Cooperative Pathfinding

Milan Rybář

<kontakt@milanrybar.cz>

Multi-agent Pathfinding

- Agenti hledají nekolidující cesty do různých cílů
- Chceme minimalizovat celkovou cenu
- Cooperative Pathfinding
  - Agent má úplnou informaci o ostatních agentech a jejich naplánovaných cestách
- Non-Cooperative Pathfinding
  - Agent nemá informaci o naplánovaných cestách ostatních agentů a musí předpovídat jejich pohyb
- Antagonistic Pathfinding
  - Agent se snaží dostat do svého cíle zatímco zabraňuje ostatním agentům dosažení jejich cílů

Multi-agent Pathfinding

- Centralised planning
  - Bere v úvahu všechny agenty najednou
  - Typicky úplné a optimální algoritmy
  - PSPACE-težké
- Decoupled (distributed) planning
  - Rozložení plánování do nezávislých nebo částečně závislých problémů pro jednoho agenta
  - Rychlé
  - Typicky neúplné a suboptimální algoritmy

Motivace a praktické využití

Motivace a praktické využití

- Herní průmysl
- Řízení dopravy (provozu)
- Plánování jízdních řádů (vlaků, ...)
- Robotika
  - Pohybující se roboti
  - Robotická ruka ve výrobní lince

Připomenutí A* algoritmu

- Informované prohledávání stavového prostoru
- Evaluační funkce f(n) = g(n) + h(n)
  - g(n) je cena cesty z kořene do uzlu n
  - h(n) heuristická funkce odhadující délku nejkratší cesty do cíle
- Přípustná heuristika h(n)
  - h(n) ≤ „nejmenší cena cesty z n do cílového uzlu“
- Monotónní (konzistentní) heuristika h(n)
  - h(n) ≤ c(n,a,n‘) + h(n‘), kde n‘ je následník n přes akci a, c(n,a,n‘) je cena přechodu
  - Jedná se o formu trojúhelníkové nerovnosti
- Je-li h(n) monotónní heuristika, potom je algoritmus A* optimální
  - Pro monotónní heuristiku jsou hodnoty f(n) podél libovolné cesty neklesající

Algoritmy pro real-time prostředí

- Více agentů v dynamickém a často přeplněném prostředí
- A* naplánuje cestu pro jednoho agenta
- A* je možno upravit, tak že přeplánuje, jakmile je potřeba

Local Repair A* (LRA*)

- Agent hledá cestu pomocí A* algoritmu a ignoruje ostatní agenty
- Půjde po cestě dokud nehrozí kolize
- Až dojde ke kolizi, agent přeplánuje zbytek cesty
- Algoritmus odpovídá prohledávání hrubou silou

Local Repair A* (LRA*)

- Nevýhody:
  - Složitější prostředí s úzkými průchody a mnoha agenty => úzké hrdlo, deadlock a cyklické opakování
    - Agenti neustále přeplánovávají
    - Vizuálně neinteligentní chování
    - Každé přeplánování je provedeno nezávisle, což vede k cyklům
  - V mnoha případech je nucen přímo game designer navrhnout prostředí, tak aby tato slabina moc neovlivnila vizuální chování agentů

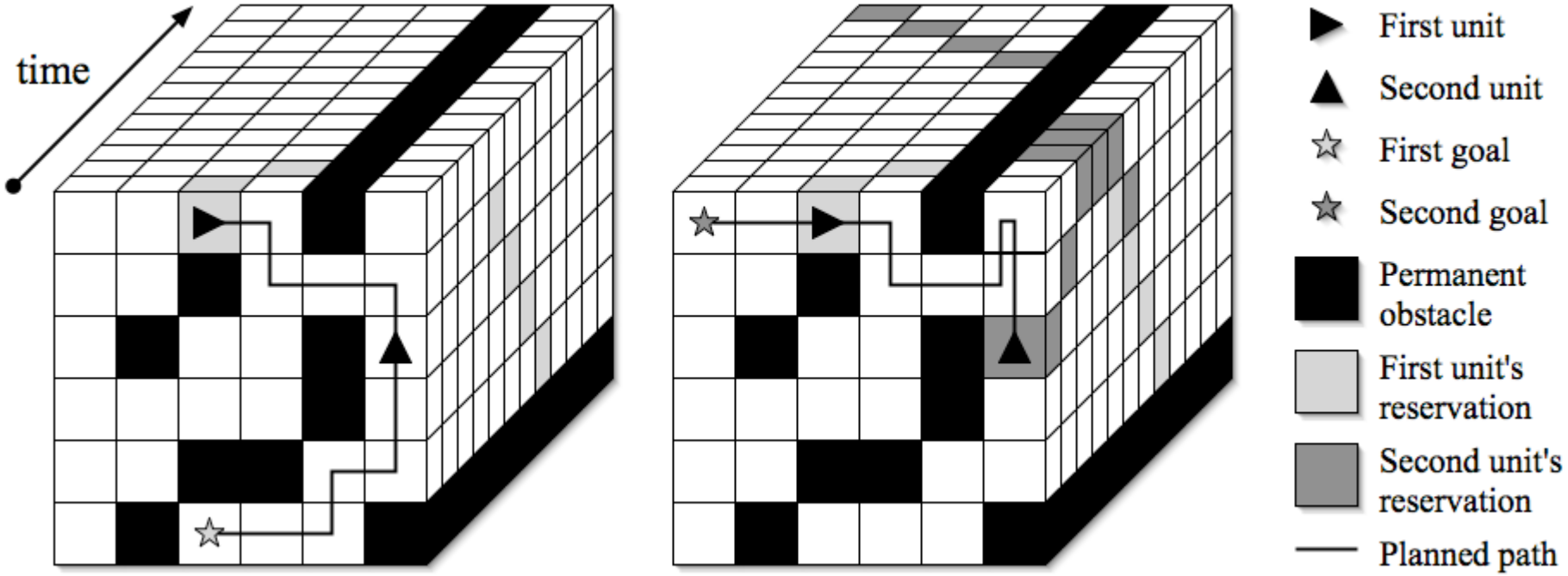
Cooperative A* (CA*)

- K prohledávacímu prostoru přidáme další rozměr - čas
- Agent má navíc akci *čekej*
- Agent nalezne cestu pomocí A* algoritmu v 3-rozměrném prostoru. Po nalezení jsou stavy na této cestě zaznamenány do rezervační tabulky (reservation table)
- Údaje v rezervační tabulce jsou považovány za neprůchodné a další agenti se jim vyhnou během dalšího prohledávání

CA* - Příklad

- Čtvercová mřížka, 4 sousední políčka
- Akce: *nahoru*, *dolu*, *doleva*, *doprava* a *čekej*
- Akce *nahoru* odpovídá přesenu agenta ze stavu (x, y, t) do stavu (x, y + 1, t + 1). Obdobně akce *čekej* (x, y, t) do (x, y, t + 1)
- Akce je povolená pokud v cílovém stavu není překážka ani jiný agent

CA* - Reservation Table



Two units pathfinding cooperatively.
(A) The first unit searches for a path and marks it into the reservation table.
(B) The second unit searches for a path, taking account of existing reservations, and also marks it into the reservation table.

CA* - Reservation Table

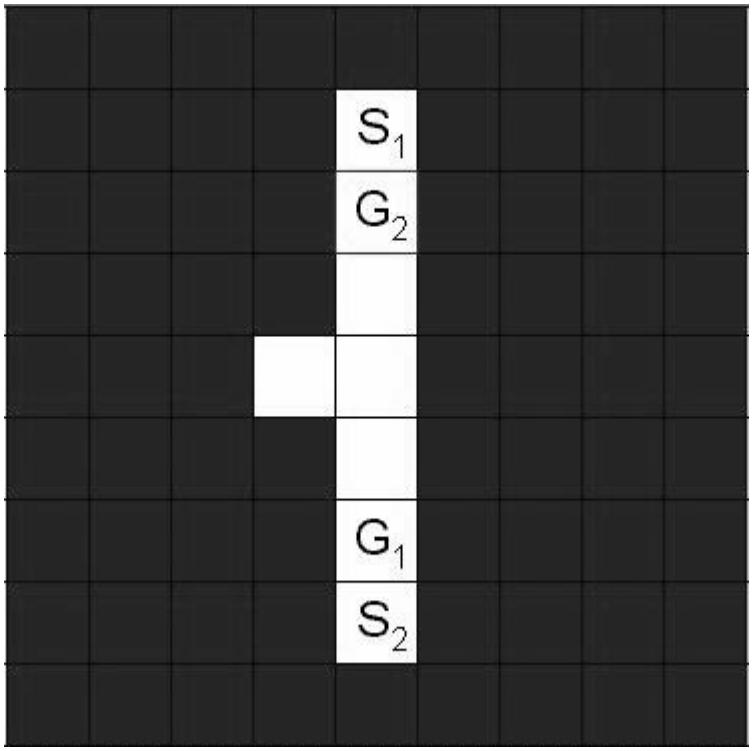
- Reprezentuje sdílené informace o naplánovaných cestách ostatních agentů
- Volba datové struktury není závislá na prohlédávaném prostoru
- Agenti se obecně mohou lišit v rychlosti a velikosti
- Pouze malá část tabulky bude využita
  - Efektivní implementace pomocí hashovací tabulky, kde (x, y, t) je klíč

CA* - Vlastnosti

- Představené použití rezervační tabulky nezabraňuje procházení dvou agentů skrz sebe
  - První agent si rezervuje (x, y, t) a (x + 1, y, t + 1) a druhý agent (x + 1, y, t) a (x, y, t + 1)
  - Řešení:
    - Provádět 2 rezervace pro oba stavy použité v akci (v čase *t* a *t+1*)
    - Explicitní identifikace a označení jako neplatné akce

CA* - Vlastnosti

- Záleží na pořadí agentů
- Kvalitu CA* ovlivňuje výběr vhodné heuristiky



CA* - Heuristika

- Manhattan distance
  - Špatná účinnost
- True Distance
  - Nejkratší vzdálenost do cíle v původním prostředí (ingorujíc ostatní agenty)
  - Tzn. velikost cesty dané pomocí A* algoritmu v původním prostředí
  - Monotonní heuristika

Hierarchical Cooperative A* (HCA*)

- Hierarchie označuje sérii abstrakcí stavového prostoru, každá obecnější než předchozí
- HCA* používá 1 abstrakci, která ignoruje čas a rezervační tabulku
  - Tzn. prostředí bez ostatních agentů
- V podstatě CA* s lepší heuristikou
- Jak nejlépe opětovně využít prohledávání v abstraktním prostoru?

Reverse Resumable A* (RRA*)

- Modifikace A* hledající v opačném směru
- Začíná se v cíli *G* agenta a hledá se agentova iniciální pozice *O*
- Nekončí se v *O*, ale pokračuje se dokud není daný uzel *N* expandován
  - Získáme optimální vzdálenost z *N* do *G*
- Použití Manhattan distance heuristiky

HCA* - Použití RRA*

- Požadována abstraktní vzdálenost z *N* do *G*
  - RRA* už *N* expandoval, tzn. vzdálenost už známe
  - Jinak pokračujeme hledání v RRA*, dokud není *N* expandován

HCA* - Nevýhody

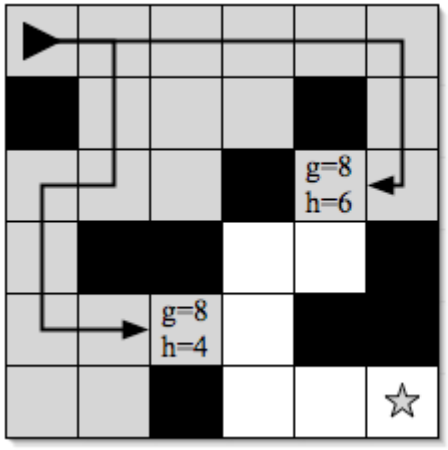
- Agent po dosažení cíle skončí
  - Může blokovat cestu ostatním agentům
    - Měl by opustit cíl, aby mohli ostatní projít
  - Agenti by měli stále spolupracovat, dokud nedosáhnout svých cílů
- Záleží na pořadí agentů
  - Někdy lze agenty globálně seřadit
  - Robustnější je dynamická varianta
    - Každý agent bude mít na krátkou dobu nejvyšší prioritu
- Počítání kompletní cesty ve velkém 3-rozměrném stavovém prostoru

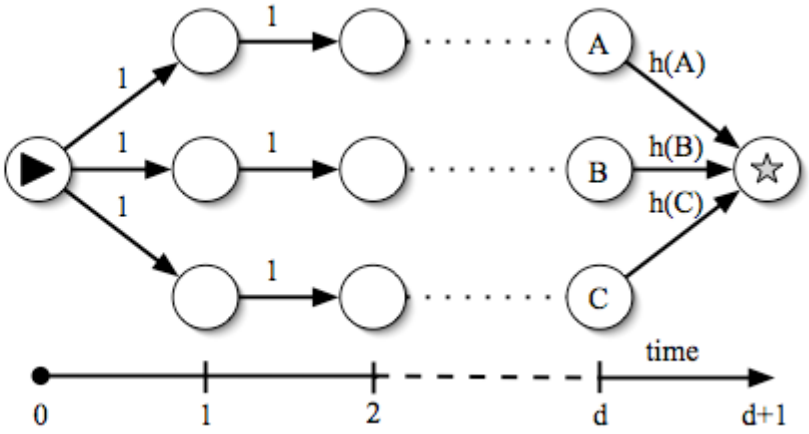
Windowed Hierarchical Cooperative A* (WHCA*)

- Přidává okénko pro prohledávání
  - Prohledávání je omezeno na fixní hloubku
- Agent hledá částecnou cestu do cíle (*d* kroků)
  - Jde po nalezené cestě a v pravidelných intervalech (např. v polovině cesty) posune okénko dopředu a hledá novou částečnou cestu
- Hloubka je omezena pouze v kooperativním prohledávání, ne v abstraktním prostoru
  - Agent míří ve správném směru
- Příklad: Pro spolupráci agentů v 8 krocích, *d=16*

WHCA* - Terminal edge

- Po uběhnutí *d* kroků jsou agenti ignorováni a prohledávací prostor je identický abstraktnímu prostoru
  - Tzn. abstraktní vzdálenost má stejnou informaci jako dokončení prohledávání





When the pathfinding depth is limited, the true distance heuristic can discriminate between partial paths.
(A) All partial paths complete after d = 8 steps.
(B) A terminal node can represent the remainder of the search in a single step.

WHCA* - Vlastnosti

- Prohledávání pokračuje i po dosažení cíle
  - Agent v cíli nezmizí. Ostatní agenti mohou chtít projít
  - Okénko má fixní velikost. Vždy se hledá *d* kroků
  - Úprava cen akcí:
    - Normální akce (jako *nahoru*) mají stále hodnotu 1
    - Akce *čekej* v cíli má hodnotu 0
- Čas pro výpočet je rozšířen mezi všechny agenty
  - *n* agentů, *d* velikost okénka, přepočet cesty v polovině
  - Pouze *2n/w* agentů plánuje ve stejný čas
  - Vytváří více robustní prohledávání
    - Dva po sobě následující agenti:
      - 1. agent rezervuje cestu od času *t* do *t+d*
      - 2. agent v dalším kroku rezervuje cestu od času *t+1* do *t+d+1*
    - Každý agent může mít alespoň jeden krok větší prioritu

Priorita agentů

- Dosud měli všichni agenti stejnou prioritu
  - To ale není případ většiny počítačových her
    - Například hlavní hrdina by měl mít největší prioritu
- Agenty seřadit podle priority
  - Bohužel díky částečným plánům mají agenti přibližně stejnou šanci na rezervaci plánu
- Uložit prioritu do rezervační tabulky
  - Agent s prioritou p uloží cestu s prioritou p
  - Když agent s prioritou $q > p$ plánuje, tak ignoruje menší priority
  - Pokud cesta přepisuje existující rezervace, tak je agent označen. Označení agenti přeplánují s novou prioritou q
- Zvýšení priority může pomoci, pokud agent nenalezne žádnou cestu

Shrnutí *CA* algoritmů

- Neúplné, ale rychlé algoritmy
- Snaží se najít nejkratší cesty, přičemž malinko delší cesty mohou mít menší šanci na neúspěch
  - Pohyb agentů může být chaotický
- Co když chceme spíše „vizuálně inteligentní“ chování skupiny agentů?

Roje (Flocking)

- Inspirace v rojích a hejnech, např. ptáků
- Agent používá malou množinu pravidel pro pohyb
- 3 pravidla (Reynolds 1987):
  - Vyhnutí se kolizi blízkých agentů (separation)
  - Přizpůsobení pohybu s blízkými agenty (alignment)
  - Pohyb do středu blízkých agentů (cohesion)

Další inspirace

- Stigmergy – nepřímá komunikace agentů pomocí změny prostředí
  - Inspirace mravenci
    - Virtuální feromon
    - Např. Ant System pro řešení TSP (obchodní cestující)
- Steering
  - Agent je parametrizován vektorem pohybu
  - Například: Path following, Flow field following

Direction Map

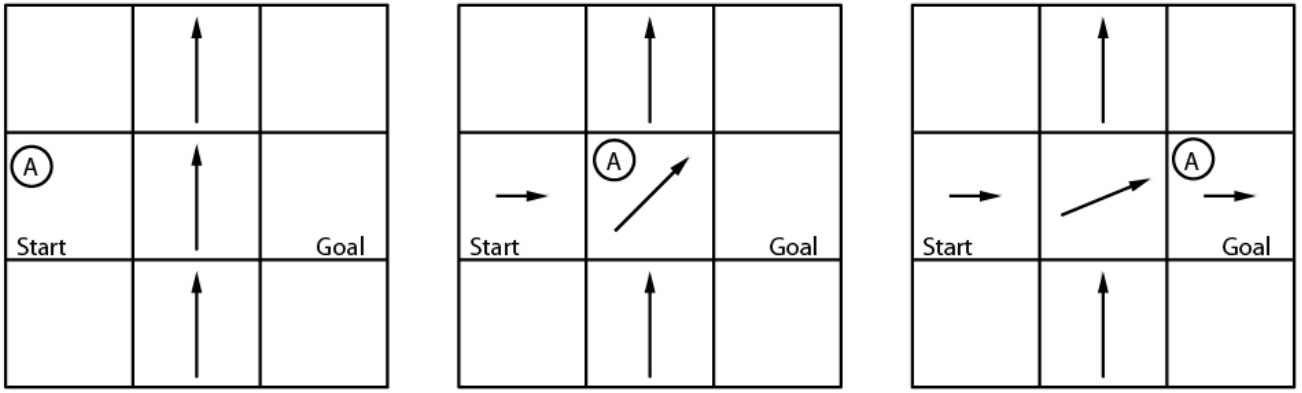
- Inspirace Flow field following
  - Často staticky dané
    - Např. ve hře Assassin's Creed (Bernard and Therien 2008)
- Automatické vytvoření a úprava flow field mapy
- Direction map (DM) uchovává směry (Direction Vector (DV)), kterými agenti prošli na částech mapy
  - DV má velikost mezi 0-1 a reprezentuje očekávaný směr, kam agenti půjdou
  - Movement Vector (MV) – pohyb agenta
- Agent tuto informaci používá při plánování cesty
  - Dostává penalizace pokud chce jít proti směru

Direction Map - Plánování

- Lze použít do libovolného prohledávání s heuristikou (např. A*)
  - Cena hrany *e* z buňky *a* do *b* je cena hrany *e* (w_{ab}) + funkce z DV pro buňky, které jsme navštívili a opustili
  - Intuitivně: penalizace pokud opustíme *a* v jiném směru, než je její DV
  - w_{max} je penalizace pro pohyb v opačném směru
- $w_{ab} + 0.25 w_{max}(2 - DV_a MV_{ab} - DV_b MV_{ab})$
- Normalizovaný skalární součin mezi 0 a 1
  - Hodnota 0: Agent následuje směr
  - Hodnota 1: Agent jde v opačném směru
- Cílem již není najít nejkratší cestu, ale cestu s nejmenší cenou, která následuje směr agentů v prostředí
- „Kooperace“ je docíleno, tím že se agenti pravděpodobně nesrazí, když jdou ve stejném směru

Direction Map – Učení

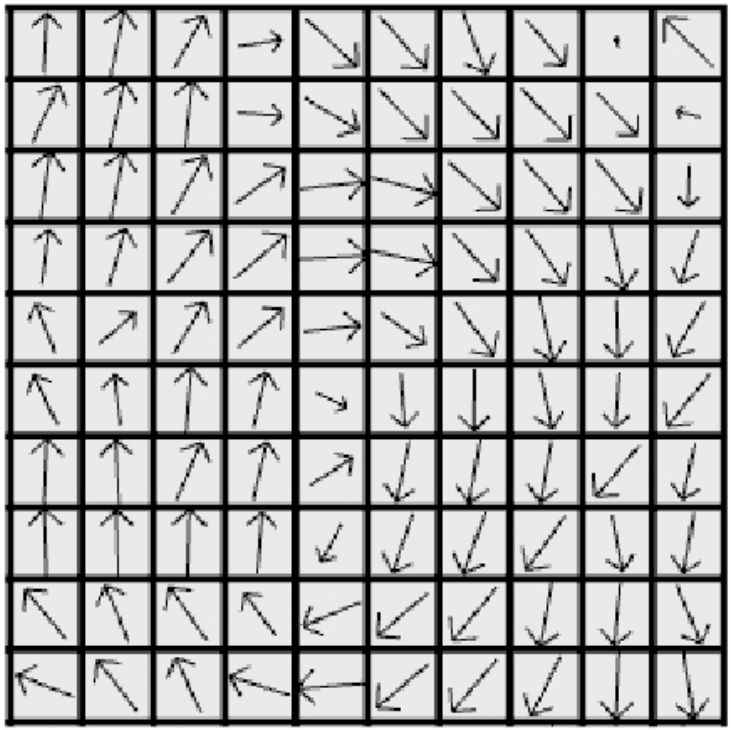
- DM se učí z pohybu agentů po mapě
- Vážený průměr MV agentů skrz buňku
  - $DV \leftarrow (1-\alpha)DV + \alpha MV$
  - α je učící parametr, DV se normalizuje
  - Aktualizace MV probíhá, když agent vstupuje a opouští buňku



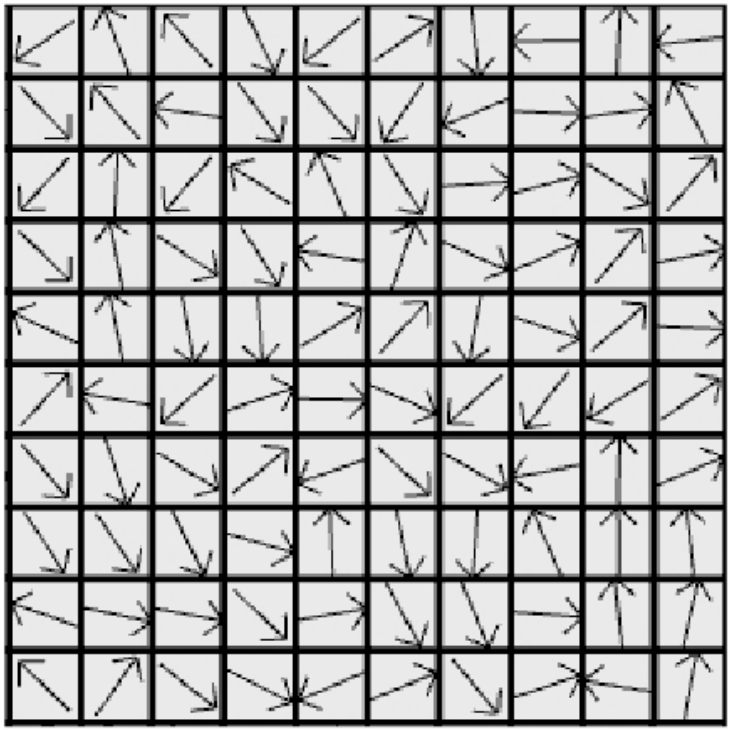
(a) Step 1 (b) Step 2 (c) Step 3

Learning DV's.

Direction Map - Příklad



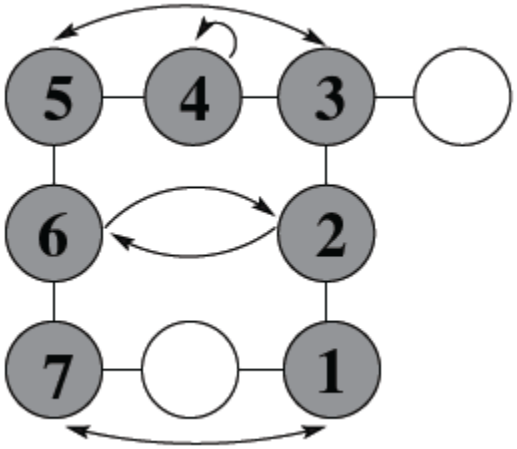
(a) Using DM

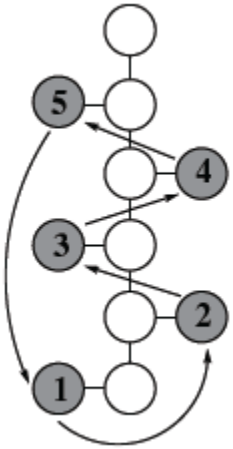


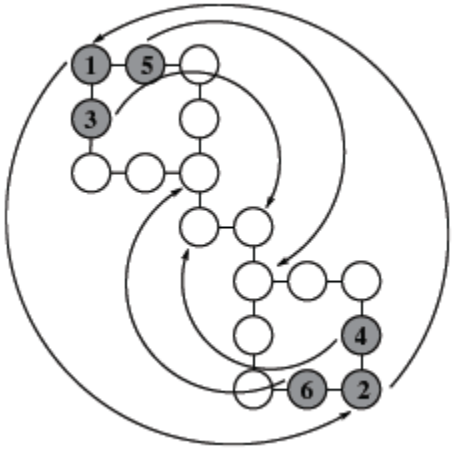
(b) Not using DM

The DM's generated when we are and when we are not using the DM for planning

Úplné algoritmy







- SAT
  - Plánování je zakódováno jako SAT problém
- Push and Swap
  - Úplný algoritmus pro maximálně *n-2* agentů v grafu s *n* vrcholy
  - 2 operace:
    - Push – agent jde směrem k cíli, dokud může
    - Swap – 2 agenti si vymění pozice beze změny konfigurace ostatních agentů
- Operator Decomposition (OD)
  - A* se speciálním operátorem a upraveným prohledávacím prostorem
  - Independent Subproblems (ID)
    - Shlukování agentů do skupin
  - Optimal Anytime (OA)
    - Vhodný na reálné použití. Nemusíme nechat doběhnout až do konce, pokud nechceme optimální řešení
    - Postupně zvyšuje maximální velikost skupiny. Tzn. pouze zlepšuje kvalitu řešení

Zdroje

- LRA*, CA*, HCA*, WHCA*
  - D. Silver. Cooperative Pathfinding. In AI Game Programming Wisdom 3, pages 99–111. Charles River Media, 2006.
  - D. Silver. Cooperative Pathfinding. In 1st Conference on Artificial Intelligence and Interactive Digital Entertainment, 2005.
- Direction Maps, Flocking, Ant System
  - A new approach to cooperative pathfinding, Renee Jansen and Nathan Sturtevant, Proc. of 7th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2008)
  - Direction Maps for Cooperative Pathfinding, Renee Jansen and Nathan Sturtevant
  - G. Rozenberg et al. (eds.), Handbook of Natural Computing, chapter Swarm Intelligence. Springer, 2012.
- SAT
  - A SAT-Based Approach to Cooperative Path-Finding Using All-Different Constraints. Pavel Surynek. Proceedings of the 5th Annual Symposium on Combinatorial Search (SoCS 2012), Niagara Falls, Ontario, Canada, AAAI Press, 2012.
  - On Improving Plan Quality via Local Enhancements. Tomáš Balyo, Roman Barták, Pavel Surynek. Proceedings of the 5th Annual Symposium on Combinatorial Search (SoCS 2012), Niagara Falls, Ontario, Canada, AAAI Press, 2012.
- Push and Swap
  - An Efficient and Complete Approach for Cooperative Path-Finding. Ryan Luna, Kostas E. Bekris.
  - Push and Swap: Fast Cooperative Path-Finding with Completeness Guarantees. Ryan Luna, Kostas E. Bekris.
- OD, ID, OA
  - Finding Optimal Solutions to Cooperative Pathfinding Problems. Trevor Standley. Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence (AAAI-10)
  - Complete Algorithms for Cooperative Pathfinding Problems. Trevor Standley, Richard Korf. Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence