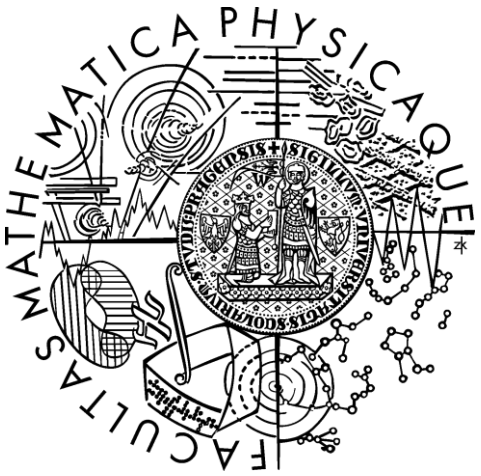


Vizualizace problémů pohybu po grafu

Pavel Surynek

Matematicko-fyzikální fakulta
Univerzita Karlova v Praze

Problémy pohybu po grafu

- **Abstrakce** k úlohám o pohybu více (autonomních nebo pasivních) entit v jistém prostředí (fyzickém či virtuálním).
  - Entity mají dané **počáteční** a **cílové** rozmístění v prostředí.
  - Chceme **naplánovat pohyby entit v čase**, aby dosáhly cílového rozmístění a přitom **respektovaly fyzikální omezení**.
- **Fyzikální omezení** jsou:
  - Entity spolu **nesmějí kolidovat**.
  - Entity **nesmějí narážet do překážek** vyskytujících se prostředí.
- Existují **dvě abstrakce** k úloze o pohybu:
  - **Pohyb kamenů po grafu** (*pebble motion on a graph*)
  - **Plánování cest pro mnoho robotů** (*multi-robot path planning*)

Pavel Surynek, 2012

Úloha o **pohybu kamenů po grafu** (1)

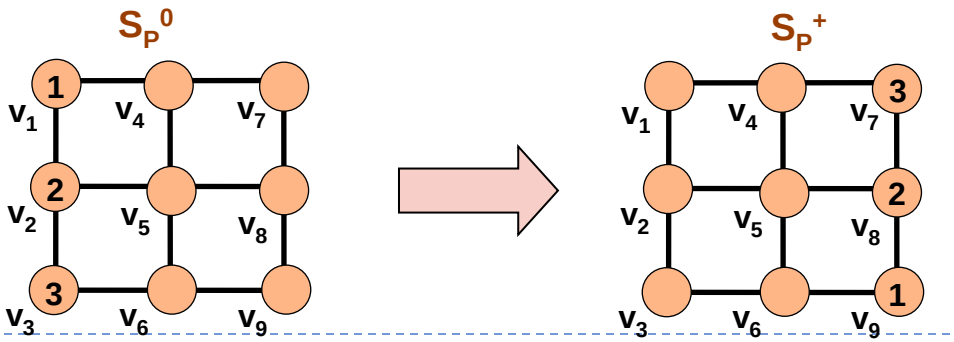
Wilson, 1974; Kornhauser et al., 1984

- Populární pohybová úloha, jež lze abstrahovat jako problém pohybu kamenů po grafu je známá jako **Lloydova patnáctka (devítka).**
  - **Entita** je **kámen** s číslem (*pebble*).
- Prostředí modelujeme jako neorientovaný graf, kde **vrcholy reprezentují pozice** v prostředí a **hrany možnost průchodu entity mezi pozicemi**.
- **Formální definice** problému pohybu kamenů po grafu:
  - Je to čtveřice $\Pi = (G, P, S_P^0, S_P^+)$, kde:
    - $G=(V,E)$ je neorientovaný **graf**,
    - $P = \{p_1,p_2,...,p_\mu\}$, kde $\mu<|V|$ je **množina kamenů**,
    - S_P^0: $P \rightarrow V$ je prostá funkce určující **počáteční rozložení** kamenů a
    - S_P^+: $P \rightarrow V$ je prostá funkce určující **cílové rozložení** kamenů.

Úloha o **pohybu kamenů po grafu** (2)

Wilson, 1974; Kornhauser et al., 1984

- Čas modelujeme diskrétně. **Časové kroky** a jejich uspořádání jsou izomorfní přirozeným číslům.
- **Dynamicita** úlohy je následující:
  - Kámen nacházející se v časovém kroku i v jistém vrcholu se může přesunout do sousedního vrcholu v časovém kroku i+1, jestliže cílový vrchol je v časovém kroku i **neobsazený** a **žádný jiný kámen** se současně nepřesouvá do stejného cílového vrcholu.
- Pro dané $\Pi = (G, P, S_P^0, S_P^+)$, chceme najít:
  - Posloupnost pohybů pro každý kámen tak, že je splněna podmínka dynamicity úlohy a kámen dosáhne cílového vrcholu.



Řešení problému pohybu po grafu, kde P={1,2,3}

délka řešení=7

$\mathbf{M_1}=[\mathbf{v_1}, v_4, v_7, v_8, v_9, v_9, \mathbf{v_9}]$
$\mathbf{M_2}=[\mathbf{v_2}, v_2, v_1, v_4, v_7, v_8, \mathbf{v_8}]$
$\mathbf{M_3}=[\mathbf{v_3}, v_3, v_3, v_2, v_1, v_4, \mathbf{v_7}]$

Časový krok: **1 2 3 4 5 6 7**

Úloha **plánování cest pro mnoho robotů** (1)

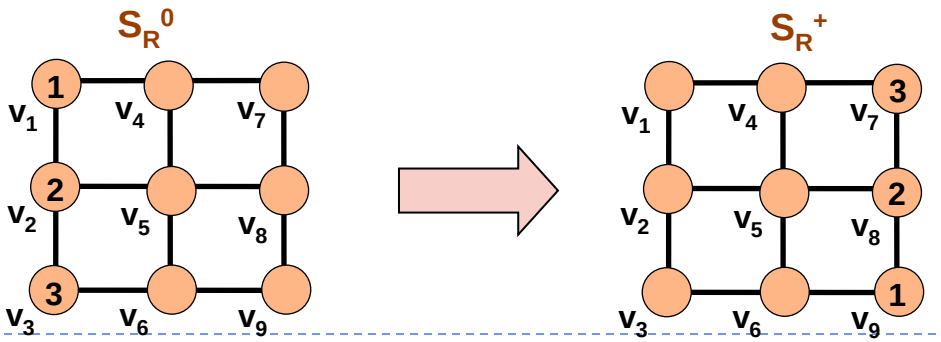
Ryan, 2007

- **Formální definice** problému plánování cest pro mnoho robotů:
  - Je to čtveřice $\Sigma = (G, R, S_R^0, S_R^+)$, kde:
    - G=(V,E) je neorientovaný **graf**,
    - $R = \{r_1, r_2, ..., r_v\}$, kde $v<|V|$ je **množina robotů**,
    - S_R^0: $R \rightarrow V$ je prostá funkce určující **počáteční rozmístění** robotů a
    - S_R^+: $R \rightarrow V$ je prostá funkce určující **cílové rozmístění** robotů.
- **Zeslabíme** podmínku na **dynamicitu**:
  - Robot nacházející se v časovém kroku i v jistém vrcholu se může přesunout do sousedního vrcholu v časovém kroku i+1, jestliže je cílový vrchol je **neobsazený** nebo jistým způsobem **právě opouštěný** a **žádný jiný robot** v témže časovém kroku nevstupuje do stejného cílového vrcholu.

Úloha **plánování cest pro mnoho robotů** (2)

Ryan, 2007

- Pro dané $\Sigma = (G, R, S_R^0, S_R^+)$, chceme najít:
  - Posloupnost pohybů pro každého robota tak, že je splněna podmínka dynamicity úlohy a robot dosáhne cílového vrcholu.
- Podmínka dynamicity pro pohyb kamenů po grafu **implikuje** podmínku dynamicity pro plánování cest pro mnoho robotů.
  - Řešení úlohy pohybu kamenů po grafu je také řešením odpovídající úlohy plánování cest pro mnoho robotů.
- **V čem je tedy rozdíl?**
  - Plánování cest pro mnoho robotů umožňuje vyšší **paralelismus**.



Řešení problému plánování cest pro mnoho robotů, kde R={1,2,3}

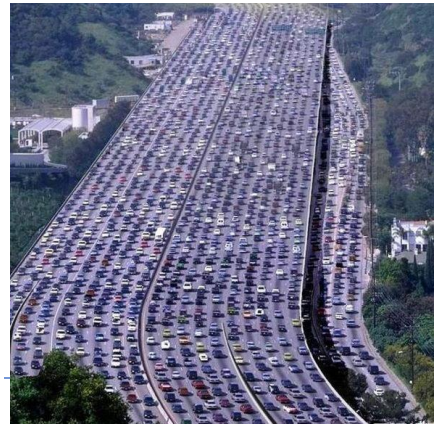
délka řešení=5

$O_1=[v_1, v_4, v_7, v_8, v_9]$
$O_2=[v_2, v_1, v_4, v_7, v_8]$
$O_3=[v_3, v_2, v_1, v_4, v_7]$

Časový krok: **1 2 3 4 5**

Má smysl řešit úlohy pohybu po grafu?

- Přesouvání kontejnerů
  (entita = **kontejner**)
- Intenzivní doprava
  (entita = **automobil** (v zácpě))
- Přesuny dat
  (entita = **datový paket**)
- Zobecněné výtahy
  (entita = **výtah**)

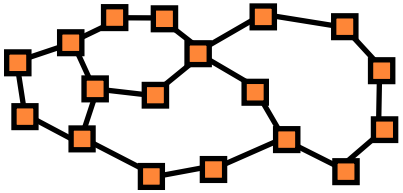


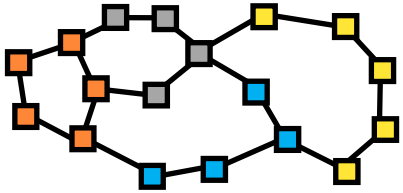
Jsou úlohy pohybu po grafu **lehké** či **těžké**?

- **Základní** varianta obou úloh je **snadno řešitelná**:
  - Existuje algoritmus pracující v polynomiálním čase ($O(|V|^3)$), který vygeneruje řešení délky $O(|V|^3)$ problému pohybu kamenů po grafu (Kornhauser et al., 1984).
  - Podle uvedených pozorování je použitelný i na plánování cest pro mnoho robotů.
- Chceme-li najít **řešení nejkratší možné délky** obtížnost roste:
  - Optimální varianta úloh pohybu po grafu je **NP-těžká** a navíc **neexistuje** pro ni **polynomiální aproximační schéma** (pokud P≠NP) (Ratner a Warmuth, 1986).
- Zaměřili jsme se na vygenerování **kvalitnějších** (tj. kratších) **neoptimálních** řešení:
  - Omezení na **2-souvislé grafy** - téměř vždy je úloha řešitelná.

Případ **2-souvislého** grafu

- Úlohy pohybu po grafu na **2-souvislých** grafech jsou z praktického hlediska nejdůležitější.
  - Téměř všechny cílová rozložení entit v grafu jsou dosažitelná z libovolného počátečního rozložení.
- **Další omezení** je, že dovolujeme pouze **jeden volný vrchol** (toto představuje neobtížnější situaci).
- Neorientovaný graf G=(V,E) je **2-souvislý**, jestliže $|V| \geq 3$ a $\forall v \in V$ je graf G=(V-{v},E'), kde E'={{x,y}∈E | x,y ≠ v}, souvislý.
- **Důležitá vlastnost:** Každý 2-souvislý graf sestrojit postupným přidáváním uch k počáteční kružnici → **rozklad na ucha**

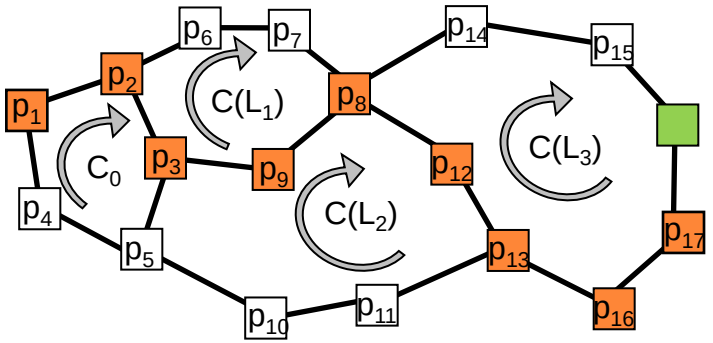




počáteční kružnice
1. ucho
2. ucho
3. ucho

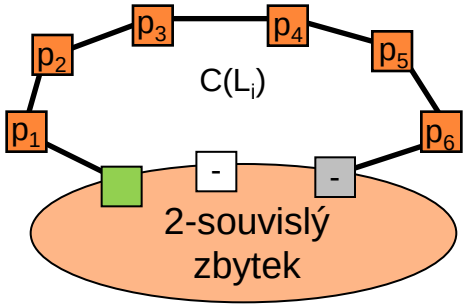
Algoritmus **BIBOX** (1)

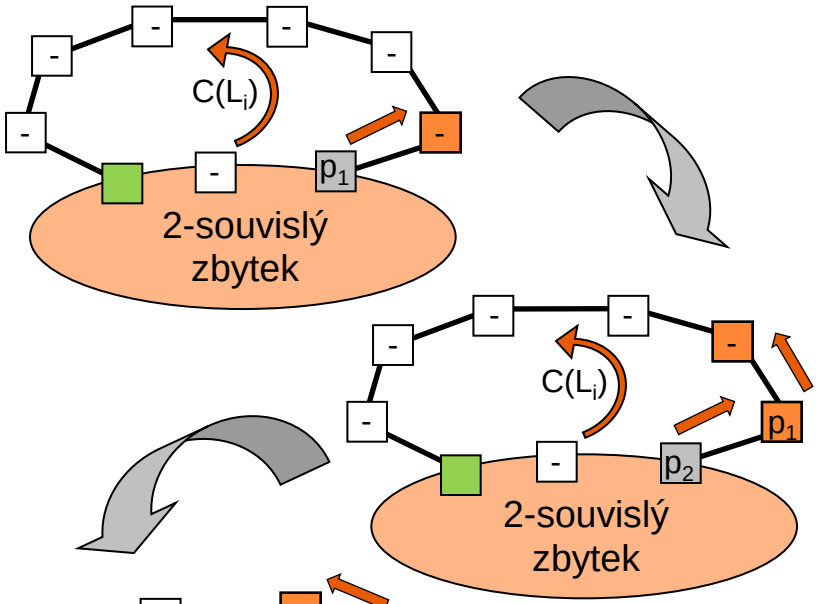
- Algoritmus **BIBOX** řeší obě úlohy pohybu po grafu.
  - Předpokládá, že prostředí je modelováno **2-souvislým** grafem.
    - Opírá se o **znalost rozkladu na ucha**.
  - Předpokládáme právě jeden neobsazený vrchol.
    - Není-li tomu tak, obsadí se vrcholy fiktivními kameny/roboty. Jejich pohyb se pak z finálního řešení odfiltruje.
  - Algoritmus opět pracuje v polynomiálním čase ($O(|V|^3)$), ovšem konstanta v odhadu je nižší než u algoritmu z (Kornhauser et al., 1984).
- Základní dovednost je přesun kamenu/robota do vybraného vrcholu grafu:
  - **Přesouvání volného** vrcholu a
  - **rotace podél uch**.



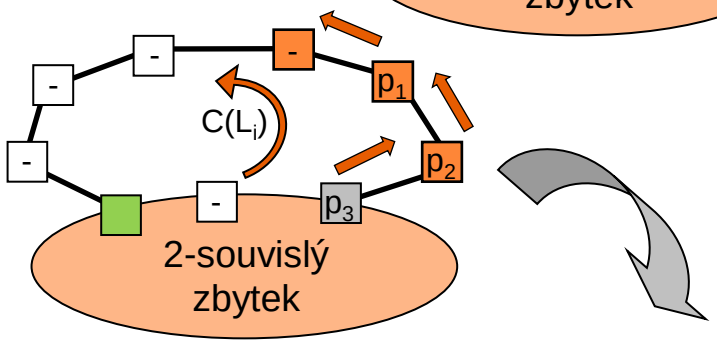
Algoritmus **BIBOX** (2)

- Zvládáme-li přesun individuálního kamenu/robota lze provádět složitější přesuny:
  - Skládání kamenů/robotů do ucha ve správném pořadí.





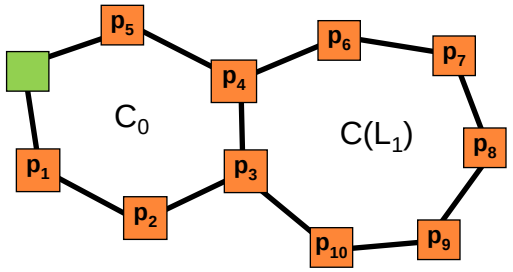
- **Zásobníkové skládání**:
  - Vezmeme **poslední ucho** rozkladu.
    - Přesuneme kámen/robota do **šedého** vrcholu.
    - Provedeme rotaci ucha (pomocí **zeleného** volného vrcholu).
    - ...



...

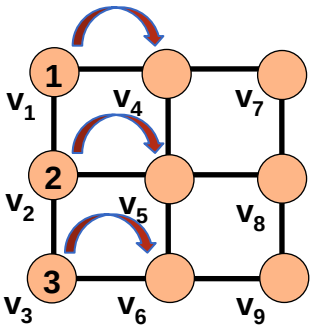
Algoritmus **BIBOX** (3)

- Počáteční kružnice a první ucho rozkladu představují zvláštní situaci.
- Zásobníkové skládání, zde **nefunguje**.
- Výslednou (sudou) **permutaci** kamenů/robotů nutno poskládat z jednotlivých **rotací podél 3-cyklů** (bez detailů).
  - **Úzké hrdlo** algoritmu, zlepšování pomocí předvypočtené databáze optimálních řešení.
  - Podstatné vylepšení: používat optimální řešení tzv. **slabých rotací podél 3-cyklů**.

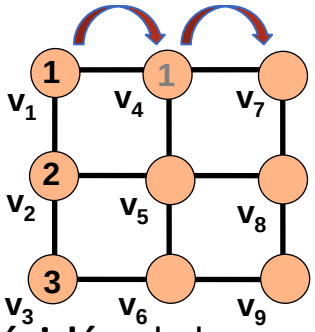


Zvyšování paralelismu v řešeních

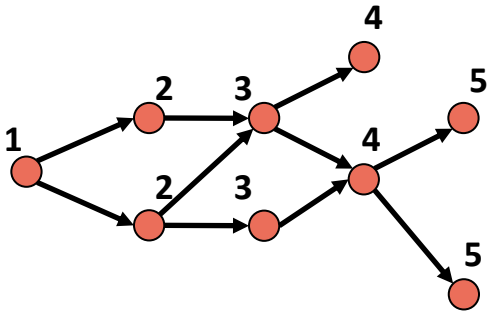
- Řešení generovaná algoritmem BIBOX nezohledňují možnosti **paralelismu** (více než jeden pohyb je proveden mezi dvojicí po sobě jdoucích časových kroků).
  - V úloze pohybu kamenů po grafu je paralelismus možný, když graf obsahuje alespoň **dva neobsazené vrcholy**.
  - U plánování cest pro mnoho robotů je **paralelismus možný vždy**.
- Jak zkonstruovat paralelní řešení?
  - Definuje se závislost mezi pohyby – závislé pohyby musejí následovat jeden po druhém.
  - Metodou **kritické cesty** spočítáme **nejdřívější čas** provedení pohybů.



Nezávislé pohyby - možno provést **paralelně**



Závislé pohyby - nutno provést **sekvenčně**



Nedostatky popsaného přístupu

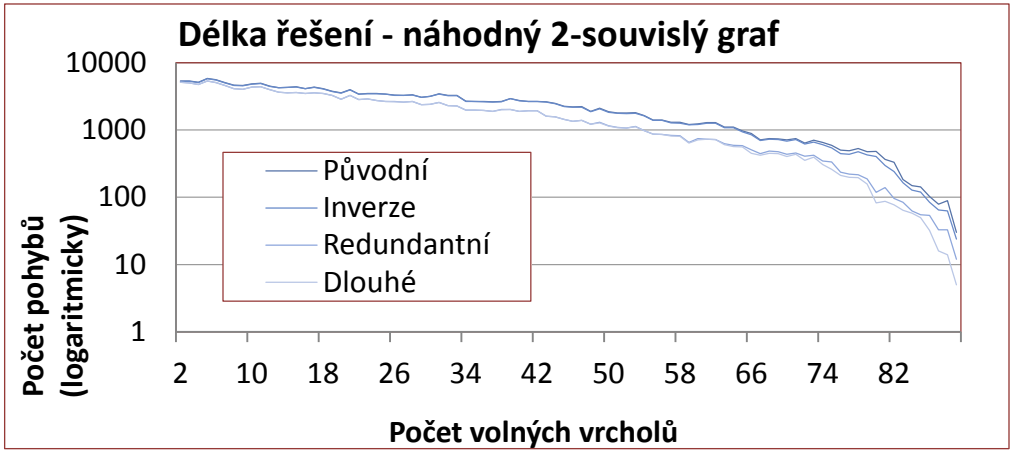
- Není-li graf zcela zaplněn kameny/roboty může navržený postup vytvářet jisté redundace v rámci řešení.
  - Přidáme fiktivní kameny/roboty, úlohu vyřešíme.
  - Z vygenerovaného řešení odstraníme pohyby fiktivních kamenů/robotů.
  - Řešení zparalelizujeme metodou kritické cesty.
- Pomocí vizualizačního programu GraphRec se podařilo identifikovat několik typů **redundancí**:
  - **(i) Inverzní pohyby** – pohyb negující bezprostředně předcházející pohyb.
  - **(ii) Redundantní pohyby** – sekvence pohybů, která kámen/robota vrátí do výchozí pozice.
  - **(iii) Dlouhá posloupnost pohybů** – sekvence pohybů, která kámen/robota přesouvá do jiného vrcholu, ale přitom existuje kratší sekvence pohybů provádějící totéž.

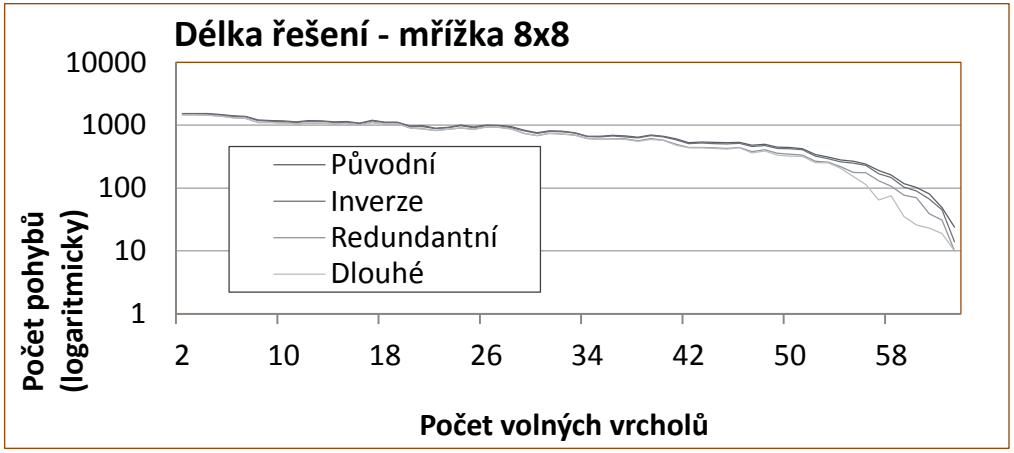
Odstraňování redundancí

- Uvedené tři druhy redundancí jsou postupnými zobecněními, tj. redundance typu **(iii) zahrnuje všechny předchozí**.
- Odstranění redundancí lze provádět přímočarými algoritmy (podle definice dané redundance).
  - **Časové nároky** algoritmů na odstranění jednotlivých typů redundancí **rostou**
    (redundance typu **(iii) má největší časové nároky** při odstraňování).
  - Proto se algoritmy **aplikují postupně**.
    - 1. odstranit inverzní pohyby (i)
    - 2. odstranit redundantní pohyby (ii)
    - 3. odstranit dlouhé posloupnosti (iii)
  - Náročnější algoritmus běží na již potenciálně zkráceném řešení.
- Provedli jsme **srovnání**, jaký přínos odstranění redundancí má.

Experimenty (pro pohyb kamenů po grafu)

- **Náhodný 2-souvislý** graf:
  - Postupné přidávání uch náhodné délky k již zkonstruovanému grafu.
  - Počáteční a cílové rozložení kamenů voleno jako náhodná permutace.
- **Mřížka 8x8**:
  - Opět náhodné počáteční a cílové rozložení.
- Postupně zmenšujeme počet kamenů.
- Obsahuje-li graf více volných vrcholů, dosahujeme až několikanásobného zkrácení řešení.





Literatura (1)

- **Wilson**, R. M., 1974. Graph Puzzles, Homotopy, and the Alternating Group. Journal of Combinatorial Theory, Ser. B 16, pp. 86-96, Elsevier.
- **Kornhauser**, D., **Miller**, G. L., **Spirakis**, P. G., 1984. Coordinating Pebble Motion on Graphs, the Diameter of Permutation Groups, and Applications. Proceedings of the 25th Annual Symposium on Foundations of Computer Science (FOCS 1984), pp. 241-250, IEEE Press.
- **Ratner**, D., **Warmuth**, M. K., 1986. Finding a Shortest Solution for the N × N Extension of the 15-PUZZLE Is Intractable. Proceedings of the 5th National Conference on Artificial Intelligence (AAAI 1986), pp. 168-172, Morgan Kaufmann Publishers.
- **Ryan**, M. R. K., 2007. Graph Decomposition for Efficient Multi-Robot Path Planning. Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI 2007), Hyderabad, India, pp. 2003-2008, 2007.

Literatura (2)

- **Surynek**, P. 2009. A Novel Approach to Path Planning for Multiple Robots in Bi-connected Graphs. Proceedings of the 2009 IEEE International Conference on Robotics and Automation (ICRA 2009), Kobe, Japan, pp. 3613-3619, IEEE Press, 2009.
- **Surynek**, P. 2009. Towards Shorter Solutions for Problems of Path Planning for Multiple Robots in θ-like Environments. Proceedings of the 22nd International Florida Artificial Intelligence Research Society Conference (FLAIRS 2009), Sanibel Island, FL, USA, pp. 207-212, AAAI Press, 2009.
- **Surynek**, P 2009. Making Solutions of Multi-robot Path Planning Problems Shorter Using Weak Transpositions and Critical Path Parallelism. Proceedings of the 2009 International Symposium on Combinatorial Search (SoCS 2009), Lake Arrowhead, CA, USA , 6 pages, University of Southern California, 2009.
- **Surynek**, P. 2009**.** An Application of Pebble Motion on Graphs to Abstract Multi-robot Path Planning. Proceedings of the 21st International Conference on Tools with Artificial Intelligence (ICTAI 2009), Newark, NJ, USA, pp. 151-158, IEEE Press, 2009, ISBN 978-0-7695-3920-1, ISSN 1082-3409.

Splniteľnosť (SAT)
Plánovanie cez SAT
Skracovanie plánov cez SAT

Tomáš Balyo

Splniteľnosť (SAT)

- Definície
  - Bool. premenná má hodnoty True a False
  - Literál je premenná alebo jej negácia
  - Klauzula je disjunkcia literálov
  - CNF formula je konjunkcia klauzul

F = (X or -Y or Z) and (-X or W) and (-W or -Z)

Splniteľnosť (SAT)

- Definície
  - **Bool. premenná** má hodnoty True a False
  - Literál je premenná alebo jej negácia
  - Klauzula je disjunkcia literálov
  - CNF formula je konjunkcia klauzul

F = (**X** or -**Y** or **Z**) and (-**X** or **W**) and (-**W** or -**Z**)

Splniteľnosť (SAT)

- Definície
  - Bool. premenná má hodnoty True a False
  - **Literál** je premenná alebo jej negácia
  - Klauzula je disjunkcia literálov
  - CNF formula je konjunkcia klauzul

F = (**X** or **-Y** or **Z**) and (**-X** or **W**) and (**-W** or **-Z**)

Splniteľnosť (SAT)

- Definície
  - Bool. premenná má hodnoty True a False
  - Literál je premenná alebo jej negácia
  - **Klauzula** je disjunkcia literálov
  - CNF formula je konjunkcia klauzul

F = **(X or -Y or Z)** and **(-X or W)** and **(-W or -Z)**

Splniteľnosť (SAT)

- Definície
  - Bool. premenná má hodnoty True a False
  - Literál je premenná alebo jej negácia
  - Klauzula je disjunkcia literálov
  - **CNF formula** je konjunkcia klauzul

F = **(X or -Y or Z) and (-X or W) and (-W or -Z)**

SAT

- Formula je splniteľná ak má splňujúce ohodnotenie – v každej klauzule je splnený aspoň jeden literál

  F = (X or -Y or Z) and (-X or W) and (-W or -Z)

  X=true, W=true, Z=false je splňujúce ohodnotenie, tzn. F je splniteľná

- **Probém splniteľnosti**: pre danú CNF formulu najdi splňujúce ohodnotenie alebo zisti, že je nesplniteľná

Riešenie splniteľnosti

- SAT je NP úplny problem
  - Nie je známy algoritmus, ktorý je v najhoršom prípade rýchlejší než 2^n
  - Ak by existoval polynomiálny algoritmus, tak existuje polyn. algoritmus na všetky problémy v NP napr. klika, TSP, CSP, farbenie grafu
  - Existujú riešiče, ktoré vyriešia mnoho vstupov veľmi rýchlo
  - Preto sa používajú v praxi na riešenie NP problémov ako napr. verifikácia obvodov a software alebo plánovanie

Riešiče SAT

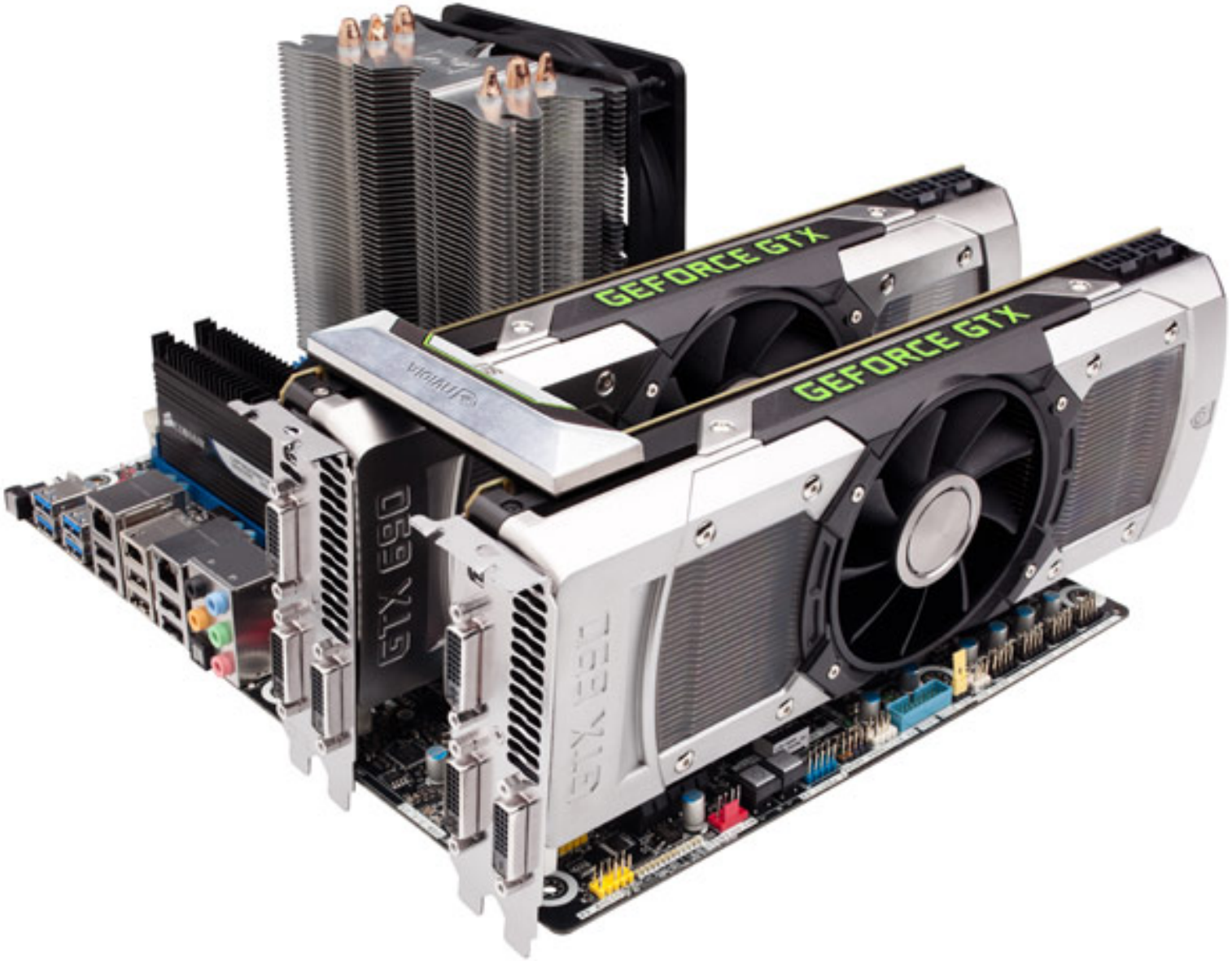
- Neúplné riešiče založené na local search
  - Začneme s náhodnym ohodnotením premenných a prepíname hodnoty podľa heuristiky (walkSat, Novelty, Gsat, …)
  - Nevedia dokázať nesplniteľnosť => neúplne
- Úplne riešiče založené na systematickom prehľadávaní (DPLL algoritmus)
  - Prehľadávanie priestoru čiastočných ohodnotení do hlbky (DFS) s orezávanim (unit propagation) a heuristikami.

SAT na KTIML

- Chceme využiť silu grafických kariet (GPU) pri riešení SAT a príbuzných problémov
  - CUDA walksat (Filip Stočes)
  - Počítanie na GPU zložitých ale dobre paralelizovateľných heuristik pre DPLL
  - Samotné DPLL sa zle paralelizuje
  - Príbuzné problémy: #SAT, Max–SAT
- Multi–vláknové SAT solvery pre viacjadrové processory vyvýjame
- Dobrá téma na BP/DP

SAT na KTIML

- Ch … pri
rie …



- Mu … é
pr…
- Do…

SAT pre KDSS

Department of Distributed and Dependable Systems D3S

- Problém: pre danú formulu najdi splňujúce ohodnotenie minimálnej dĺžky
  - Tzn. Ohodnoť minimálne množstvo premenných tak aby bola formula splnená (každá klauzula)
- Použije sa pre SMT solver pre SPL
  - Programátor deklaruje výkonnostné predpoklady k metódam triedy
  - Použitím experimentov sa predpoklady overia
  - Aby sa minimalizoval počet potrebných experimentov pre overenie všetkých predpokladov tak sa hodí minimálne splňujúce ohodnotenie

Plánovanie cez SAT

- Ako previesť plánovanie na SAT?
  - Plánovanie je PSPACE úplné
  - Obmedzíme sa na plány konečnej dĺžky
  - “existuje plán dĺžky ***k***?” je NP problém
- Iteratívny prevod na SAT
  - Pre plánovací problem ***P*** a číslo ***k*** zostrojíme formulu ***F(P,k)***, ktorá je splniteľná práve vtedy, keď existuje plán pre ***P*** dĺžky ***k*** alebo kratší
  - Postupne riešime ***F(P,1)***, ***F(P,2)***, … až kým sa neobjaví splniteľná formula

SAT plánovač na KTIML

- Vyvýjame plánovač v jazyku JAVA
  - Open source projekt **freelunch**
  - Plánovacia knižnica pre programátorov Java aplikácii a hier
  - Požíva viachodnotový SAS+ formalismus
  - Generuje optimálne paralelné plány
  - Rožširujeme o neoptimálny plánovač
  - V budúcnosti chceme pridať možnosti temporálneho plánovania a zdroje

- Témy na BP/DP: rozšírte alebo aplikujte fl

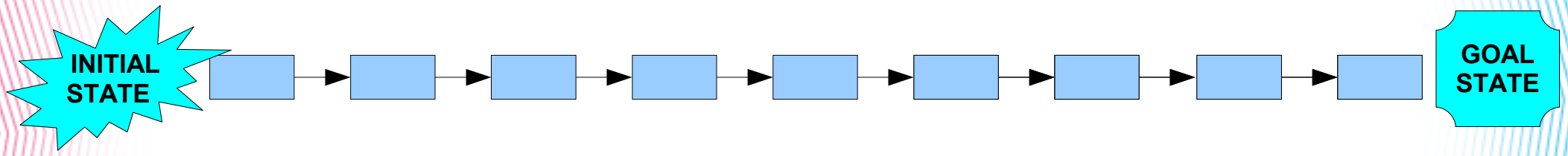
Zlepšovač plánov

- Existujú 2 typy plánovačov
  - Optimálne
    - + kvalitný plán
    - + efektívny na malé úlohy
    - – nezvláda ťažšie úlohy
  - Neoptimálne
    - + rýchlejšie
    - + zvláda väčšie a ťažšie úlohy
    - – nekvalitné (dlhé) plány
- Ako ich kombinovať a získať výhody oboch?

Kombinovaný plánovač

- Kombinujeme rýchly plánovač s optimálnym

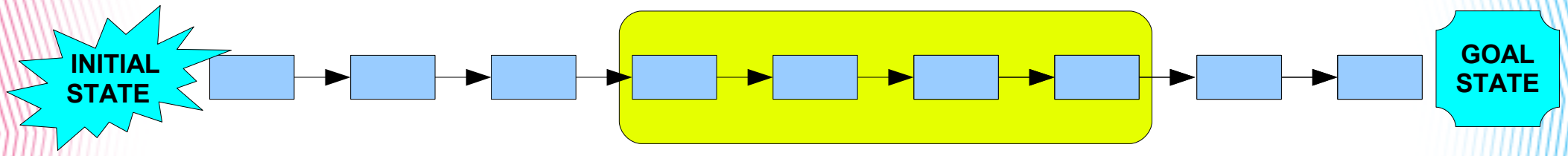
  1. Nájdi plán s rýchlym plánovačom



Kombinovaný plánovač

- Kombinujeme rýchly plánovač s optimálnym

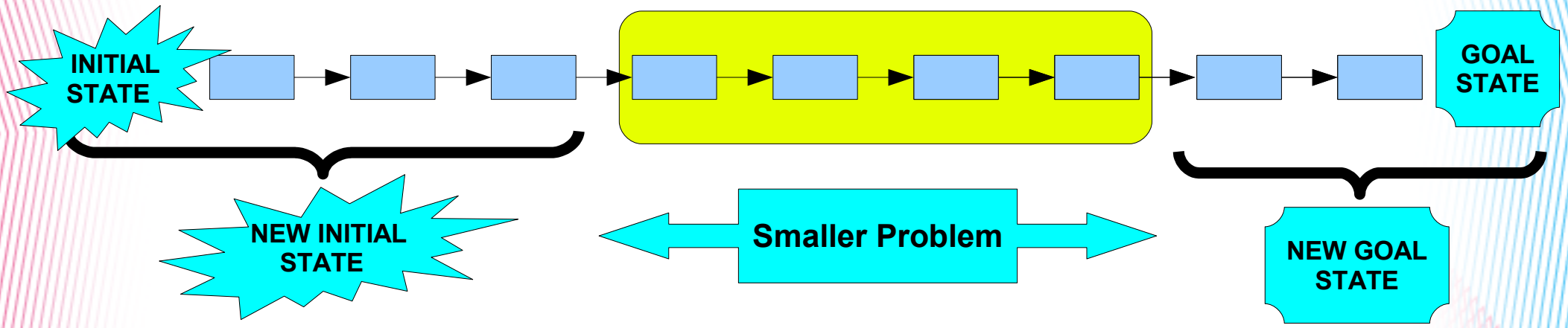
  2. Vyber nejaký podplán (podpost. akcií)



Kombinovaný plánovač

- Kombinujeme rýchly plánovač s optimálnym

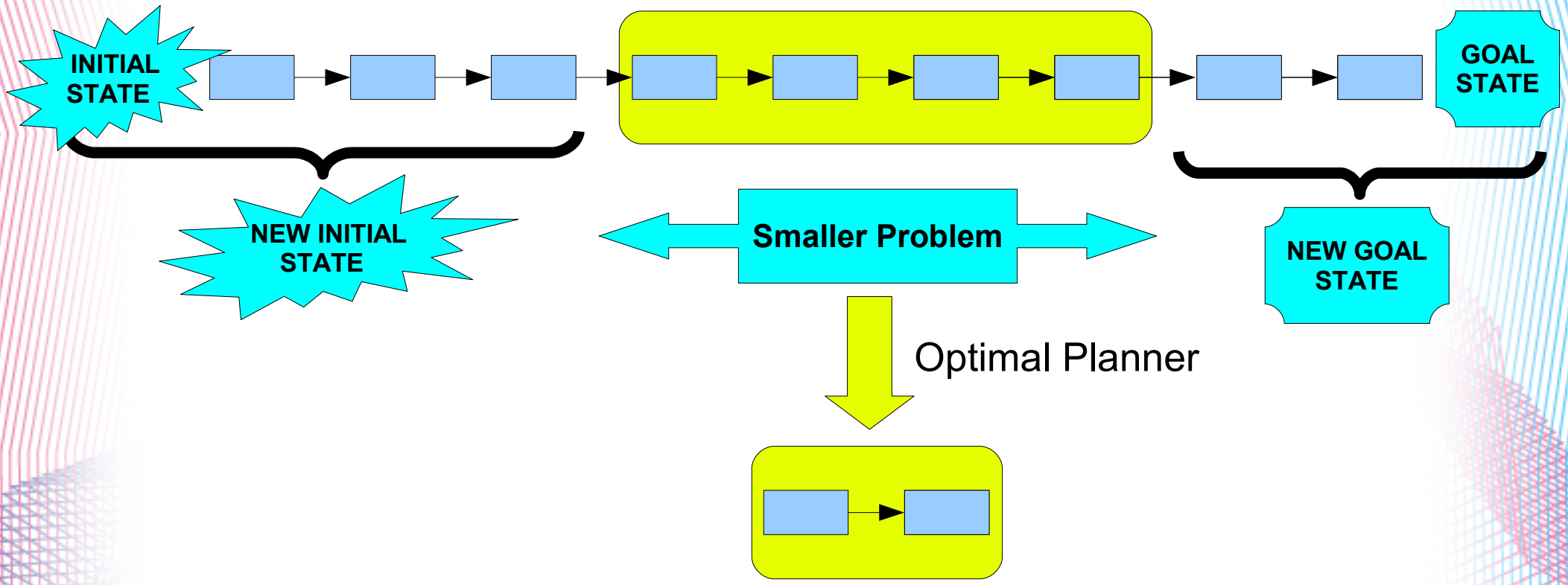
3. Vygeneruj malý plánovací problem



Kombinovaný plánovač

- Kombinujeme rýchly plánovač s optimálnym

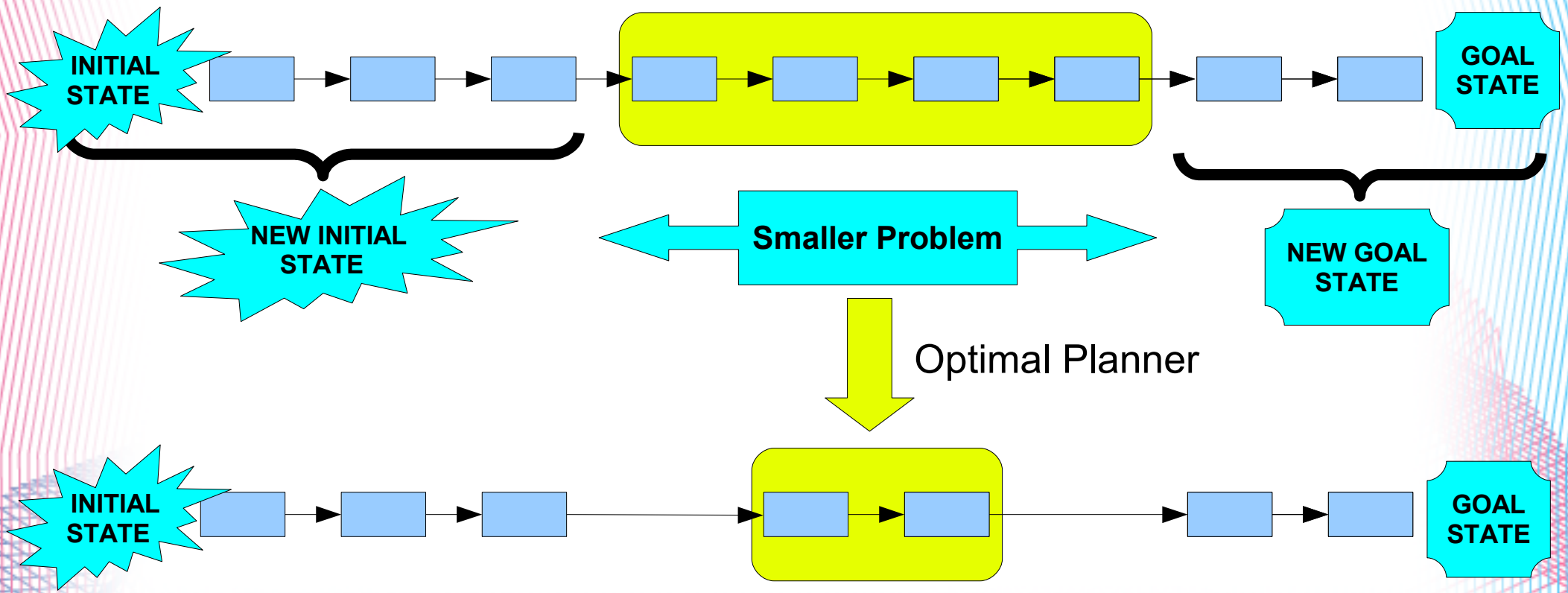
4. Vyrieš malý problem optimálnym plánovačom



Kombinovaný plánovač

- Kombinujeme rýchly plánovač s optimálnym

5. Nahraď podplán optimálnym



Kombinovaný plánovač

- Zlepšuj podplány pokiaľ nenastane timelimit
- Postupne sa vyberajú dlhšie podplány – ťažšie problémy pre optimálny plánovač
- Podplány sa vyberajú postupným posúvanim okienka cez celý plán
  - Mnoho priestoru na vylepšenie (BP/DP)
- Zlepšovač plánov je súčasťou freelunch

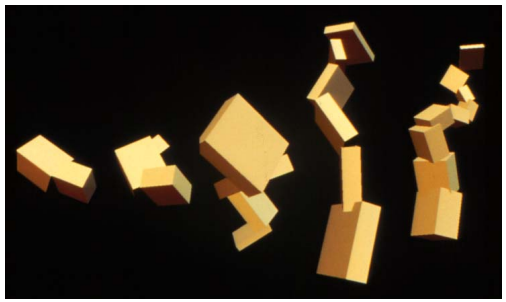
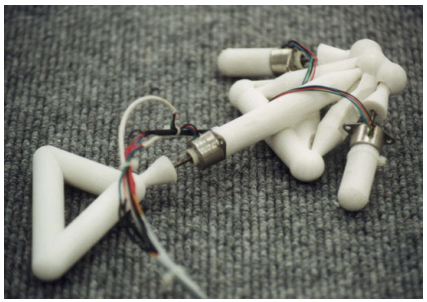
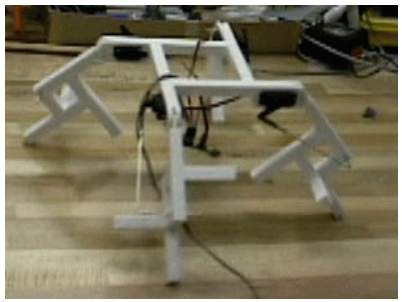
KONIEC

- Dakujem za pozornosť
- Otazky?
- Námety?
- Pripomienky?
- Záujemci o BP/DP?

- biotomas@gmail.com
- http://ktiml.mff.cuni.cz/~balyo

Introduction

- **Goal**: to automatically design hardware and software for robots capable of performing given tasks
- **Previous works**: Virtual Creatures (Sims 1994), GOLEM (Lipson, Pollack 2000), Genobots (Hornby, Pollack, 2002), Miconi (2006)

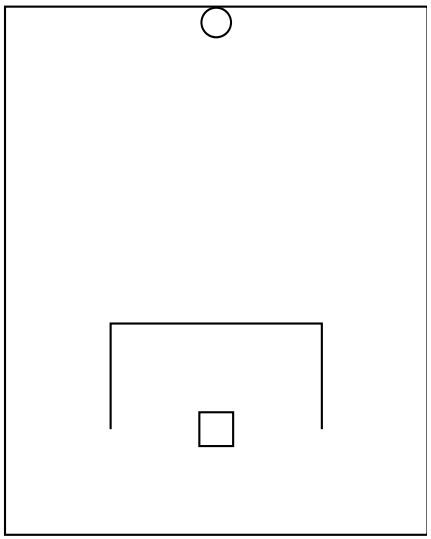
- Fitness evaluation done in **simulation**
- Body and brain of a robot is co-evolved
- Problem: **premature convergence** to local optima
- **Symptoms:** loss of diversity within a population, no beneficial mutations, stagnation

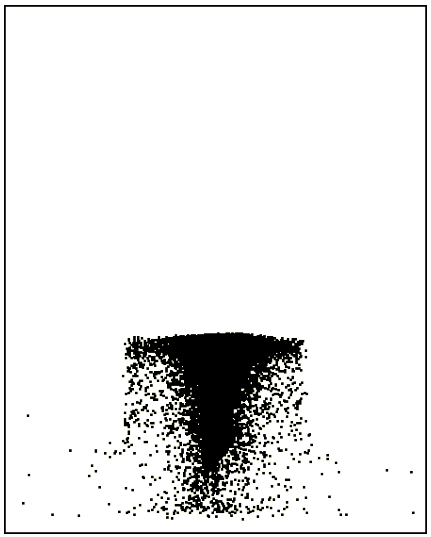
Problem Definition

- **Example:** barrier avoidance task
  - Robot (square) is rewarded for getting closer to the goal (circle):

$$f = \max(0, d(p_{start}, p_{target}) - d(p_{final}, p_{target})) \quad (1)$$

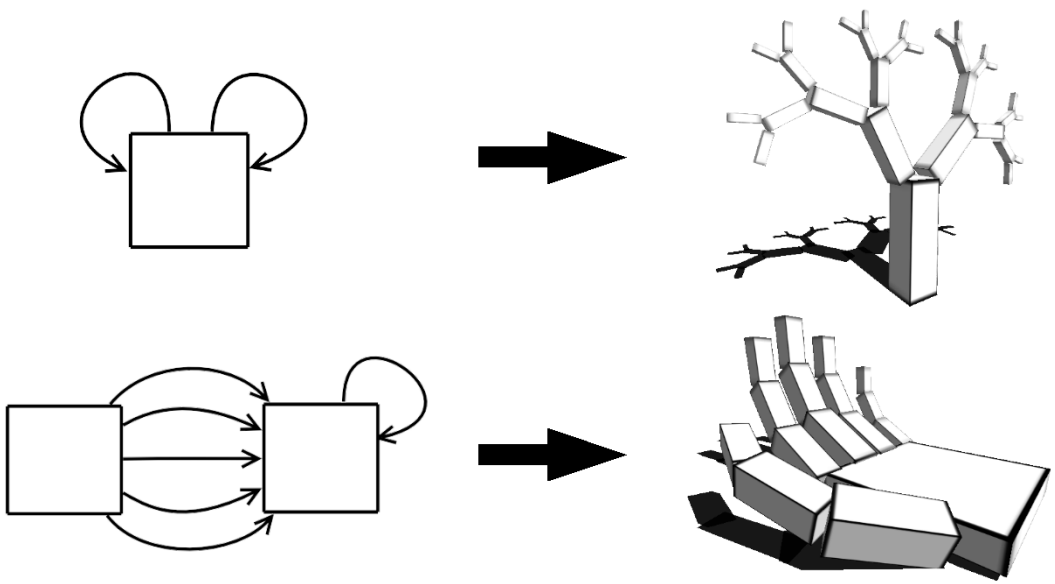
- **Problem:** Fitness-based search is prone to become trapped in local optima of the fitness function
- **Standard solutions:** diversity maintenance techniques (speciation, fitness sharing, ALPS) - still based on fitness





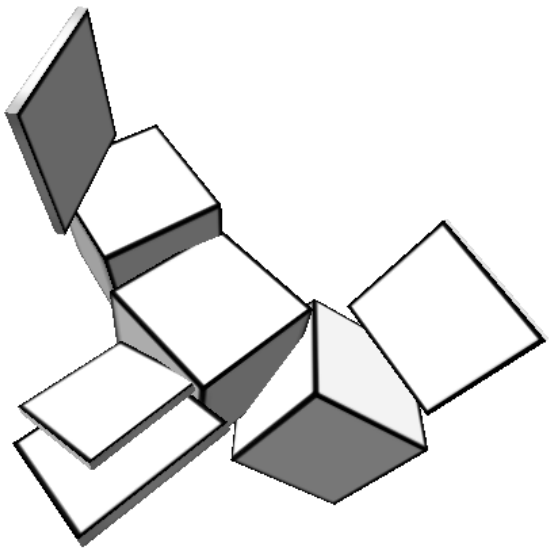
Robot Morphology

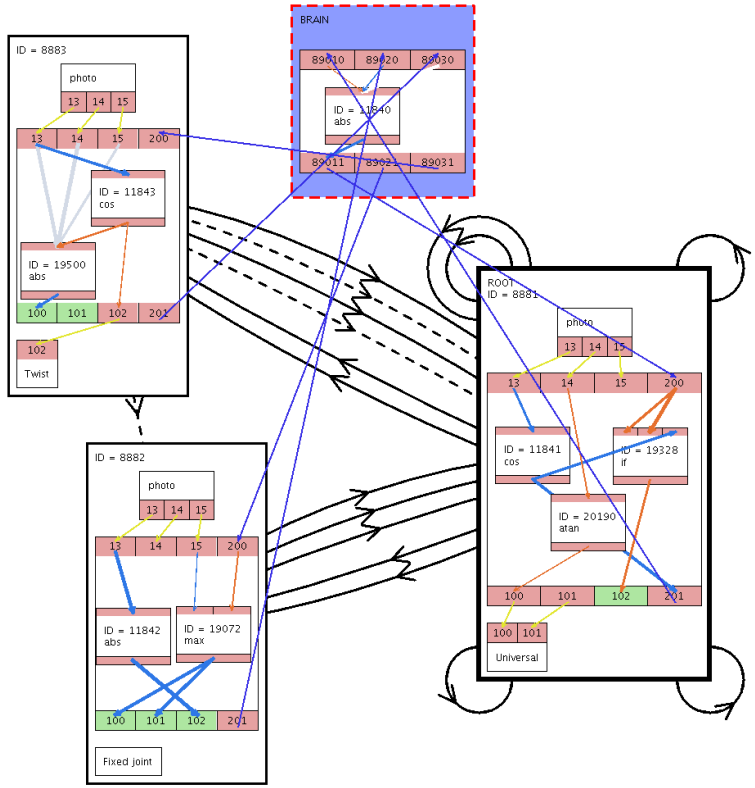
- **Morphology** represented by a directed graph
  - Each **node** represents a body part (box)
  - Each **edge** represents a physical joint between two parts
- **Generative encoding** inspired by Sims' Virtual Creatures
  - Phenotype is constructed by **recursive traversal** of edges of the genotype
  - **Recursive limit** in each node prevents infinite cycles



Robot Control System

- **Control system** is distributed along the robot body
- Each body part contains a **local feedforward neural network**, **sensors** and an **effector**
- Each neuron has one of **22 transfer functions**
- Single independent neural network for **central coordination**





Novelty Search (Lehman, Stanley, 2008)

- Avoids problem of premature convergence by ignoring the objective
- Instead searches for **any novel behaviors**

- Robot behavior needs to be described using a vector of real values
- Fitness function is replaced with a novelty metric computed using an **archive** of previously discovered individuals:

$$\rho(x) = \frac{1}{k} \sum_{i=0}^{k} dist(x, \mu_i),$$

where μ_i is the ith-nearest neighbor of x with respect to distance measure $dist$.

- Individuals with novelty metric above a threshold value are added to the archive

Novelty Search (Lehman, Stanley, 2008)

- Avoids problem of premature convergence by ignoring the objective
- Instead searches for **any novel behaviors**

- Robot behavior needs to be described using a vector of real values
- Fitness function is replaced with a novelty metric computed using an **archive** of previously discovered individuals:

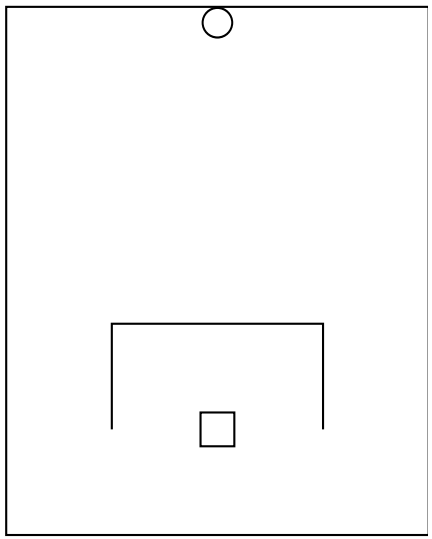
$$\rho(x) = \frac{1}{k} \sum_{i=0}^{k} dist(x, \mu_i),$$

where μ_i is the ith-nearest neighbor of x with respect to distance measure *dist*.

- Individuals with novelty metric above a threshold value are added to the archive

Experimental Setup

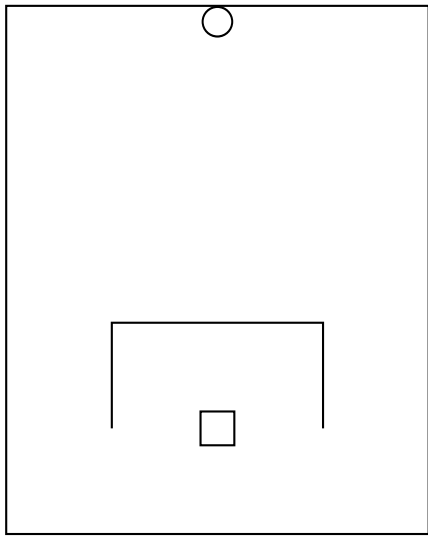
- **Barrier Avoidance:**
  - Container 20m x 20m x 25m, barrier 10m x 10m x 5m
  - Goal is to reach the target within a timeout of 60 seconds
  - Barrier obstructs direct path to the goal



- **Algorithms:** Novelty Search (NS), Fitness-Based Search (FBS), Random Search, **Combinations of NS and FBS** (NS to get around the barrier, FBS to optimize quickly afterwards)
- Each configuration tested **25 times** for **150 generations**
- **Statistical significance** of the differences verified ($p < 0.01$)

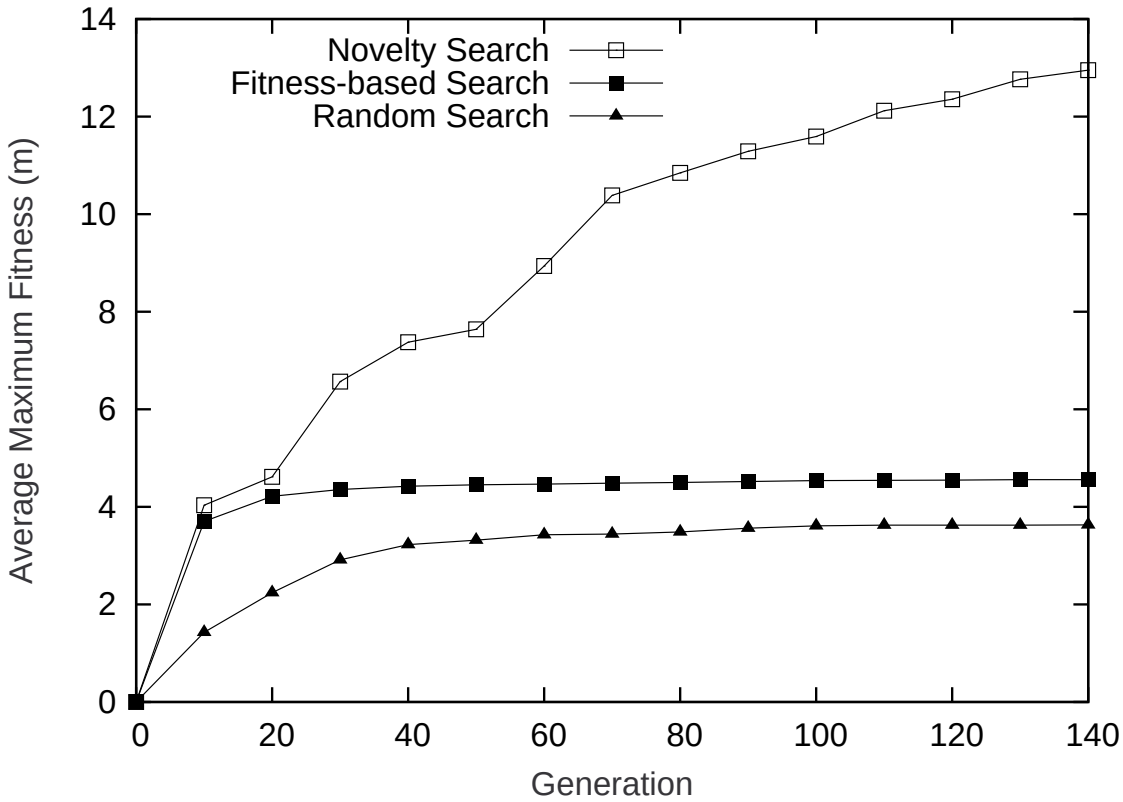
Experimental Setup

- **Barrier Avoidance:**
  - Container 20m x 20m x 25m, barrier 10m x 10m x 5m
  - Goal is to reach the target within a timeout of 60 seconds
  - Barrier obstructs direct path to the goal



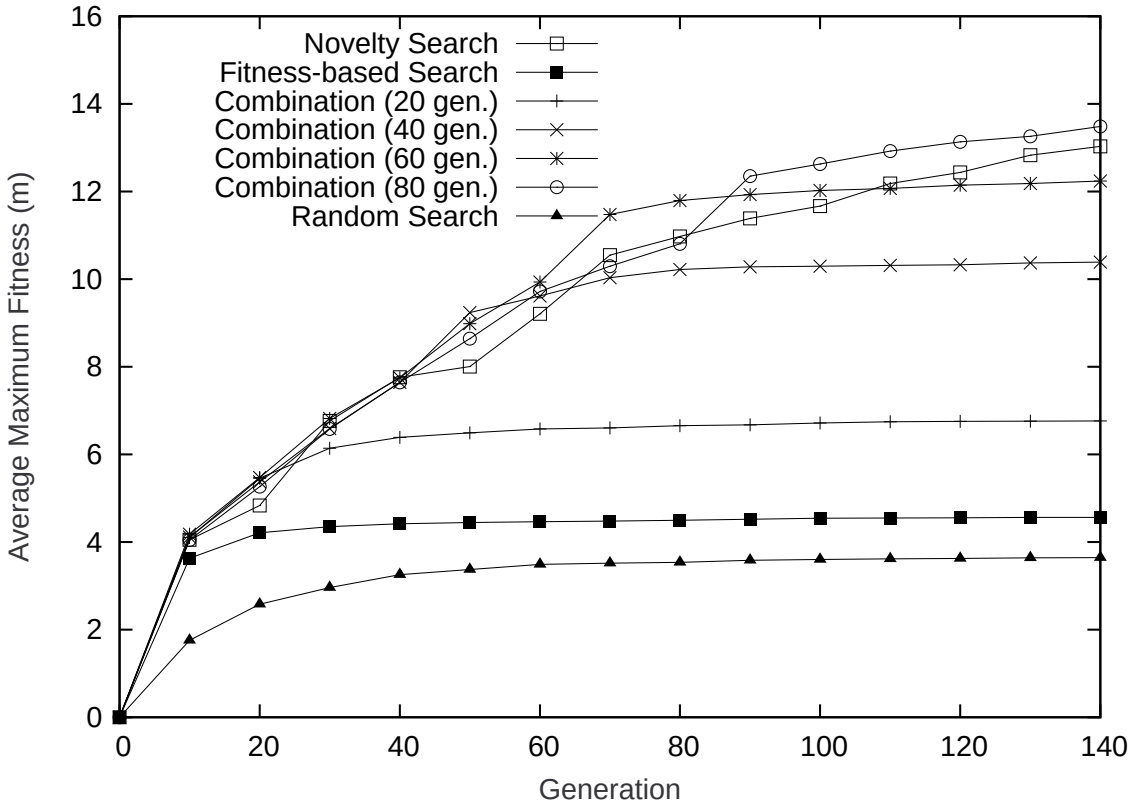
- **Algorithms:** Novelty Search (NS), Fitness-Based Search (FBS), Random Search, **Combinations of NS and FBS** (NS to get around the barrier, FBS to optimize quickly afterwards)
- Each configuration tested **25 times** for **150 generations**
- **Statistical significance** of the differences verified ($p < 0.01$)

Comparison of Maximum Achieved Fitness



- Random and fitness-based search never escape the barrier
- Novelty Search consistently escapes the barrier

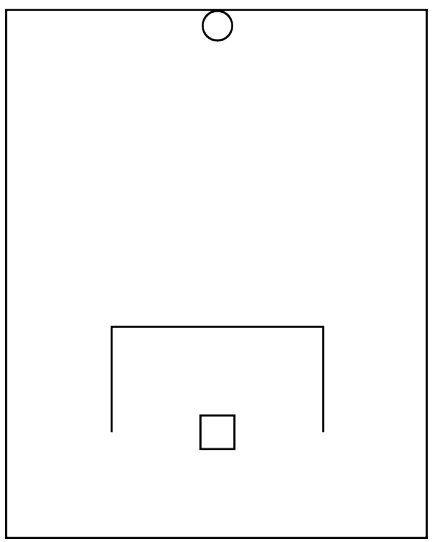
Comparison of Maximum Achieved Fitness

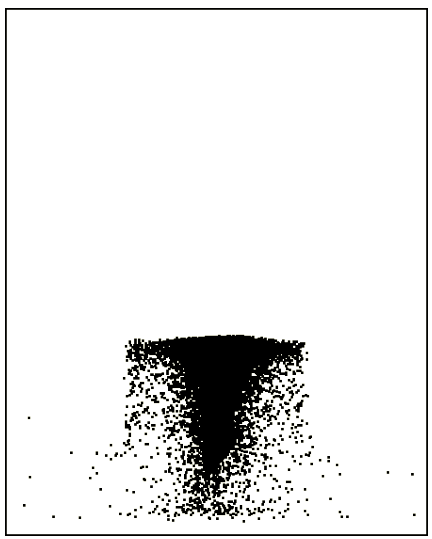


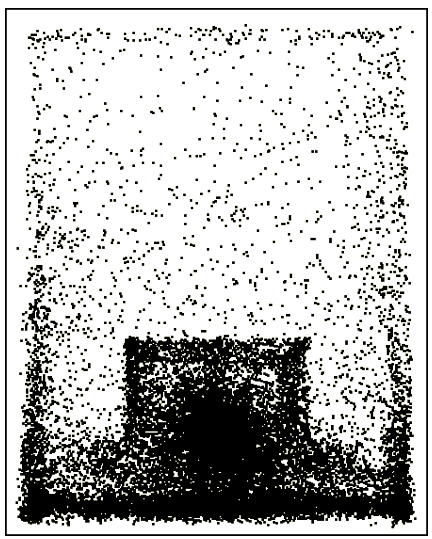
- Random and fitness-based search never escape the barrier
- Novelty Search consistently escapes the barrier
- Longer period of Novelty Search leads to higher fitness

Examples of Behavioral Diversity

- Did novelty search discover more distinct behaviors?
- Final positions of all robots in selected runs
- From left to right: experiment plan, fitness-based search, novelty search
- **Result:** Novelty Search explores behaviors more evenly than fitness-based search

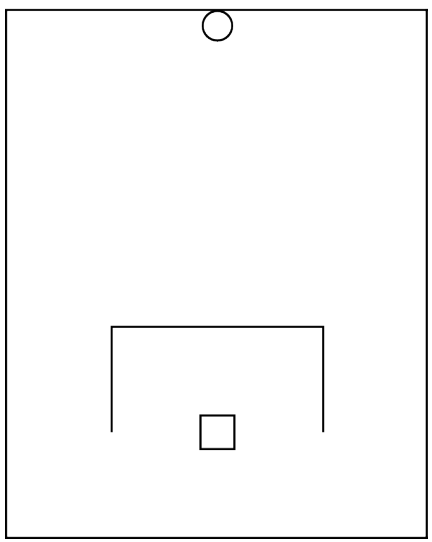
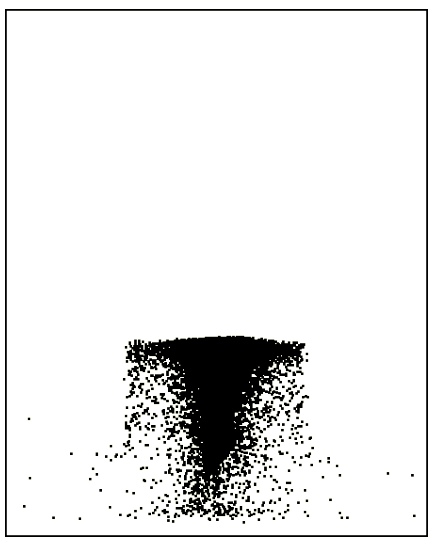
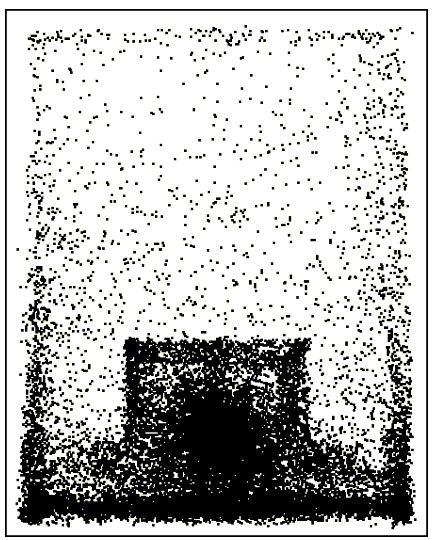






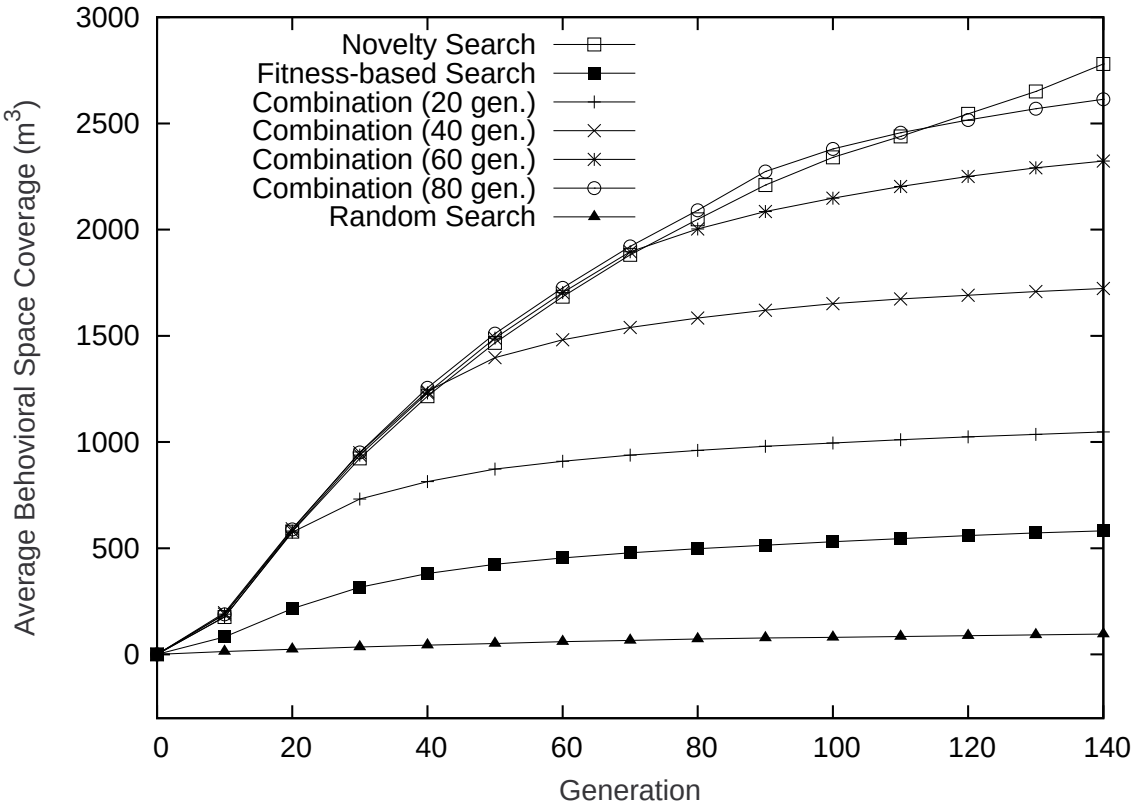
Examples of Behavioral Diversity

- Did novelty search discover more distinct behaviors?
- Final positions of all robots in selected runs
- From left to right: experiment plan, fitness-based search, novelty search
- **Result:** Novelty Search explores behaviors more evenly than fitness-based search

Behavioral Diversity

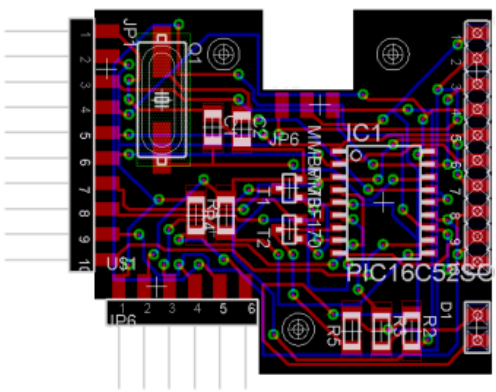
- **Metric**: Total number of cells from a 1m x 1m x 1m grid that contained the final position of at least one robot
- **Result**: Novelty search discovers more behaviors



Video

Conclusion

- **Contributions**
  - Independent verification of recently published algorithm
  - Novelty Search first time applied to the complex domain of body-brain co-evolution
  - Very successful for deceitful problem of barrier avoidance
  - Combining novelty search with fitness-search does not provide long term advantage
- **Future works**
  - Investigate further which problems are suitable for NS
  - Analyze long term behavior
  - Construct evolved robots in reality



Thank you for your attention.