Problemområden i Starcraft II

...och AI-tekniker för att tackla problemen

Mattias Tiger
Fredrik Präntare
Jonas Kvarnström

Introduction and motivation

For your individual task, you will **define a problem** and **select one or more AI technologies** that could be used to solve this problem.

- There are a lot of *domain-specific problems* in StarCraft II to solve.
  You don't have to solve them *all*, but choose those that can contribute to a good agent.
- There are a lot of AI technologies. They can replace each other, complement each other or be integrated to solve larger problems.

To support you, we will now give you overview **examples of some domain-specific problems** within StarCraft II and overview **examples of some AI Techniques**.

- It is up to you to **look for** further information and to **choose** what you want to do.

What is not covered today!

- Today we don’t:

  - Discuss and define all problems in detail:
    You will explore and create your own variations of these general problems
    (we will see great variation between different groups - good!)
  - Discuss and specify solution techniques in detail:
    We give tips, you explore solutions (according to course objectives)
    (Once you have read up, you can raise your questions!)

Building placement

Building placement

- When you build new buildings, where should they be located?
  - May involve geometric reasoning
  - Have to think about what the map looks like - adapt to different maps
- Possible goals:
  - Avoid blocking your own units - let them move freely in your base.
  - Block places where the enemy might want to move.
  - Build walls to make it more difficult for the enemy to enter your base.
  - Find locations for multiple bases for decentralized production.

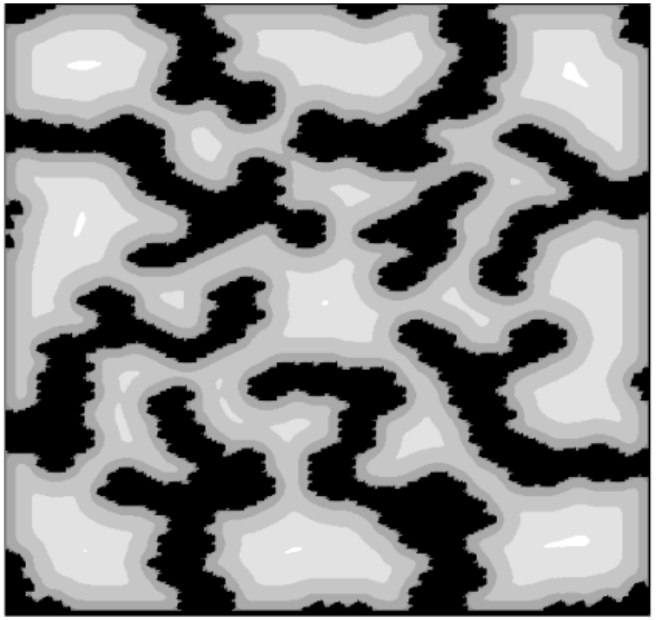
Building placement

- **Possible** solution for **some** objectives:
  - We may want to know where there are **free regions**
  - **and crowded areas**

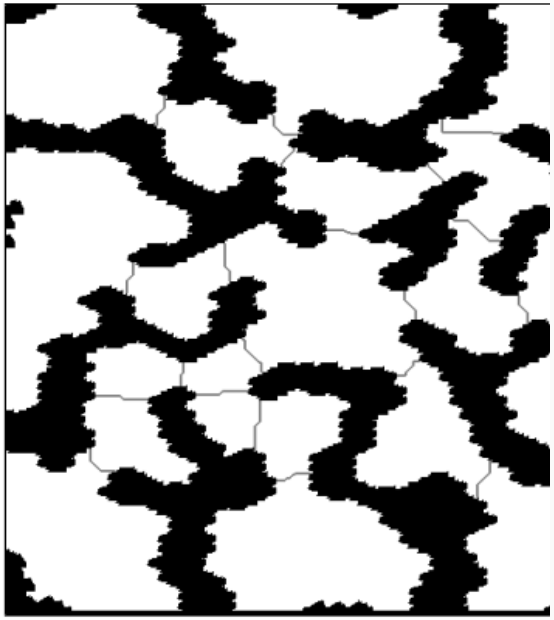
Find choke points

- Many AI-algorithms (litterature list in wiki):
  Halldórsson, Kári, and Yngvi Björnsson. **"Automated decomposition of game maps."**

Accessible terrain



Flood fill



Choke areas

Find choke points

- An other approach:: Using voronoidiagrams (Perkins, Luke. **"Terrain Analysis in Real-Time Strategy Games: An Integrated Approach to Choke Point Detection and Region Decomposition."**)

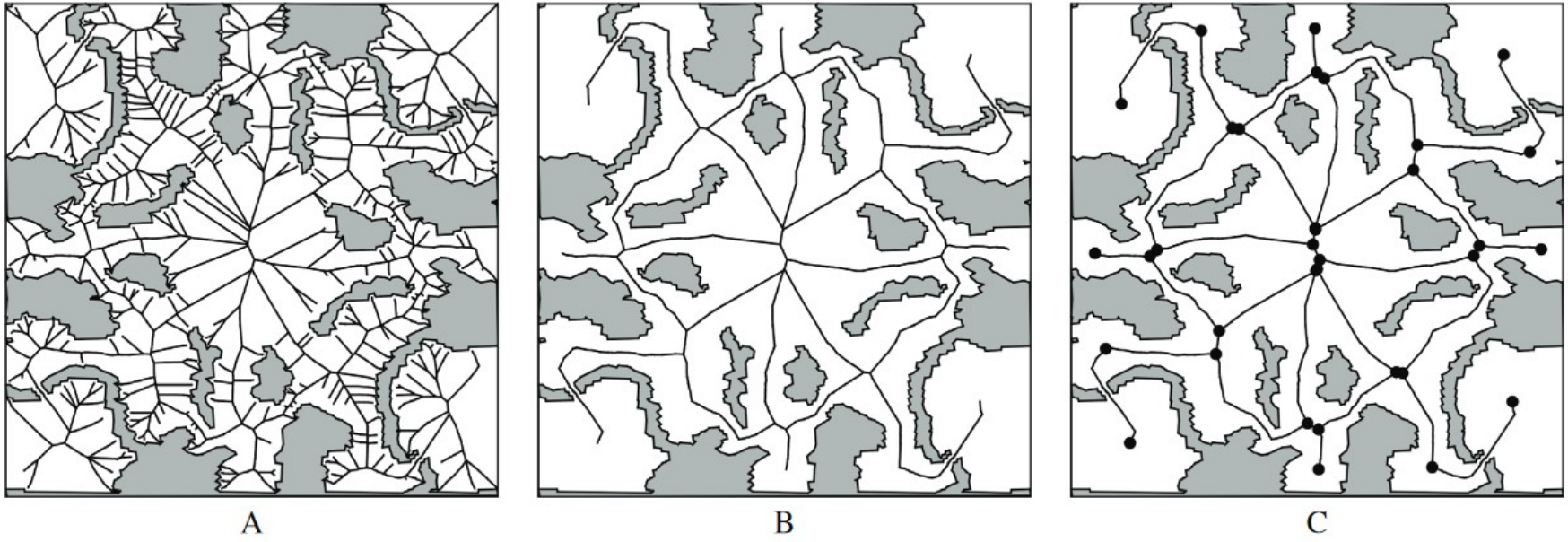


Figure 3: Steps one through four (A) Computed Voronoi diagram (B) Pruned Voronoi diagram (C) Identified region nodes

Production planning, building order

Production planning, building order

Given **priorities**, **overall objectives/strategy** and **current knowledge**:

- **What** should the agent build / produce / upgrade, and in what **order**?

Possible approach:

- **How do we best achieve a particular production target?**
  Input: Production goals (e.g. we want to produce an army to counter flying units as quickly as possible), budget/income, current buildings, etcetera.
  Output: Best possible plan/policy to achieve the production target.
- **Which production plan is best to counter the enemy's strategy?**
  Input and solution (algorithm) also need to take into account the adversary -- e.g. building defenses against attack or to counter flying units.

Production planning, building order

- Tech Tree shows in some sense what you need first
  - Cannot build B without A first

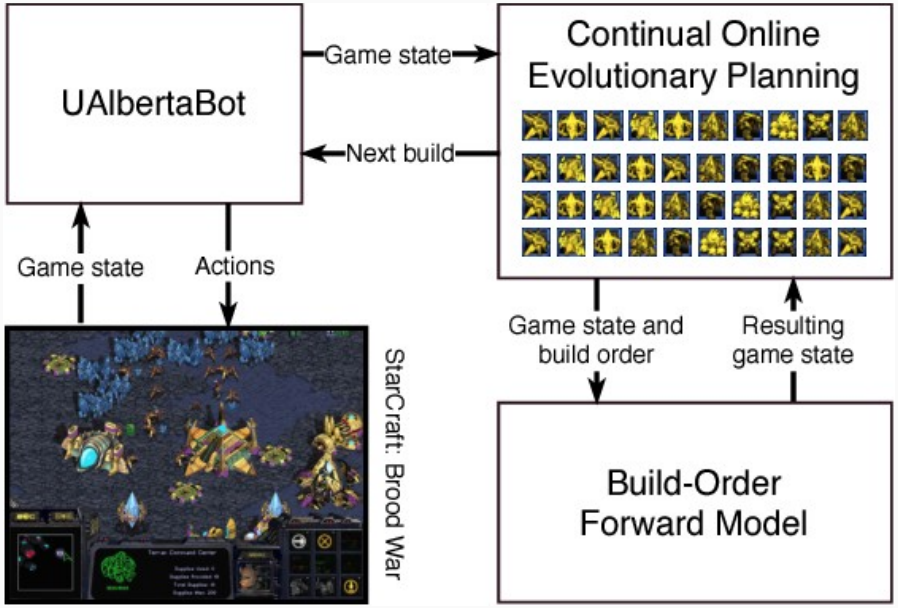
Production planning, building order

- Just **looking** at the Tech Tree is a too trivial task
  - Even if you use a search algoritm
- Must take into account more! Example:
  - How many units do we intend to build?
  - What resources are available? What *will be* available at the next stage?
  - How long does it takes to execute the different production plans?

Production planning, building order

Time	Action	Supply
0:17	Supply Depot	14 / 15
0:24	SCV	15 / 15
0:38	SCV	16 / 23
0:40	Barracks	16 / 23
0:44	Refinery	16 / 23
0:51	SCV	17 / 23
0:53	Refinery	17 / 23
1:03	SCV	18 / 23
1:15	SCV	19 / 23
1:27	Reaper	20 / 23

A common "*opening move*" -- a building order the player makes at the beginning of the game before the opponent has been scouted.



UAlbertaBot makes a very simplified simulation of StarCraft to approximate the future ("I'm producing X, so in 10 steps I'll have the resources to produce Y...")

Production planning, building order

- **Technical example**: Search, planing
  - Build an appropriate **model** of the **state** of the world and of the possible **actions**
  - Search through the admissible state:
    What can be done?
    What is the expected conscequence?
  - Find a plan that leads to reach the goals

Exploration of the map (scouting)

Problem: **Efficient exploration of the map (scout)**

Given **priorities, goal** and **current knowledge:**

- which part of the map should we scout?
- when should we scoun?
- in which order?
- …and when do we need to scout again?

Example:

- Find and track your opponent's army.
- Find out how many bases your opponent has and where their location.
- Keep an eye out for sneak attacks (Pearl Harbour).

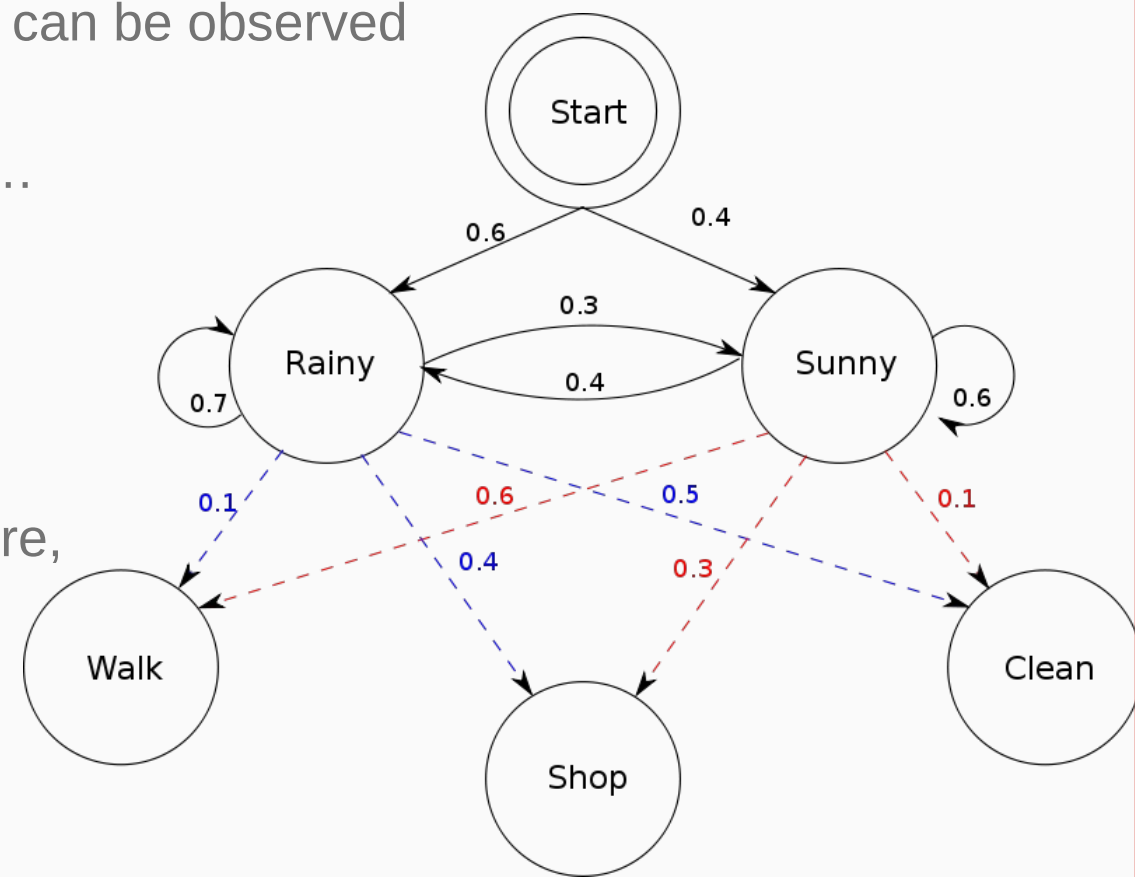
Input: Previous observations of the opponent's army, the Scout's (or Scouts') position and the map.

Output: A sequence of positions on the map to which the scout should move.

Example: Hidden Markov Models

- Some aspects of the world can be observed directly
- We are interested in other… that *affect* the observables

- StarCraft:
  “If we saw enemy units there, what can we conclude…”

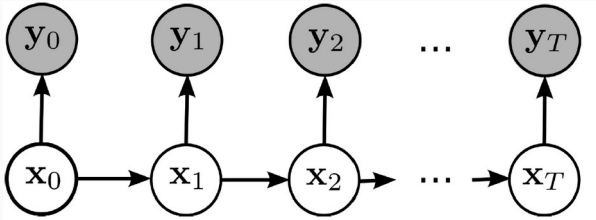


Problem: **Efficient exploration of the map (scout)**

Hidden Markov Models (HMMs).

Given the **current observation**
and what was **observed in the past**,

- P(opponent_base | base_location) = ?,
- P(opponent_units | position) = ?,
- P(opponent_combat_power | position) = ?,
- …

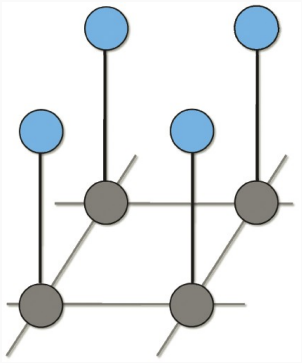


Exempel på temporal HMM.

“When was location **x** observed last?”

vs

“How <u>likely</u> is that there is something <u>new</u> to observe at location **x**?”



Exempel på spatial HMM (på en grid).

Problem: **Efficient exploration of the map (scout)**

Some other useful technics:

- Search
- Cost fields
- Potential fields

Find a safe path

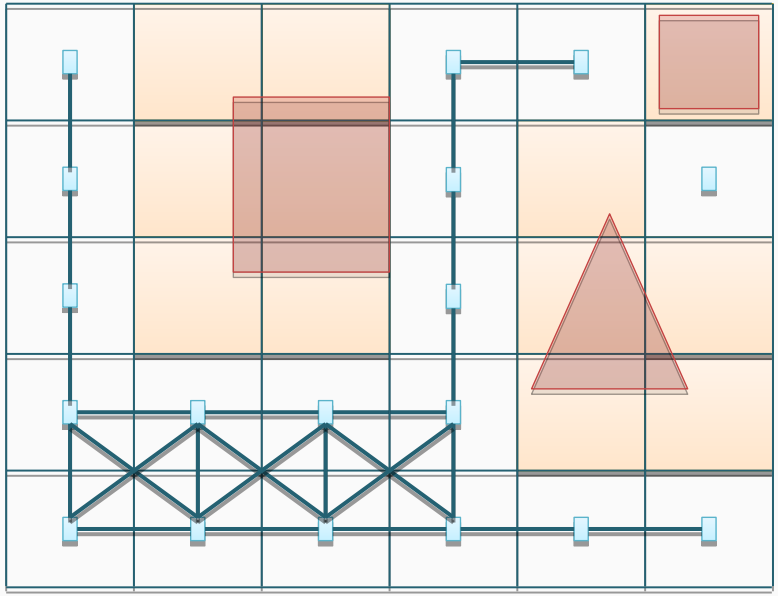
Find a safe path

- StarCraft has a built-in functionality to go from A to B
  - It does *not* take into account the location of ennemies
  - Avoid shelling? Stay out of enemy range
  - Avoid *detection*? Stay further away
  - (Require information from exploration, may require scouting…)

Find a safe path

- Example: Search
  - Can create **nodes** according to a **grid**
  - Connect nearby nodes with **arcs**
  - Find a path through the grid
    (i.e. with A* and a reasonable heuristic)

- Keep in mind:
  - Arcs should have a **cost**
  - Expensive arcs on location to avoid
    (i.e. high risk of shelling)
  - What is **arc cost function?**

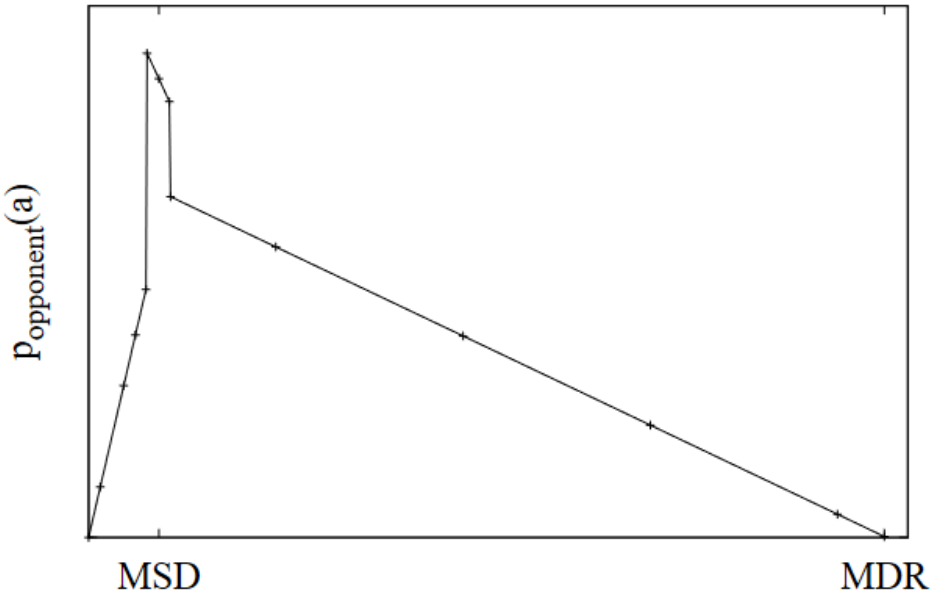


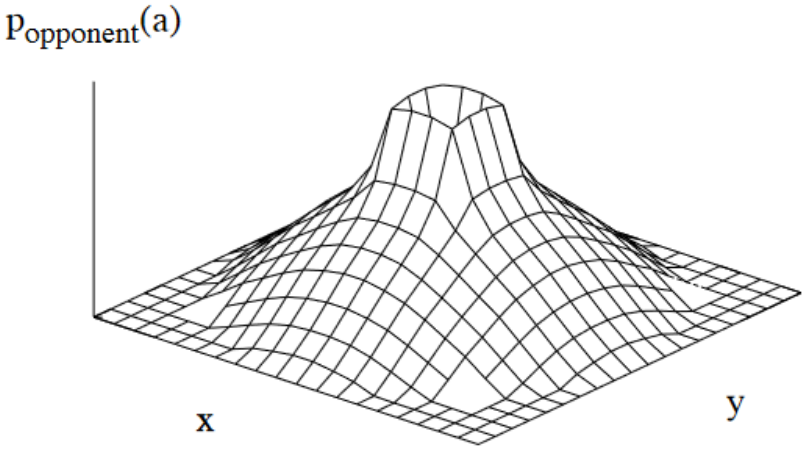
Find a safe path

- **Potential fields**
  - Potential: Each point is attractive / repulsive potential
  - Can be combined with search: Low potential defines a cost function

Find a safe path

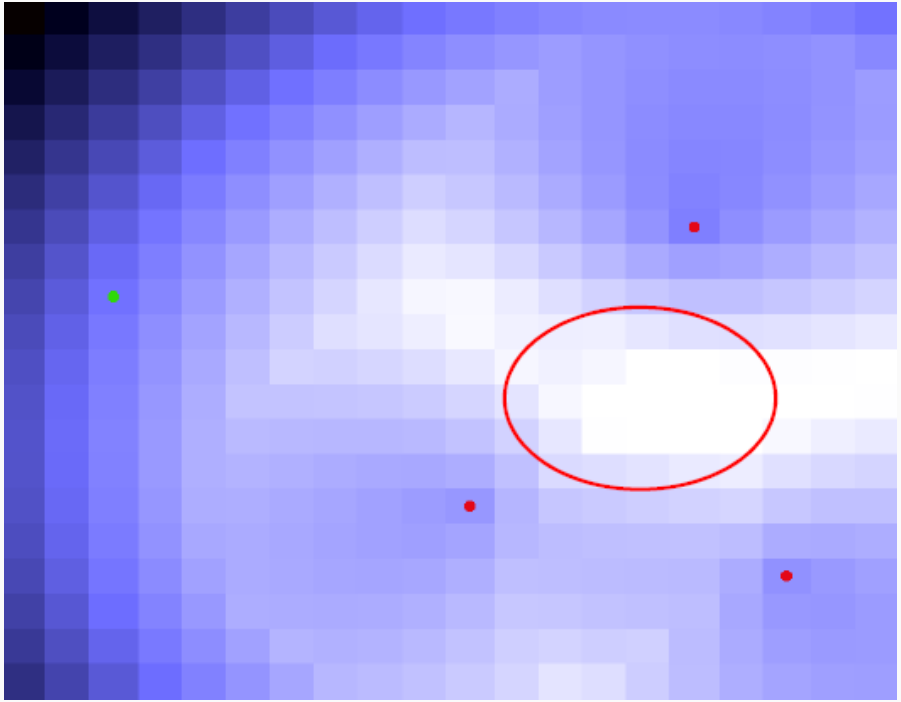
- **Potential fields -- example**
  - It is bad to be closed to an enemy's MSD, Maximum Shooting Distance.
  - Good to be right at the edge of the MSD.
  - Outside of the Maximum Detection Range you are not affected

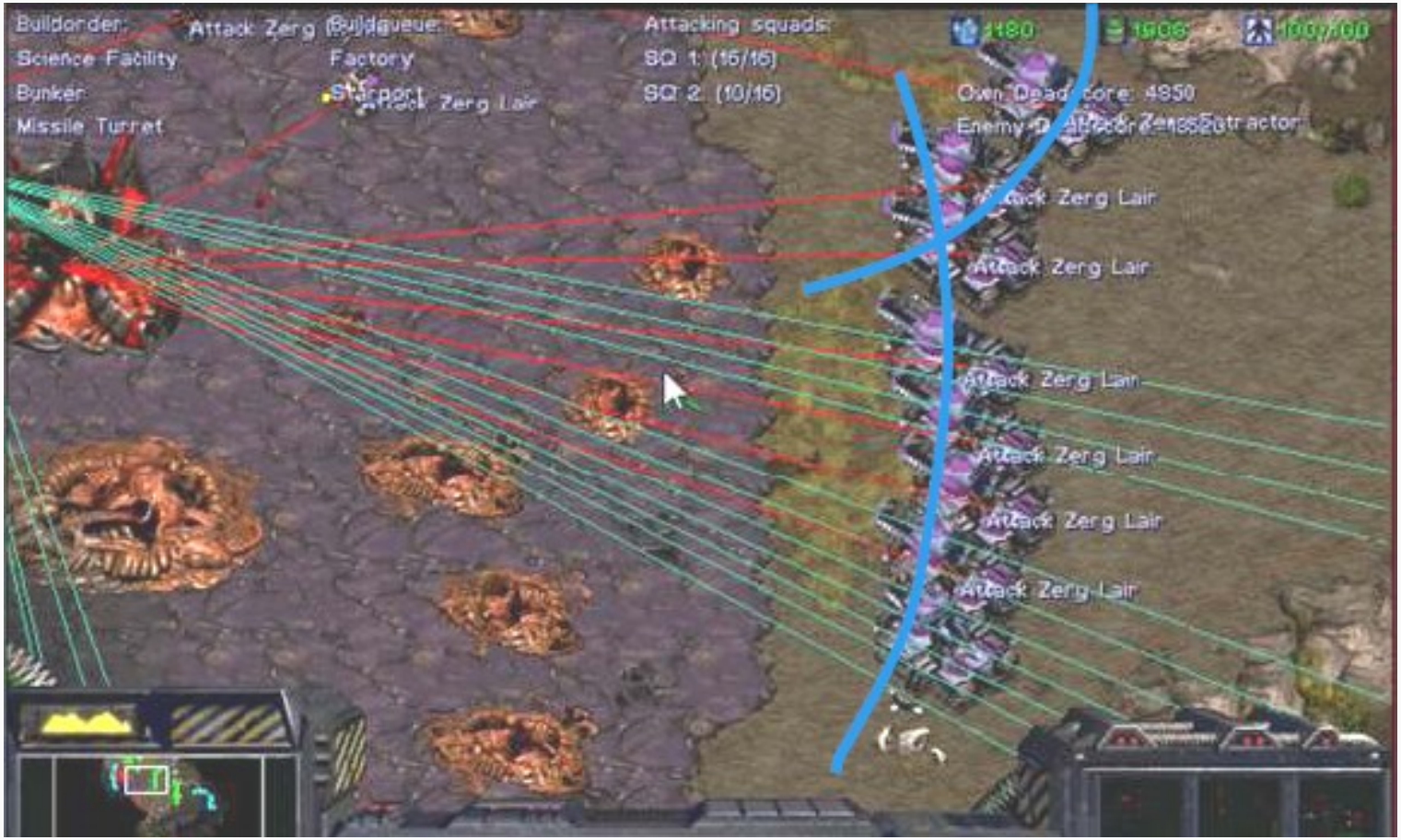




Find a safe path

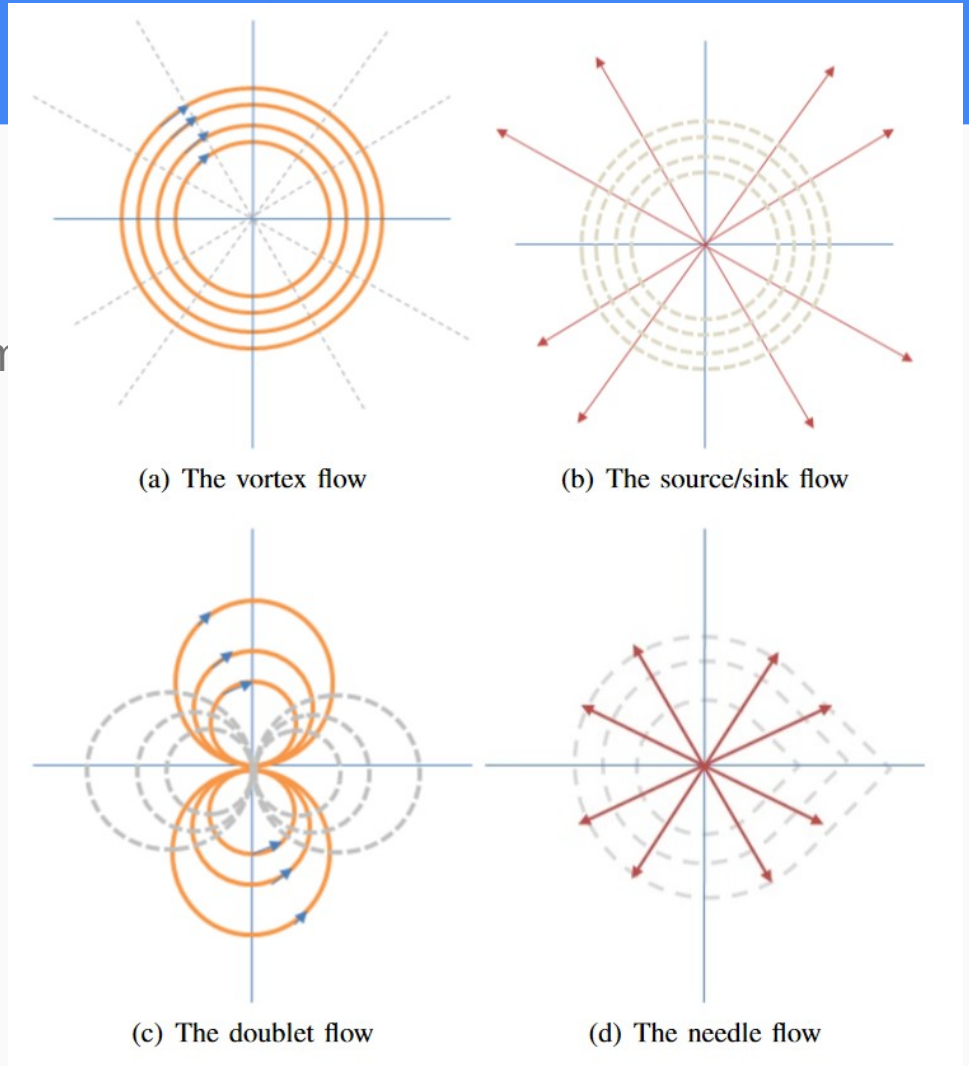
- Adding potentials' from different attracting and repelling positions
- In each step, go to the best potential within n cells from where you are




Builderorder:
Science Facility
Bunker
Missile Turret
Attack Zerg Buildqueue:
Factory
Starport
Attack Zerg Lair
Attacking squads:
SQ 1: (15/16)
SQ 2: (10/16)
1180
1900
100/200
Own Dead core: 4850
Attack Zerg Lair
Attack Zerg Lair
Attack Zerg Lair
Attack Zerg Lair
Attack Zerg Lair
Attack Zerg Lair

Find a safe path

- **<u>Flow fields</u> –**
  similar, but:
  - Potential field may have local optim (where you are stuck)
  - Flow fields use other ”components” to guarantee that there are no local optima



Choosing strategy/tactics

Problem: **Choosing strategy/tactics**

- Example: we need to change the general strategy for an agent
  - Should the agent be expending? (build up economie and industrie), Defend, or launch an Offensive right now?
- And/or: We want to choose tactics
  - How do we implement a given strategy?

Problem: **Choosing strategy/tactics**

Technic example: Supervised Learning with Bayesian Networks (supervised ≅ generalisation from example from correct answers)

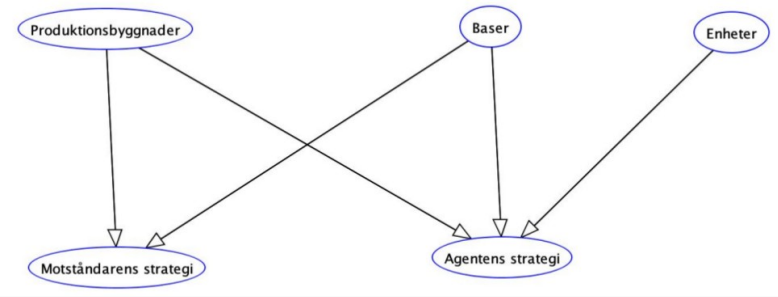
Example:

- strategy ∈ {Expand, Defend, Offensive}
- state ∈ {#base, #building of type x, #offsenive units of type y, …}
- observations: of the opponent state (what is intresting?)

P(own_strategy | observation, own_state) = ?
P(opponent_strategy | observation) = ?

Choose/adopt the strategy with highest probability.

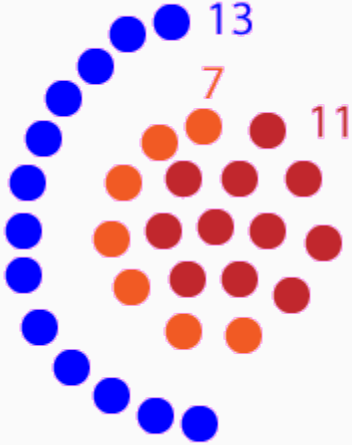
Can use replays fpr training



Troop formations and attack patterns

Troop formations and attack patterns

- In many situations you need to control the position for each **members of group**
  - Example: Troop formation, ex. to attack in circle
  - Exempel: Micromanagement – focused firing, kiting, exploiting strength/weakness…



Troop formation, attack pattern

Example of technic: Flocking/Swarming with Flow fields / Potential fields

Different behaviours are good for different tactics

- Deal maximun damage/time unit
- Avoid damage (range, melee, AoE)
- Motion formation (avoid detection)

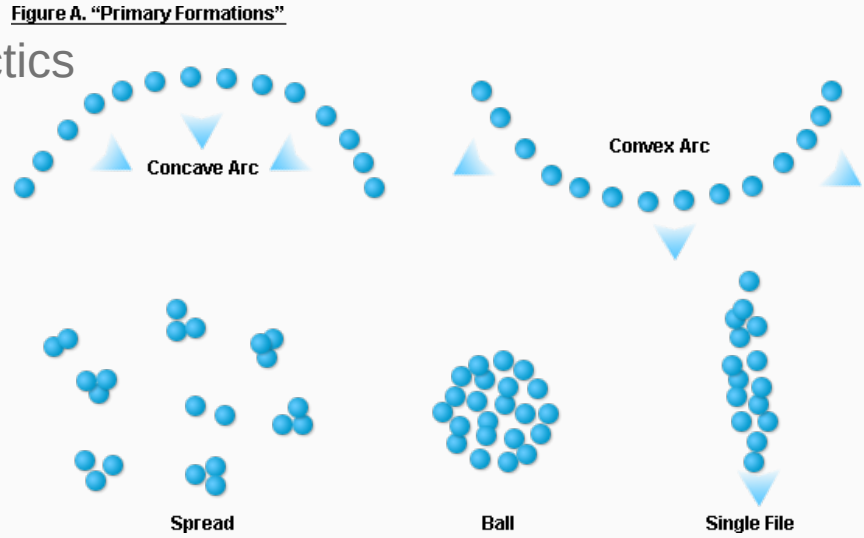
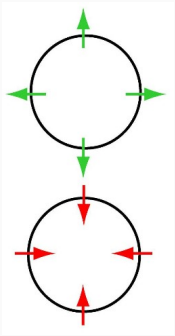


Figure From:
http://www.teamliquid.net/forum/sc2-strategy/187892-positioning-formations-and-tactics

Attraction and repelant



You can also use more general vector fields...

Sub-problem: **Micromanagement (with attack patterns)**

Example of technique: Reinforcement Learning with Q-learning

(For each decision there is a reward/"punishment"; which strategy is best in the long run?)

The units are independent of each other

- Original behavior

- Avoid death
- Focused firing (law HP)

Sub-problem: **Micromanagement (with attack patterns)**

Technic: Reinforcement Learning med Q-learning

1) Select interesting features

- Example: own health, opponent's health, relative distance, combat power...

2) Discrete features

- Example: own health $\in$ {0, 10, 20, 30, …, 90, 100} (percentage)

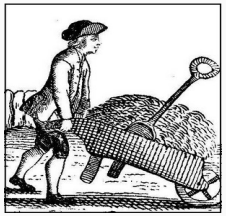
3) Select utility function:

Example: 100*#*own_troop* + #*own_health* – 100*#*opponent_troop* - #*opponent_helath*

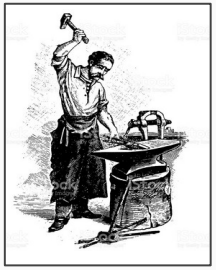
4) Construct representative scenario and train from Q-tabel: **P**(action | state)

Assignment of agents to tasks

The allocation problem



Task 1

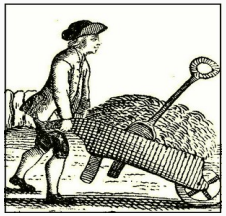


Task 2

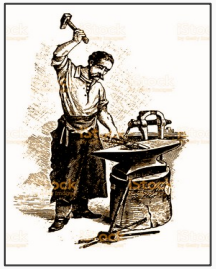
Task 3

Agents/units with different skills and abilities

The allocation problem



Task 1



Task 2

Task 3

Agents/units with different skills and abilities

The allocation problem

Technic: The Hungarian method, min-cost max-flow, local search, ant-colony optimization

StarCraft-example: Assign units to "jobs", for example:

- $A = \{SCV_1, SCV_2, SCV_3, Marine_1, Marine_2\}$
- $B = \{$build, collect mineral, collect gas, scout, attack$\}$
- **Constraints:** All units must be given exactly one job each.
- **Costfunction**: *Distance* between agent $a \in A$ and task $b \in B$.

Example of assignemnt:

$\{(SCV_1$,build), (SCV_2,mineral), (SCV_3,gaz), ($Marine_1$,scout), ($Marine_2$,attack)$\}$

The allocation problem

Also need to be handled in StarCraft:

- **Micro:** Assign targets in combat, focus firepower.
- **Produktion:** Decide which building will produce what.
- **“Strategy”:** Assign abstract tasks (e.g. "attack base" or "defend area") to groups of units.

... and much more!

The grouping problem

The grouping problem



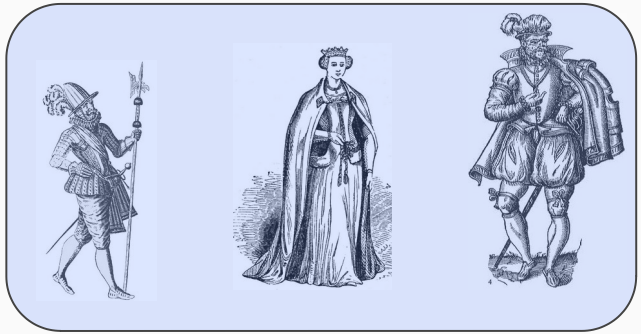




Agents/units with different skills and abilities

The grouping problem

Coalition Blue



Value: 3

Coalition Red

Value: 1

Coalition Green

Value: 5

Total value:

3 + 1 + 5 = 9

The grouping problem

Technic: Coalition structure generation, team formation, optimering, branch-and-bound

StarCraft example: Create effective teams/groups that have good synergies (e.g. medivac that can heal troops + troops that can fight)

- $A = \{Marauder_1, Marauder_2, Medivac_1, Medivac_2, Marine_1, Marine_2, Marine_3, Marine_4, Marine_5, Tank_1\}$
- **Constraint:** Each entity must be a member of exactly one team.
- **Value function**: Many different aspect to define...

Tips and Summary

Tips

Think about:

- … that there is a good **bibliography** with relevant references on the course **wiki.**
- … that you do not to choose any of the problems we have used as example or solve them with the technics we mentioned.
- … how your agent will perform if you don’t have a good solution to a problem saknar en bra lösning.

Tips

Think about:

- … that a solution requires *input data*
  - Search methods over a graph. What does it contains? How are they defined?
  - Potential fields requires attractors / repelants. How are they defined?
- ...and give an *output*, or *control* something
  - Potential fields gives a vector field. How do you use it?

Summary

There are many interesting AI problems to solve and many techniques and algorithms to experiment with!

Think about what you want to do! Complement each other!