Path-finding

Petr Michalík
(m.peta@centrum.cz)
MFF UK 18.12.2009

Osnova

- Seznámení s problémem
  - o co jde?
- Metody řešení
  - abstrakce světa
  - základní algoritmy
  - vylepšení
- Praxe
  - srovnání metod, výsledky

O co jde?

- nalezení co nejkratší cesty z místa A na místo B (případně přes C)

- typickou aplikací jsou počítačové hry

- v daném umělém světě (na mapě) je potřeba najít takovou cestu, aby se jednotka(y) dostala(y) na místo určení co nejrychleji a vyhnula(y) se všem překážkám

- typicky se uvažuje předem známá 2D mapa (i v případě 3D her)

- problém se tak redukuje hledání (nejkratší) cesty v grafu

- typický požadavek na rychlost (real-time)

X

velké mapy s množstvím obejktů

Aplikace :

- počítačové hry
- robotika
- navigace, doprava
- animace ve filmu

Řešené problémy

2 hlavní části:

1. Abstrakce světa (převod mapy na graf)

2. Hledání cesty v grafu

Řešené problémy

2 hlavní části:

1. Abstrakce světa (převod mapy na graf)
- automaticky
- manuálně

2. Hledání cesty v grafu
- hledání nejlepšího řešení
- vylepšení
- aproximační algoritmy

Abstrakce mapy

Automaticky

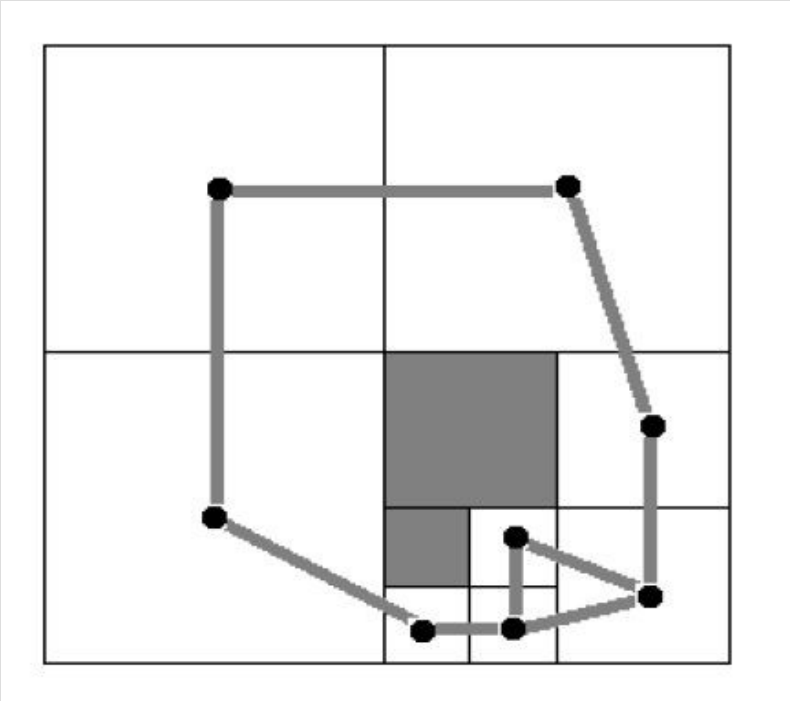
Standardní postup:

- mapa se proloží dostatečně hustou sítí bodů
- sousední body se spojí hranami
- tam, kde je překážka, tam nebude ani hrana (bod)
- může to přijít dost draho: na mapě 256x256 dostaneme 65536 vrcholů grafu

Automaticky

Vyplatí se rozdělit mapu na co největší konvexní polygony, unvitř kterých platí, že cena pohybu uvnitř polygonu je uniformní.

Potom se hledá v duálním grafu:

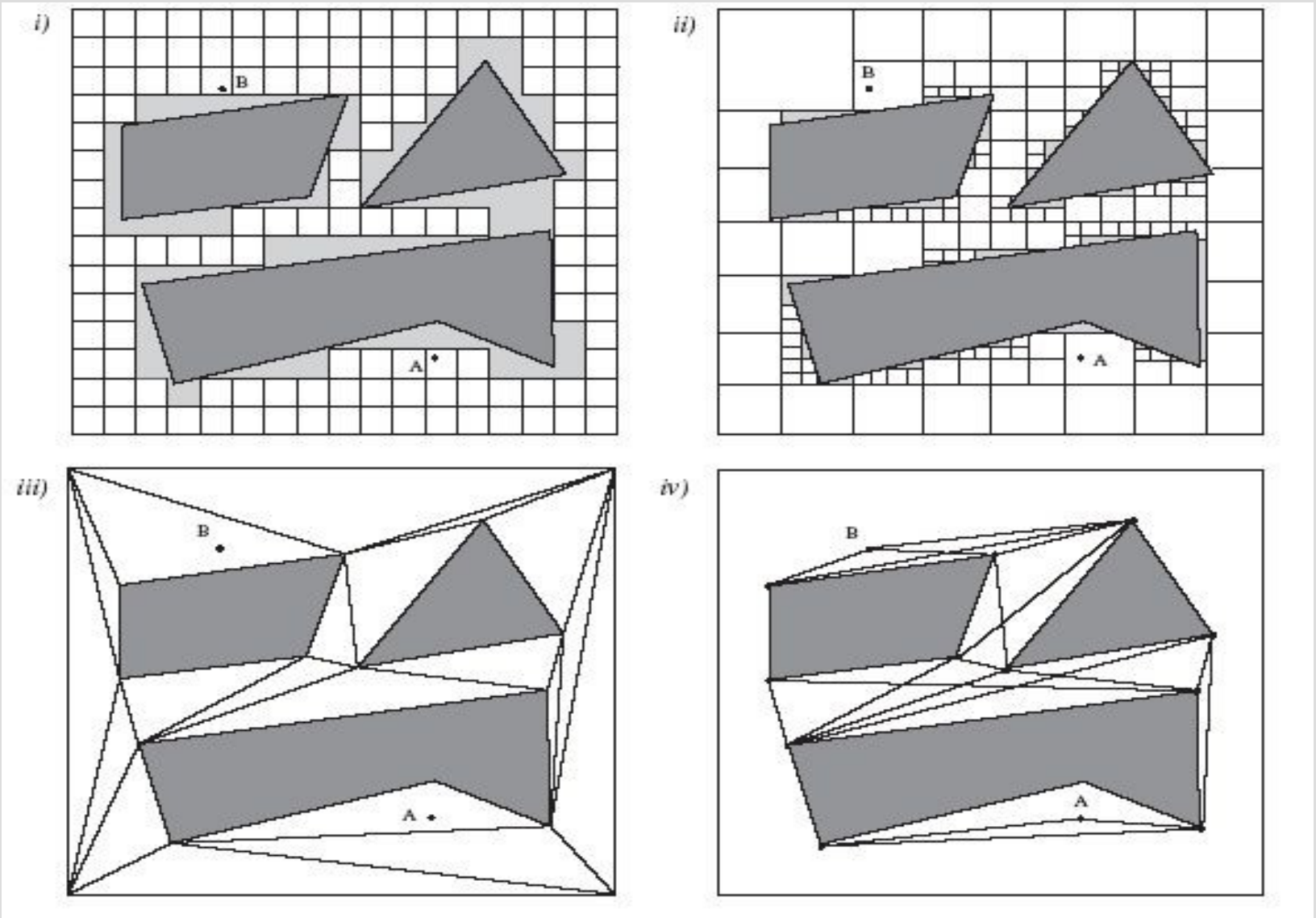


Abstrakce mapy

Automaticky

Metody rozdělení mapy

- Quadtrees
- Optimálni polygony (triangulace)
- Využití „points of visibility“


i)
B
A
ii)
B
A
iii)
B
A
iv)
B
A

Manuálně

Práce pro programátora / designéra: „ručně“ vytvořit aspoň „kostru“ grafu.
Označí se důležitá místa, určí se, jak hustá má být síť bodů v určitých oblastech apod.

Využívá se zejména u 3D akčních her (místa, kde se objevují zbraně, dobíjí zdraví...)

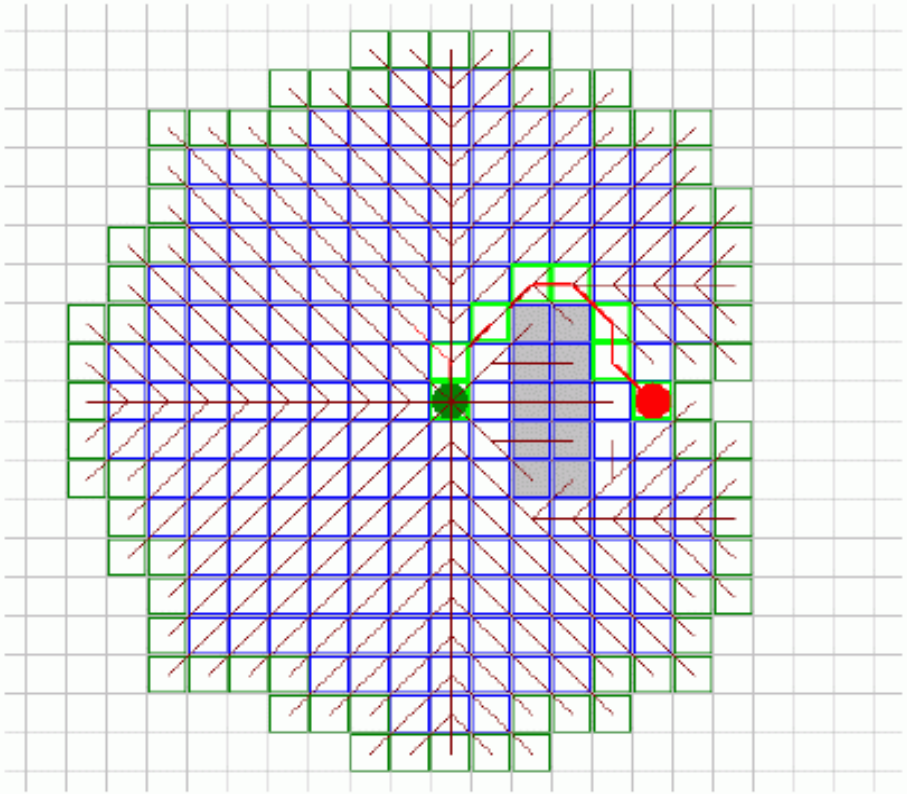
Hledání cesty v grafu

Hledání cesty v (obecném) grafu je známý problém.

Je ale nutné si uvědomit:

- řešení je třeba najít rychle - ne „asymptoticky rychle“, ale opravdu real-time (Dijkstra dělá moc práce)
- hledáme řešení pro objekt, ne pro hmotný bod
  → collision checking

Dijkstra



[Dijkstra, E.W. (1959)]

A*

Algoritmus A*

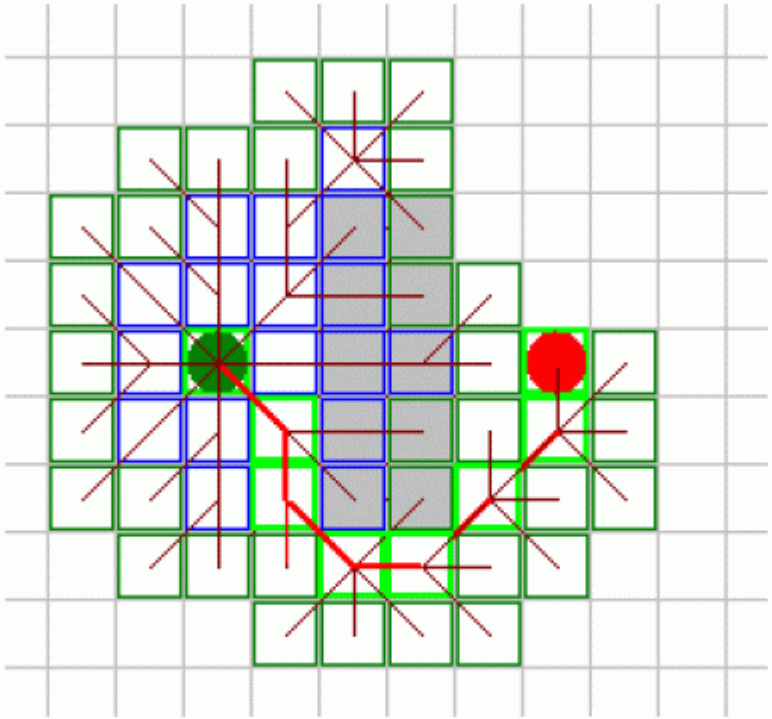
- zřejmě nejznámější algoritmus prohledávání s cenou
- hledá za pomoci heuristiky a prohledá nejslibnější vrchol
- přesněji takový ještě nenavštívený vrchol, pro který je minimální:

$$f(x) = g(x) + h(x)$$

kde

$g(x)$ je vzdálenost od startu do bodu x
$h(x)$ je odhad vzdálenosti z x do cíle

Algoritmus A*



[Hart, P., Nilsson, N., & Raphael, B. (1968)]

Heuristiky

Pro heuristiku *h* platí: $h(g) = 0$ pro cílový bod g
$h(x) >= 0$ pro všechny body x

Heuristika je **přípustná**, pokud je dolním odhadem skutečné ceny nejkratší cesty z x do cíle.

Heuristika je **monotónní**, pokud pro každé dva sousední body x a x' platí:

$$h(x) \leq c(x,x') + h(x')$$

kde

$c(x,x')$ je cena za přechod mezi x a x'

(taková trojúhelníková nerovnost)

Pozorování 1: Monotónní heuristika je přípustná

Pozorování 2: Pro monotónní heuristiku jsou hodnoty f(n) podél libovolné cesty neklesající

Věta:
Je-le h monotónní heuristika, potom algoritmus A* najde optimální řešení

Heuristiky:

- Eukleidovská vzdálenost
- Maximová metrika ($h(x) = \max(dx, dy)$)
- L_1 (Manhattonská) metrika ($h(x) = dx + dy$)

- Extrémní případ: $h(x) = 0$ pro všechna x

→ *to je Dijsktra !!!*

Vlastnosti A* :

+ pokud existuje řešení, tak ho najde, navíc optimální
+ snadná implementace
+ nezávislost na grafu
+ velmi snadno se zapracují různé druhy terénu (~ ceny hran)

- počet otevřených vrcholů může být velmi vysoký
- zpoždění (nejprve je nutné spočítat celou cestu, až potom lze udělat první krok)
- nepočítá s dynamikou prostředí (D*)

Hierarchické plánování

Hierarchický přístup nabízí možnost, jak získat vyšší výkon za cenu ztráty optimality

Motivace:

Hledání nejkratší cesty z Malostranského náměstí v Praze na Zeppelinstraße v Norimberku

- člověk asi nebude hledat řešení na mapě s měřítkem 1:50 000

Obecné vlastnosti:

\+ výrazná úspora zdrojů
\+ obvykle rychlejší odezva („on-line“)

\- nemusí najít optimální cestu
\- náročnější implementace a nastavení/ladění parametrů
\- může (ale nemusí!) být závislé na mapě, může vyžadovat práci navíc

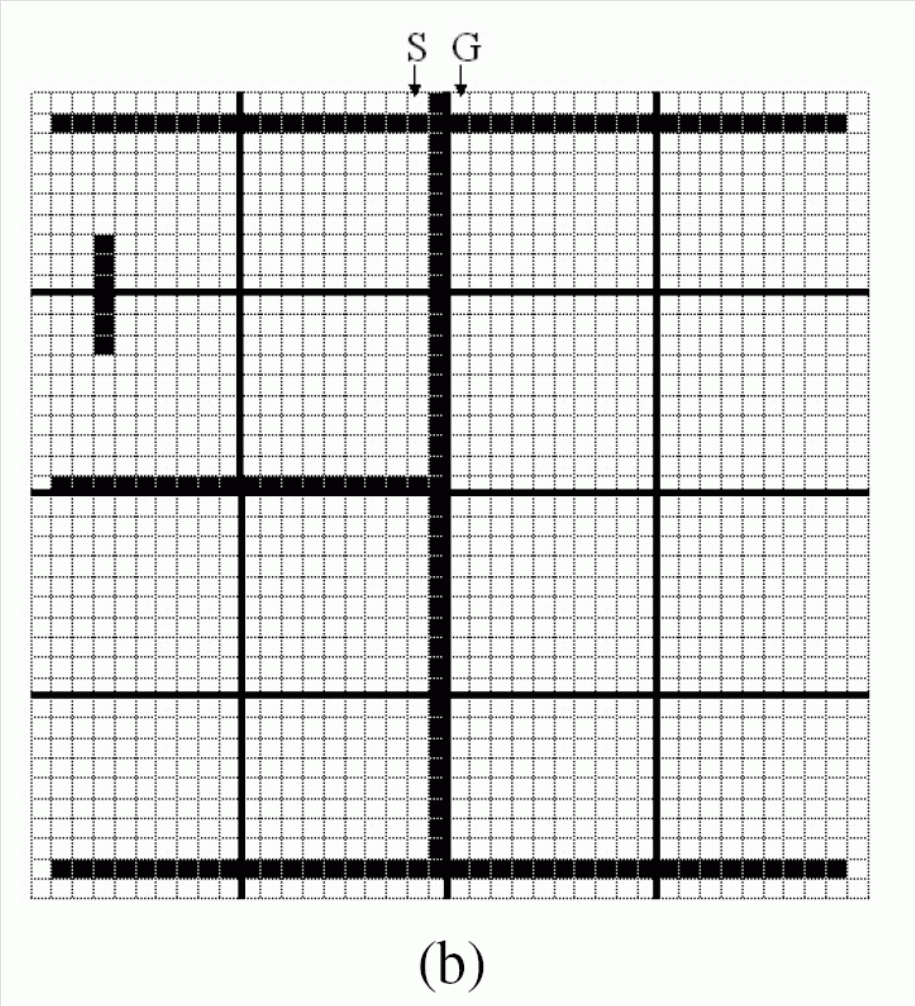
HPA*

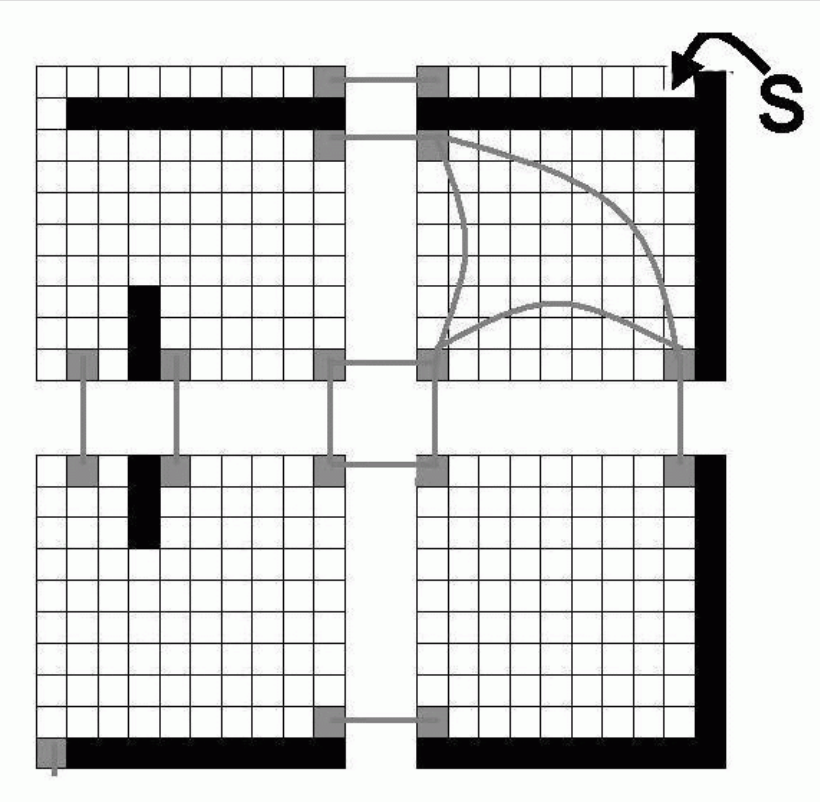
Relativně jednoduchá, snadno implementovatelná a obecná technika, umožňující zohlednit různé druhy terénů a do jisté míry schopná vypořádat se s dynamikou prostředí

Cesta se hledá na několika úrovních (typicky 2, možno více)

Graf se rozdělí na *clastery* (10 x 10 vrcholů), určí se *průchody* mezi nimi

Cesta se poté hledá na úrovni clasterů


S
G
(b)


S

Alg. podrobněji:

1) Vytvořit abstraktní graf: vrcholy odpovídají *průchodům*, přidají se „vnitřní hrany“ (v rámci *clasteru*, jejich délka se předpočítá) a „vnější hrany“ (mezi clastery, jednotková délka)

2) Přidat startovního a cíloveho bodu do grafu

3) Pomocí stadnartního A* najít optimální cestu v abstaktním grafu

4) „Zjemnit“ vnitřní hrany na cesty v rámci *clasteru* (netřeba dělat předem, ale až v případě potřeby)

Alg. podrobněji:

Evidentně nemusí najít optimální řešení.

Možné vylepšení: „vyhlazování cesty“

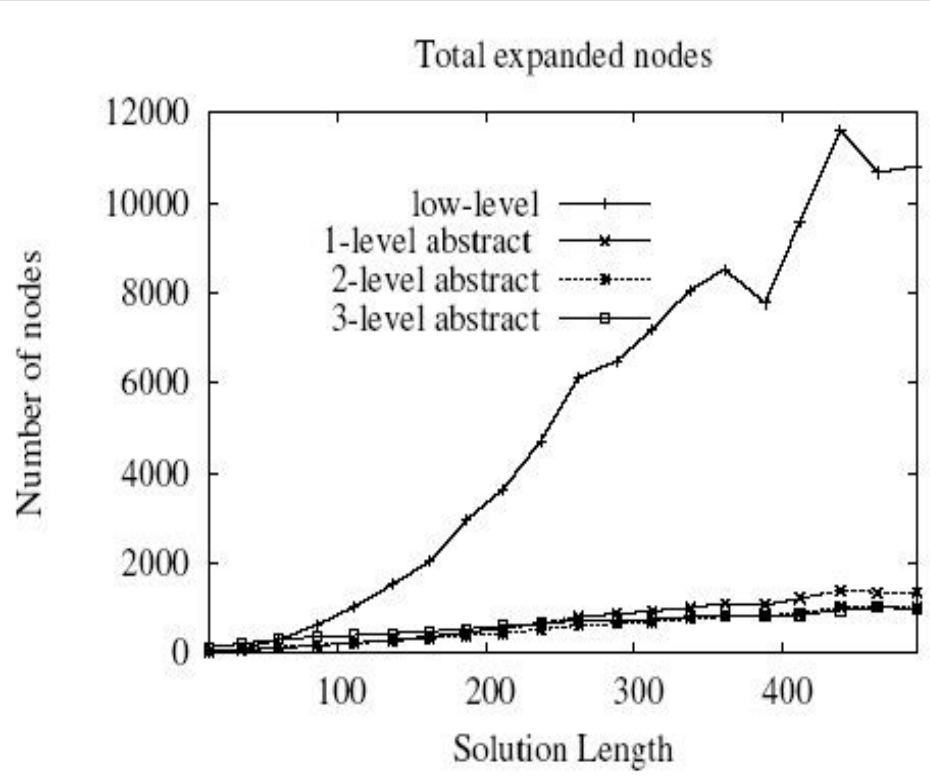
- pokud *x,y,z* jsou 3 vrcholy abstraktní cesty a existuje přímá cesta mezi *x* a *z*, pak je možné *y* vynechat a jít přímo
- aplikuje se odzadu, dokud to jde

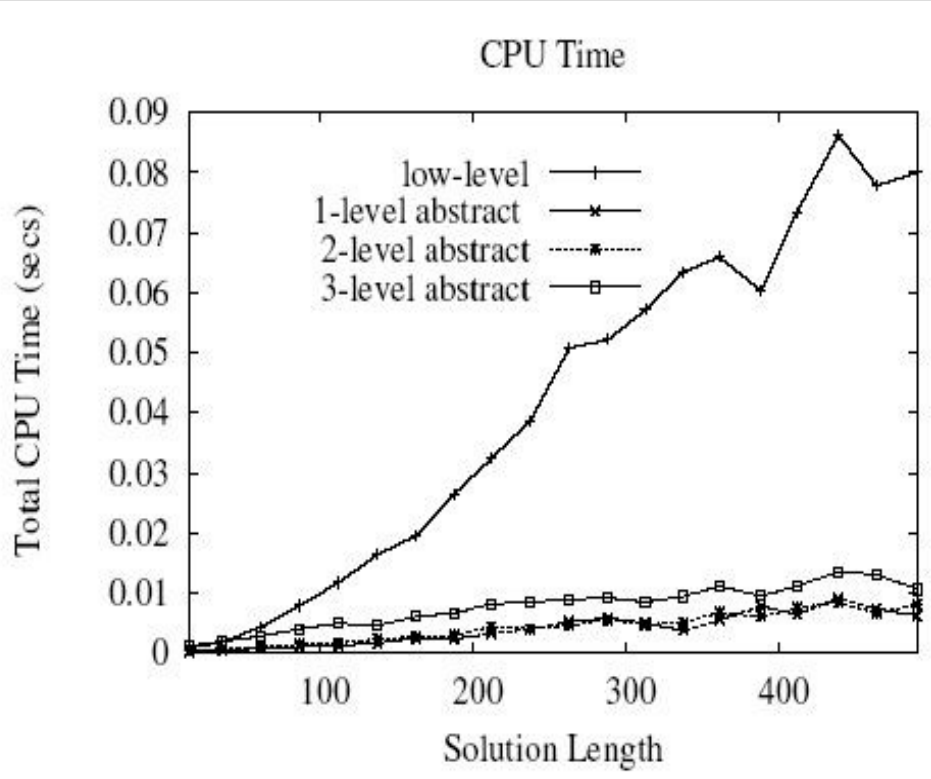
Vlastnosti HPA* :

+ nezávislé na mapě
+ existuje rozšíření na vícevrstvé hierarchie pro velké mapy
+ řádově 10-krát rychlejší než „obyčejný“ A*
+ není nutné získat celou cestu, stačí jen začátek (zbytek on-line)
+ na používaných mapách (Baldur's Gate) cca o 1% horší řešení než optimum (s vyhlazováním)

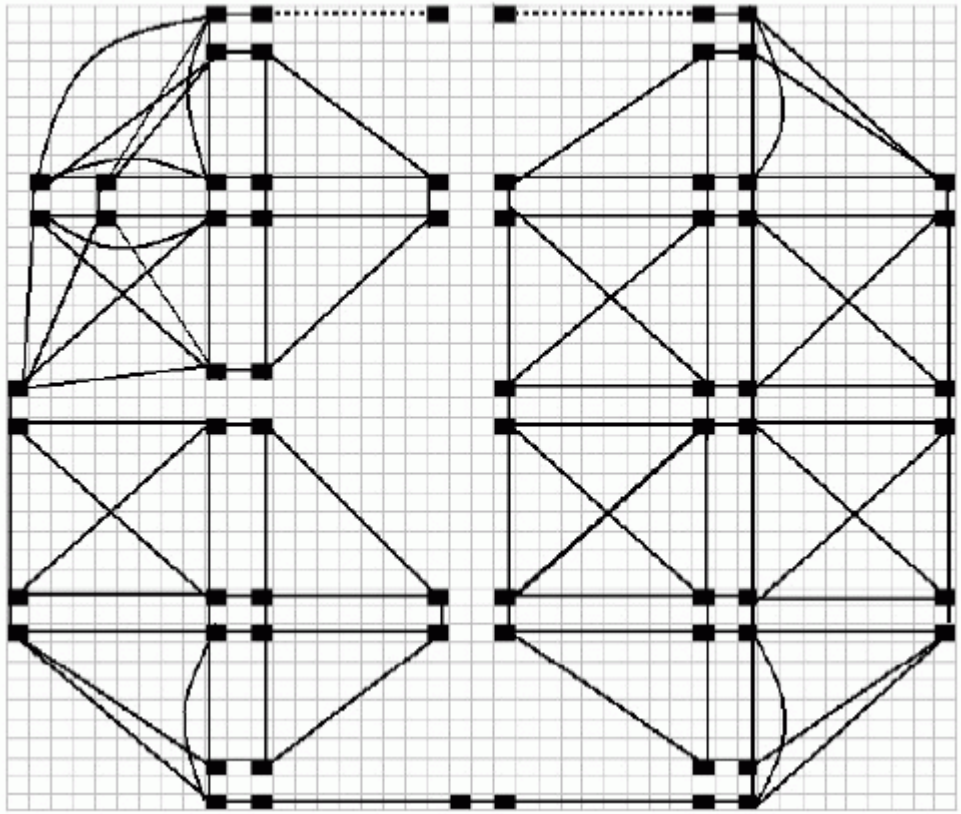
- nemusí najít optimální cestu
- vyhlazování na hustých mapách (hodně překážek) nefunguje
  → až o 10% horší řešení
- nutné nastavovat parametry (např. velikost *clasteru,* jak dělat *průchody*)
  → metoda pokus-omyl

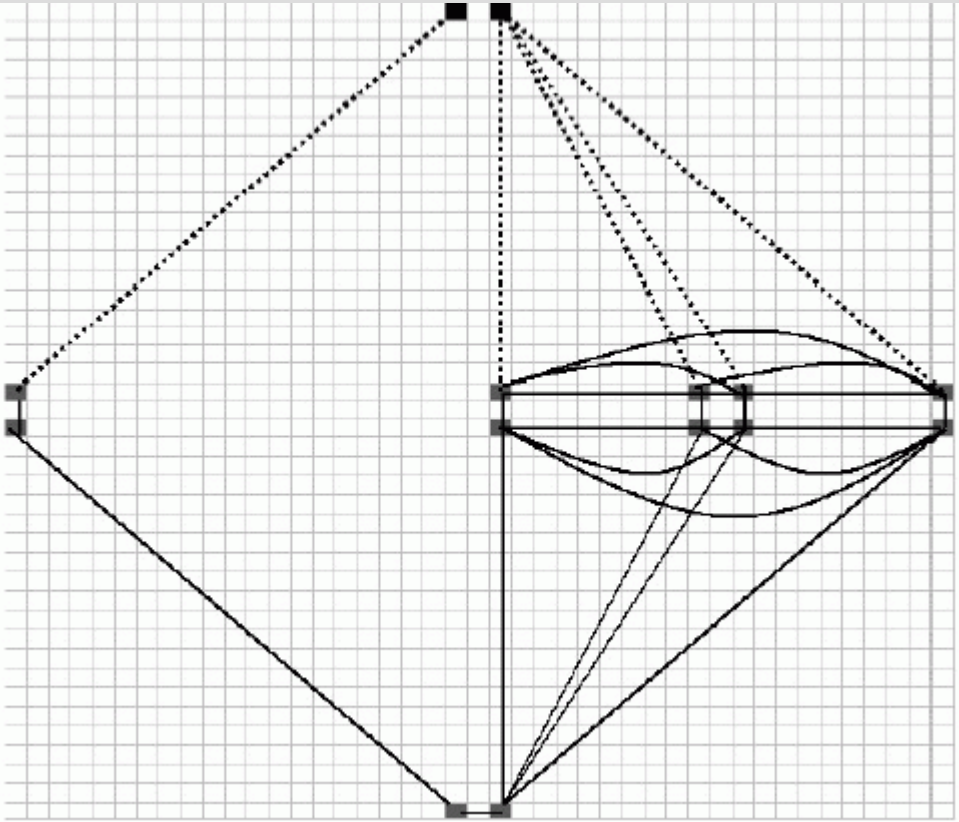
Srovnání HPA* a A*





Vícevrstvá hierarchie





Vícevrstvá hierarchie

Search Technique	*SG*	Main	**Abstract**	Refinement
L-0	0	1,462	**1,462**	0
L-1	16	67	**83**	145
L-2	41	7	**48**	161

SG … přidání startu a cíle do abs. Grafu
Main … počet navštívených vrchulů při hlavním prohledáváni
Abstract … součet
Refinement … námaha na rekonstrukci celé cesty z abstraktní

Další vylepšení

Pro účely využití ve virtuálních světech lze přidat mnohá obecná i specializovaná vylepšení (např. Caching).

Záleží, jaké vlastnosti bude svět mít:

- dynamika (statická mapa / více postav, jednotek / pohybující se překážky ...)
- geometrie (z hlediska tvarů překážek, typ mapy: vnitřní/vnější)
- nejistota (kompletní informace / „fog of war“ / neznámé prostředí)
- další faktory (omezené zdroje, pohyb formací, kinematika, interakce – vyhýbání se, nebo naopak spolupráce)

Caching

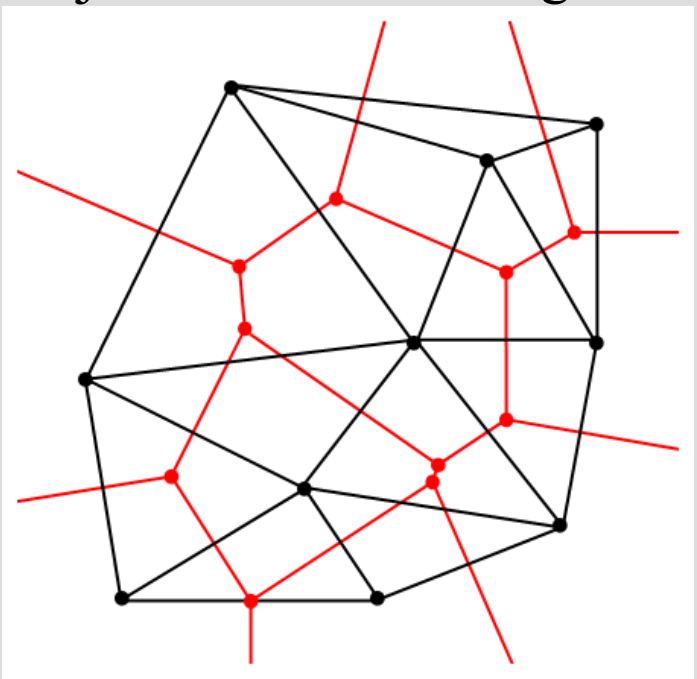
1) Path caching
   - buď jako předpočítání krátkých cest (HPA*) → je možné udělat předem
   - nebo si ukládat informace o nalezených cestách

2) Collision caching
   - každý pohyb v grafu vyžaduje collision check
   - ale ptát se na to enginu je obvykle drahé!
   - cachování této informace (pro statické světy a symetrické jednotky) přinese výrazné zrychlení

Voronoiovo zónování

Trochu jiná metoda, jak rozdělit na zóny (v HPA* *clastery*)

- na základě dat, získaných např. při testování hry/světa, se získají *body zájmu* (místa, kam hráči často chodí) a zkostruuje se Voronoiův diagram
- waypointy (*průchody* v HPA*)
  odpovídají středům hran
  korespondující triangulace
- opět se aplikuje myšlenka hierarchie
- jen trochu pozměněná



Ghost player

Speciální koncept prohledávání, kde není potřeba explicitně budovat celý graf

- *ghost player* = zvláštní entita udržovaná enginem, platí pro ni stejná pravidla, jako pro jednotky, ale je neviditelná a dočasná
- prohledávání probíhá pomocí pohybování tímto objektem po mapě v diskrétních krocích
- velikost kroku lze parametrizovat (v závislosti na hustotě, nebo tvaru překážek)
- menší paměťové nároky (pouze skutečně prohledaná oblast)
- snadno se zapracují změny mapy (nemusí se měnit žádný graf)

Výsledky

Přehledový článek:

Marc Lanctot, Nicolas Ng Man Sun, Clark Verbrugge:
Path-finding for large scale multiplayer computer games

-srovnání několika různým metod

-využili prostředí on-line projektu Mamoth

Porovnávané metody

Algoritmy:

- statické zónování – využití hierarchie s rozdělením světa na zóny daným samotnou mapou
- roadmaps – využití přímých cest mezi 2 body
- Voronoiovo zónování
- „klasický“ A*

Vylepšení:

-collision caching
-path caching

Náhodné cesty

Test	Nodes/ Search	Dist	Total Time	Speed units/ms	Delay ms
SZ	240	2088	517.3s	0.4	1469
SZ+CC	241	2088	221.3s	0.94	628
SZ+CC+PC	230	2102	212.2s	0.99	602
SZ+CC +sPC	195	2143	150.2s	1.42	424
GR	2031	2076	1585.7s	0.13	15857
GR+CC	2146	2053	660.3s	0.31	6603
GR+CC+PC	1979	2062	618.3s	0.33	6183
GR+CC +sPC	1753	2069	482.4s	0.42	4824
RD	197	2114	936.4s	0.22	931
RD+CC	185	2130	489.2s	0.43	486
VZ	332	2317	2195s	0.1	2469
VZ+CC	332	2317	749.5s	0.3	843
VZ+CC+PC	230	2399	559.3s	0.42	591

SZ ... statické zónování
GR ... nevylepšený A*
RD ... roadmaps
VZ ... V. Zónování

CC ... collision caching
PC ... path caching

Extrahované cesty

Test	Nodes/ Search	Dist	Total Time	Speed units/ms	Delay ms
SZ	386	2052	966.8s	0.21	2222
SZ+CC	503	2051	551.3s	0.37	1388
SZ+CC+PC	534	2052	595.9s	0.34	1475
SZ+CC+ sPC	273	2052	203.9s	1	635
GR	1527	2050	1598.5s	0.12	11841
GR+CC	1887	2062	768.4s	0.26	5528
GR+CC+PC	1267	2053	437.7s	0.46	3647
GR+CC+ sPC	932	2063	292.4s	0.7	2249
RD	186	2050	1064.1s	0.19	1019
RD+CC	196	2065	547.3s	0.37	526
VZ	599	2052	3901.4s	0.05	4763
VZ+CC	461	2053	930.2s	0.22	1177
VZ+CC+PC	588	2066	1298.5s	0.15	1603

SZ ... statické zónování
GR ... nevylepšený A*
RD ... roadmaps
VZ ... V. Zónování

CC ... collision caching
PC ... path caching

Postřehy

- Hierarchie nepoměrně vylepší poměr cena/výkon
- CC se vždy vyplatí (záleží na ceně collision checking) → rychlejší
  - 1,8 – 4,2 – krát rychlejší
  - roadmaps mají nejmenší užitek
- Přínos PC je také zřetelný – ale musí se na to opatrně!
  - u reálných cest může být přínos tak malý, že ani nevyrovná režii!
  - u hierarchického pf. se ale dá dosáhnout dalšího 1,5 – 2 – násobného zlepšení

Odkazy

- A.Botea, M. Müller, J. Shaeffer: *Near Optimal Hierarchical Path-Finding*
  pěkný článek o základech hierarchického patfindingu, HPA*

- M.Lanctot, N. Ng Man Sun, C. Verbrugge: *Path-finding for large scale multiplayer games*
  srovnávací článek

- B. Reese, B.Stout: *Finding a Pathfinder*
  obecně o problémech s pathfindingem

- R.Barták: materiály k předmětu *Umělá inteligence I*

Otázky

Šťastné a veselé

Děkuji za pozornost