HALMSTAD UNIVERSITY

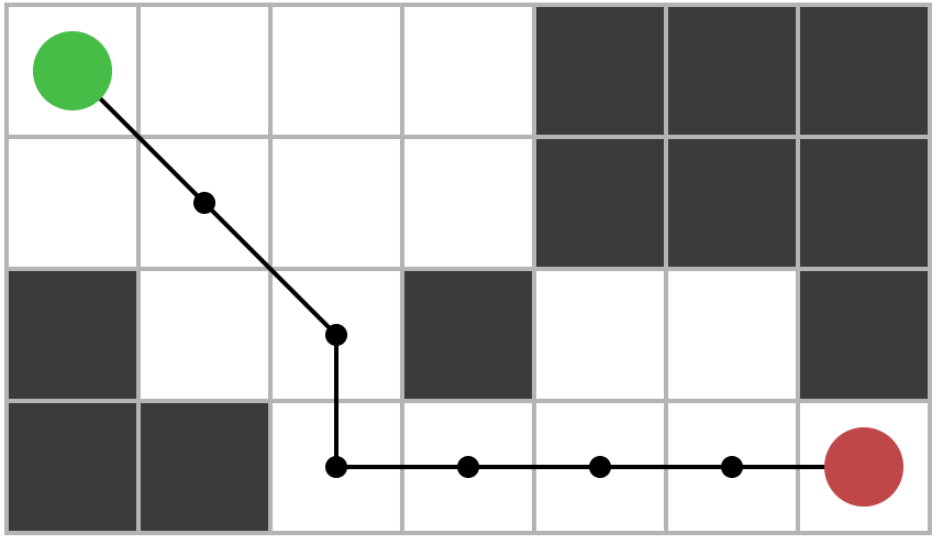
Artificial Intelligence Laboratory 2: Search Algorithms

DT8012 (HT17) Halmstad University

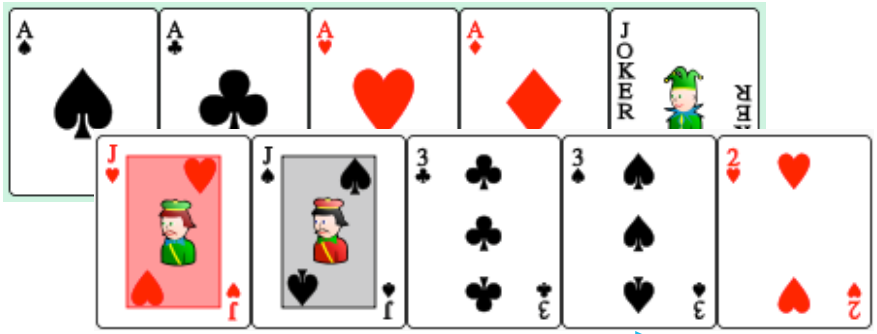
Nov 2017

Lab 2

- Path Planning
  - Find a shortest path



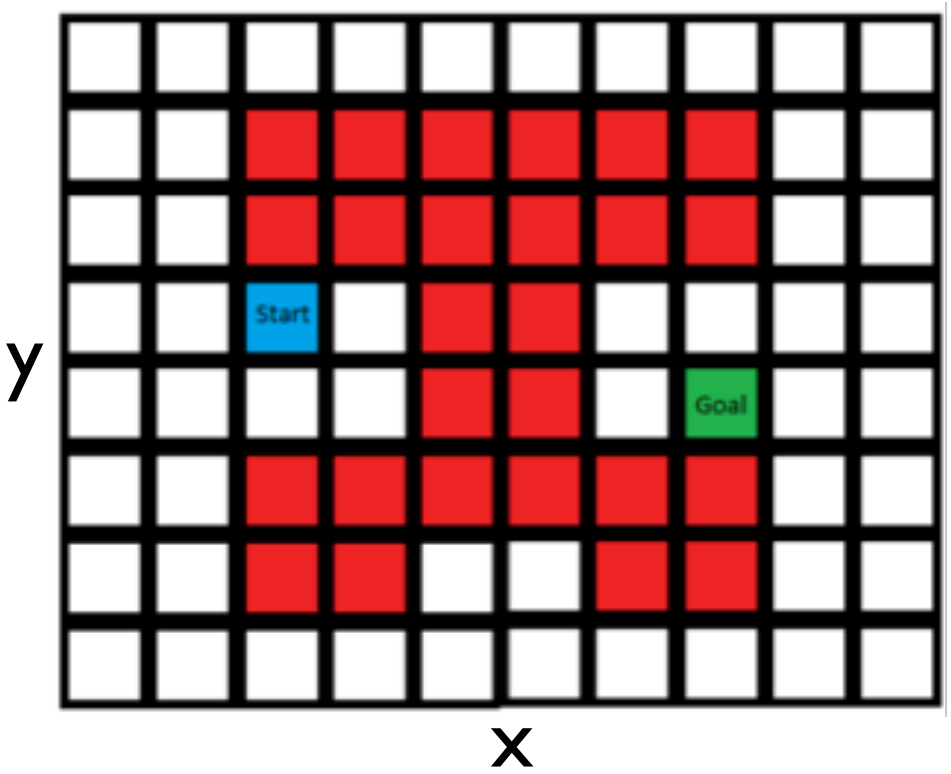
- Simplified Poker game
  - Find optimal sequence of actions



General Objective

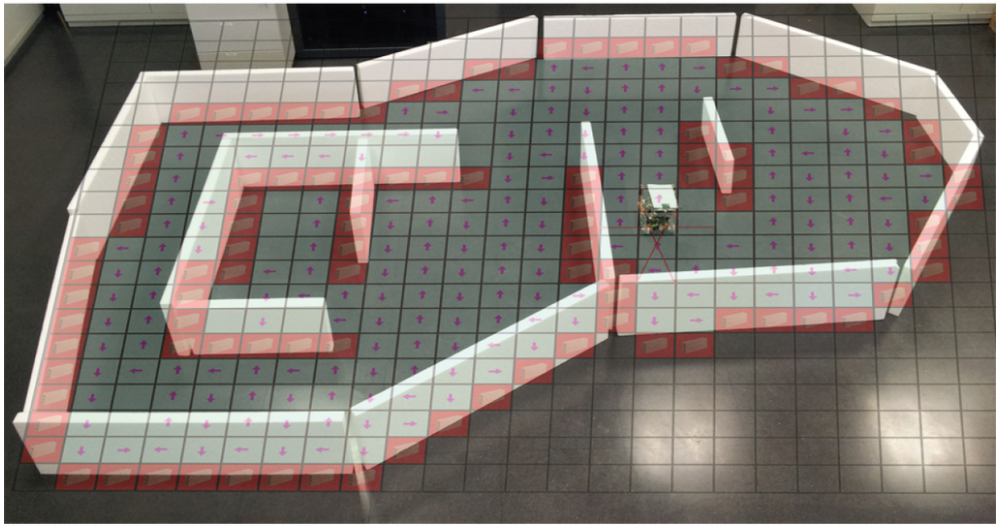
- Implement search algorithms
  - Random search
  - Exhaustive search (BFS & DFS)
  - Greedy search
  - A* algorithm
- Design and investigate different heuristic functions for informed search algorithms
  - Reduce the search space
  - based on available information of the problem

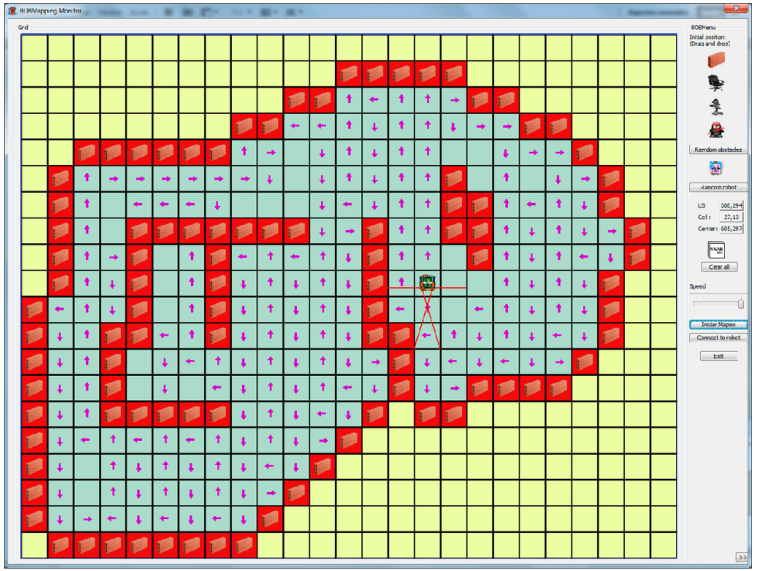
Path Planning



- Red block: obstacles
- Starting point
- Goal

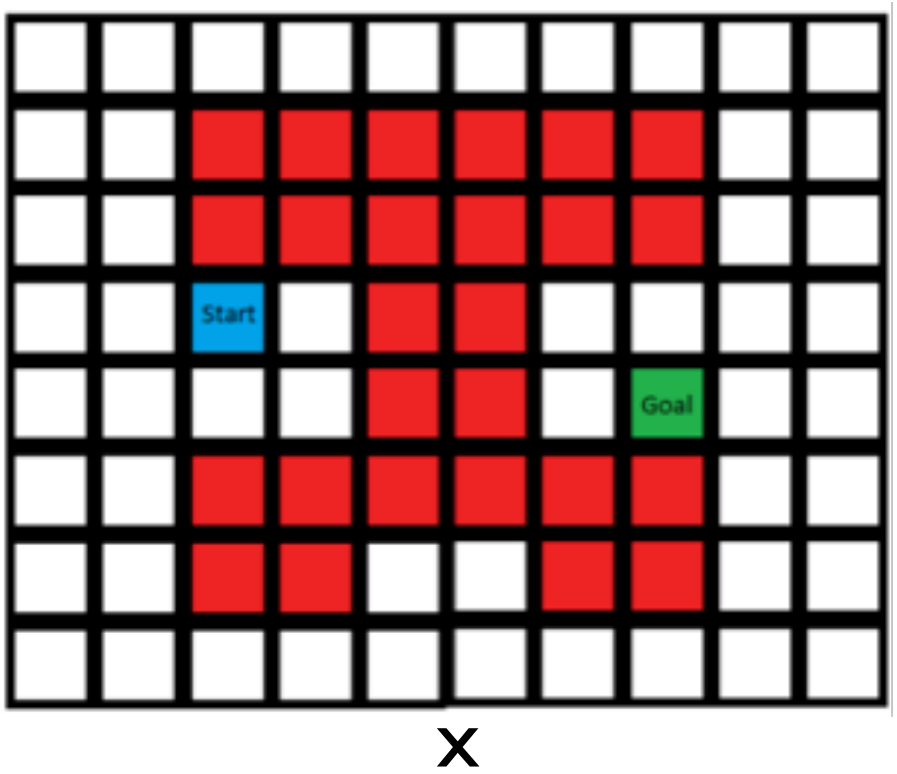
Occupancy Grid Map





Gonzalez-Arjona, David, et al. "Simplified occupancy grid indoor mapping optimized for low-cost robots." *ISPRS International Journal of Geo-Information* 2.4 (2013): 959-977.

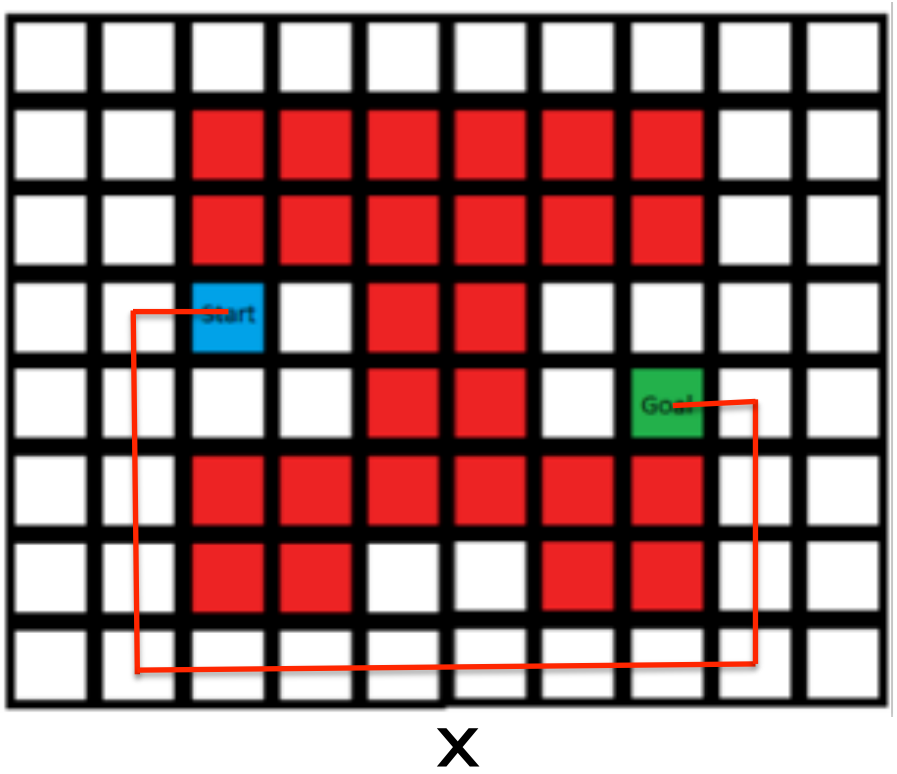
Path Planning



- Red block: obstacles
- Starting point
- Goal

1	1	1	1	1	1	1	1	1	1
1	1	-1	-1	-1	-1	-1	-1	1	1
1	1	-1	-1	-1	-1	-1	-1	1	1
1	1	-2	1	-1	-1	1	1	1	1
1	1	1	1	-1	-1	1	-3	1	1
1	1	-1	-1	-1	-1	-1	-1	1	1
1	1	-1	-1	1	1	-1	-1	1	1
1	1	1	1	1	1	1	1	1	1

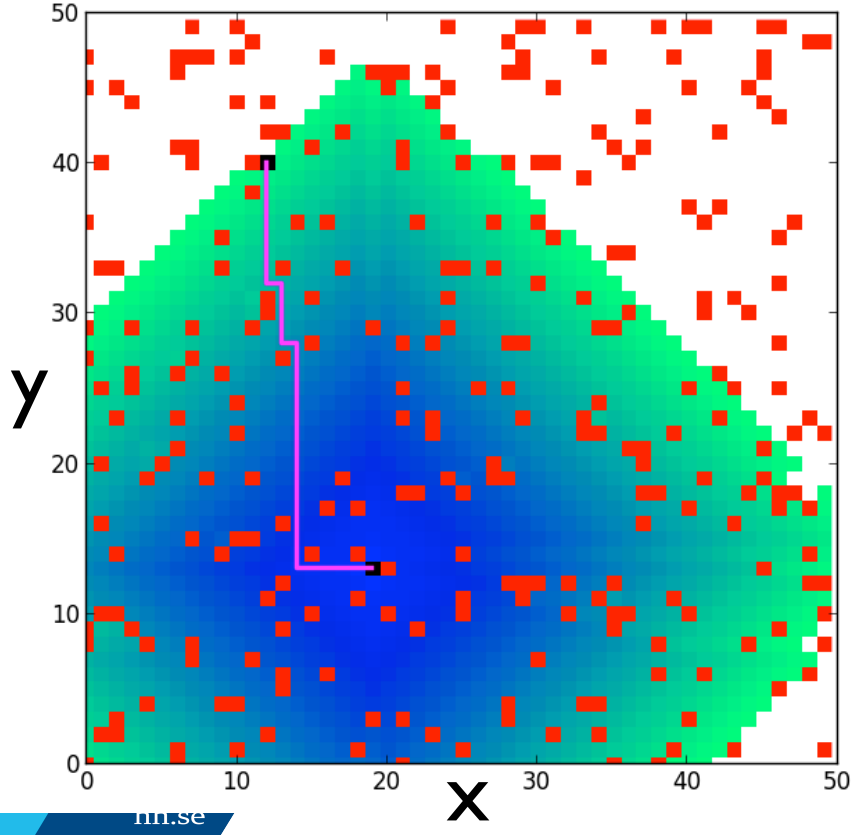
Path Planning



- Red block: obstacles
- Starting point
- Goal

- Find short path

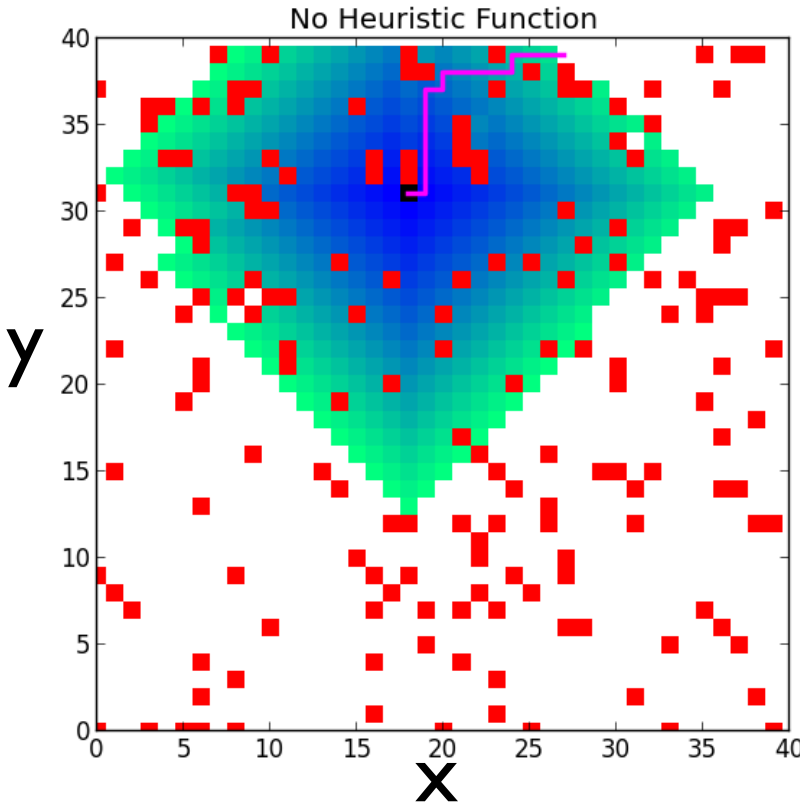
Path Planning



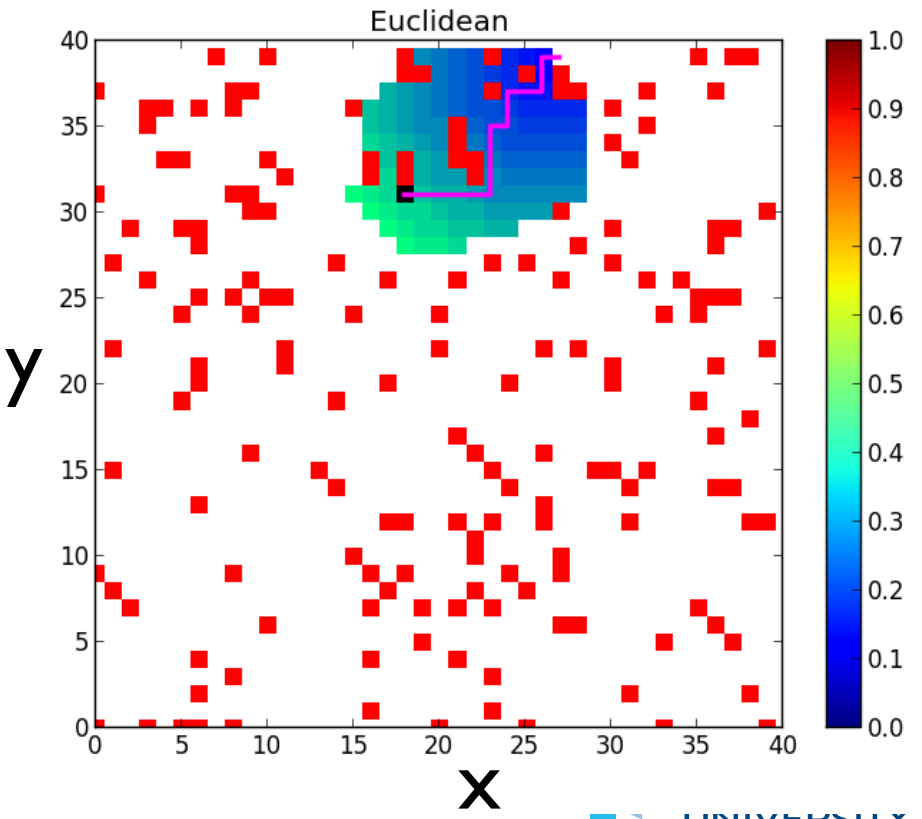
Red block: obstacles
Starting point
Goal

Find short path
– Reduce search space!

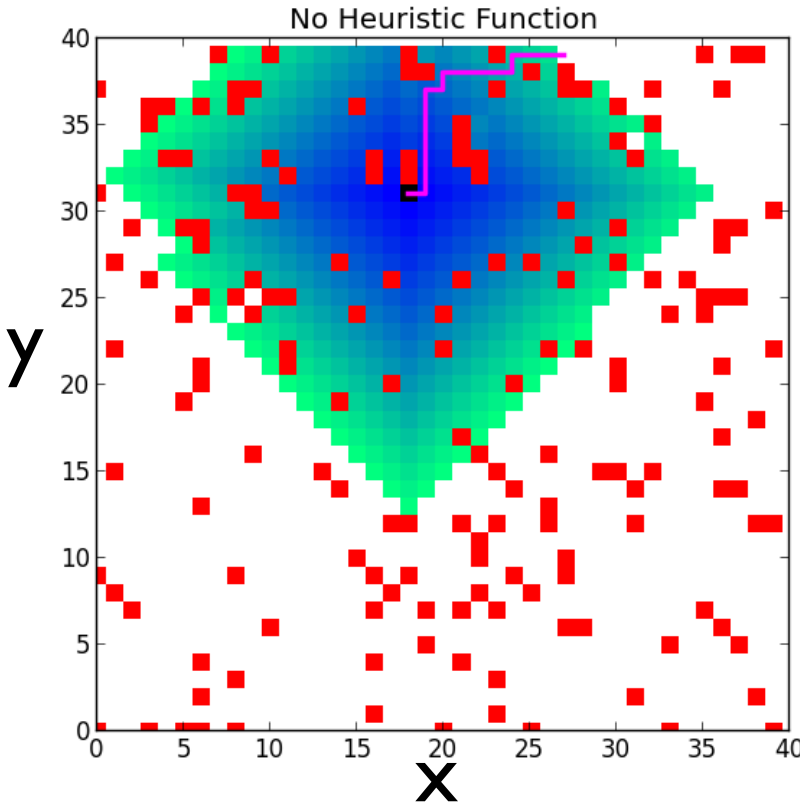
Breadth-first search



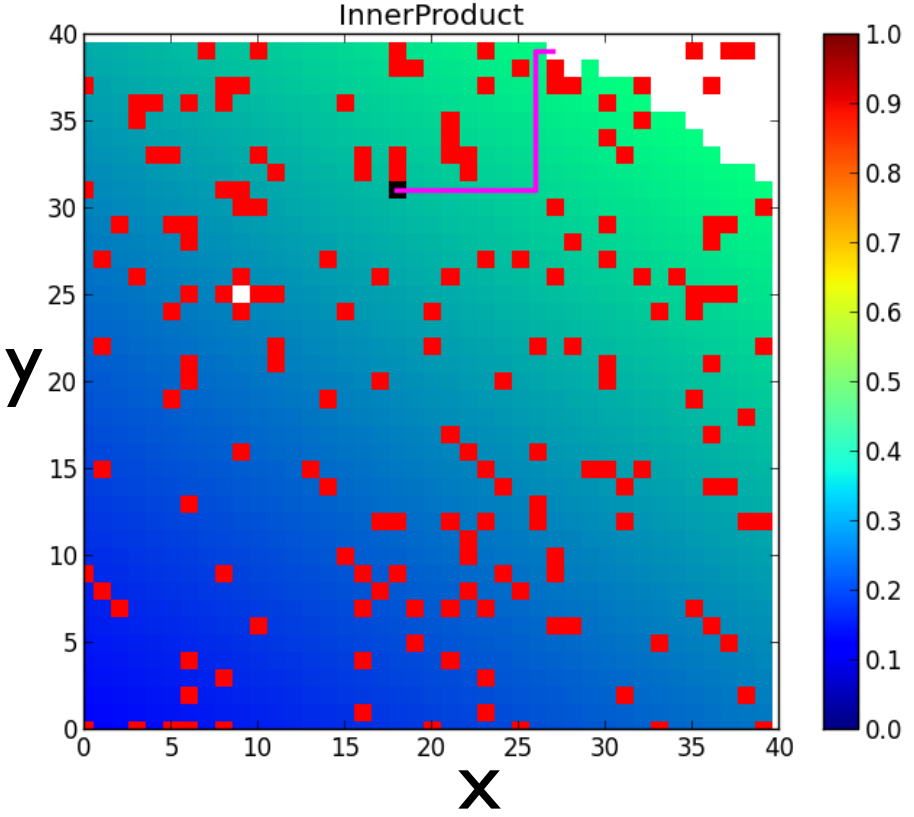
A* with Manhattan distance as heuristic



Breadth-first search

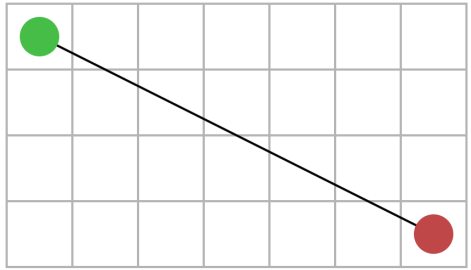


A* with Inner Product distance as heuristic

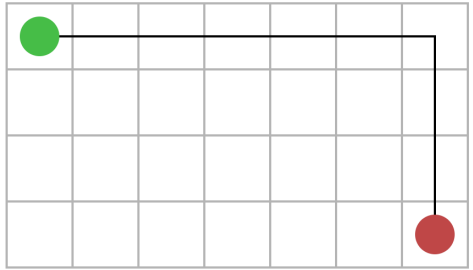


Path Planning

$$h(n)^2 = (n.x - goal.x)^2 + (n.y - goal.y)^2$$

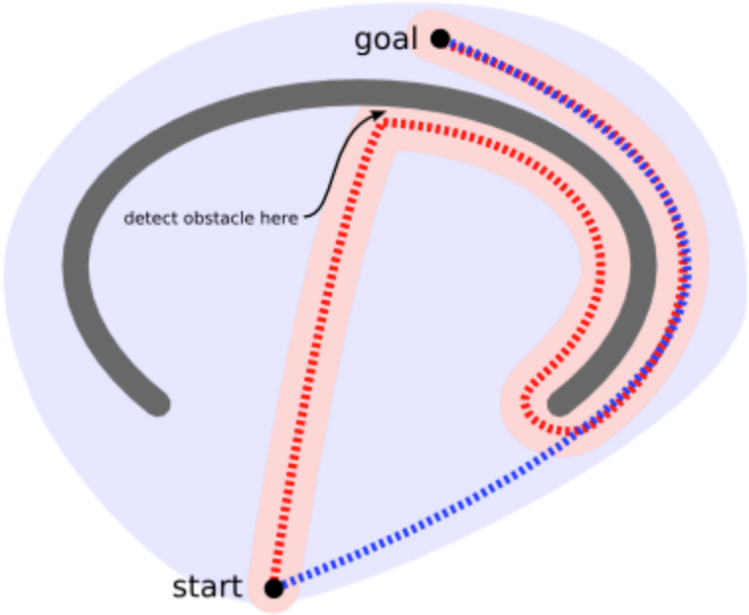


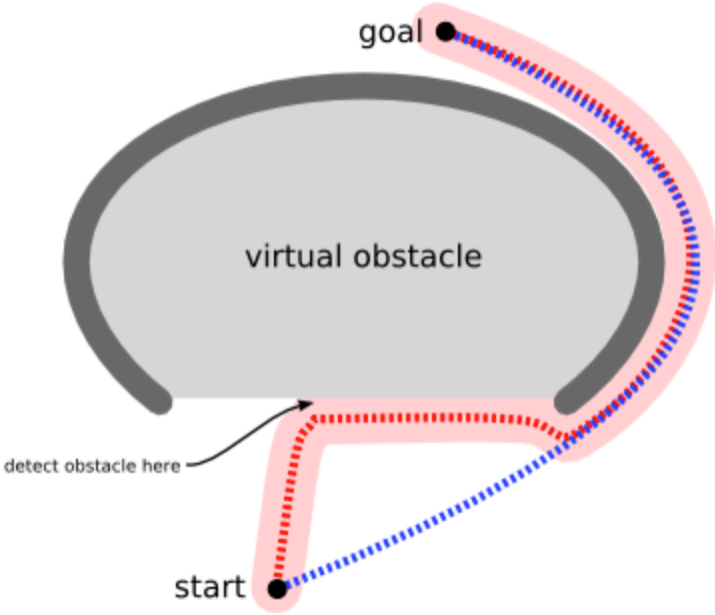
$$h(n) = |n.x - goal.x| + |n.y - goal.y|$$



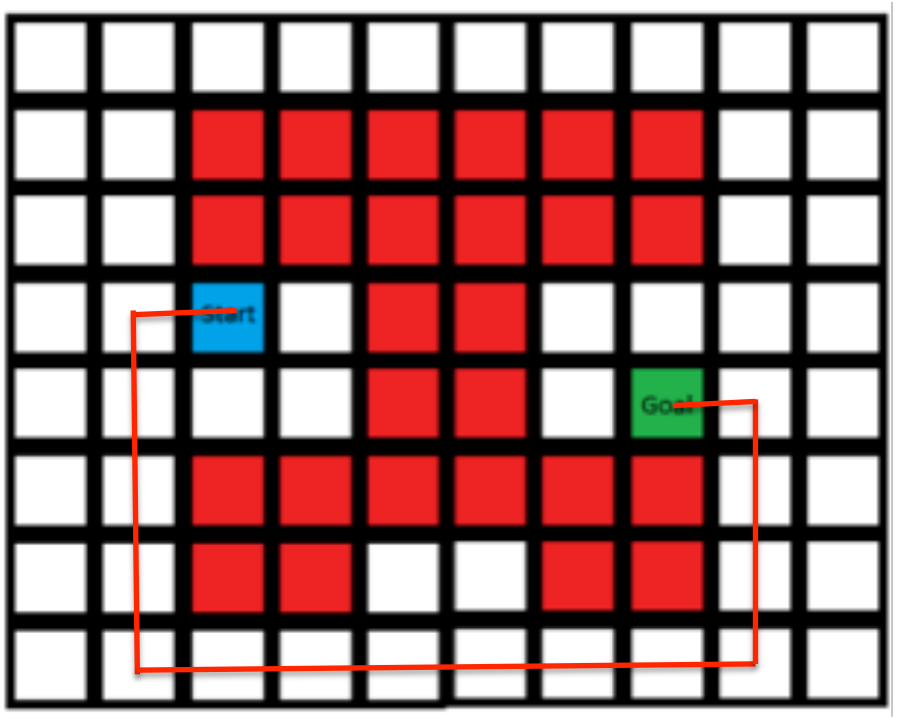
- Heuristics
  - Euclidean distance
  - Manhattan distance
- General-purpose heuristics for 2d grid map

Reduce search space using A* algorithm



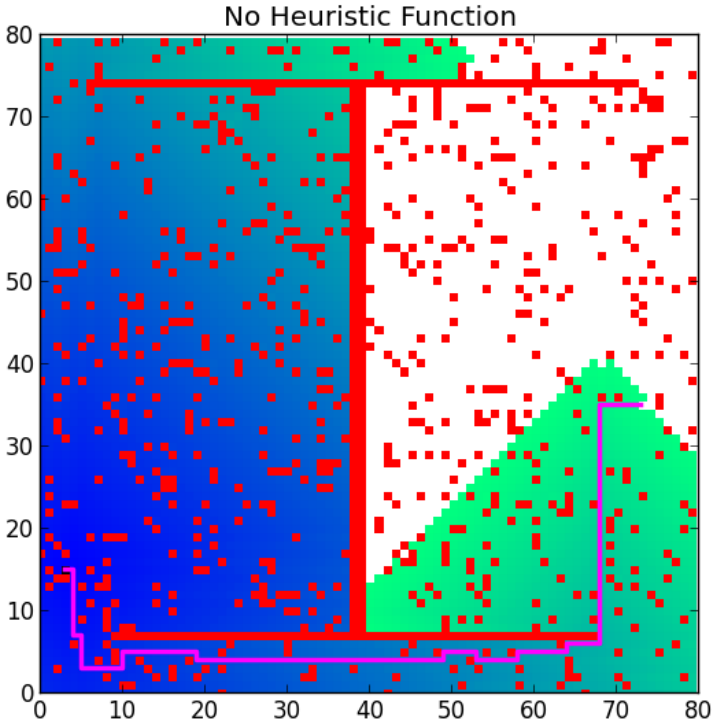


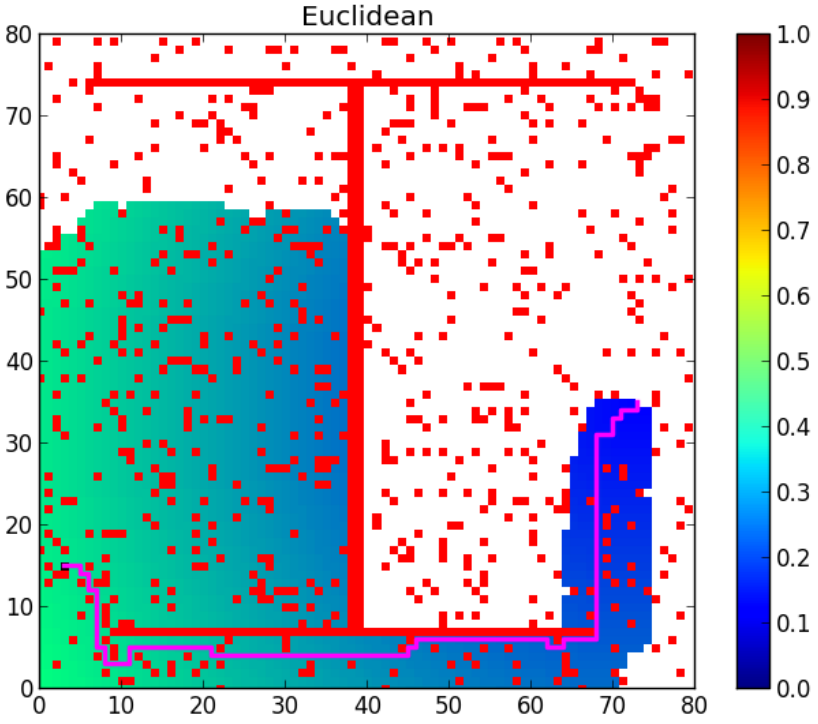
Environment-specific Heuristic



- Env. Information
  - Starting point at left side and ending point at right side
  - The environment contains a '工' shaped obstacle
    - Y coordinate of upper and lower edges of '工'
    - X coordinate of the vertical wall

Path Planning in ‘工’ environment





Expectation

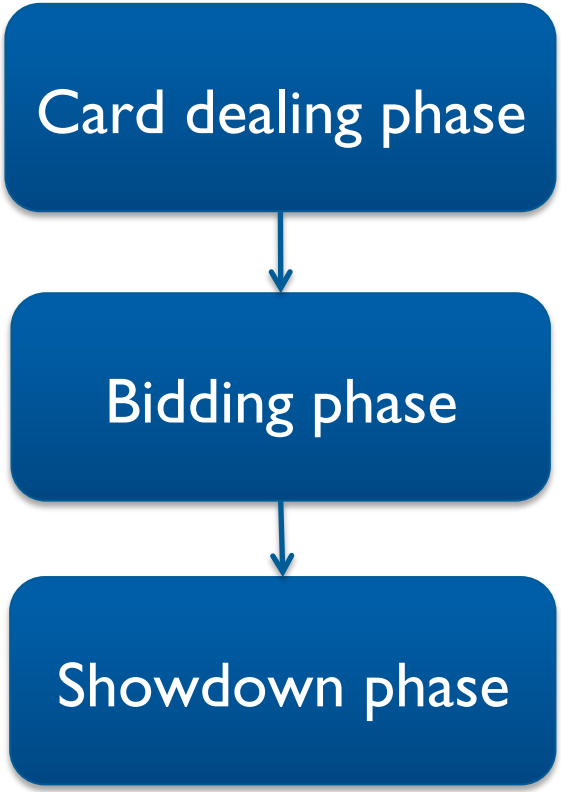
- A* Star algorithm
  - Try different type heuristics
- 2 types of environment
  - Map with obstacles randomly placed
  - Map with ‘工’ obstacles and randomly placed obstacles
- Comparison with random search, exhaustive search and greedy search
  - Modify cost function

Task 2 Poker game

- Rules: slightly more complex than the first lab!
- Search optimal solution
  - Random, exhaustive and Greedy search
  - A algorithm with heuristic
- Objective
  - Design a special heuristic function that reduces the search space

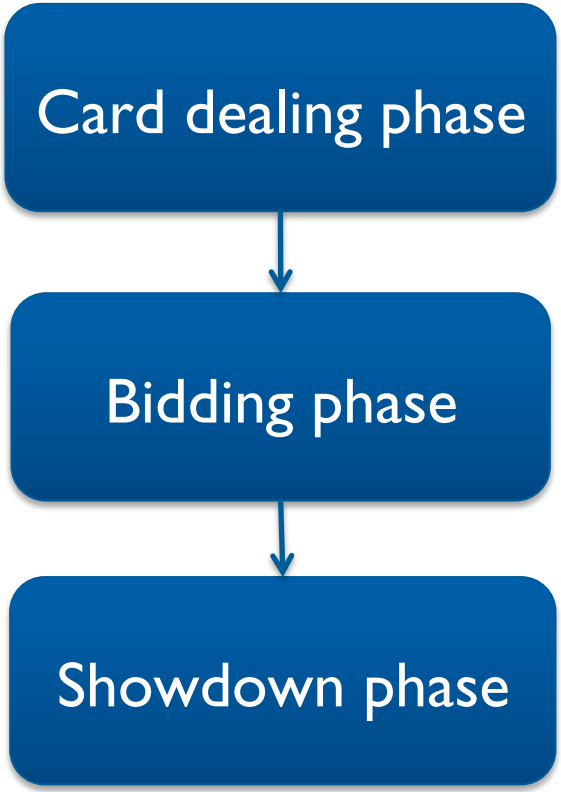
Game flow (1st lab)

- Card dealing phase
  - Assign 3 cards to agents
- Bidding phase
  - Amount $0-50
  - Regardless of how much other players bet
- Showdown phase



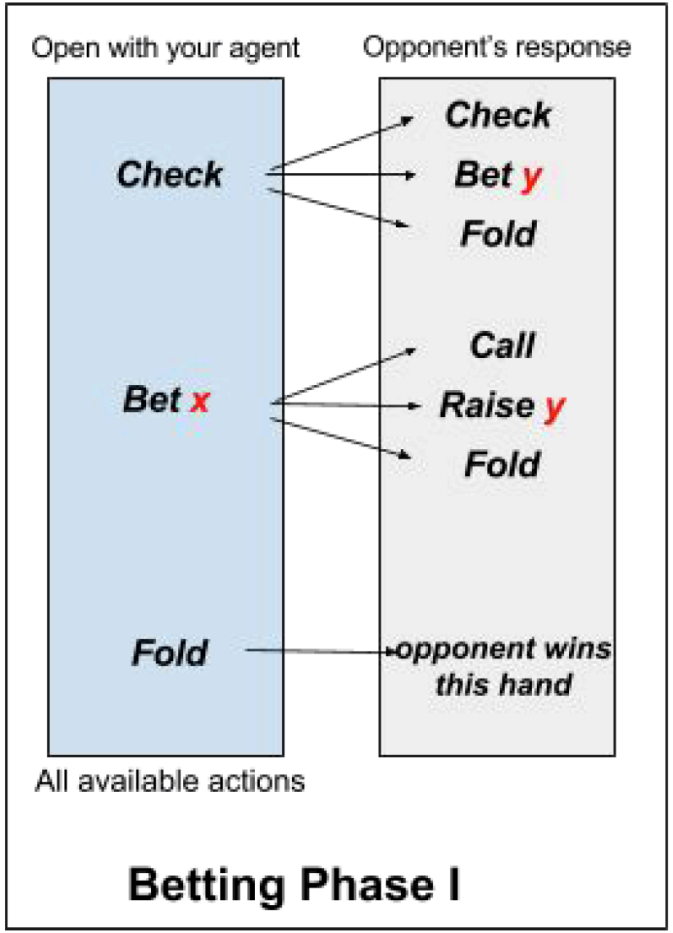
Game flow

- Card dealing phase
  - Assign 5 cards to players
- Bidding phase
  - Search sequence of actions
- Showdown phase



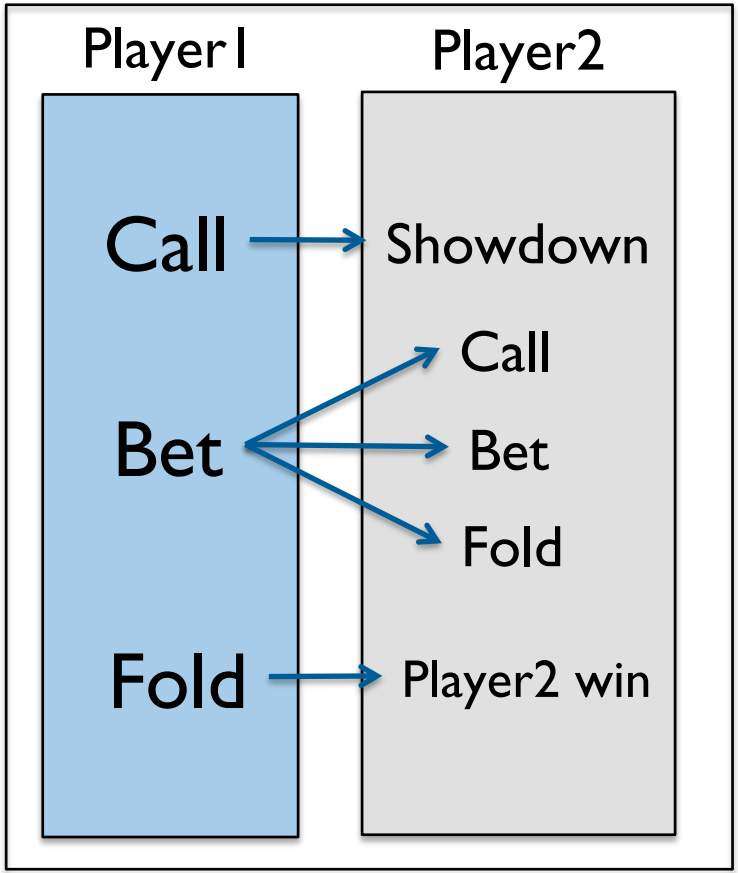
Traditional Poker game

- Actions Available
  - Bet
  - Raise
  - All In
  - Check
  - Call
  - Fold
  - Showdown



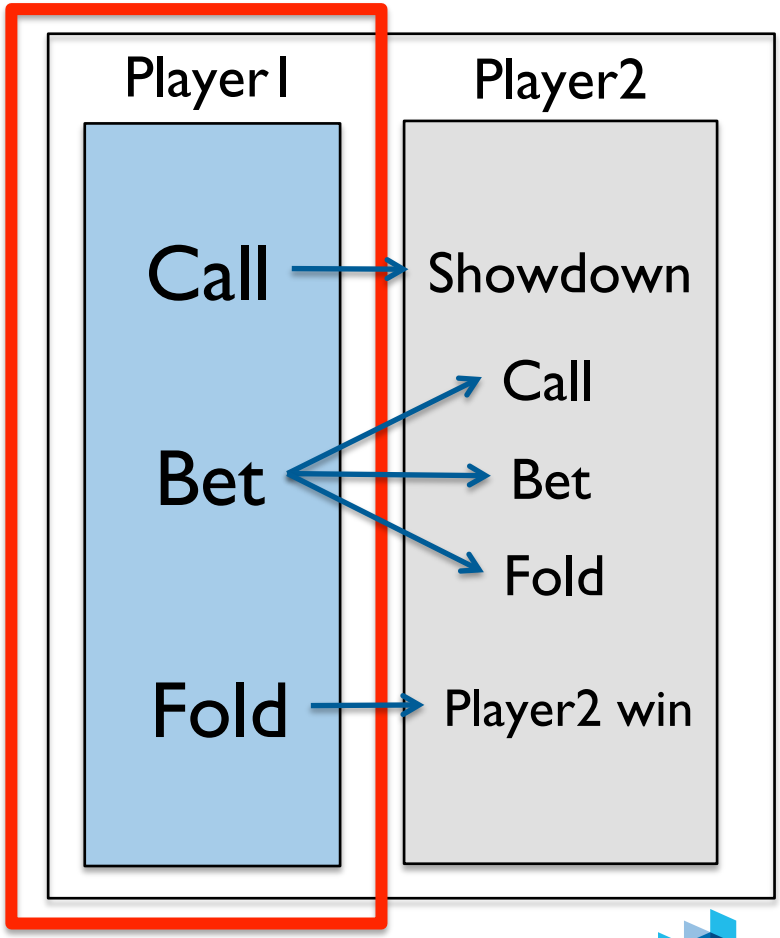
Simple Poker Game

- Action Available
  - Bet x coins
    - 5, 10 or 25 coins
    - Regardless of how much opponent bet
  - Call
    - Putting 5 coins in the pot and show hand
  - Fold



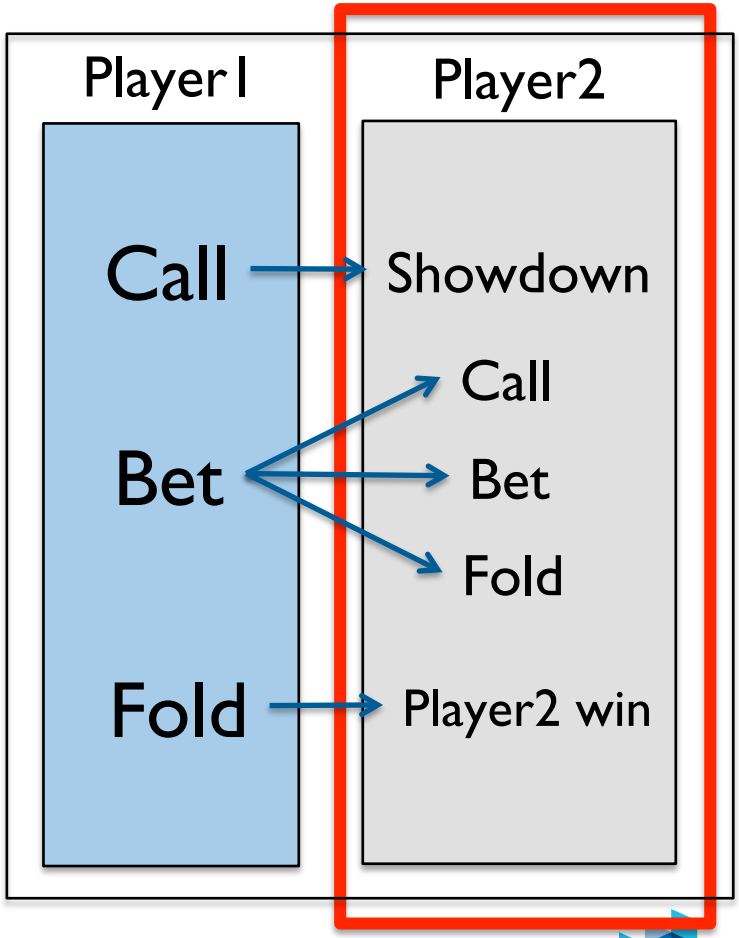
Simple Poker Game

- Always start with your agent
- Action Available
  - Bet x coins
  - Call
  - Fold



Simple Poker Game

- Always start with your agent
- Action Available
  - Bet x coins
  - Call
  - Fold



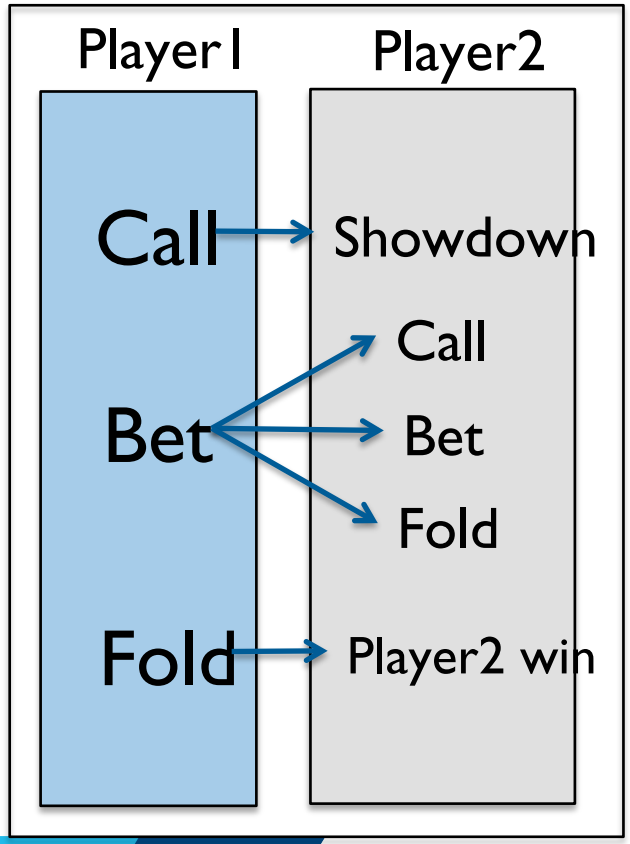
Tasks

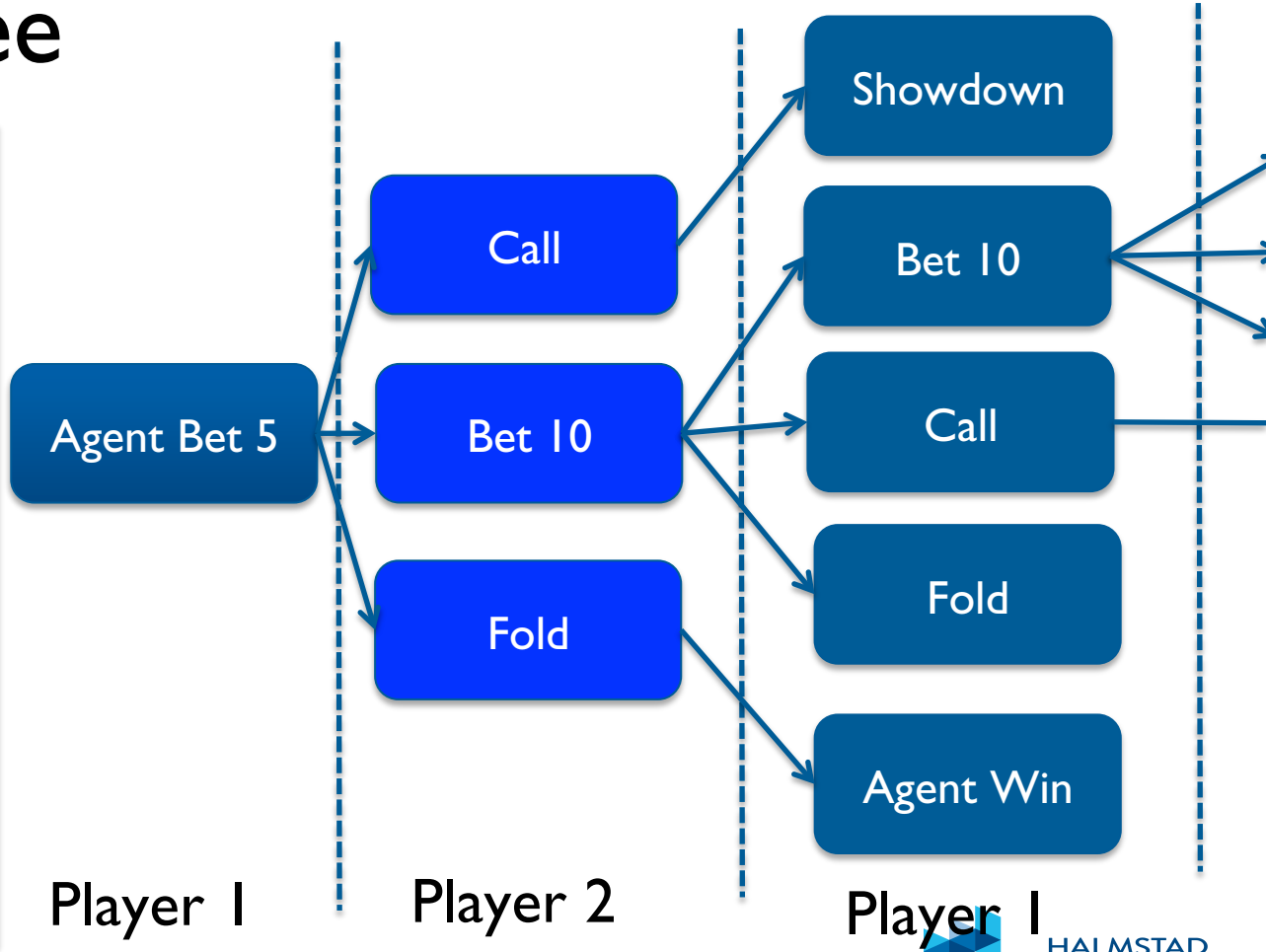
- Build environment of the game
  - Hand evaluation function for 5 cards
  - Function for updating the state of the game
    - Number of hands played, coins left for both agent, coins in the pot, current hand for both agent etc.
- Implement two fixed agents playing against each other

Expectation / Tasks

- Use search algorithm to find a series of actions for your agent to win more than 100 coins within 4 hands
  - given known strategy of the opponent and complete information of the game
  - Start with exhaustive search
- Design heuristic function that reduce the search space

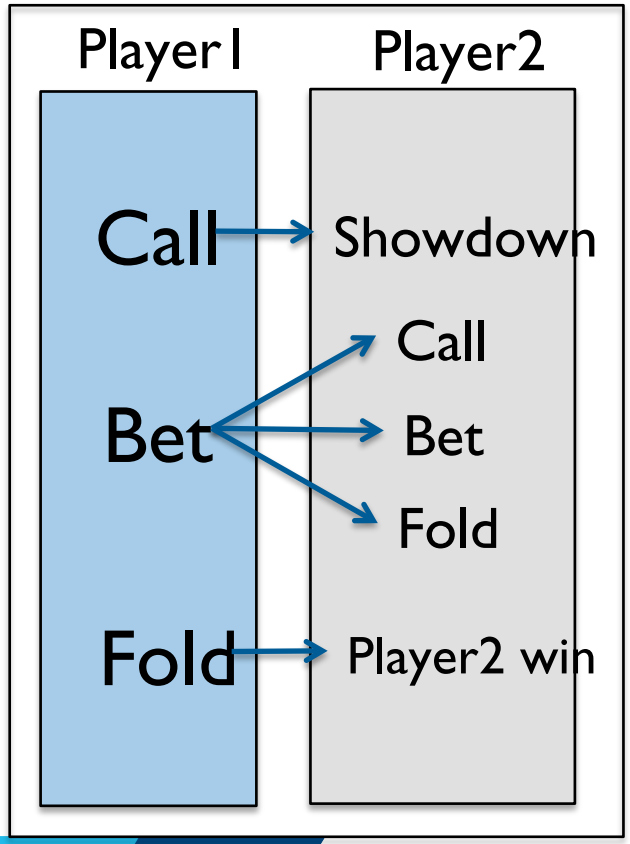
Decision Tree

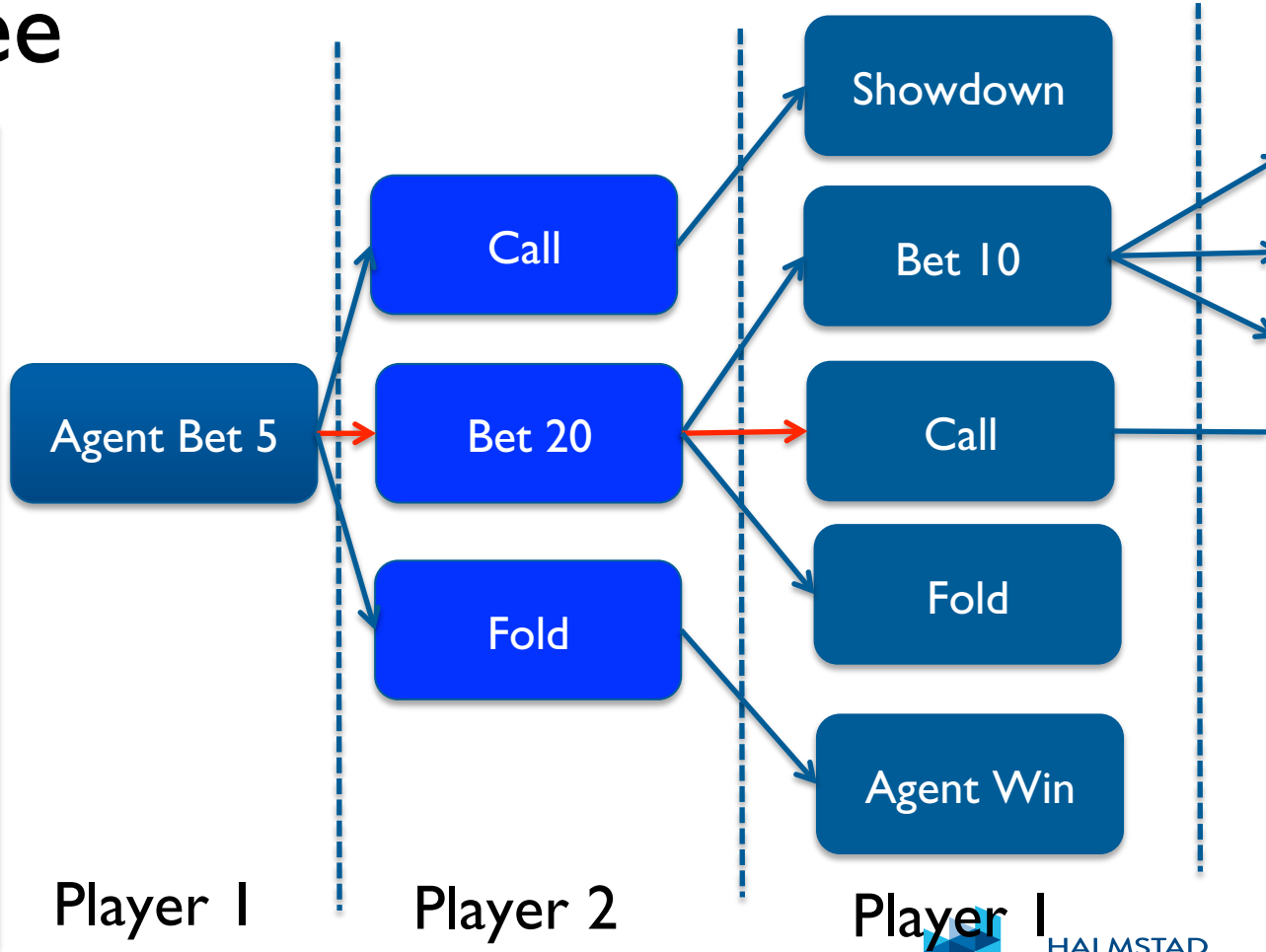




HALMSTAD UNIVERSITY

Decision Tree

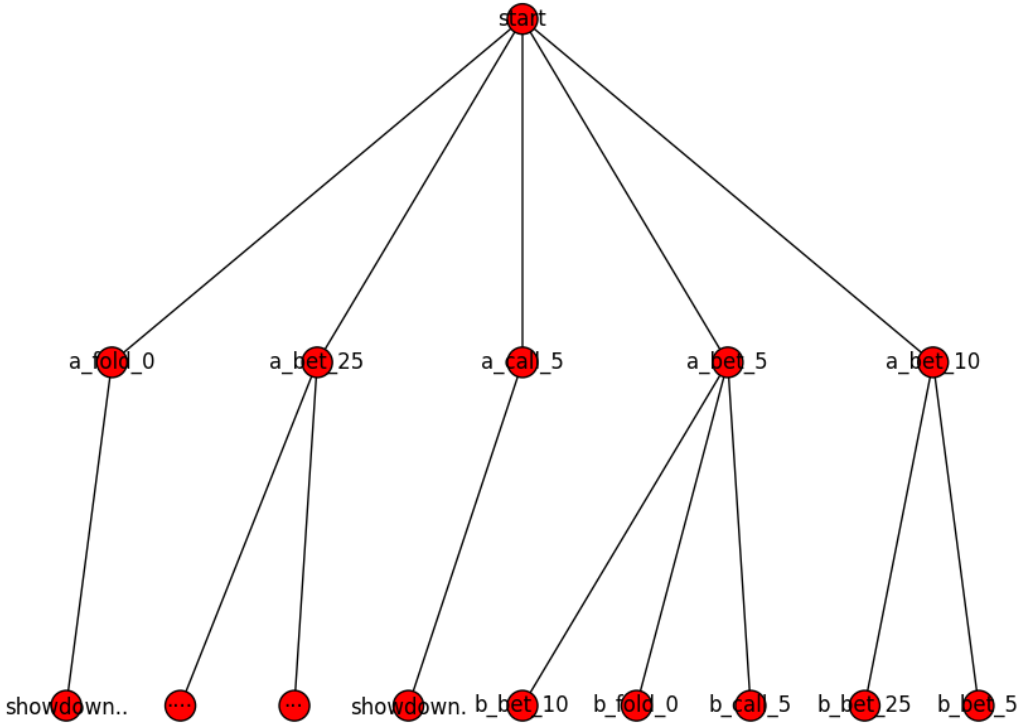




poker_strategy_example(...)

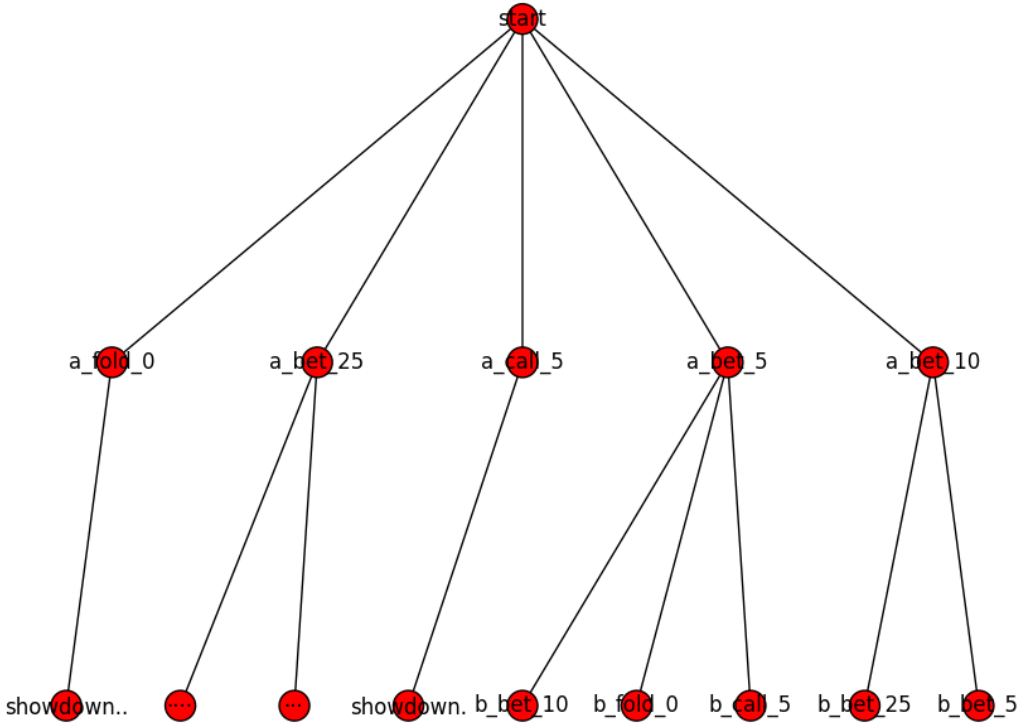
Information/input	Details
opponent_hand	type of opponent's hand
opponent_hand_rank	rank of opponent's hand
opponent_stack	total amount of coins opponent has
agent_action	agent's action: Bet, Call or Fold
agent_action_value	the amount of coins agent used to Bet or Call
agent_stack	total amount of coins the agent has
current_pot	total amount of coins currently in bidding
bidding_nr	numbers of times both player has bidded

Start by generating tree for one hand



- Generate a tree
- Give each node an identifier
  - Depth
  - Round
  - Agent actions
  - …
- Generate all edges

Start by generating tree for one hand



- State
  - Depth
    - Hand/round nr.
    - Bidding nr.
  - Current pot
  - Agent
    - Hand
    - Stack
    - Action, value
    - Coins bidded in current hand
  - Cost/heuristics
  - …

Expectation / Tasks

- Implement environment of the game
  - Evaluation function
  - Update state of the game
  - 2 fixed agent playing against each other
- Implement random search, exhaustive search and greedy search to find shortest sequence of actions
- Heuristic function: Find optimal solution as exhaustive search provided and expand less nodes

Grading

- Pass/fail/extra credits
- Submit your lab to yuantao.fan@hh.se
  - a short report about what you have done
  - Code