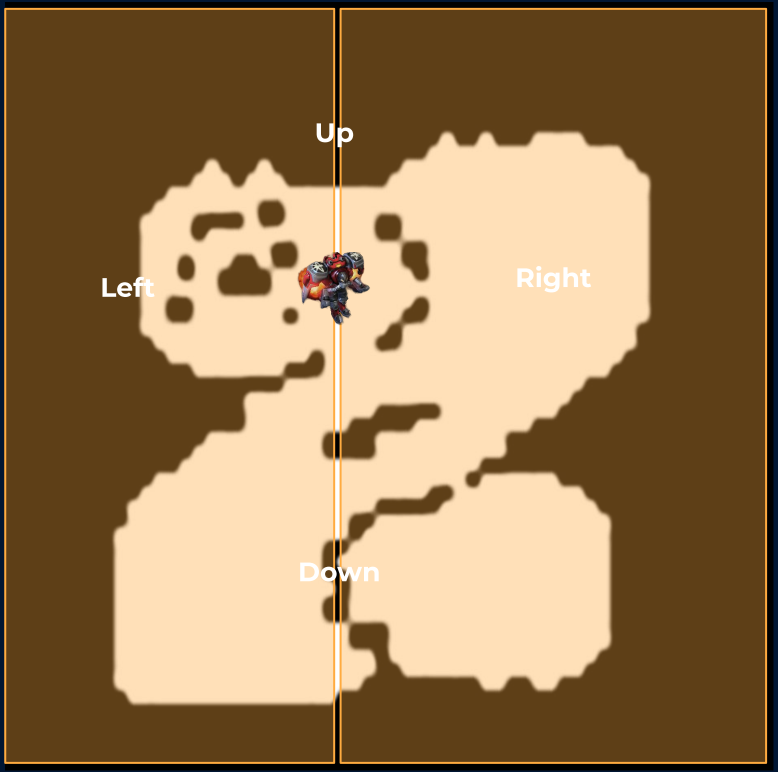
Pixels total : pathable

$$\textbf{Percentage explored} = \frac{\textbf{Pixels explored}}{\textbf{Pixels total}}$$







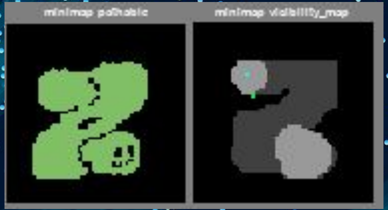


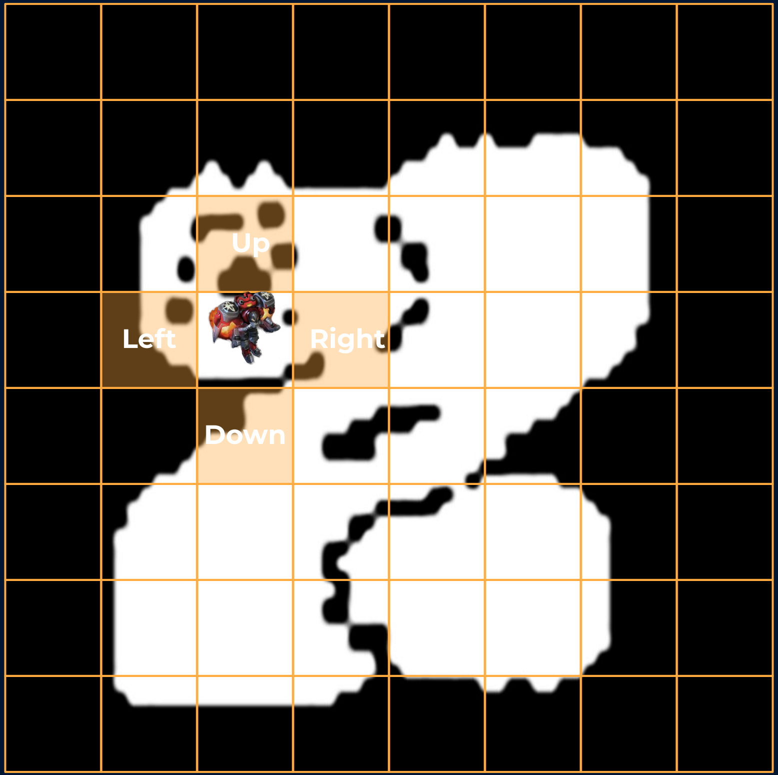
Calculating 'Explored Percentage' v01

Pixels explored : pathable and explored

Pixels total : pathable

$$\text{Percentage explored} = \frac{\text{Pixels explored}}{\text{Pixels total}}$$

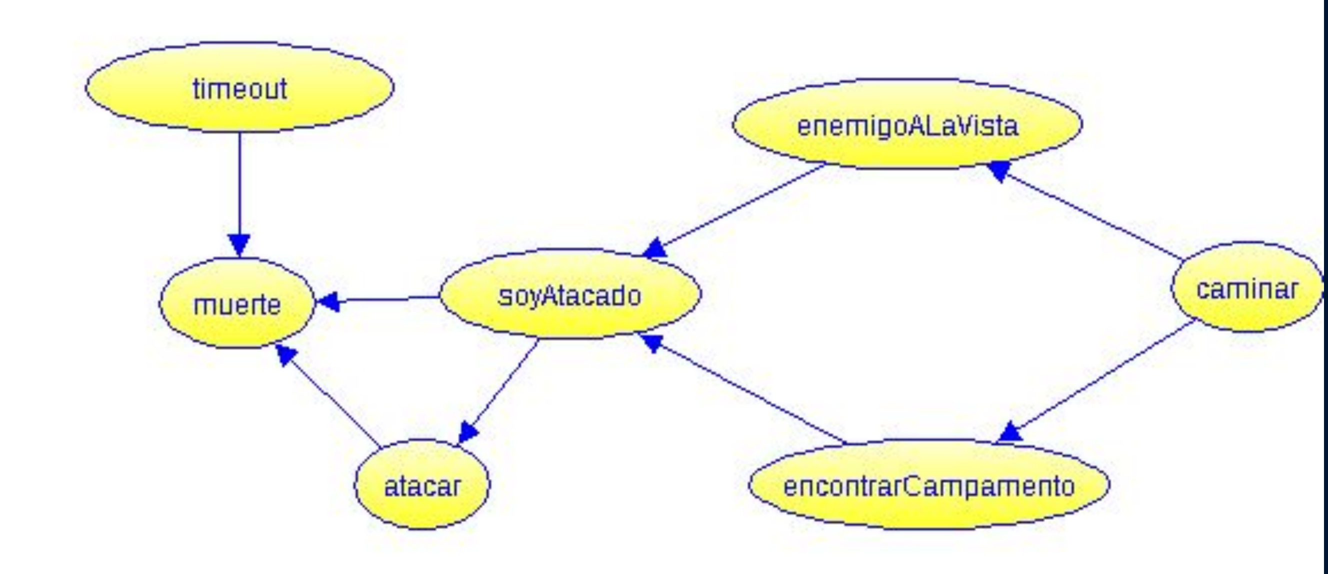


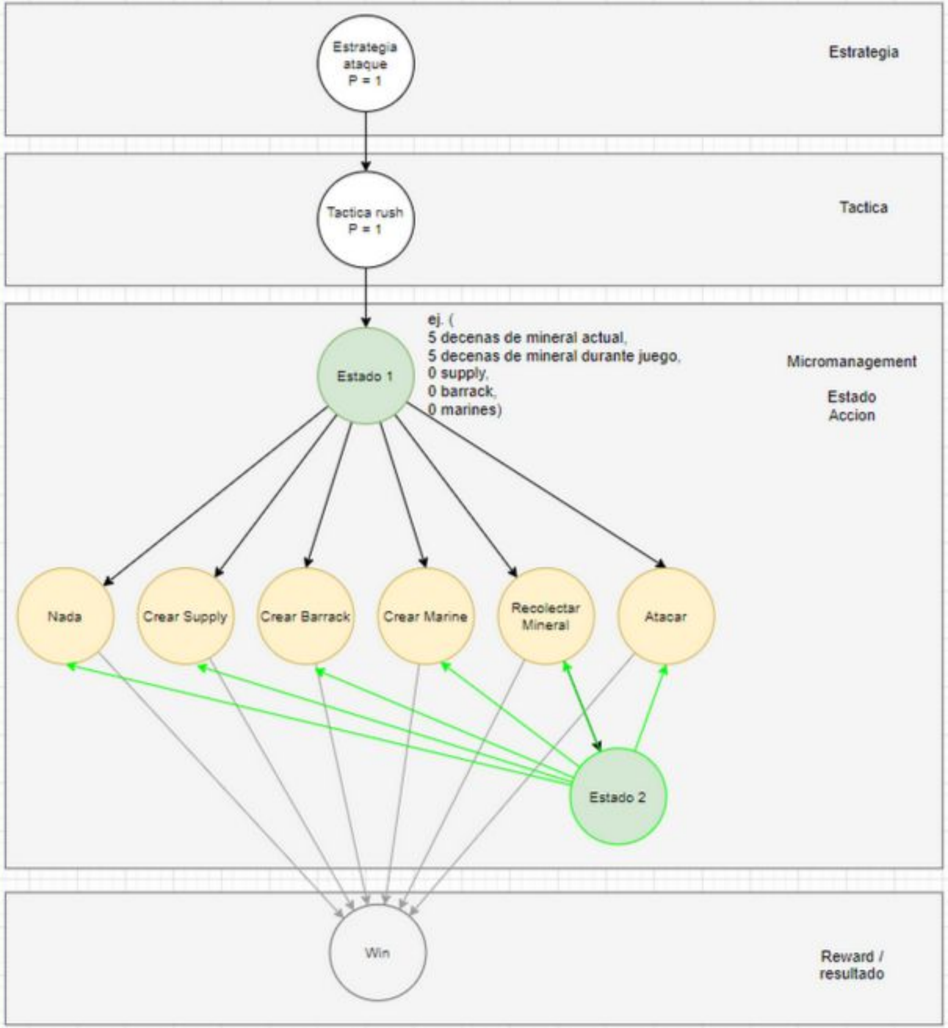


Bayesian Networks: other ideas and approaches

Not implemented

Another explorer





Vision:

Probabilidad de x accion dado y estado

Probabilidad de x estado dada y accion

Probabilidad de ganar dada cierta accion

Tomar el ESTADO: como una tupla de elementos medibles que consideramos afectan a esta etsrategia

Notas1:

Consideracion. si se decide crear un barrack (u otra cosa), pero.... no se puede, la accion ejecutada será 'Nada'. Y se deberá considerar el nodo al que se dirigió es 'nada' (Generar los estados despues de saber si se tomo 'nada')

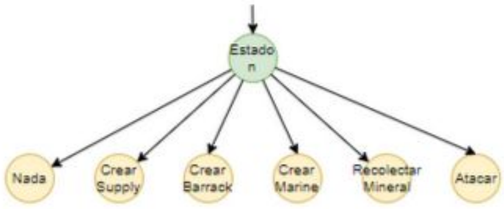
con (50 mineral, x barrackas, x monos, etc etc.).. se hará 'nada'

Notas2:

- P(rush|est.ataque) =1
- P(estado1|rush) = 1 (porque iniciamos siempre igual en este juego)
- P(crearBarrack|estado1) = ?
- P(estadoX|crearBarrack) ?
- P(win|nada) = 0 (me imagino)
- P(win|attack) = ? (dependerá de las acciones previas al attack)

DECIDIR - Durante el juego (En cada step)
Dependiendo que tenemos, ver que estado de la red bayesiana es lo que tenemos. Y decidir en base a eso.

CREAR -Durante el juego (En cada step)
Generar y conectar los nuevos Estados si se necesita



Cada estado nuevo se conecta:

- desde la accion que viene
- hacia todas las acciones

APRENDER - Al terminar cada partida
Por backpropagation modificar las probabildiades de nuestra red dependiendo los nodos tocados.

Conclusions

- Designing Bayesian Networks can be a hard job

- Assigning the probabilities is not so intuitive for us humans
- Training is required to help the Bayesian Network to work optimally

- Bayesian Network could be mixed with other strategies, like A* and Alpha Beta pruning.

References:

https://www.ijcai.org/Proceedings/11/Papers/364.pdf

https://projekter.aau.dk/projekter/files/61058274/1055315209.pdf

https://www.researchgate.net/publication/278641976_Bayesian_Programming_and_Learning_for_Multi-Player_Video_Games_Application_to_RTS_AI