**IV126 – Umělá inteligence II**

**Většina slidů ze semestru jaro 2017 + pár poznámek navíc.**

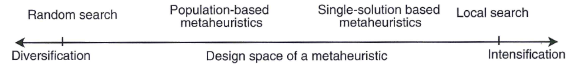
**Červeně zaznačené věci, které už se vyskytly na zkoušce.**

**Opravujte, doplňujte ;)**

**Metaheuristika**

Metodologie vyšší obecnější úrovně, která může být použita jako řídící strategie pro návrh základní heuristiky pro řešení specifických optimalizačních problémů. Reprezentují obecný přístup aplikovatelný na různé problémy. Je nutné aplikovat průzkum (diversifikace) a zužitkování (intersifikace).

* rozvrhování předmětů: simulované žíhání, výměna dvou předmětů.



**Iterativní**: začneme s úplným řešením a provádíme jeho transformaci (většina metaheuristik)

**Hladové**: začneme z prázdného řešení a v každém kroku přiřadíme jednu rozhodovací proměnnou, dokud nenalezneme úplné řešení

**Reprezentace** problému musí být úplná, dosažitelná a efektivní.

Lineární: binární kódování, diskrétní kódování, permutace, vektor diskrétních hodnot

Nelineární: stromy, grafy

**Binární kódování**

S každou rozhodovací proměnou je spojená binární hodnota. Reprezentace: vektor bitových hodnot.

**Př**.: problém batohu, SAT problém, problém celočíselného programování s binárními proměnnými.

**Diskrétní kódování**

Zobecněné binární kódování. Reprezentace: vektor diskrétních hodnot.

**Př**.: kombinatorické problémy, problém přiřazení.

**Reprezentace permutací**

Problémy: seřazení, směrování, plánování. Každý element problému se musí v reprezentaci objevit pouze jednou.

**Př**.: problém obchodního cestujícího, plánování úloh bez překrývání na jeden stroj.

**Účelová funkce**

**Soběstačná účelová fce** – přímo vychází z formulace problému, ideální situace,

**Př.:** problém obchodního cestujícího, **ne**vhodné použití SAT

**Směrující účelová fce** – případ, kdy je nutné účelovou fci transformovat kvůli lepší konvergenci metaheuristiky, nová účelová fce bude směřovat k efektivnějšímu prohledávání, **Př**.: k-SAT

**Relativní a kompetitivní účelová fce** – pokud nelze přiřadit fci absolutní hodnotu ke všem řešením, **Př**.: teorie her, strategie A, B, C

**Meta modelování** – při extrémně výpočetně náročné účelové fci je navržen přibližný model a přibližná účelová fce, **Př**.: návrh telekomunikační sítě

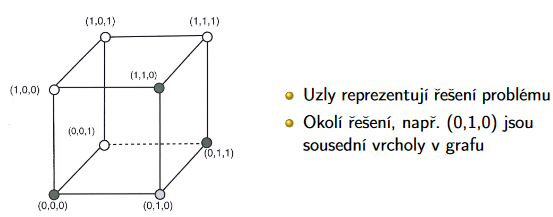
**Manipulace s omezeními**

**Zamítací strategie** – uchováváme pouze splnitelná řešení

**Penalizující strategie** – původní účelová fce rozšířená o penalizaci nesplnitelných řešení, uvažovány váhy nesplněných omezení, řádově vyšší váha za nesplněná omezení ve srovnání s hodnotou účelové fce

**Opravné strategie** – opravují nesplnitelná řešení, aplikovány opravné strategie na vygenerované nesplnitelné přiřazení, které tedy opravíme a pokračujeme se splnitelným řešením

**Okolí**

Funkce okolí *N* je mapování *N: S→2S*, která přiřadí každému řešení *s*z *S* množinu řešení *N(s)⊂S*.

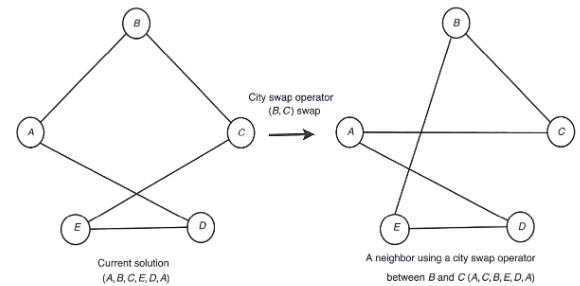
**Lokální optimum**

Řešení *s ∈ S*je lokální optimum vzhledem k dané fci okolí *N*, pokud má lepší kvalitu než všichni jeho sousedi.

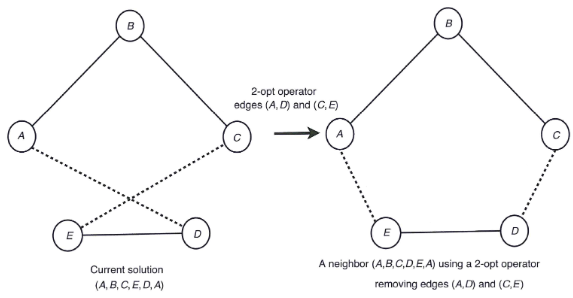
**K-distance**

Permutační problémy. **Př**.: problém obchodního cestujícího

Velikost okolí pro 2-distance: *n*(*n*-1)/2, kde *n* je počet měst.

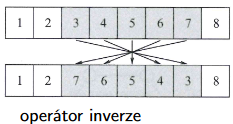


**K-opt**

Smaže *k* hran a nahradí je dalšími *k* hranami. Permutační problémy. **Př**.: problém obchodního cestujícího. Velikost okolí pro 2-opt: *n*(*n*-3)/2, kde *n* je počet měst.

**Permutační okolí pro rozvrhovací problémy**

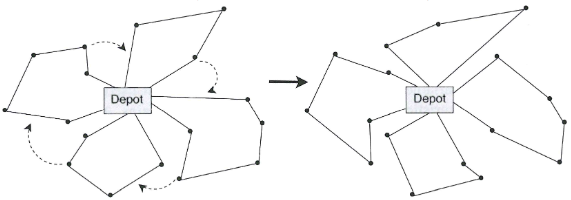




**Rozsáhlé okolí**

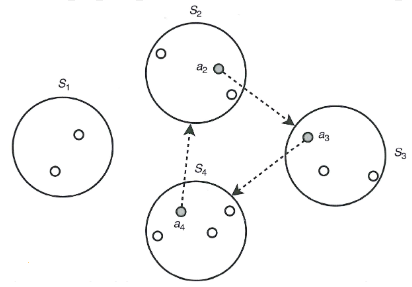
**Řetězec odstranění (ejection chain)**

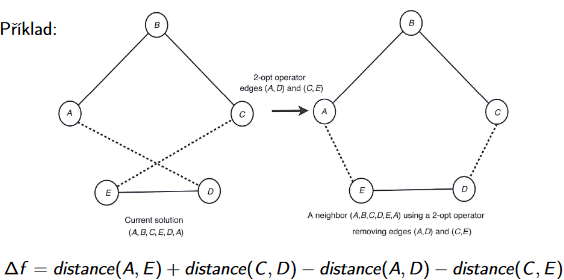
Vhodné pro VRP – posloupnost přesunů zákazníka z jednoho okruhu na následující



**Cyklická výměna (cyclic exchange)**

Problém rozdělení *n* prvků do *q* skupin.

Cyklická výměna 3 prvků (3-okolí): prvek a2 ∈ S2 je přesunut do S3, a3 ∈ S3 je přesunut do S4, a4 ∈ S4 přesunut do počáteční skupiny S2.

**Inkrementální účelová fce**

**Iniciální řešení**

Náhodné řešení, heuristické řešení (pomocí hladového algoritmu), částečně nebo úplně inicializované uživatelem. Použití lepšího iniciálního řešení nepovede vždy k lepšímu lokálnímu optimu. Generování náhodných řešení může být obtížné pro vysoce omezené problémy.

**Lokální prohledávání**

Nejstarší a nejjednodušší metaheuristika. Nahrazuje aktuální řešení zlepšujícím řešením.

Nejlepší zlepšující **x** první zlepšující **x** náhodný výběr.

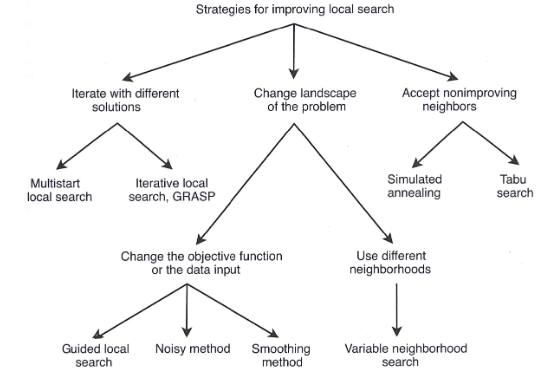
**Prohledávácí prostor**

Je definován orientovaným grafem *G = (S, E)*, kde množina vrcholů *S* odpovídá řešením problému definovaným jejich reprezentací a množina hran *E* je určena operátorem změny použitým pro generování sousedů z okolí. Sousedním vrcholem řešení v *G* jsou sousedé z okolí řešení.

**Krajina (vhodnosti)**

Je definována dvojicí*〈G, f〉*, kde *G* je prohledávací prostor a *f* účelová funkce, která řídí prohledávání.

**Pokročilé lokální prohledávání**



**Simulované žíhání**

Náhodný výběr souseda z okolí. Výběr lepšího souseda je vždy akceptován jako u hill climbing. Akceptuje i výběr horšího souseda. Na začátku prohledávání je povolena větší pravděpodobnost zhoršení vybraného řešení, postupně se povoluje menší a menší pravděpodobnost zhoršení vybraného řešení. Deterministická varianta: algoritmy založené na přijetí prahem.

**Tabu prohledávání**

Deterministické rozšíření základního algoritmu lokálního prohledávání + další algoritmus. Umožňuje akceptovat i horší řešení. Řešení horší/stejné kvality akceptováno, pokud neexistuje lepší. Cyklení je zabráněno prostřednictvím tabu seznamu. **Tabu seznam** je několik posledních změn, které jsou zakázány (typicky 5-9). **Aspirační kritérium** je podmínka, za které je možné realizovat změny z tabu seznamu.

**Iterativní lokální prohledávání**

Opakování lokálních prohledávání. Nalezené lokální optimum změněno a použito při dalším lokálním prohledávání. Lze použít libovolný algoritmus lokálního prohledávání.

**Řízené lokální prohledávání**

Opakovaná lokální prohledávání se změněnou hodnotou účelové fce. Snaha o prohledávání částí stavového prostoru s nižší cenou charakteristik.

**Hyper-heuristiky**

Prvotní myšlenka: heuristika na výběr heuristik. Rozšířeno na: prohledávácí metoda nebo učící mechanismus pro výběr nebo generování heuristik pro řešení obtížných prohledávácích problémů. Pracují nad prohledávacím prostorem heuristik (a ne přímo nad prohledávacím prostorem řešeného problému).

**Výběr heuristik**: snaha o nalezení vhodné sekvence aplikace existujících heuristik pro efektivní řešení problému.

**Generování heuristik**: snaha o vývoj nových heuristik na základě komponent známých heuristik, učení heuristik pomocí neuronových sítí, evolučních algoritmů…

**Metaheuristiky s populacemi**

Evoluční algoritmy, optimalizace mravenčí kolonie, optimalizace jedinců hejna, včelí úl, umělé imunitní systémy, odhad distribučními algoritmy. Základní rozdělení algoritmů podle využití paměti při prohledávání. Populace/generace -> uchovávaná množina řešení.

**Generování počáteční populace**

1. **Sekvenční diversifikace** – řešení generovaná postupně s maximální odlišností
2. **Paralelní diversifikace** – řešení generovaná nezávisle paralelně se snahou o celkovou maximální odlišnost řešení v populaci, může být obtížnější než řešení původního problému
3. **Heuristická inicializace** – jednotlivá řešení generovaná libovolnými heuristickými algoritmy (např. lokální prohledávání), nebezpečí v malé odlišnosti řešení v populaci

**Podmínky ukončení**

1. **Statická procedura** – konec prohledávání je předem znám, př.: pevný počet iterací, limit na CPU zdroje, maximální počet vyhodnocení účelové fce
2. **Adaptivní procedura** – konec prohledávání předem neznámý, př.: pevný počet iterací bez zlepšení, vypočítáno dostatečně kvalitní řešení, malá odlišnost řešení v populaci

**Evoluční algoritmy**

**Generické algoritmy** – významná role operátoru křížení, použití mutace

**Evoluční strategie** – většinou aplikovaný na spojité optimalizace s vektory reálných hodnot, křížení využito zřídka

**Evoluční programování** – spojitá optimalizace, menší použití pro velkou podobnost s evolučními strategiemi

**Genetické programování** – jedinci jsou programy, automatické generování programů řešících danou úlohu, př.: nalezení programu odpovídajícího dané matematické rovnici

**Strategie výběru**

**Výběr ruletovým kolem**

Nejpoužívanější strategie výběru. Ruletové kolo s díly pro všechny jedince v populaci. Problémy: příliš velká snaha vybírat kvalitní jedince -> předčasná konvergence. Řeší to pravděpodobností univerzální vzorkování.

**Pravděpodobnostní univerzální vzorkování**

Náhodně vybírá pouze první vzorek, všechny další vzorky se poté vybírají po obvodu ruletového kola v pravidelných intervalech 1/N. Garantuje, že vzorky s vahou alespoň *C*/N, budou v nové reprezentaci zastoupeny alespoň *c*-krát.

**Turnajový výběr**

Náhodný výběr *k* jedinců. Z těchto *k* jedinců je vybrán nejlepší jedinec.

**Výběr rankováním**

Pro každého jedince spočítáme rank a dle něj jsou vybírání jedinci. Rank může např. škálovat lineárně se závislostí ve snaze o výběr nejlepšího jedince. Rank použit pro výpočet pravděpodobnosti a aplikován stejně jako u ruletového kola.

**Mutace**

Operátor mutace mění jedince v populaci a způsobí jeho malou změnu.

V binární reprezentaci – prohození hodnoty binární proměnné

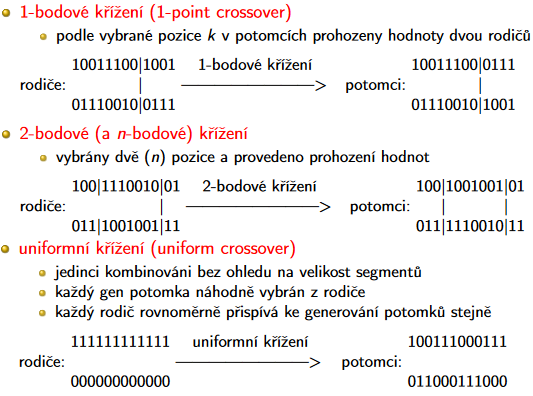
V diskrétní reprezentaci – změna hodnoty prvku za jinou hodnotu v abecedě

V permutacích – vložení, výměna, inverze hodnoty

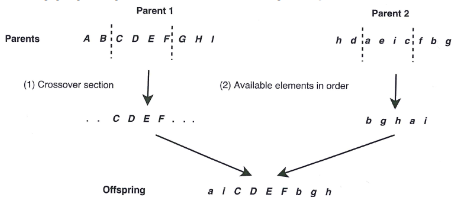
**Křížení**

Binární operátor. Cíl: zdědit vlastnosti rodičů potomkem.

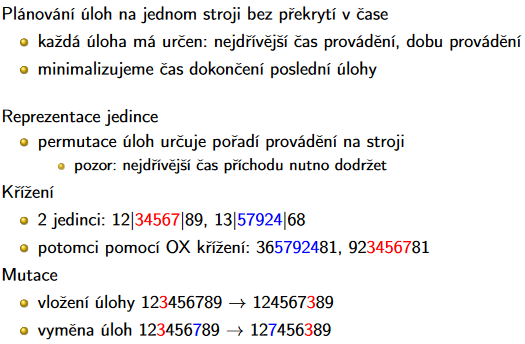
**Lineární reprezentace (vyjma permutací)**



**Reprezentace permutací**



**Cvičení:**



**Inteligence hejna**

Algoritmy inspirované skupinovým chováním druhů jako jsou mravenci, včely, vosy…

Původ v sociálním chování těchto druhů při hledání potravy.

**Optimalizace mravenčí kolonie**

Mravenčí kolonie je schopna najít nejkratší cestu mezi dvěma body. Mravenci během cesty nechávají na zemi chemickou stopu. Feromony vedou mravence k cíli. Feromony se postupně vypařují.

**Rozvrhování**

Rozhodování, jak zpracovat dané akce použitím omezených zdrojů a času.

**Plánování**

Rozhodování, jaké akce jsou potřeba pro dosažení daných cílů. Zabývá se volbou a organizací akcí, které mění stav systému. Cílem plánování je zjistit, jaké akce a na které stavy se mají aplikovat, abychom za dané situace dosáhli požadovaných cílů. Cílem plánování je cílový stav, splnění dané podmínky, optimalizace dané objektivní funkce.

**Systém** je konečný, plně pozorovatelný, deterministický a statický. **Cíle** jsou omezené. **Plány** jsou sekvenční. **Čas** je implicitní a plánujeme offline.

**Klasické plánování**

Stav systému je popsán množinou výroků. Každá akce je syntaktický výraz specifikující, jaké výroky musí patřit do stavu, aby na něj byla akce aplikovatelná a jaké výroky akce přidá nebo smaže, aby vytvořila nový stav.

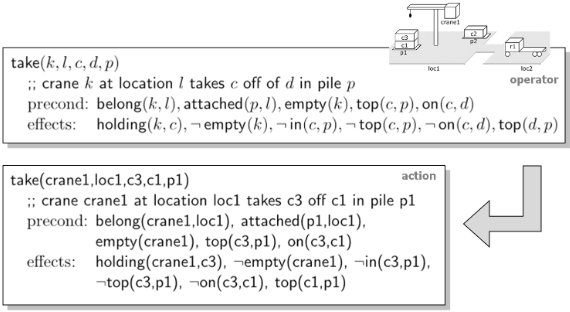
**Množinová reprezentace**

Srozumitelná – přehlednější než výčet stavů. Přechodová fce se snadno realizuje pomocí množinových operací. Některé množiny výroků neodpovídají žádnému stavu. Některé stavové prostory stejně mají obrovskou množinovou reprezentaci. **Stav** je množina výroků. **Akce** je trojice množiny výroků: akce, effects-, effects+.

**Klasická reprezentace**

Zobecňuje množinovou reprezentaci směrem k predikátové logice.

**Stavy** jsou množiny logických instanciovaných **atomů**, které jsou v dané interpretaci buď pravda nebo nepravda a je jich konečně mnoho. Pravdivostní hodnota některých atomů se mění. Atom, který není ve stavu explicitně uveden, neplatí.

**Akce** jsou reprezentovány plánovacími operátory, které mění pravdivostní hodnotu těchto atomů. Za proměnné jsou dosazeny konstanty. **Operátor** je trojice: jméno operátoru, předpoklady a důsledky. Předpoklady a důsledky jsou množiny literálů.

Vyjadřovací síla obou reprezentací je stejná. Při převodu z klasické na množinovou reprezentaci ale může dojít k exponenciálnímu nárůstu velkosti.

**Plánování ve stavovém prostoru**

Prohledávací prostor odpovídá stavovému prostoru plánovacího problému. Uzly odpovídají stavům, hrany odpovídají stavovým přechodům pomocí akcí a cílem je najít cestu mezi počátečním stavem a některým koncovým stavem. Problémem je velký větvící faktor a alternativní pořadí akcí vedoucí k neúspěchu.

**Dopředné prohledávání**

Začínáme v počátečním stavu a jdeme k některému s cílových stavů. Je potřeba umět rozhodnout, zda je daný stav cílový nebo ne, najít množinu akcí aplikovatelných na daný stav a vypočítat stav, do kterého se dostaneme aplikací akce.

Procedura dopředného plánování je **korektní**. Pokud vrátí nějaký plán, potom je řešením.

Procedura dopředného plánování je **úplná**. Pokud existuje plán, potom ho alespoň jedna z nedeterministických větví najde indukcí podle délky plánu. V každém kroku má algoritmus šanci zvolit správnou akci.

Algoritmus dopředného vyhledávání můžeme implementovat **deterministicky**: prohledávání do šířky, prohledávání do hloubky, A\* algoritmus.

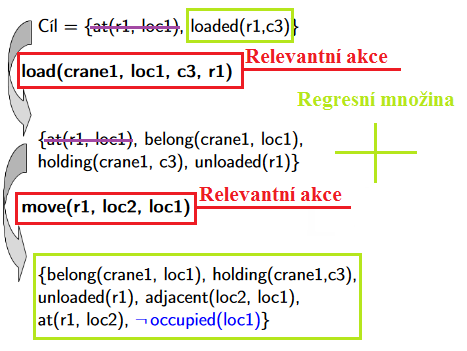
Problémem dopředného vyhledávání je **větvení**. To vadí u determinismu, který může ztrácet čas zkoušením irelevantních akcí. Řešením je heuristika doporučující výběr akce nebo ořezávání prohledávacího prostoru.

**Zpětné plánování**

Začínáme s cílem a jdeme přes podcíle k počátečnímu stavu. Je potřeba umět zjistit, zda daný stav odpovídá cíli, pro daný cíl najít relevantní akce a vypočítat podcíl umožňující aplikovat danou akci. Procedura zpětného plánování je **korektní a úplná**. Můžeme ji implementovat **deterministicky** – pro úplnost potřebujeme detekci cyklů. **Větvení** může být menší než u dopředného prohledávání, pořád ale zbytečně velké.

**Relevantní akce** - akce, která přispívá k cíli *g* a její efekty nejsou v konfliktu s *g*.

**Regresní množina** – množina předpokladů pro relevantní akci sjednocená s cílovou množinou mínus efekty po relevantní akci.



**STRIPS algoritmus**

Redukuje prohledávaný prostor zpětného plánování. Z podcílů řeší vždy jen část odpovídající předpokladům poslední přidané akce. Pokud aktuální stav splňuje všechny předpoklady operátorů, daný operátor se použije a tento závazek nebude rušen při backtrackingu. STRIPSneumí efektivně řešit Sussmanovu anomálii.

**Plánování v prostoru plánů**

Princip podobný **zpětnému plánování** ve stavovém prostoru. Začínáme z prázdného plánu, obsahujícího popis počátečního stavu a cíle. Přidáváme další akce, které plní dosud nesplněné cíle. Případně přidáváme vazby mezi již přítomnými akcemi. Přecházíme od jednoho částečného plánu k dalšímu, dokud nenajdeme úplný plán.

**Možné úpravy plánu**: přidání akce, svázání proměnných, přidání podmínky uspořádání, přidání kauzální vazby.

Počáteční plán i cíl zakódujeme jako speciální akce, které jsou v prvotním částečném plánu:

* akce *a0* reprezentuje počáteční stav tak, že nemá žádné předpoklady a počáteční stav je zakódován jako efekt; tato akce je před všemi ostatními akcemi
* akce *a∞* reprezentuje cíl, který je zakódován jako předpoklad, efekt akce je prázdný; takto akce je za všemi ostatními akcemi

Plánování bude založeno na odstraňování kazů v částečném plánu. Budeme přecházet od jednoho částečného plánu k dalšímu, dokud nenajdeme řešící plán.

**Uzly** prohledávaného prostoru jsou tvořeny částečnými plány.

**Částečný plán je čtveřice**:

* množina částečně iniciovaných plánovacích operátorů
* částečné uspořádání na první množinu
* množina vazeb
* množina kauzálních vztahů

**Kazy plánu:**

1. **Otevřený cíl** – předpoklad nějakého operátoru, o kterém zatím nebylo rozhodnuto, jak ho splnit. Lze je efektivně zjistit udržováním agendy předpokladů akcí. Řešení: najít operátor -> svázat proměnné -> vytvořit kauzální vazbu
2. **Hrozba** – jedná se o akci, která může porušit kauzální vazbu. Lze je najít prozkoumáním všech trojic akcí. Řešení: uspořádání nebo navázání proměnných.

Pro odstranění otevřených cílů a hrozeb se používají pouze konzistentní zjemnění plánu.

**Řešící plán**

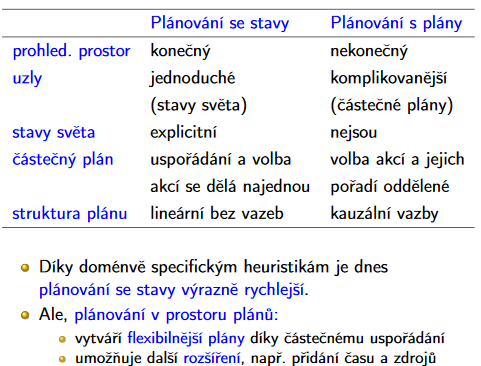
Částečný plán je řešící, pokud nemá žádné kazy a částečné uspořádání a množina vazeb jsou globálně konzistentní.

Algoritmus plánování v prostoru plánů je:

**Korektní** – pokud program skončí, pak v něm není žádný kaz a plán je konzistentní

**Úplný** – pokud existuje plán, má algoritmus vždy možnost vybrat ty správné akce

**Pozor na deterministickou implementaci -> prostor plánů není konečný.**

**Srovnání technik:**

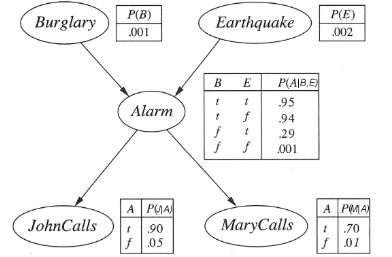
**Bayesovská síť**

Zachycuje závislosti mezi náhodnými proměnnými, pomocí orientovaného acyklického grafu. Každý uzel odpovídá náhodné proměnné, předchůdci uzlu v grafu se nazývají rodiče a každý uzel má přiřazenou tabulku podmíněné pravděpodobnostní distribuce.

Náhodné **boolovské proměnné** reprezentují možné události – některé události ignorujeme. **Pravděpodobnostní tabulky** reprezentují vztah podmíněné pravděpodobnosti – stačí reprezentovat hodnoty true.

Bayesovská síť kompaktním způsobem reprezentuje úplnou sdruženou distribuci.

**Příklad:**





**Konstrukce sítě**

Síť je z principu konstrukce acyklická a neobsahuje redundantní informaci, a tudíž je vždy konzistentní. Síť může být mnohem **kompaktnější** než úplná sdružená distribuce, pokud je síť řídká. Náhodná proměnná často závisí jen na omezeném počtu jiných proměnných. Můžeme ignorovat **slabé vazby**, čímž budeme mít menší přesnost reprezentace, a reprezentace bude kompaktnější. Kompaktnost sítě hodně závisí na vhodném uspořádání proměnných.

**Uzly**

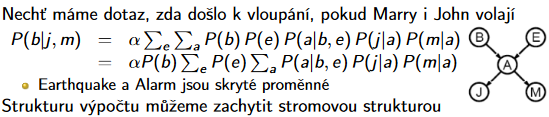
Rozhodněte, jaké náhodné proměnné jsou potřeba a uspořádejte je. Funguje libovolné uspořádání, ale pro různá uspořádání dostaneme různě kompaktní sítě. Doporučené uspořádání je takové, kdy příčiny předcházejí efekty.

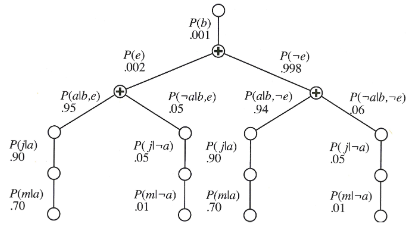
**Hrany**

Bereme proměnné v daném pořadí od *1* do *n*. Z rodičů vedeme hranu do proměnné. Vypočteme podmíněné pravděpodobnostní tabulky.

**Odvozování enumerace**

Některé části výpočtu se opakují. Enumerace se provádí shora dolů násobením hodnot po každé cestě a sečítáním hodnot v „+“ uzlech.





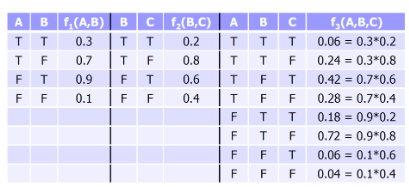
**Řešení jednotlivých složek:**

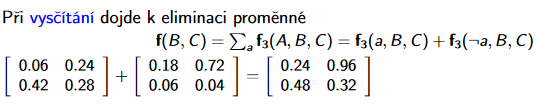
**Obecné řešení:**

Alfa je konstanta, která "upravuje" složky tak, že jejich součet bude roven 1.

**Eliminace proměnných**

Vyhodnocení provedeme zprava doleva. Násobení činitelů je násobení po prvcích. Vysčítáním činitelů eliminujeme příslušnou proměnnou. Na závěr provedeme normalizaci.





**Vzorkovací metody**

**Přímé vzorkování**

Vzorek pro nás bude ohodnocení náhodných proměnných. Vzorek je potřeba generovat tak, aby odpovídal tabulkám v Bayesovské síti. **Uzly** bereme v topologickém uspořádání. **Ohodnocení rodičů** nám dá pravděpodobnostní distribuci hodnot aktuální náhodné proměnné. Náhodně vybereme hodnotu podle této distribuce.

**Vzorkování se zamítáním**

Ze vzorků, které vygenerujeme, vezmeme jen ty, které jsou kompatibilní s uzlem, který nás zajímá. Ostatní zamítneme. Hlavní nevýhoda metody je zamítání příliš mnoha vzorků.

**Vážení věrohodnosti**

Místo zamítání vzorků je efektivnější generovat pouze vzorky vyhovující pozorování *e*. Zafixujeme hodnoty z pozorování *e* a vzorkujeme pouze ostatní proměnné.

**Přechodový model**

Popisuje, jak je stav ovlivněn předchozími stavy. Používá se **Markovský předpoklad** – současný stav závisí pouze na pevně daném konečném počtu předchozích stavů a **předpoklad stacionárního procesu** – stav se vždy mění podle stejných pevně daných pravidel.

**Senzorický model**

Popisuje, na čem závisí pozorované náhodné proměnné. Ty mohou záviset i na proměnných z předchozích stavů, ale uděláme **Markovský senzorický předpoklad** – pozorované proměnné závisí pouze na nepozorovatelných proměnných ze stejného stavu.

Přechodový a senzorický model můžeme popsat Bayesovskou sítí.

Markovský proces prvního řádu předpokládá, že stavové proměnné obsahují veškerou informaci pro charakteristiku pravděpodobnostní distribuce dalšího stavu. Pokud je tento předpoklad nepřesný, můžeme zvýšit řád Markovského procesu nebo můžeme rozšířit množinu stavových proměnných.

**Filtrace/odhad stavů**

Cílem je zjištění pravděpodobnosti aktuálního stavu na základě dosavadních pozorování. Dobrý filtrační algoritmus odhaduje aktuální stav z odhadu předchozího stavu a aktuálního pozorování.



**Predikce**

Cílem je zjištění pravděpodobnosti budoucího stavu základě dosavadních pozorování. Jedná se v podstatě o filtraci bez přidávání dalších pozorování. Po určité době konverguje předpovězená distribuce ke stacionární distribuci a nadále zůstane stejná.



**Vyhlazování**

Cílem je zjištění pravděpodobnosti minulého stavu na základě dosavadních pozorování. Použijeme rekurzivní předávání zpráv ve dvou směrech.



**Nejpravděpodobnější průchod**

Na základě posloupnosti pozorování chceme zjistit nejpravděpodobnější posloupnost stavů, která tato pozorování generuje.



**Teorie pravděpodobnosti** – říká, čemu máme věřit na základě pozorování

**Teorie užitku** – popisuje, co chceme a jak máme ohodnotit rozhodnutí

**Teorie rozhodování** – spojuje obě teorie dohromady a popisuje, jak bychom měli vybrat akci s největším očekávaným užitkem.

**Teorie užitku**

Preference lze zaznamenat **fci užitku**, která mapuje stavy na reálné číslo. Očekávaný užitek potom spočteme jako průměrný užitek přes všechny možné stavy vážené pravděpodobností výsledku.

**Víceatributová fce užitku**

Budeme uvažovat, že každý atribut má definované preferované pořadí hodnot, vyšší hodnoty odpovídají lepšímu řešení.

**Rozhodovací síť**

Popisuje vztahy mezi vstupy, rozhodnutími a užitkem.

**Náhodné uzly** – reprezentují náhodné proměnné stejně jako v Bayesovské síti.

**Rozhodovací uzly** – popisují rozhodnutí, které může agent udělat

**Uzly užitku** – popisují funkci užitku

**Markovský rozhodovací proces**

Sekvenční rozhodovací problém v plně pozorovatelném stochastickém prostředí s Markovským přechodovým modelem a aditivní funkcí užitku. Tvořen množinou stavů, množinou akcí v každém stavu, přechodovým modelem a oceněním.

Ve stochastickém prostředí nemůže pracovat s pevným pořadím akcí. Protože předem nevíme, kam akce agenta zavede, potřebujeme v každém stavu vědět, co dělat. Hledáme strategii s největším očekávaným užitkem – **optimální strategie** – maximalizuje očekávaný užitek.

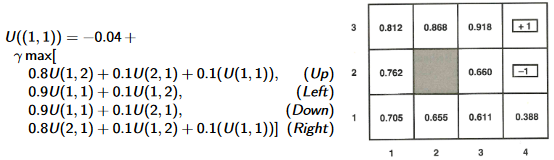
**Konečný horizont** – máme daný pevný termín a po něm už na ničem nezáleží

**Nekonečný horizont** – není důvod se ve stejném stavu chovat v různé časy různě. Optimální politika je stacionární. To neznamená nutně nekonečné posloupnosti stavů, pouze zde není žádný deadline.

**Užitek v čase**

**Kumulovaná funcke užitku**

Pro světy bez cílového stavu a nekonečný horizont by aditivní fce užitku byla problémová. Užitek v kumulované funkci užitku je konečný. Pokud má prostředí cílový stav, do kterého se agent garantovaně může dostat, nebudeme potřebovat pracovat s nekonečnými posloupnostmi. U nekonečné posloupnosti lze porovnat průměrné ocenění.

**Bellmanova rovnice**

Užitek stavů přímo závisí na užitku okolních stavů.

**Robot**

Fyzický agent, který vykonává úlohy manipulací s fyzickým světem. Pro realizaci úloh je robot vybaven **efektory** – nohy, klouby, kola… Účelem efektoru je uplatnění fyzických sil na prostředí. Dále je robot vybaven **senzory** – kamery, lasery, … které umožňují vnímání.

**Manipulátor** – fyzicky připevněné k pracovnímu místu – robotická ruka.

**Mobilní robot** – pohybují se v prostředí pomocí kol, nohou apod. Př.: bezpilotní pozemní vozidlo pro autonomní jízdu, planetární vozidlo, bezpilotní letadlo, bezpilotní ponorka

**Mobilní manipulátor** – kombinace mobility s manipulací – humanoidní roboti

**Robotický hardware**

**Senzory**

Představují rozhraní mezi robotem a prostředím. Senzory dělíme v závislosti na tom, zda jsou určeny pro vnímání prostředí (dálkoměry), robotova umístění (lokační senzory) nebo robotovy vnitřní konfigurace.

**Pasivní senzory** – slouží k pozorování prostředí, kdy zachycují signály generovaným dalšími zdroji v prostředí.

**Aktivní senzory** – vysílají energii do prostředí, založeny na tom, že energie je reflektována senzoru zpět, poskytují více informací než pasivní senzory, náročnější na energii než pasivní senzory, hrozí nebezpečí interference při použití více aktivních senzorů zároveň.

**Efektory**

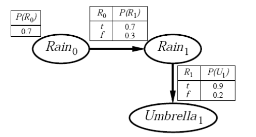
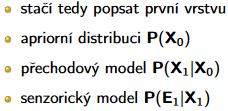
Umožňují robotovi pohyb a změnu tvaru těla.

**Stupeň volnosti** – umožňuje pochopit návrh efektoru, jeden stupeň volnosti pro každý nezávislý směr, ve kterém se robot nebo jeden z jeho efektorů může pohybovat.

**Vnímání**

Proces, kdy robot mapuje měření senzorů na vnitřní reprezentaci prostředí. Obtížné, protože senzory jsou nepřesné, prostředí je částečně pozorovatelné, nepředvídatelné a často dynamické. Robot řeší problémy filtrace/odhadu stavu, kdy je cílem zjištění pravděpodobnosti aktuálního stav na základě dosavadních pozorování. Robotovo vnímání můžeme chápat jako temporální odvozování z posloupnosti akcí a pozorování.

**Dynamické Bayesovské sítě**

Reprezentuje temporální pravděpodobnostní model. Skládá se z opakujících se vrstev proměnných. Dle Markovských předpokladů má každá proměnná rodiče buď ve stejné nebo předchozí vrstvě.

**Plánování pohybu robota**

**Pohyb z bodu do bodu** – přesun robota do cíle

**Koordinovaný pohyb** – robot se přesunuje v kontaktu s předmětem

**Konfigurační prostor** – používá se pro reprezentaci plánovacího problému, je to prostor stavů robota definovaný jeho umístěním, orientací a úhly kloubů, vhodnější reprezentace než původní 3D prostor.

**Plánování cesty** – problém nalezení cesty z jedné konfigurace do druhé v konfiguračním prostoru, na rozdíl od klasického hledání cesty v diskrétním prostoru se pohybuje ve spojitém prostoru.

**Reprezentace pracovního prostoru**

Souřadnice určují reprezentaci pracovního prostoru – reálného prostoru, kdy jsou souřadnice robota zadány stejným souřadným systémem jako objekty, kterými manipulujeme. Reprezentace vhodná pro kontrolu kolizí zejména robota a objektů zadaných jednoduchými polygonálními modely. Problémy: všechny souřadnice pracovního prostoru nejsou dosažitelné, generování cest, které dodržují tato omezení je velmi náročné.

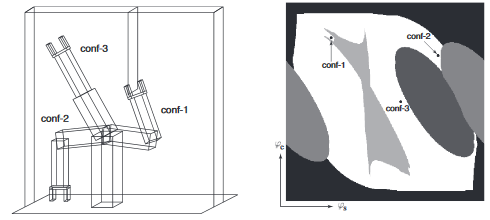
**Reprezentace konfiguračního prostoru**

Reprezentace pomocí robotových kloubů.

**Plánování cesty**: spojíme aktuální pozici a cíl rovnou čarou a robot se po ní pohybuje konstantní rychlostí, dokud nedorazí do cíle

**Problém**: úloha robota je většinou zadána v pracovním, a ne v konfiguračním prostoru.

**Kinematika**: transformace souřadnic z konfiguračního prostoru do pracovního prostoru je jednoduchá – lineární transformace pro otočné klouby

**Inverzní kinematika**: opačná transformace – obtížná, zřídka jedno řešení, např. pozice chapadla umožňují dvě konfigurace

Pracovní prostor Konfigurační prostor

**Buňková dekompozice**

Dekompozice volného prostoru na konečný počet spojitých regionů nazývaných buňky. Plánování cesty redukováno na prohledávání v diskrétním grafovém prostoru. Základní grafová dekompozice: pravidelná mřížka – úroveň šedi buňky určuje hodnotu buňky = cenu nejkratší cesty do cíle. Hodnoty spočítáme deterministickou verzí algoritmu iterací hodnot nebo A\* algoritmem.

**Potenciální pole**

Dodatečná cenová fce. Hodnota potenciálu pole narůstá ze vzdáleností od nejbližší překážky. Potenciál pole je pak využit jako dodatečná cenová fce při výpočtu nejkratší vzdálenosti.

**Skeletonizace**

Alternativní přístup pro převedení plánování cesty na prohledávání v diskrétním grafovém prostoru. Robotův volný prostor je redukován na jednodimenzionální reprezentaci nazvanou **skeleton** = Voroného diagram – množina všech bodů, které mají stejné vzdálenosti od dvou nebo více překážek. Plánování cesty redukováno na hledání nejkratší cesty ve Voroného diagramu.

**Dynamický stav**

Přechodový model pro dynamický stav zahrnuje účinek sil na rychlost změny. Nutné modely vyjádřené diferenciálními rovnicemi, které provážou kvantitu s odpovídající změnou v čase. Pokud bychom uměli generovat takové plány, roboti by měli skvělý výkon. Dynamický stav má však více dimenzí než kinematický, plánovací algoritmy by byly použitelné pouze pro nejjednodušší roboty. Robotické systémy v praxi se spoléhají na jednodušší kinematické plánovače cest.

**Regulátor**

Kompenzuje limitace kinematických plánů, aby byla zachována cesta.

**Referenční regulátor** – snaží se robota udržet na naplánované cestě.

**Proporční derivační regulátor** – nejjednodušší regulátor, který zachovává striktní stabilitu v naší doméně

**Integrační proporční derivační regulátor** – umožňuje odstranit chybové chování PD regulátoru v některých případech. Chybové hlášení jako důsledek systematické vnější síly, která není součástí modelu. Dlouhotrvající odchylky mezi referenčním a aktuálním stavem opraveny. Regulátor tak řeší systematické chyby na úkor zvýšeného nebezpečí oscilací

**Další způsoby řízení pohybu**

**Řízení pomocí potenciálu pole** – podobně jako při plánování pohybu

**Reaktivní řízení** – můžeme specifikovat regulátor bez explicitního modelu prostředí, nejprve si určíme vzor posunu končetin a pak už reaktivně reagujeme na daný stav

**Zpětnovazební učení**