Problemområden i Starcraft II

...och AI-tekniker för att tackla problemen

Mattias Tiger
Fredrik Präntare
Jonas Kvarnström

Introduktion och motivering

Ni ska inför er individuella uppgift **definiera ett problem** och **välja ut en eller flera AI-tekniker** som kan tänkas användas för att lösa detta problem.

- Det finns väldigt många *domänspecifika problem* i StarCraft II att lösa.
  Man behöver inte lösa *alla*, utan välja sådana som kan bidra till en bra agent.
- Det finns väldigt många AI-tekniker. De kan ersätta varandra, komplettera varandra eller integreras för att lösa större problem.

För att stödja er ska vi nu ge er översiktliga **exempel på några domänspecifika problem** inom StarCraft II och översiktliga **exempel på några AI-Tekniker**.

- Det är upp till er att **leta** ytterligare information och att **välja** vad ni vill göra.

Vad vi inte gör idag

- Idag ska vi inte:
  - Diskutera och definiera alla problem i detalj:
    Ni ska själva utforska och skapa egna varianter av dessa generella problem
    (vi ser stor variation mellan olika grupper – bra!)
  - Diskutera och specificera lösningstekniker i detalj:
    Vi ger tips, ni utforskar lösningar (enligt kursmålen)
    (När ni har läst på kan ni ta upp era frågor!)

Byggnadsplacering

Byggnadsplacering

- När du bygger nya byggnader, var ska de placeras?
  - Kan innebära geometriska resonemang
  - Måste tänka på hur kartan ser ut – anpassa sig till olika kartor

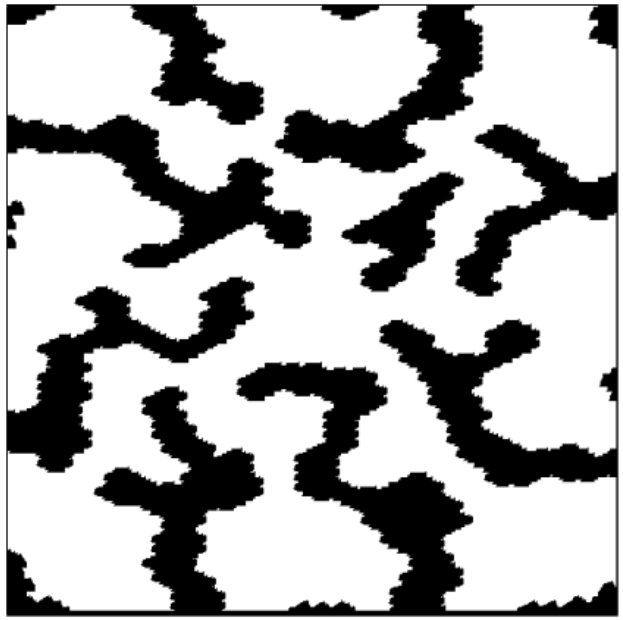
- Möjliga mål:
  - Undvika att blockera de egna enheterna – låt dem röra sig fritt i er bas.
  - Blockera platser där fienden kan tänkas vilja röra sig.
  - Bygga murar för att göra det svårare för fienden att ta sig in i er bas.
  - Hitta platser för flera baser för decentraliserad produktion.

Byggnadsplacering

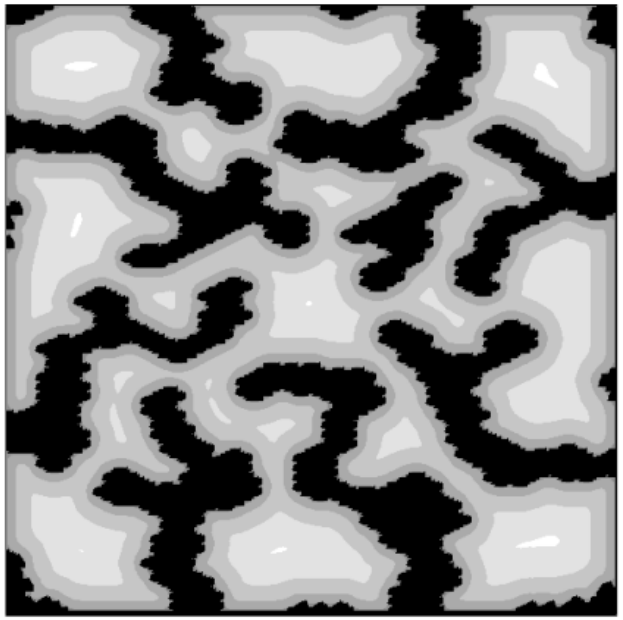
- **Del** av lösningen för **vissa** mål:
  - Vi kan tänkas vilja veta var det finns **fria regioner** och **trånga områden**

Hitta trånga områden (choke points)

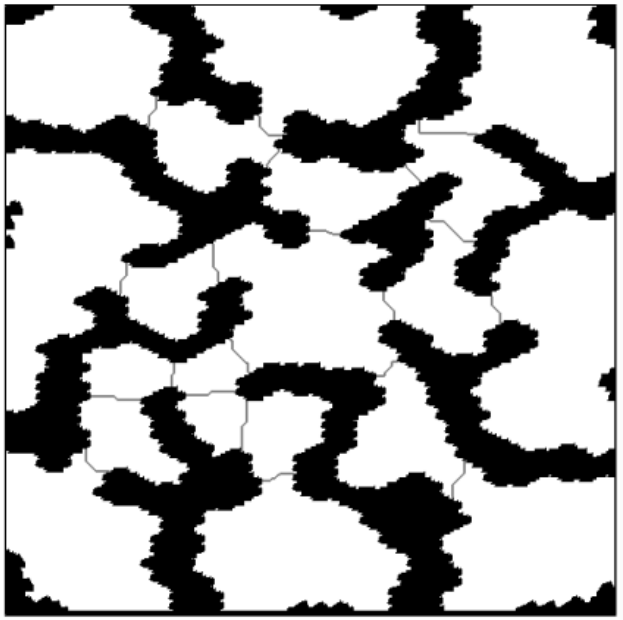
- En av flera AI-algoritmer (litteraturlistan i wikin):
  Halldórsson, Kári, and Yngvi Björnsson. "Automated decomposition of game maps."



Accessible terrain



Flood fill



Choke areas

Hitta trånga områden (choke points)

- Annan teknik: Med Voronoidiagram (Perkins, Luke. **"Terrain Analysis in Real-Time Strategy Games: An Integrated Approach to Choke Point Detection and Region Decomposition."**)

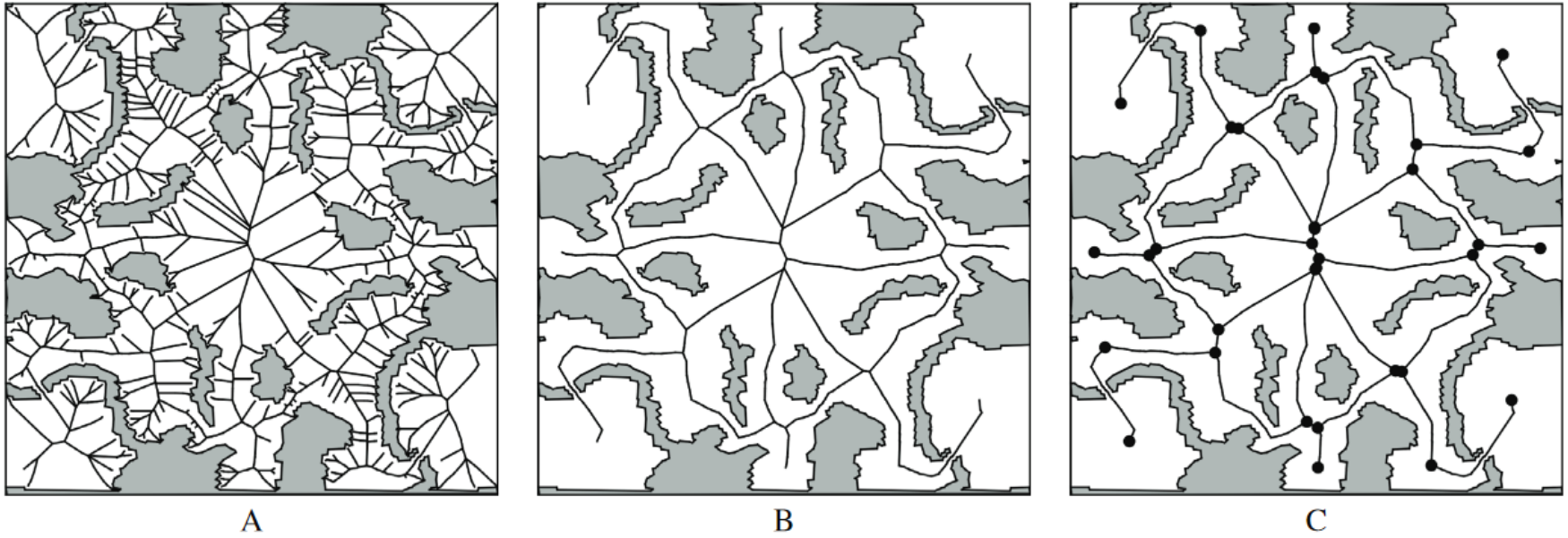


Figure 3: Steps one through four (A) Computed Voronoi diagram (B) Pruned Voronoi diagram (C) Identified region nodes

Produktionsplanering, byggnadsordning

Produktionsplanering och byggnadsordning

Givet **prioriteringar, övergripande mål/strategi** och **nuvarande kunskap:**

- **Vad** ska byggas / produceras / uppgraderas, och i vilken **ordning**?

Tänkbara konkretiseringar:

- **Hur uppnår vi ett visst produktionsmål på bästa sätt?**
  Input: Produktionsmål (t.ex. vi vill producera en armé för att kontra flygande enheter så snabbt som möjligt), budget/inkomst, nuvarande byggnader, etcetera.
  Output: Bästa möjliga plan/policy för att uppnå produktionsmålet.
- **Vilken produktionsplan är bäst för att kontra fiendens strategi?**
  Input och lösning (algoritm) behöver också ta hänsyn till motståndaren -- t.ex. bygga försvar mot anfall eller för att kontra flygande enheter.

Produktionsplanering och byggnadsordning

- Tech Tree visar i någon mening vad man behöver först
  - Kan inte bygga B innan man har A

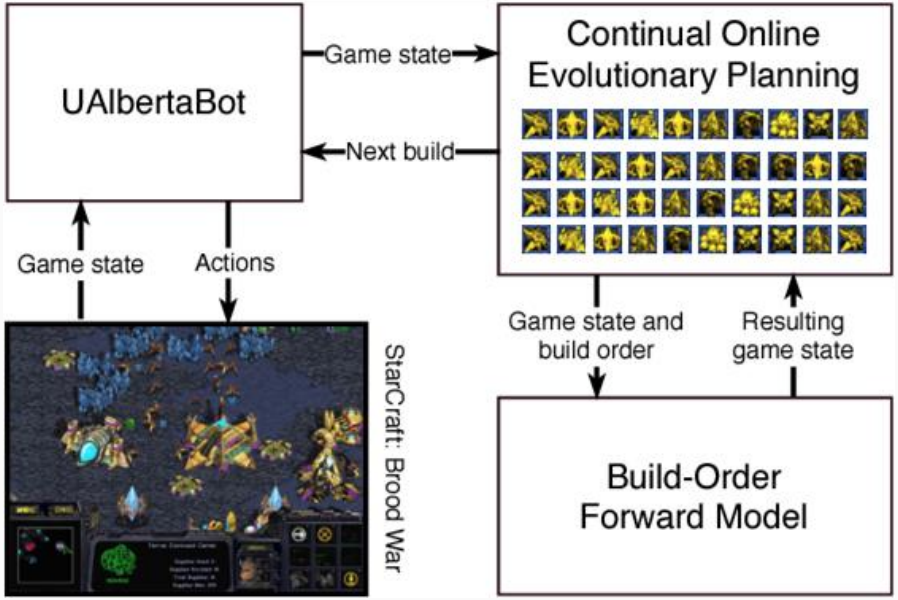
Produktionsplanering och byggnadsordning

- Att **bara** titta på Tech Tree är för trivialt
  - Även om man t.ex. använder en sökalgoritm
- Måste ta hänsyn till mer! Exempel:
  - Hur många enheter vill vi bygga?
  - Vilka resurser har vi tillgängliga? Vilka *kommer* vi att ha tillgängliga när vi bygger nästa steg?
  - Hur lång tid tar det att genomföra alternativa produktionsplaner?

Produktionsplanering och byggnadsordning

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

En vanlig “*opener*” -- en byggnadsordning spelaren gör i början av spelet innan motståndaren har scoutats.



UAlbertaBot gör en mycket förenklad simulering av StarCraft för att approximera framtiden (“Jag håller på att producera X, så om 10 steg har jag resurser för att producera Y…”)

Produktionsplanering och byggnadsordning

- **Teknikexempel**: Sökning, planering
  - Bygg upp en lämplig **modell** av världens **tillstånd** och av de **handlingar** som kan utföras
  - Sök genom en tillståndsrymd:
    Vad kan man göra?
    Vad händer då?
  - Hitta en plan som leder till att man har det som önskades

Utforskning av kartan (scouting)

Problem: **Utforska kartan effektivt (scout)**

Givet **prioriteringar, mål** och **nuvarande kunskap:**

- vilka delar av kartan bör vi scouta,
- när bör vi scouta och
- i vilken ordning?
- …och när behöver vi göra det igen?

Exempel:

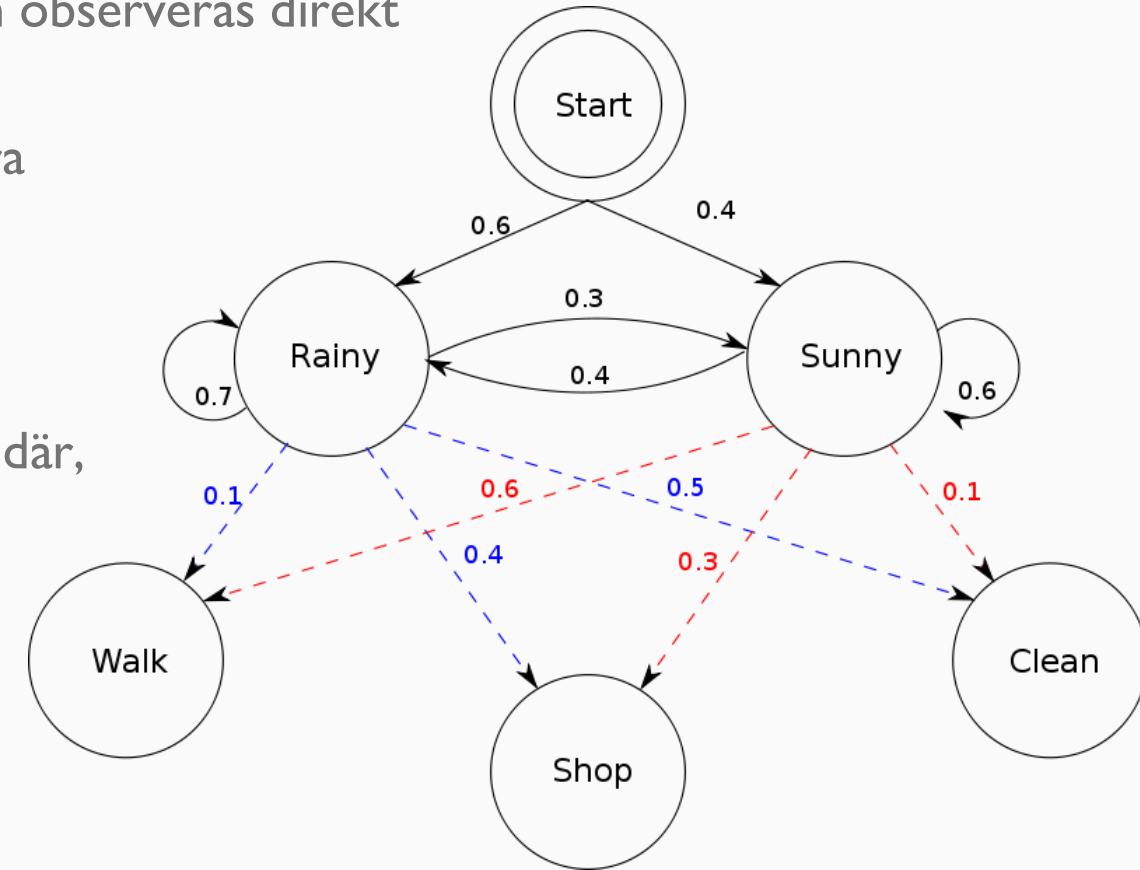
- Hitta och spåra motståndarens armé.
- Ta reda på hur många baser motståndaren har och var de finns.
- Håll utkik efter smyganfall.

Input: Tidigare observationer av motståndarens armé, scoutens (eller scouternas) position och kartan.
Output: En sekvens av positioner på kartan som scouten skall röra sig till.

Exempel: Hidden Markov Models

- Vissa aspekter av världen kan observeras direkt
- Vi är intresserade av andra…
  som *påverkar* de observerbara

- StarCraft:
  "Om vi såg fiendens enheter där,
  så tror vi…"



Problem: **Utforska kartan effektivt (scout)**

Hidden Markov Models (HMMs).

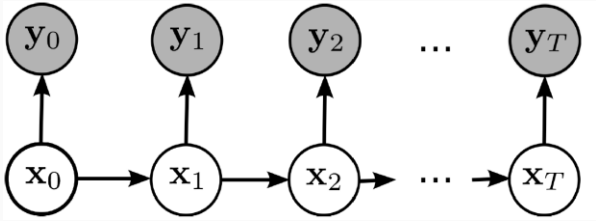
Givet vad **ni observerar just nu**
och vad **ni har observerat tidigare**,

- P(motståndar_bas | bas_plats) = ?,
- P(motståndarenheter | position) = ?,
- P(motståndarstridskraft | position) = ?,
- …

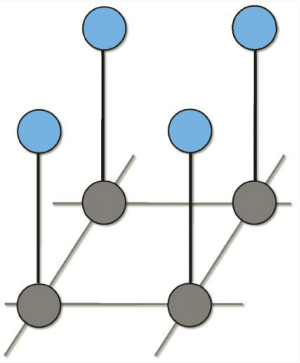
“När utforskades plats **x** sist?”

vs

“Hur troligt är det att det finns något nytt att observera på plats **x**?”



Exempel på temporal HMM.



Exempel på spatial HMM (på en grid).

Problem: Utforska kartan effektivt (scout)

Fler användbara tekniker:

- Sökning
- Cost fields
- Potentialflöden

Hitta säker väg

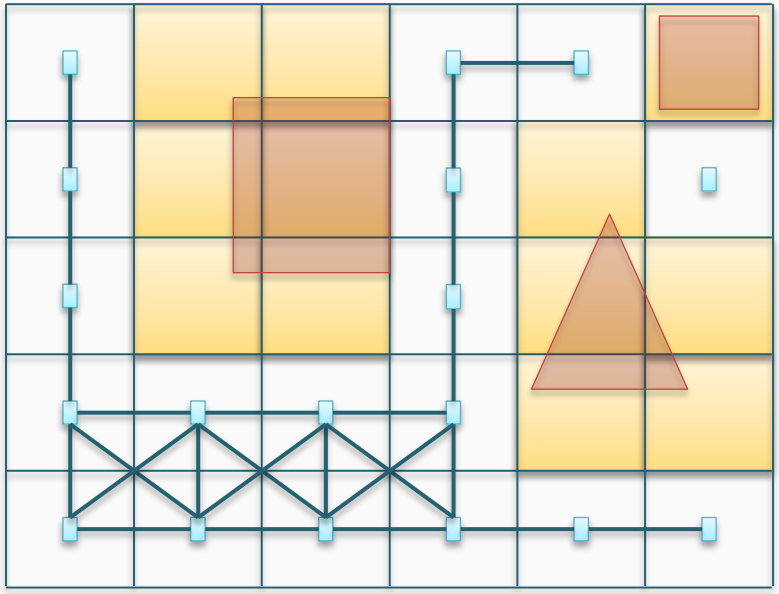
Hitta säker väg

- StarCraft har en inbyggd funktionalitet för att gå från A till B
  - Den tar *inte* hänsyn till var fiender finns
  - Undvik beskjutning? Håll dig utanför fiendens räckvidd
  - Undvik att bli *upptäckt*? Håll dig ännu längre bort
  - (Kan kräva information från scoutingen…)

Hitta säker väg

- Exempel: Sökning
  - Kan skapa **noder** motsvarande ett **rutnät**
  - Koppla ihop närliggande noder med **bågar**
  - Sök en väg genom rutnätet
    (med t.ex. A* och en rimlig heuristik)

- Att tänka på:
  - Bågar ska ha **kostnader**
  - Dyra bågar där vi vill undvika att gå
    (t.ex. trolig risk att bli beskjuten)
  - Så vad är **bågkostnadsfunktionen?**

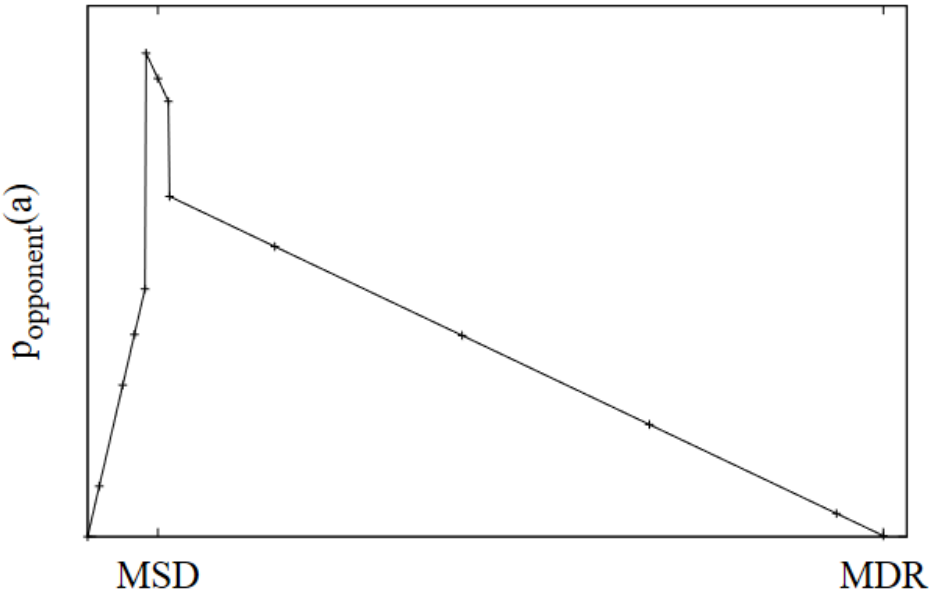


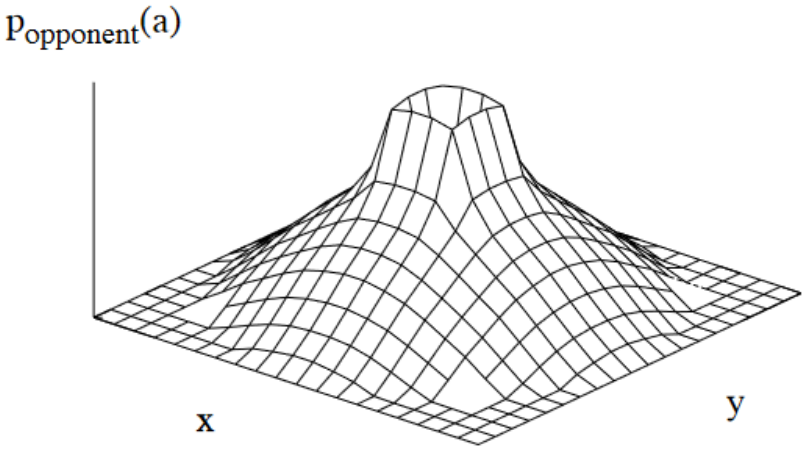
Hitta säker väg

- **Potentialfält**
  - Potential, inte potentiellt: Varje punkt har en attraherande / repellerande potential (tänk elektriskt: *spänning* är *skillnad i potential)*
  - Kan kombineras med sökning: Låt potentialfält definiera bågkostnader

Hitta säker väg

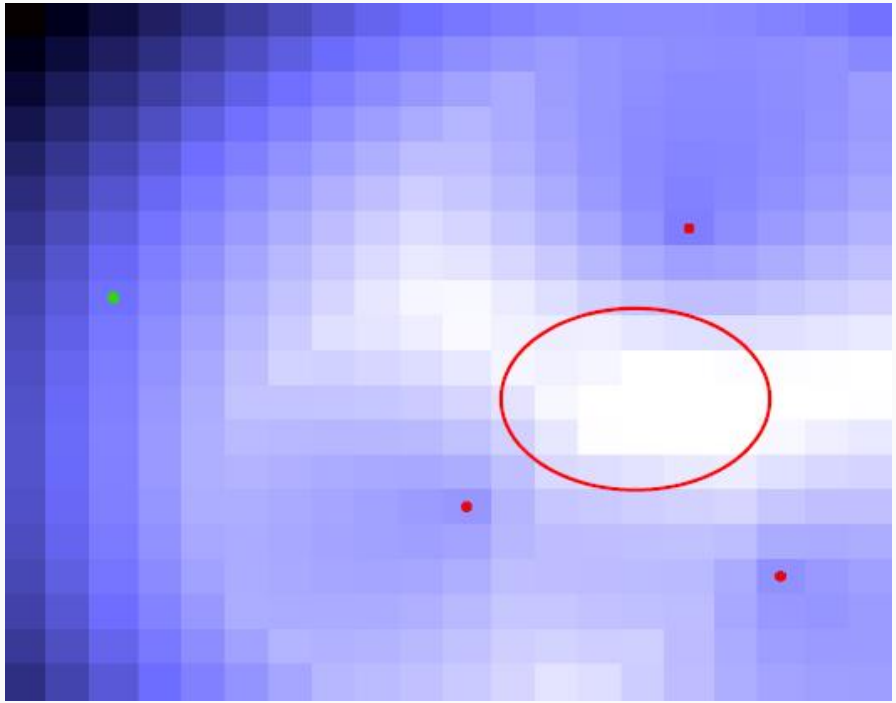
- **Potentialfält -- exempel**
  - Dåligt (onödigt) att vara närmare fienden än MSD, Maximum Shooting Distance.
  - Bra att vara precis vid MSD.
  - Utanför Maximum Detection Range påverkas man inte (man ser inte fienden)





Hitta säker väg

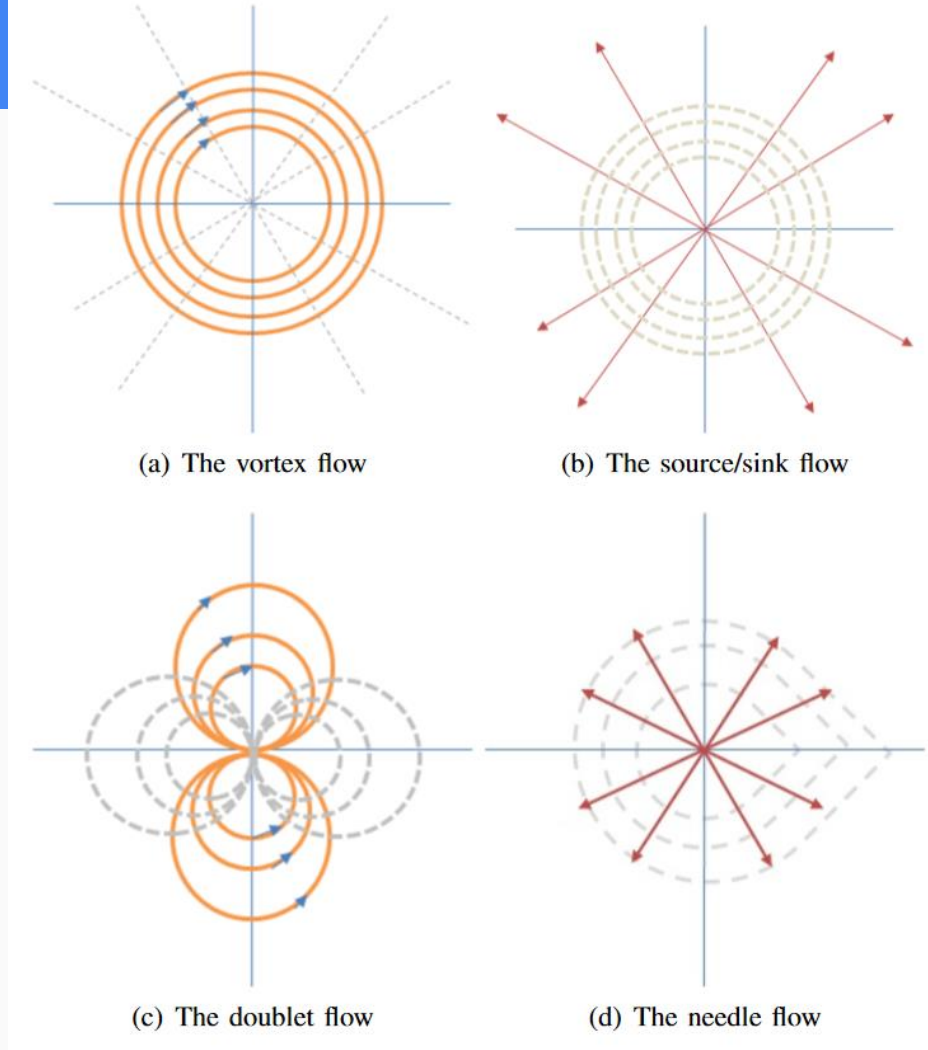
- Addera potentialer´från olika attraherande och repellerande positioner
- I varje steg, gå till den bästa potentialen inom n celler från där du är




Buildorder:
Science Facility
Bunker
Missile Turret
Attack Zerg
Buildqueue:
Factory
Starport
Attack Zerg Lair
Attacking squads:
SQ 1: (15/16)
SQ 2: (10/16)
1180
1906
Own Dead Score: 4850
Attack Zerg Lair
Attack Zerg Lair
Attack Zerg Lair
Attack Zerg Lair
Attack Zerg Lair
Attack Zerg Lair

Hitta säker väg

- **<u>Potentialflöden / Flow fields</u> – liknande, men:**
  - Potentialfält kan ha lokala optima (där man ”fastnar”)
  - Potentialflöden skapas från andra ”komponenter” på ett sätt som garanterar att inga lokala optima finns



Välja strategi / taktik

Problem: Välja strategi, taktik

- Exempel: Vi vill välja en övergripande strategi för vår agent
  - Ska agenten vara Expansiv (bygga upp ekonomi och industri), Defensiv, eller Offensiv just nu?
- Och/eller: Vi vill välja taktik
  - Hur ska vi genomföra den valda strategin?

Problem: Välja strategi, taktik

Teknikexempel: Supervised learning med Bayesian Networks

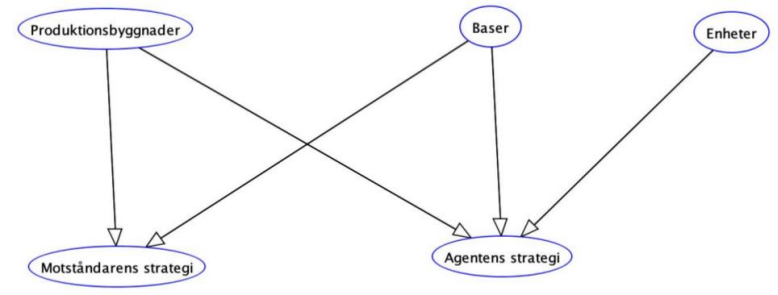
Exempel:

- strategi ∈ {Expansiv, Defensiv, Offensiv}
- tillstånd ∈ {#baser, #byggnader av typ x, #försvarande enheter av typ y, …}
- observationer: iakttagelser av motståndarens tillstånd (vad är intressant?)

P(egen_strategi | observationer, eget_tillstånd) = ?
P(motståndare_strategi | observationer) = ?

Välj/anta strategi med högst sannolikhet.

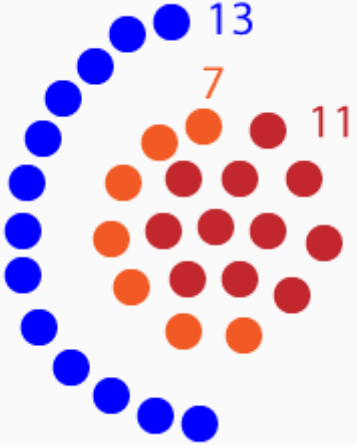
Kan använda replays för att träna (vissa risker)



Truppformationer och attackmönster

Truppformationer och attackmönster

- I många lägen behöver man **detaljstyra positionen** för **medlemmar av en grupp**
  - Exempel: Truppformationer, t.ex. att attackera i en konkav båge
  - Exempel: Micromanagement – fokuserad eldgivning, kiting, utnyttja styrkor/svagheter, …



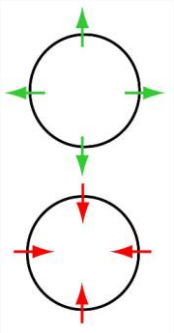
Truppformationer, attackmönster

Teknikexempel: Flocking/Swarming genom Flow fields / Potential fields

Olika beteenden är bra för olika taktiker

- Utdela maximal skada/tid
- Undvika skada (range, melee, AoE)
- Marschering (undvik att bli upptäckt)

Attraktorer och repellerare



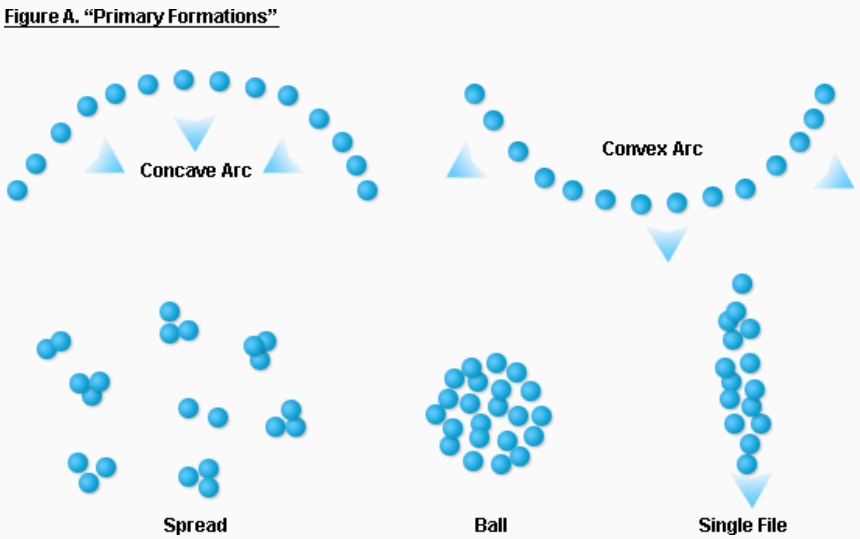


Figure From: http://www.teamliquid.net/forum/sc2-strategy/187892-positioning-formations-and-tactics

Man kan även använda mer generella vektorfält...

Delproblem: **Micromanagement (inom attackmönster)**

Teknikexempel: Reinforcement Learning med Q-learning

(För varje beslut får man belöning/”straff”; vilken strategi blir bäst i längden?)

Enheterna är oberoende av varandra.

- Ursprungligt beteende

- Undviker att dö
- Fokuserad eldgivning (lägst HP)

Delproblem: **Micromanagement (inom attackmönster)**

Teknik: Reinforcement Learning med Q-learning

1) Välj ut intressanta features

- Exempel: egen hälsa, motståndarhälsa, relativt avstånd, stridskraft…

2) Diskretisera features

- Exempel: egen hälsa ∈ {0, 10, 20, 30, …, 90, 100} (procent)

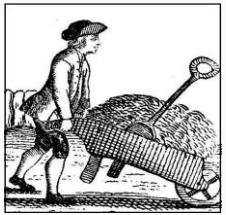
3) Välj nyttofunktion (utility) att maximera

- Exempel: 100*#*egna_trupper* + #*egen_hälsa* - 100*#*motståndartrupper* - #*motståndarhälsa*

4) Konstruera representativa scenarier och träna fram Q-tabellen: **P**(action | state)

Tilldelning av agenter till uppgifter

Tilldelningsproblemet



Task 1

Task 2

Task 3

Agents/units with different skills and abilities

Tilldelningsproblemet

Task 1

Task 2

Task 3

Agents/units with different skills and abilities

Tilldelningsproblemet

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

StarCraft-exempel: Tilldela enheter till "jobb", exempelvis:

- ***A* = {*SCV*$_1$, *SCV*$_2$, *SCV*$_3$, *Marine*$_1$, *Marine*$_2$}**
- ***B* = {bygga, samla mineraler, samla gas, scouta, anfalla}**
- **Constraint:** Alla enheter måste ges exakt ett arbete var.
- **Kostnadsfunktion**: *Avståndet* mellan agent ***a*∈*A*** och uppgift ***b*∈*B***.

Exempel på lösning:

{(*SCV*$_1$,bygga), (*SCV*$_2$,mineraler), (*SCV*$_3$,gas), (*Marine*$_1$,scout), (*Marine*$_2$,anfalla)**}**

Tilldelningsproblemet

Kan, i StarCraft, också användas till:

- **Micro:** Tilldela mål i strid, fokusera eldkraft.
- **Produktion:** Bestämma vilken byggnad som skall producera vad.
- **“Strategi”:** Tilldela abstrakta uppgifter (t.ex. “anfall bas” eller “försvara område”) till grupper av enheter.

… och mycket mer!

Grupperingsproblemet

Grupperingsproblemet







Agents/units with different skills and abilities

Grupperingsproblemet

Coalition Blue

Värde: 3

Coalition Red

Värde: 1

Coalition Green

Värde: 5

Totalt värde:

3 + 1 + 5 = 9

Grupperingsproblemet

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

StarCraft-exempel: Skapa effektiva team/grupper som har goda synergier (t.ex. medivac som kan hela trupper + trupper som kan strida).

- $A = \{Marauder_1, Marauder_2, Medivac_1, Medivac_2, Marine_1, Marine_2, Marine_3, Marine_4, Marine_5, Tank_1\}$
- **Constraint:** Varje enhet måste vara medlem i exakt ett team.
- **Värdefunktion**: Många olika sätt att definiera...

Tips och Sammanfattning

Tips

Tänk på:

- … att det finns en **litteraturlista** med relevanta referenser på kursens **wiki-sida.**
- … att ni inte måste välja något av de problem som vi exemplifierade, eller lösa dem med de tekniker vi tog upp.
- … vad er agent får för problem om ni saknar en bra lösning.

Tips

Tänk på:

- … att en lösningsteknik kräver *indata*
  - Sökmetoder kräver en graf. Vad innehåller den? Var kommer den ifrån?
  - Potential fields kräver attraktorer / repellerare. Var kommer de ifrån?
- ...och ger *utdata*, eller *styr* något
  - Potential fields ger ett vektorfält. Hur använder man detta?

Sammanfattning

Det finns många intressanta AI-problem att lösa och många tekniker och algoritmer att experimentera med!

Fundera på vad ni vill göra! Komplettera varandra!