

- ©Jan Schmidt 2011
Katedra číslicového návrhu
Fakulta informačních technologií
České vysoké učení technické v Praze
- Zimní semestr 2013/14



EVROPSKÁ
UNIE

EVROPSKÝ SOCIÁLNÍ FOND
PRAHA & EU: INVESTUJEME
DO VAŠÍ BUDOUCNOSTI

MI-PAA

7. Lokální metody

- Princip lokálních metod
- Stavový prostor (state space) a jeho graf
- Strategie pohybu stavovým prostorem
- Prostor prohledávání (search space)
- Exaktní metody
- Základní heuristické metody

Problém batohu

Jsou dána přirozená čísla n , M , c_1, c_2, \dots, c_n , w_1, w_2, \dots, w_n . Nalezněte čísla x_1, x_2, \dots, x_n z množiny $\{0,1\}$ tak, aby

$$\sum_{i=1}^n x_i w_i \leq M \qquad \sum_{i=1}^n x_i c_i = \max.$$

Je dáno n věcí, i -tá věc má váhu w_i a cenu c_i , dále batoh s nosností M . Nalezněte takovou sestavu věcí v batohu, aby nebyl přetížen a cena věcí byla maximální.

Instance a konfigurace

instance $n=3$, $M=6$, $C=\{10, 20, 30\}$, $W=\{2, 3, 5\}$

všechny
konfigurace

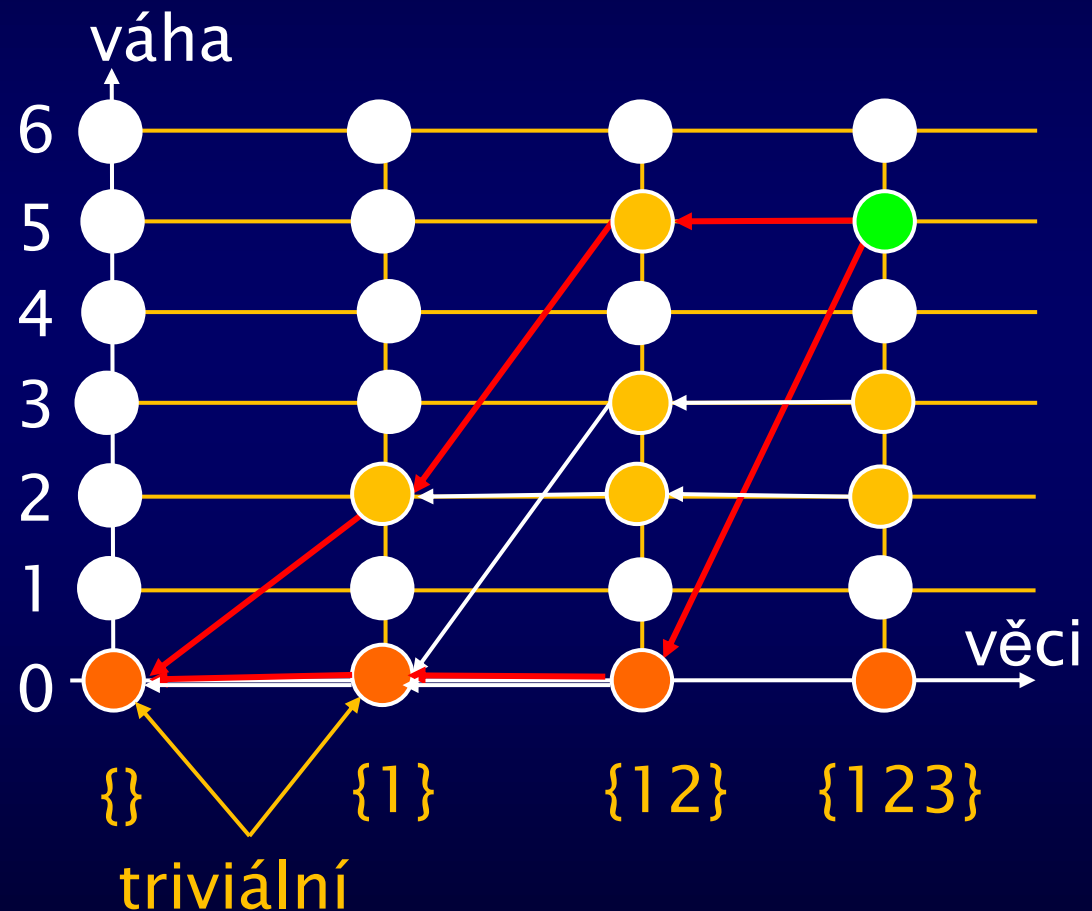
	x_1	x_2	x_3	$\sum x_i c_i$	$\sum x_i w_i$	
	0	0	0	0	0	✓ triviální
	0	0	1	30	5	✓ ✓
	0	1	0	20	3	✓
	0	1	1	50	8	✗
řešení ✓	1	0	0	10	2	✓
	1	0	1	40	7	✗
optimální řešení ✓	1	1	0	30	5	✓ ✓
	1	1	1	60	10	✗

Dílčí instance při řešení instance problému batohu dynamickým programováním

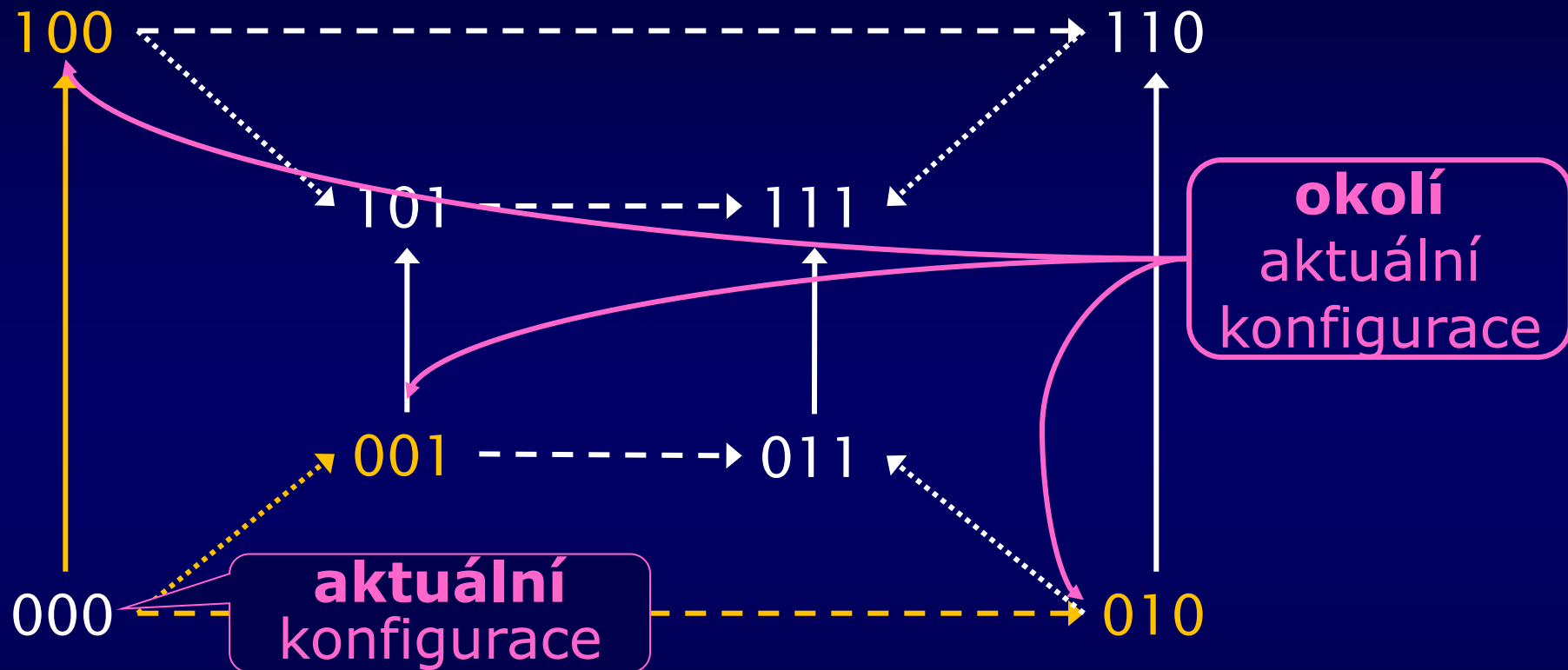
instance $n=3$, $M=6$, $C=\{10, 20, 30\}$, $W=\{2, 3, 5\}$

Řešení
zadané instance
konstruujeme
z řešení dílčích
instancí

Princip
globálních metod



Konfigurace instance problému batohu při řešení hladovým algoritmem



Věnujeme se jedné (aktuální) konfiguraci a vybíráme příští z jejích sousedů

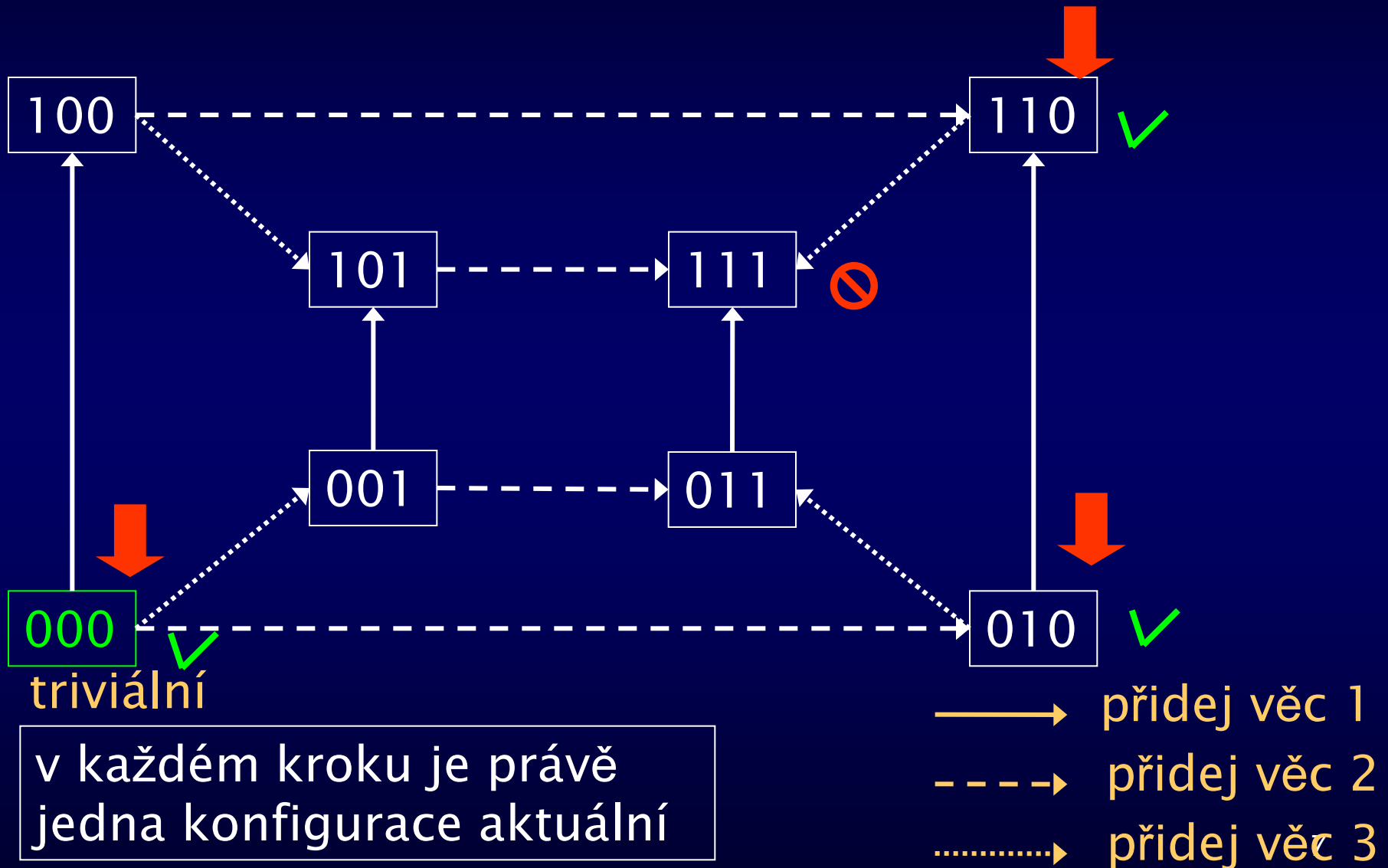
Princip **lokálních metod**

—→ přidej věc 1
- - -→ přidej věc 2
.....→ přidej věc 3

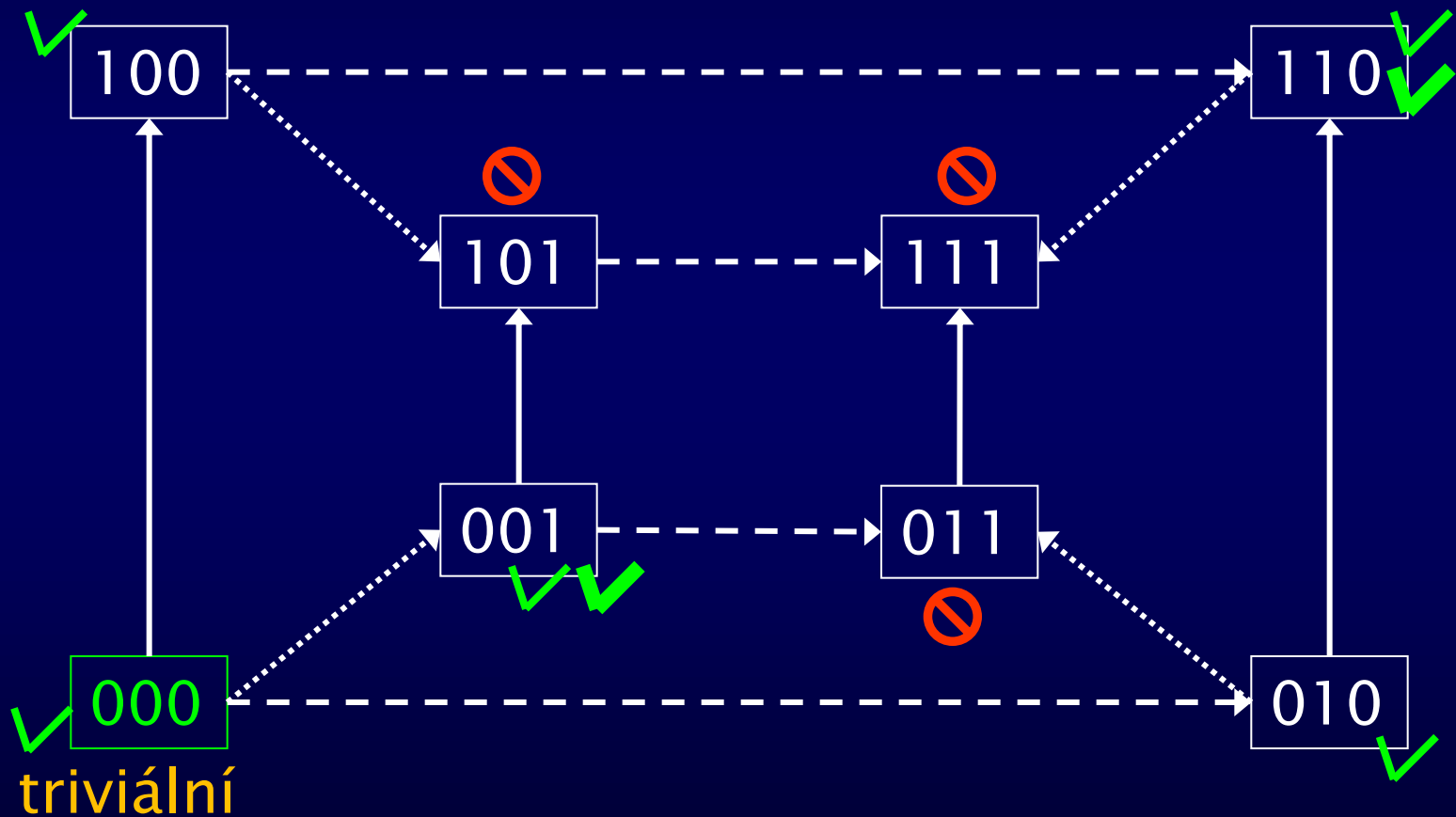
Stavový prostor (state space) a strategie pohybu v něm

- Stavový prostor, graf
- Strategie pohybu stavovým prostorem
- Prostor prohledávání

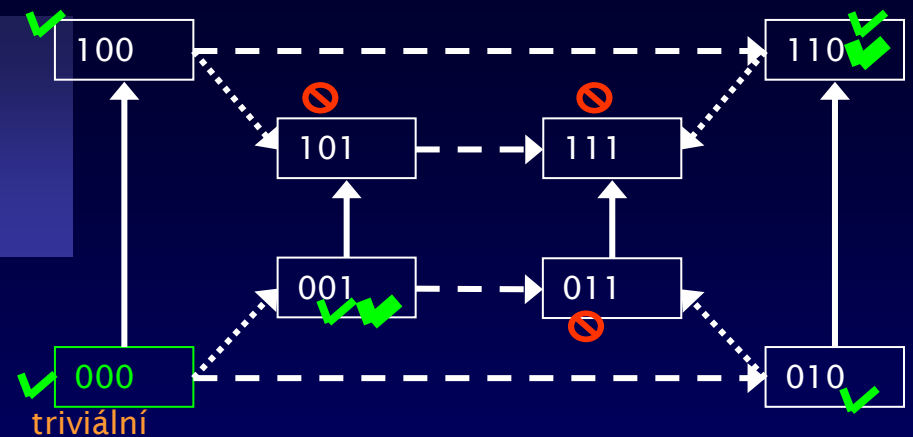
Průchod hladovým algoritmem



Hodnocení konfigurací



Vlastnosti grafu



- Vloženou věc nelze vyjmout
- V grafu **nelze zabloudit**
- Následník každé konfigurace, pokud existuje, má větší celkovou váhu i cenu
- Následník každé nepřípustné konfigurace je nepřípustná konfigurace
- **Odráží charakteristiky problému i algoritmu, který jej řeší**

Stavový prostor

state space

Typicky: učící
se algoritmy



- **Definice: stav**

Nechť $X = \{x_1, x_2, \dots, x_n\}$ jsou konfigurační proměnné problému Π . Nechť $Z = \{z_1, z_2, \dots, z_m\}$ jsou vnitřní proměnné algoritmu A řešícího instanci I problému Π . Pak každé ohodnocení s proměnných $X \cup Z$ je stav algoritmu A řešícího I .

- **Definice: stavový prostor**

Nechť $S = \{s_i\}$ je množina všech stavů algoritmu A řešícího I . Nechť $Q = \{q_j\}$ je množina operátorů $S \rightarrow S$ takových, že $q_j(s_i) \neq s_i$ pro všechna s_i, q_j . Pak dvojici (S, Q) nazveme stavovým prostorem algoritmu A řešícího I .

Graf stavového prostoru

- Formální obrat, který dovolí přenést grafovou terminologii a algoritmy na stavový prostor
- **Definice: akce**  
Nechť $s \in S$ je (jeden konkrétní) stav a $q \in Q$ operátor. Pak aplikace q na s se nazývá akce.
- **Definice: graf stavového prostoru**
Nechť (S, Q) je stavový prostor algoritmu řešícího instanci problému. Pak orientovaný graf $H = (S, E)$, kde hrana $(e_i, e_j) \in E$ odpovídá akci $s_j = q(s_i)$ pro $q \in Q$ se nazývá grafem stavového prostoru algoritmu.

Okolí stavu, sousední stav

kam se dostanu jedním krokem

- **Definice:**

okolí stavu $s \in S$ je množina stavů, dosažitelných z s aplikací některé operace $q \in Q$.

kam se dostanu nejvýše k kroky

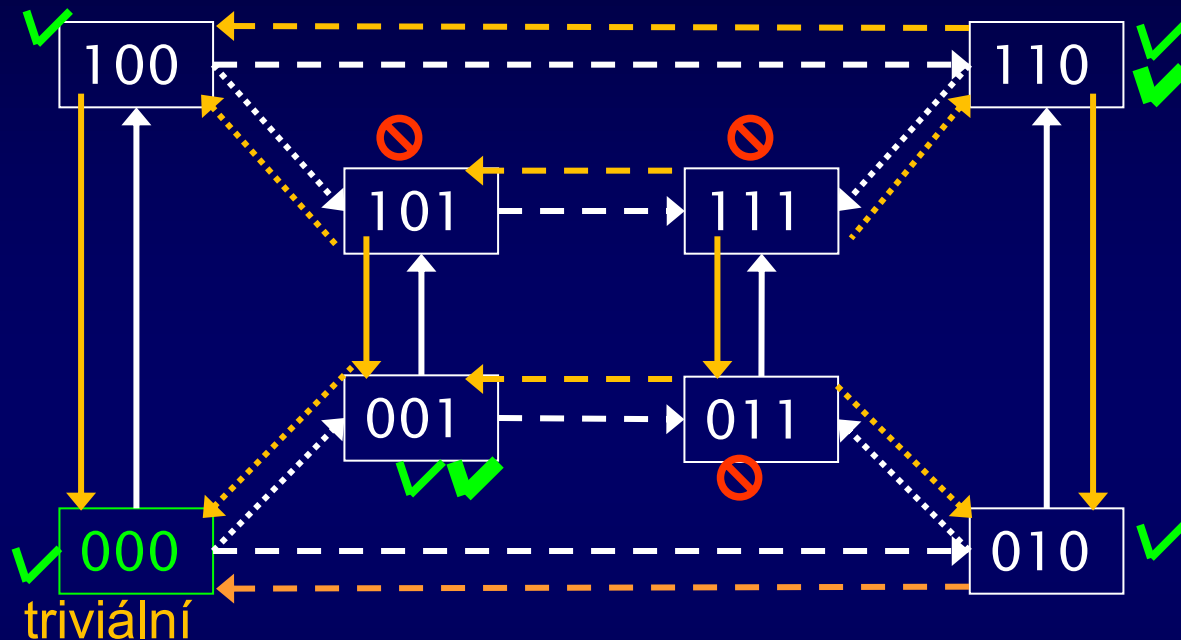
- **Definice:**

k -okolí stavu $s \in S$ je množina stavů, dosažitelných z s aplikací nejméně jedné a nejvýše k operací $q \in Q$.

- **Definice:**

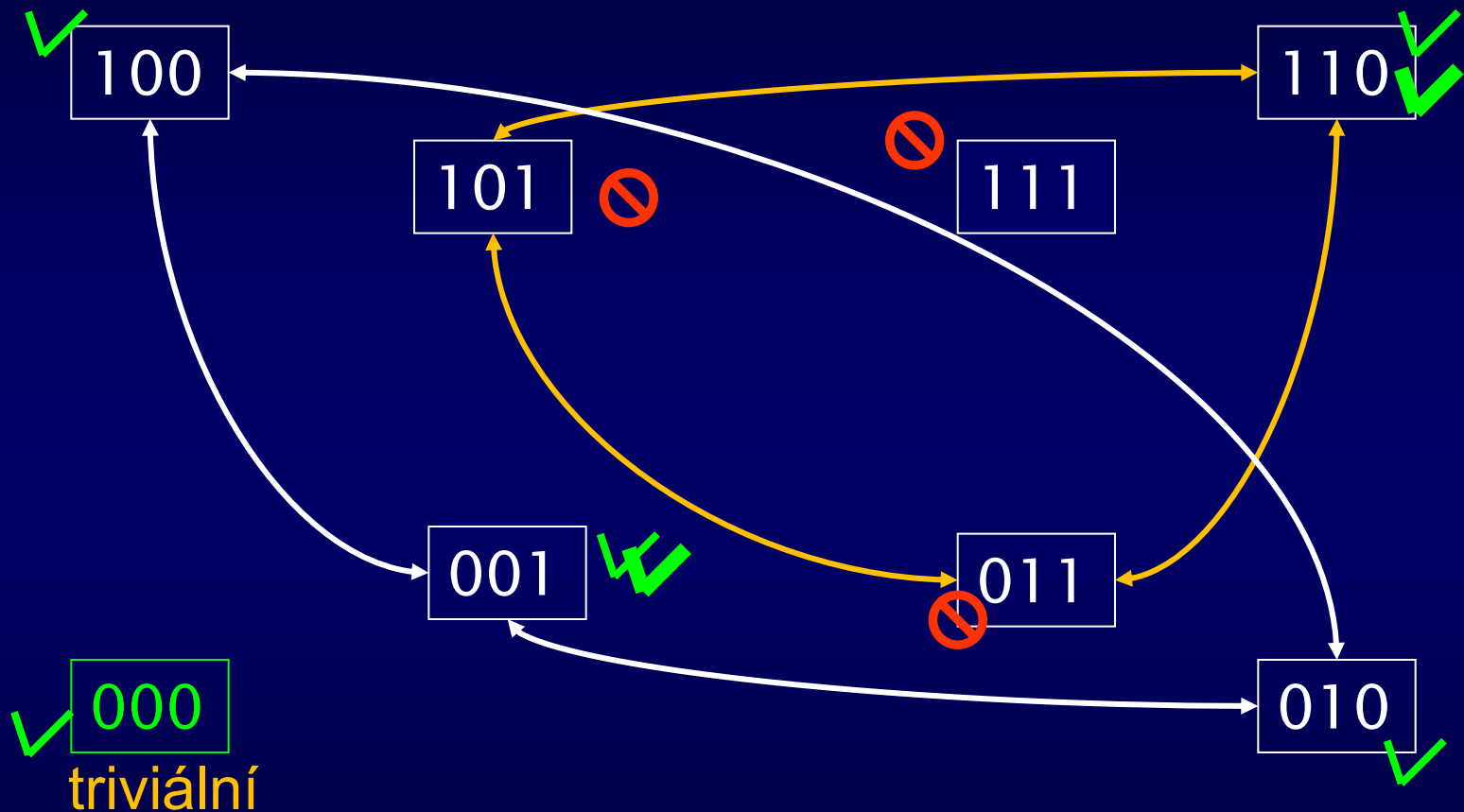
stavy z okolí stavu $s \in S$ se nazývají sousední stavy (sousedé) stavu s .

Inverzní operátory



- libovolnou vloženou věc lze vyjmout
- v grafu lze libovolně dlouho **bloudit**
- o následníku daného stavu nelze nic říci

Výměny



Stavový prostor je nesouvislý a proto bez doplnění dalšími operátory neužitečný.

Problém obchodního cestujícího (TSP)

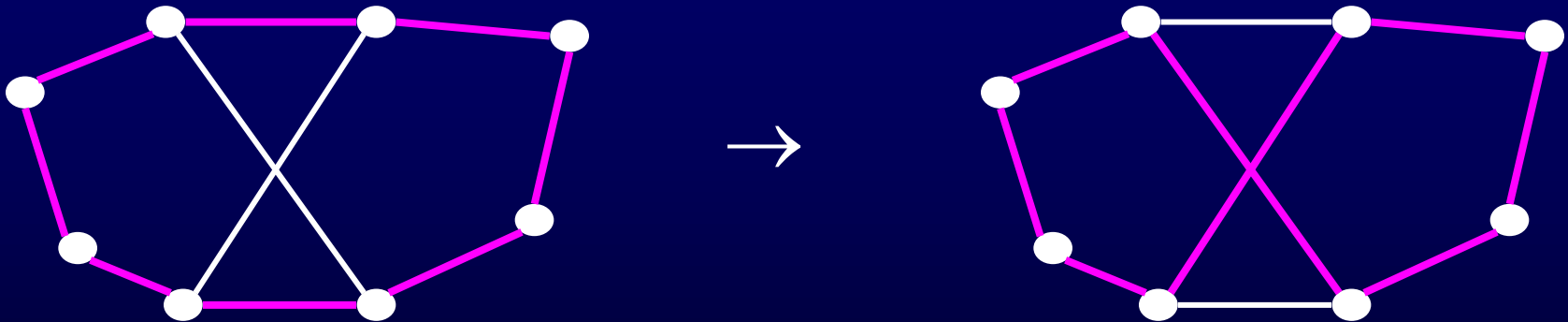
Dána množina n měst $C=\{c_1, c_2, \dots, c_n\}$. Pro každá dvě města c_i, c_j je dána vzdálenost $d(c_i, c_j) > 0$. Nalezněte uzavřenou túru, která prochází každým městem právě jednou a má nejmenší délku.

Dán graf $G=(V, E)$. Každá hrana (v_i, v_j) je ohodnocena vzdáleností $d(v_i, v_j)$. Nalezněte Hamiltonovu kružnici s minimálním součtem ohodnocení hran.

... optimalizační konstruktivní verze

Stavový prostor TSP

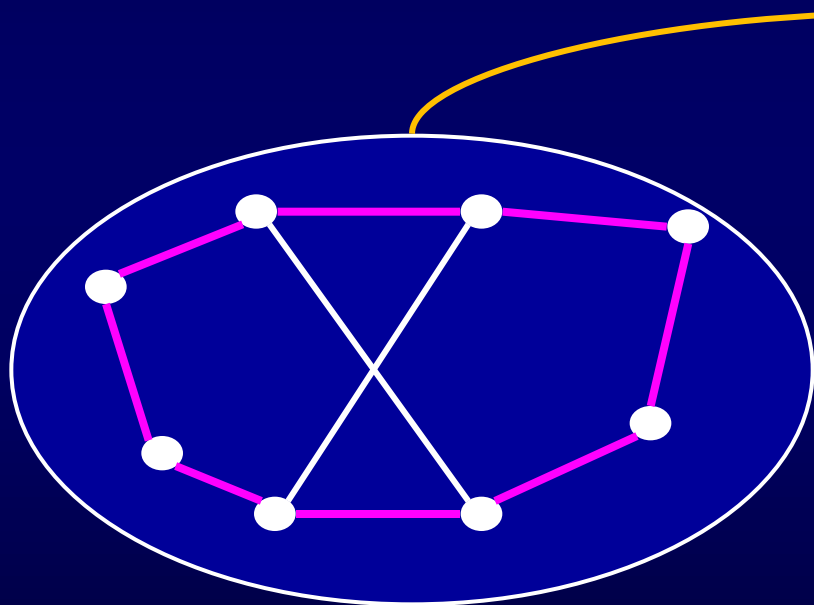
- **Konfigurace:**
obecně podgraf grafu G (v závislosti na algoritmu kružnice, H. kružnice, cesta...)
- Uzel stavového grafu = podgraf G
- **Operátor:**
např. dvojzáměna na hranách



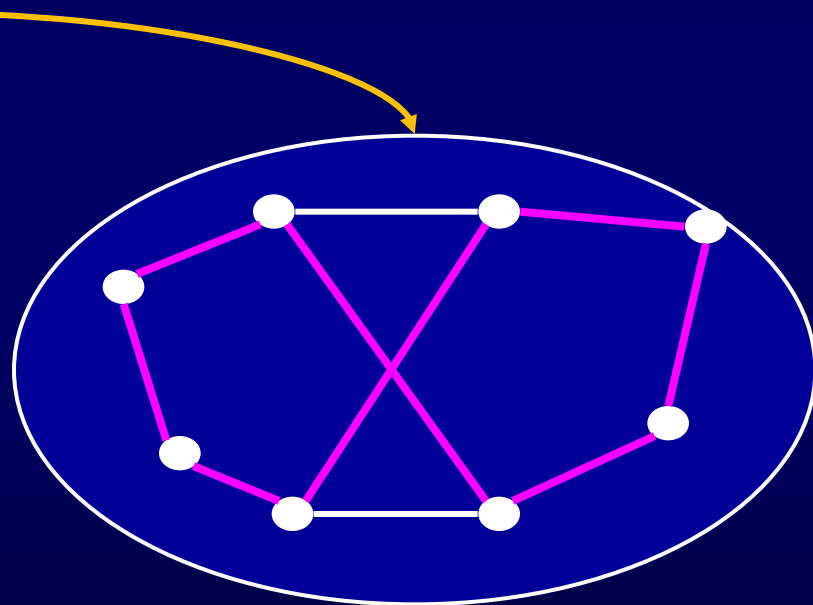
Hrana stavového grafu = dvojzáměna nad G

Stavový prostor TSP

Hrana stavového grafu



Uzel stavového grafu



Uzel stavového grafu

Stav nebo cesta?

- Dosud probrané pojmy: úkolem je nalézt **stav**, který odpovídá (optimálnímu) řešení
- Některé úlohy (zejména UI): úkolem je nalézt **cestu** ke stavu (např. robotická manipulace scény)
- Množina stavů je tedy množina **posloupností akcí**
- Operace nad stavem typu „**přidej na konec**“
- Graf stavového prostoru potom „**vypadá stejně**“ jako graf možných manipulací
- Typický příklad této situace: Stavový prostor 8-problému



BI-GRA 11 snímek 5

Strategie pohybu stavovým prostorem



BI-GRA 11

- úplná, systematická
- do hloubky, do šířky, nejlepší nejdříve
- metoda větví a hranic (branch and bound)

nepřipadá Vám to
povědomé?

Pohyb stavovým prostorem

- **Aktuální** stav, konfigurace příslušející aktuálnímu stavu
- **Transformace** aktuálního stavu pomocí operátorů (→pohyb)
- Transformace nutno řídit →
→ **strategie prohledávání**
- Charakteristika algoritmu
 - stavový prostor
 - strategie prohledávání a ukončení
 - strategie prořezávání

Strategie

- **Úplná strategie:** navštívit všechny stavy kromě těch, o kterých víme, že nedávají (optimální) řešení

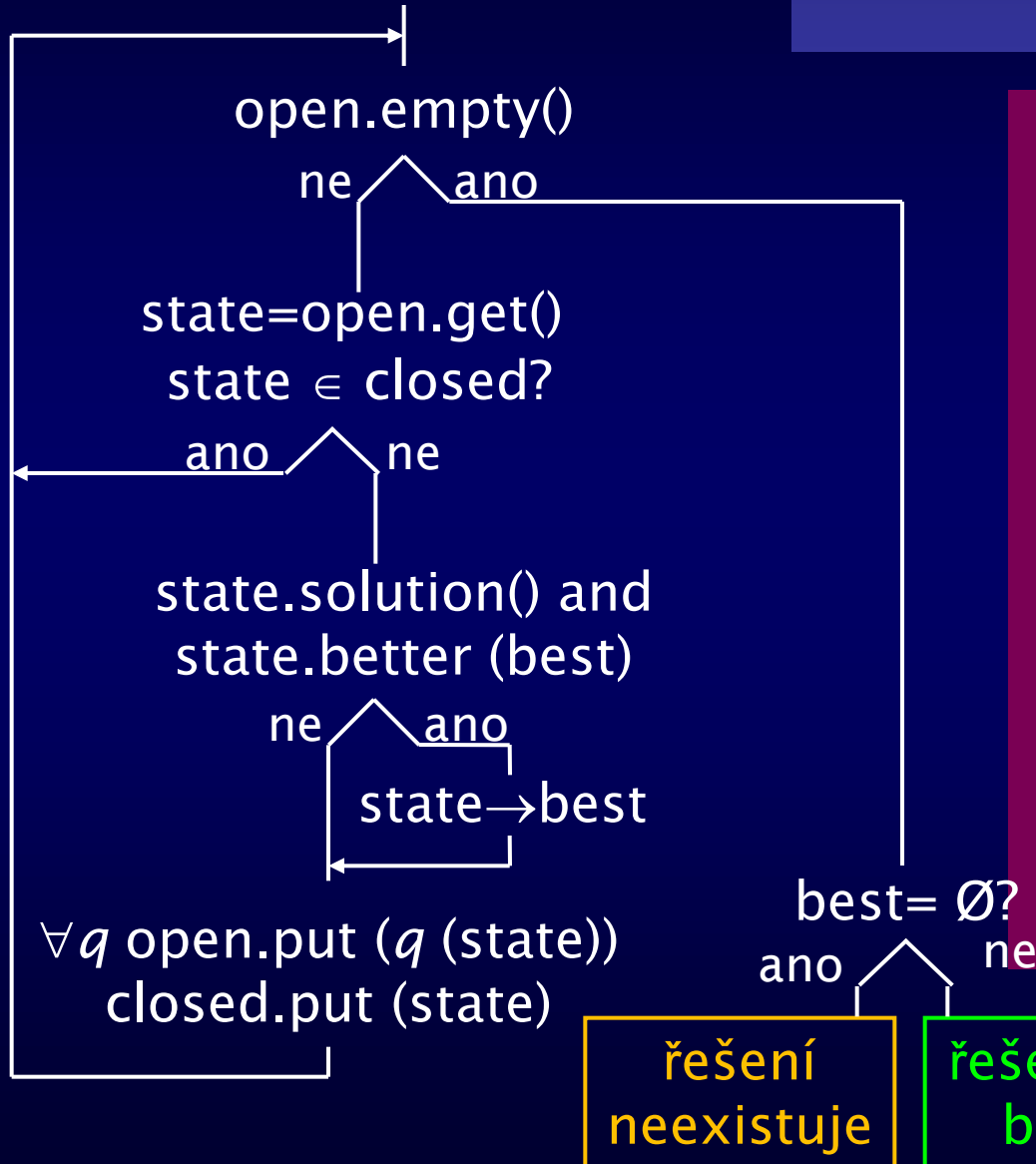
Do not leave any stone unturned, unless you are sure there is nothing under it (Pearl)

- **Systematická strategie:** úplná, navštívit každý stav nejvýše jednou

Do not turn any stone more than once (Pearl)

{počáteční stav} → open
∅ → closed
∅ → best

Typická systematická strategie



- open, closed: množiny stavů
- open: neprozkoumaní sousedé, o kterých víme
- closed: prozkoumané stavy
- best, state: stavy, ∅ znamená „žádný stav“
- metoda better srovnává optimalizační kritéria; každý stav je lepší než ∅
- zajímá nás cena stavu, nikoli cesty k němu – na rozdíl od prohledávání uspořádaným výběrem

◀ BI-GRA 11 snímek 9

{počáteční stav} → open
∅ → closed
∅ → best

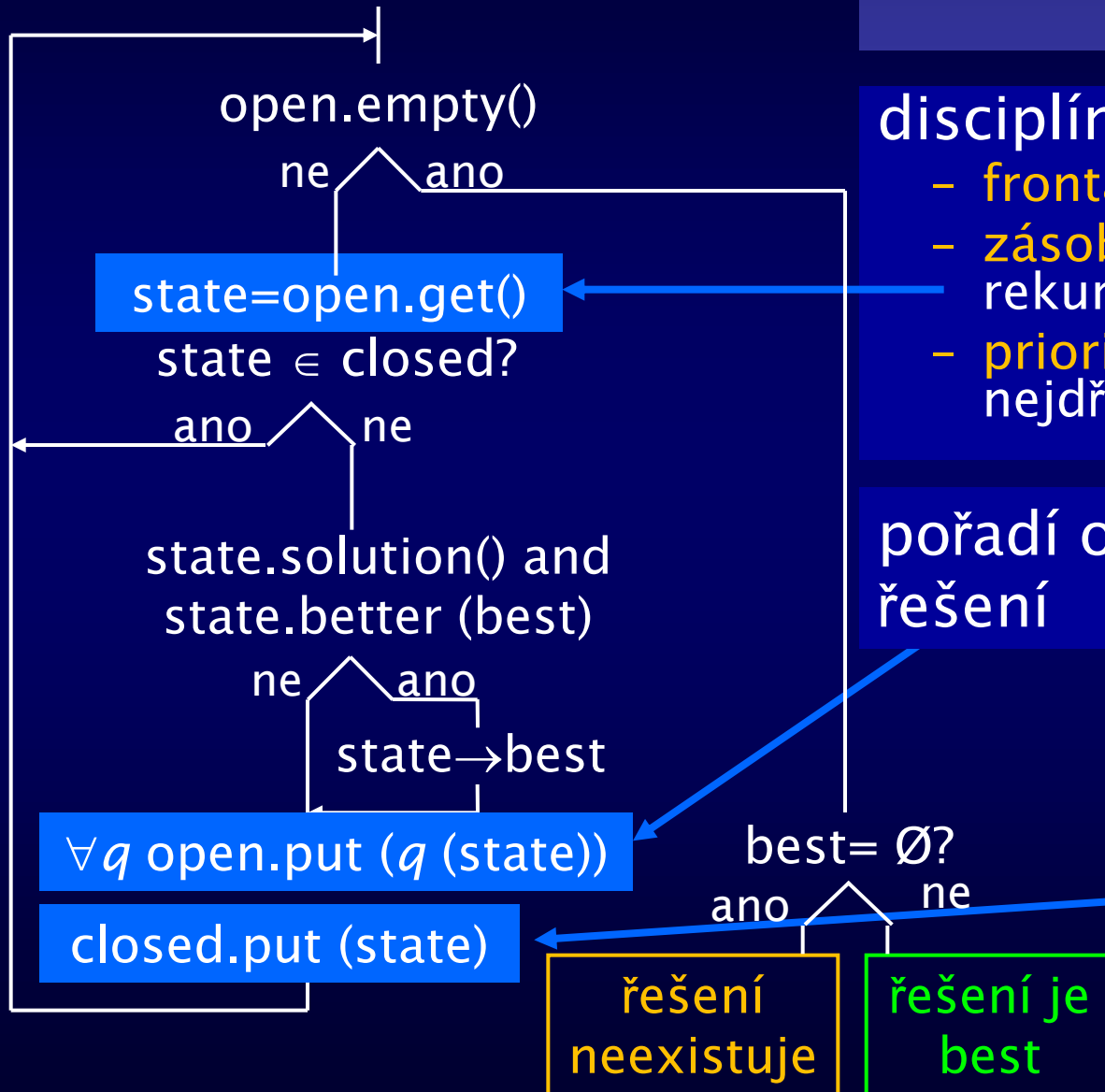
Sémantika

disciplína struktury open:

- **fronta**: do šířky
- **zásobník**: do hloubky (jako rekurzivní formulace)
- **prioritní fronta**: nejlepší nejdříve

pořadí ovlivní průměrný čas řešení

graf stavového prostoru je strom
→ není třeba closed



Vlastnosti systematických strategií

- Nejhorší případ roven hrubé síle
- V případě neexistujícího řešení tento případ nastane
- Naleznou řešení, existuje-li
- Naleznou optimální řešení

Prořezávání

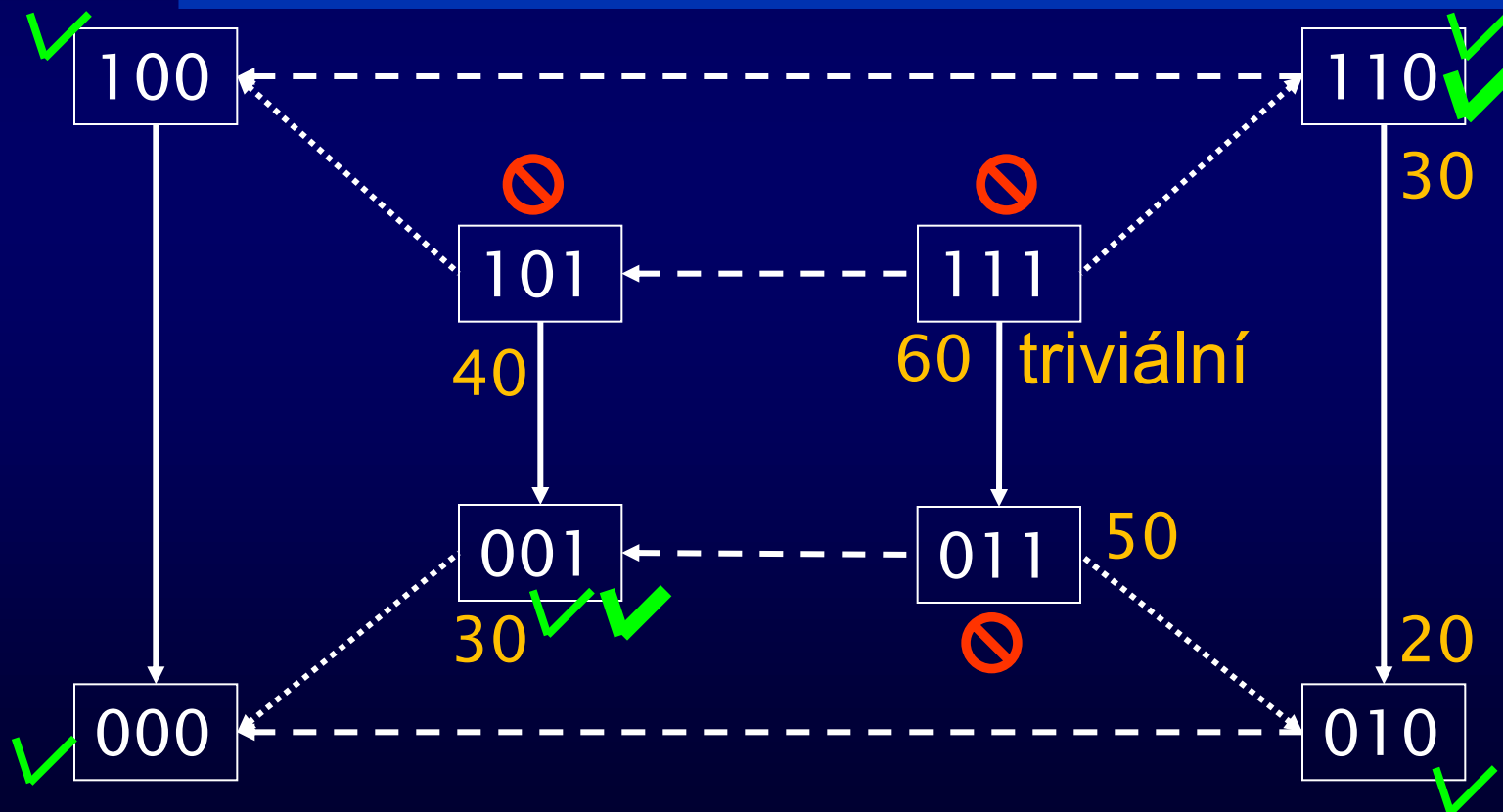
ve stavu s není řešení a
cena < 30 , stop této větvi
...proč?

111: není řešení, cena 60

110: je řešení, cena 30, stop této větvi, návrat

011: není řešení, cena $50 > 30$, pokračuj

010: je řešení, cena $20 < 30$, stop této větvi, návrat



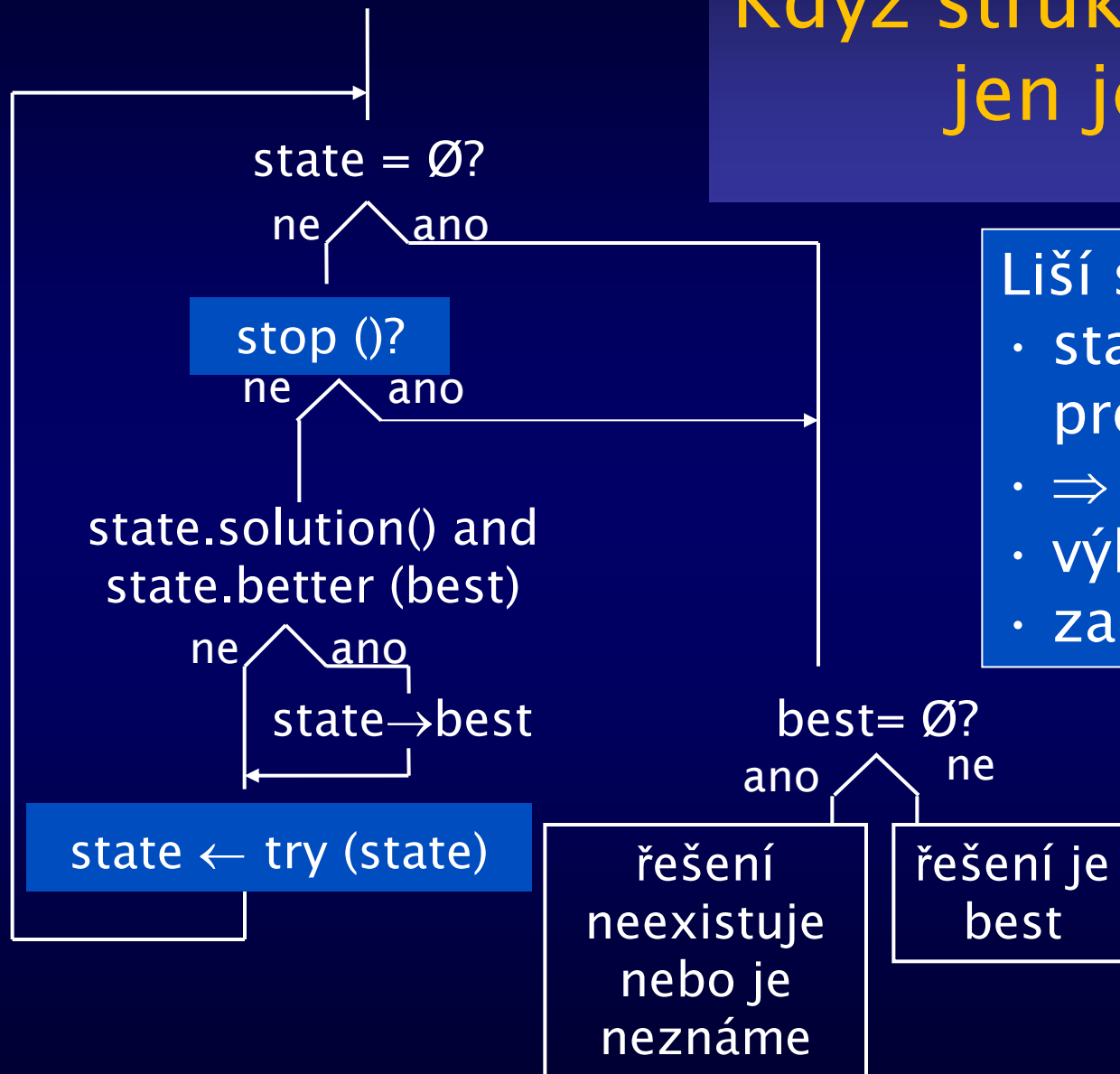
Prořezávání

- Na základě
 - dosud dosažené hodnoty optimalizačního kritéria
 - splnění omezujících podmínek
 - znalostí optimalizačního kritéria a omezujících podmínek
- můžeme vyloučit (prořezat) určité oblasti stavového prostoru (větve hledání)
- Odhady dolní/horní meze optimalizačního kritéria mohou pocházet i z jiných metod

Lokální heuristické metody

{počáteční stav} \rightarrow state
 $\emptyset \rightarrow$ best

Když struktura *open* má jen jednu položku

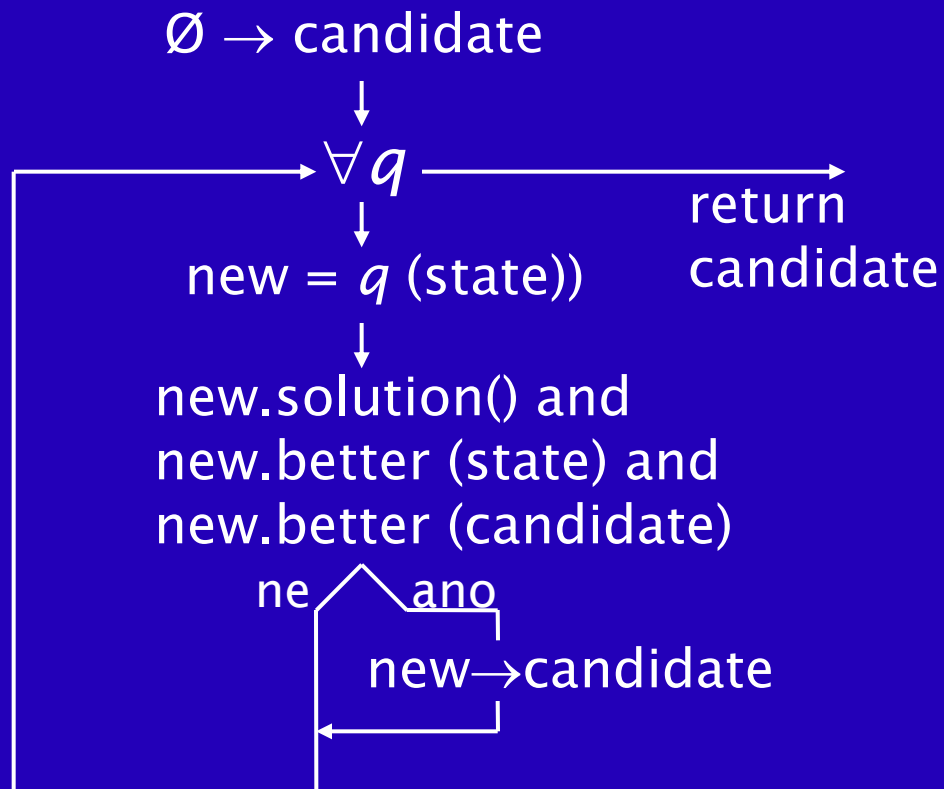


Liší se navzájem:

- stavovým prostorem
- \Rightarrow okolím
- výběrem z okolí
- zastavením

Metoda pouze nejlepší (best only)

try (state)

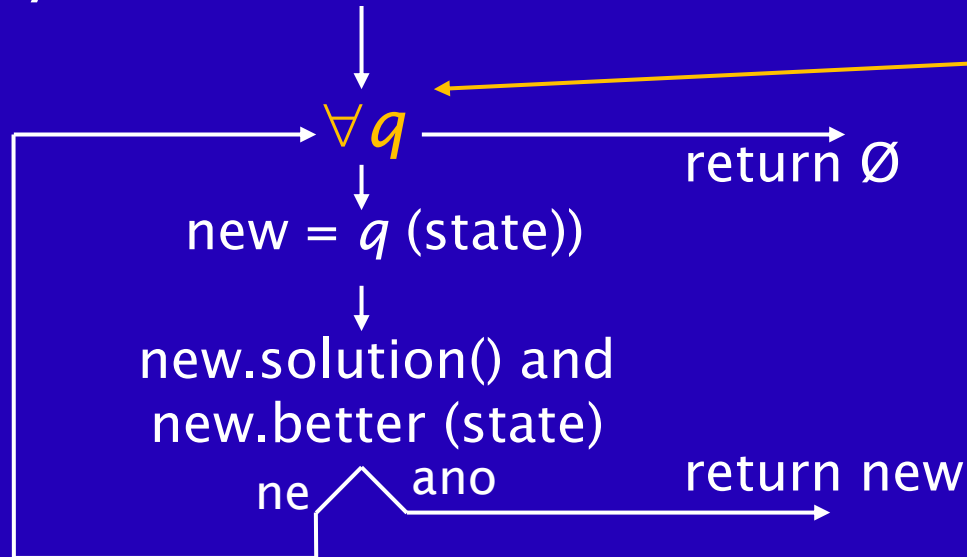


Metoda pouze nejlepší

- celá heuristika se zastaví, jestliže v nějakém stavu neexistuje zlepšující tah
- pokud `better()` používá jiné hodnocení než optimalizační kritérium, `solution()` může chybět
- jiné názvy: metoda nejrychlejšího sestupu/vzestupu, horolezecký algoritmus
- na pořadí prohledávání okolí nezáleží

Metoda prvního zlepšení (first improvement)

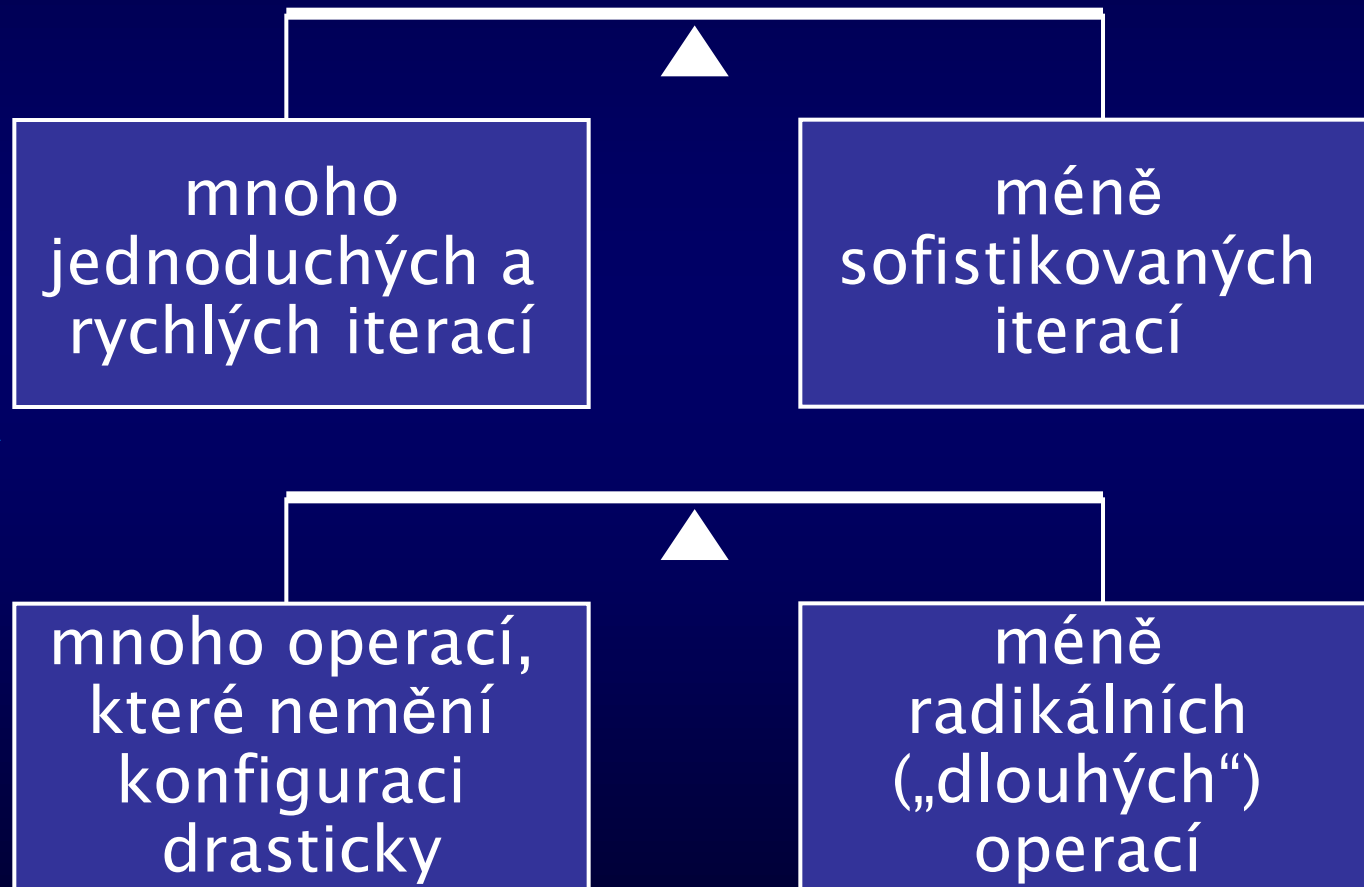
try (state)



záleží na pořadí
→ randomizace
→ náhodné pořadí

- celá heuristika se zastaví, jestliže v nějakém stavu neexistuje zlepšující tah
- pokud better() používá jiné hodnocení než optimalizační kritérium, solution() může chybně

Návrh heuristiky a jejího stavového prostoru



občas

Okolí heuristik Kernighan-Lin

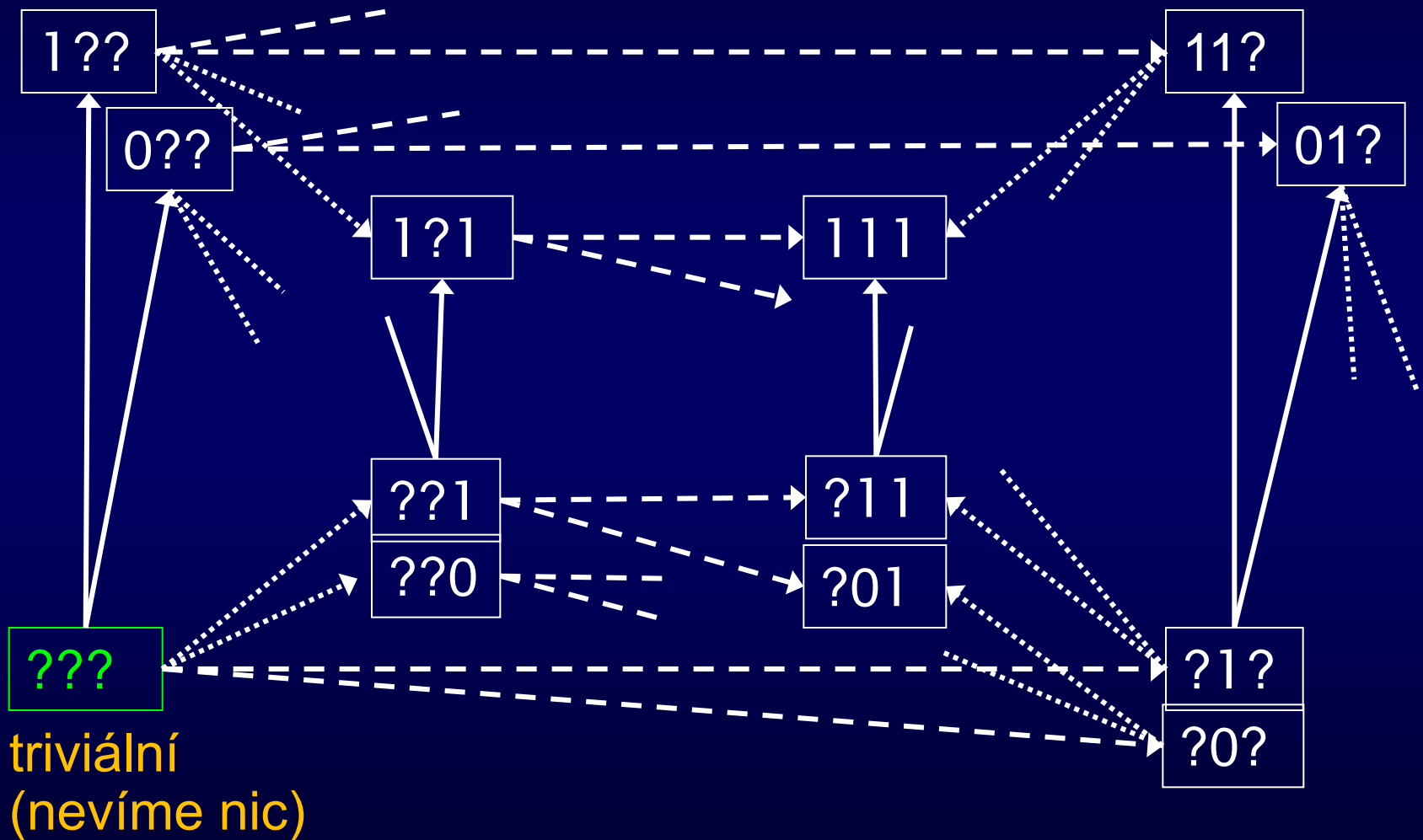
aplikuj opakovaně danou transformaci
až do stop podmínky bez ohledu
na optimalizační kritérium nebo
heuristickou funkci



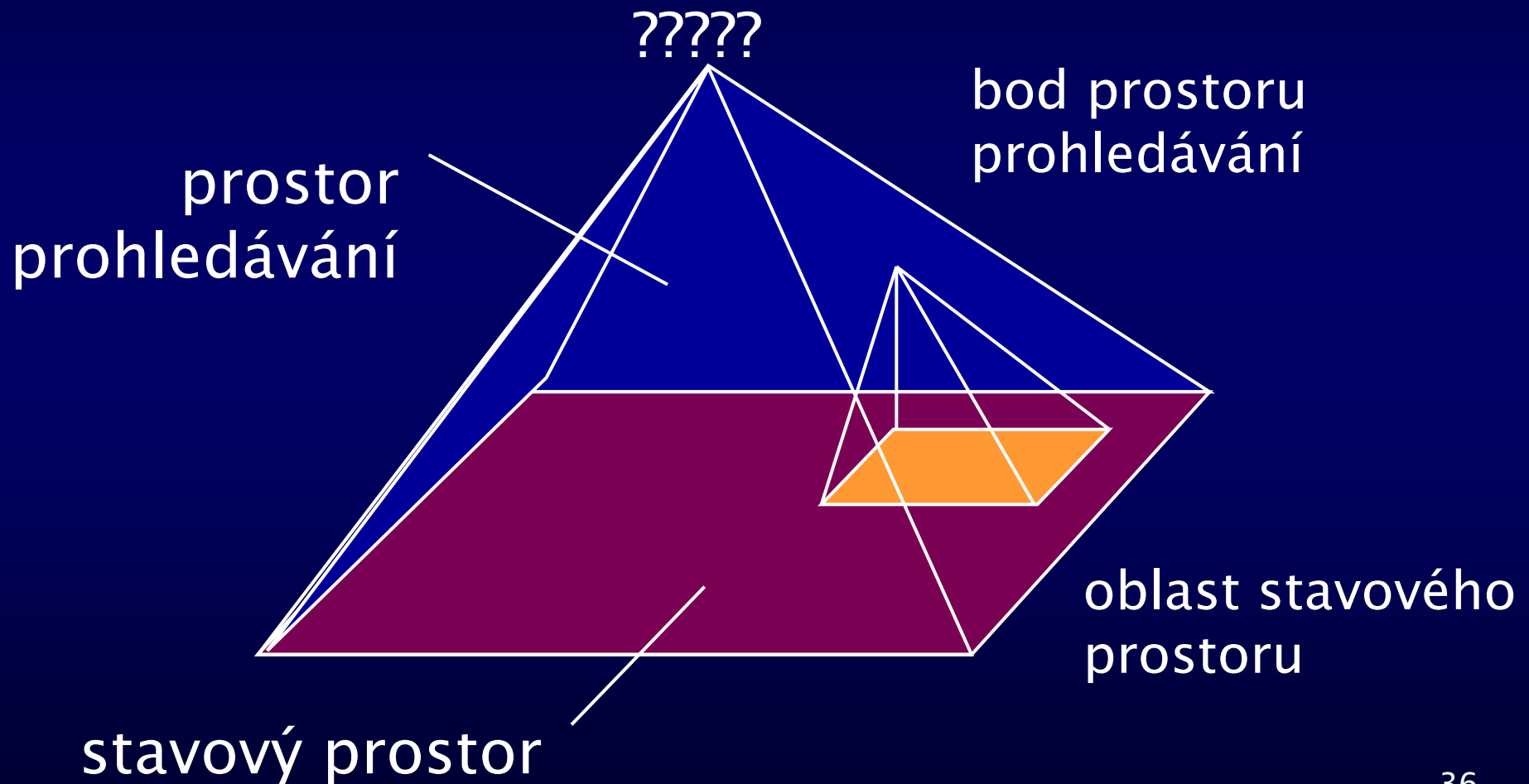
„Kernighan-Lin“:
řada heuristik
původně pro TSP,
později hlavně pro
bisekci grafu

Prostor prohledávání (search space)

Postupné ohodnocování proměnných



Vztah ke stavovému prostoru



Pohyb v prohledávacím prostoru

- Typické strategie nebývají úplné, natož systematické
- Typický krok prohledávání:
 - vyber proměnnou
 - vyber hodnotu proměnné
- Prořezávání se vztahuje na **oblast** stavového prostoru
- Možnost odvolat nastavení proměnné (backtracking)

Příklad: problém diskrétního rozmístění a jeho heuristiky

A takhle se kdysi navrhovaly
integrované obvody...

Problém diskrétního rozmístění

- Dáno:

- množina n modulů $K = \{k_1, \dots, k_n\}$
- množina m pozic $P = \{p_1, \dots, p_m\}$, $m \geq n$
- propojení modulů jako hypergraf $G(K, N)$, kde N je množina spojů n
- cenová funkce $W(R, n)$, která pro každé přiřazení $R: K \rightarrow P$ odhadne cenu realizace spoje n .

- Nalézt:

- prosté přiřazení $R: K \rightarrow P$ (rozmístění), které minimalizuje součet ohodnocení $W(R, n)$ přes všechny spoje.

Strategie

- **Reprezentace R :**
zobrazení R můžeme chápat jako
 - jednu proměnnou, s triviální hodnotou \emptyset , kterou měníme, až splní omezení (prosté zobrazení)
 - pro každý modul $k_i \in K$ jednu proměnnou r_i a jednu proměnnou z_i , která udává, zda r_i má platnou hodnotu (modul je rozmístěn)
 - pro každý modul $k_i \in K$ jednu proměnnou r_i v pohledávacím prostoru
- **Strategie:**
 - na začátku není rozmístěno nic
 - vybereme modul
 - vybereme pro něj nejlepší pozici
 - opakujeme, dokud nejsou rozmístěny všechny moduly

Algoritmus max konjunkce min disjunkce

- Necht' $H(G, k_1, k_2)$ je heuristická funkce, která na základě grafu G odhadne stupeň vazby modulů k_1 a k_2 .
- Necht' $K^+(R)$ označuje množinu modulů, které jsou právě rozmístěny (mají dvojici v R), $K^-(R)$ množinu právě nerozmístěných modulů.
- Obdobně definujeme množinu obsazených pozic $P^+(R)$ a volných pozic $P^-(R)$.
- $R \leftarrow (k_i, p_j)$, buď ze zadání nebo pomocnou heuristikou.

Algoritmus max konjunkce min disjunkce

1. Vyber modul $k \in K^-(R)$ takový, že

$$\sum_{l \in K^+(R)} H(G, k, l) = \max.$$

max. přidrátovaný
k rozmístěným
modulům

2. Je-li jich více, vyber z nich modul k takový, že

$$\sum_{l \in K^-(R)} H(G, k, l) = \min.$$

min. přidrátovaný
k nerozmístěným
modulům

3. Najdi pozici $p \in P(R)$ takovou, že

$$\sum_n W(R \cup (k, p), n) = \min.$$

pozice pro
nejlevnější dráty

kde suma je přes všechny spoje incidentní s k .

4. $R \leftarrow R \cup (k, p)$

5. Opakuj 1., dokud $K^-(R)$ není prázdná.

Okolí

- Krok: dáme nerozmístěný prvek na volnou pozici.
- Vezmeme $K(R)$, najdeme nejlepší prvek podle heuristické funkce.
- Vezmeme $P(R)$, najdeme nejlepší prvek podle optimalizačního kritéria.
- „Šidíme“ tak hledání nejlepšího prvku $(k,p) \in K(R) \times P(R)$ (nepotřebovali bychom heuristickou funkci!).

Stavový prostor je acyklický. Algoritmus se zastaví po n krocích.

Konstruktivní heuristika

- Začali jsme ze (skoro) triviální konfigurace
- Dopracovali jsme se k řešení (konfiguraci, která vyhovuje omezením – zde prosté zobrazení R)
- → konstruktivní heuristika

Iterativní zlepšování

1. Najít pozice p_1 a p_2 takové, že vzájemným prohozením jejich „obsahu“ (modul nebo nic) se nejvíce zlepší hodnota optimalizačního kritéria.
2. Není taková \Rightarrow stop.
3. Provést záměnu, opakovat 1.

Stavový prostor je acyklický (nemohu provést záměnu, která by nezlepšila optimalizační kritérium – nemohu se vrátit)

Okolí

- Vezmu $P \times P$, najdeme nejlepší prvek podle optimalizačního kritéria
- Můžeme to „šidit“:
 - najdeme pozici p_1 , která „toho má nejvíc zapotřebí“ (heuristická funkce)
 - najdeme pozici p_2 , kde je zlepšení největší
- Jiná možnost: hledáme v $P \times P$, ale spokojíme se s první záměnou, která zlepší optimalizační kritérium

Iterativní heuristika

- Začali jsme z řešení
- Dopracovali jsme se k lepšímu řešení
- → iterativní heuristika

Dvojfázové heuristiky

- Prvá fáze
 - konstruktivní
 - náhodná, více náhodných řešení (viz GSAT)
- Druhá fáze iterativní

Problém lokálních extrémů

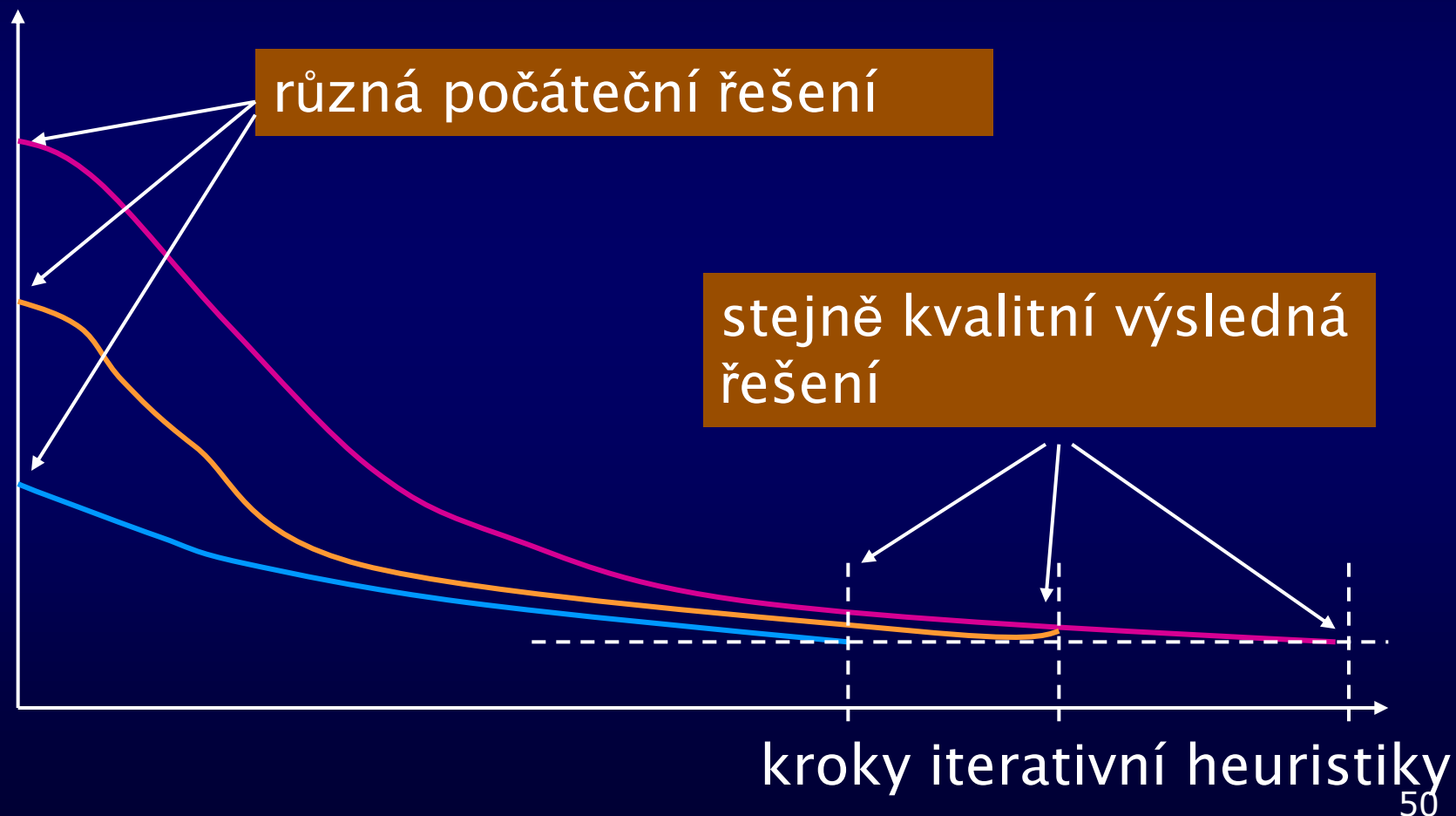
Lokální minimum

hodnota optimalizačního kritéria



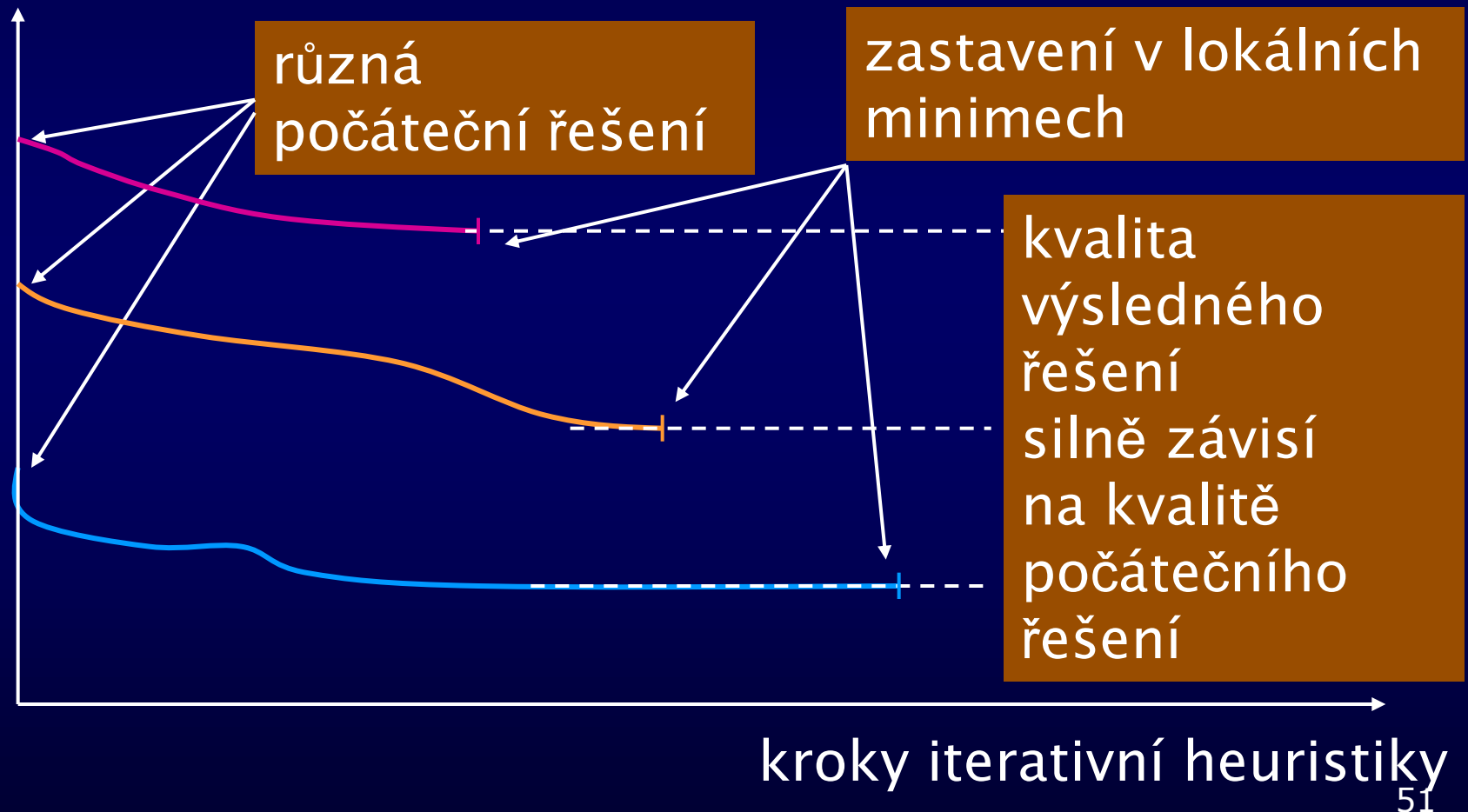
Správný průběh iterativní heuristiky

hodnota optimalizačního kritéria



Uvážnutí v lokálních minimech

hodnota optimalizačního kritéria



Problém lokálních extrémů

- Heuristiky, které neřeší uvážnutí v lokálním extrému (pouze nejlepší, první zlepšení a jim podobné) se zastaví v lokálním minimu
- Projev: závislost výsledného řešení na počátečním, tzv. nedostatečná iterační síla
- Centrální problém lokálních heuristik

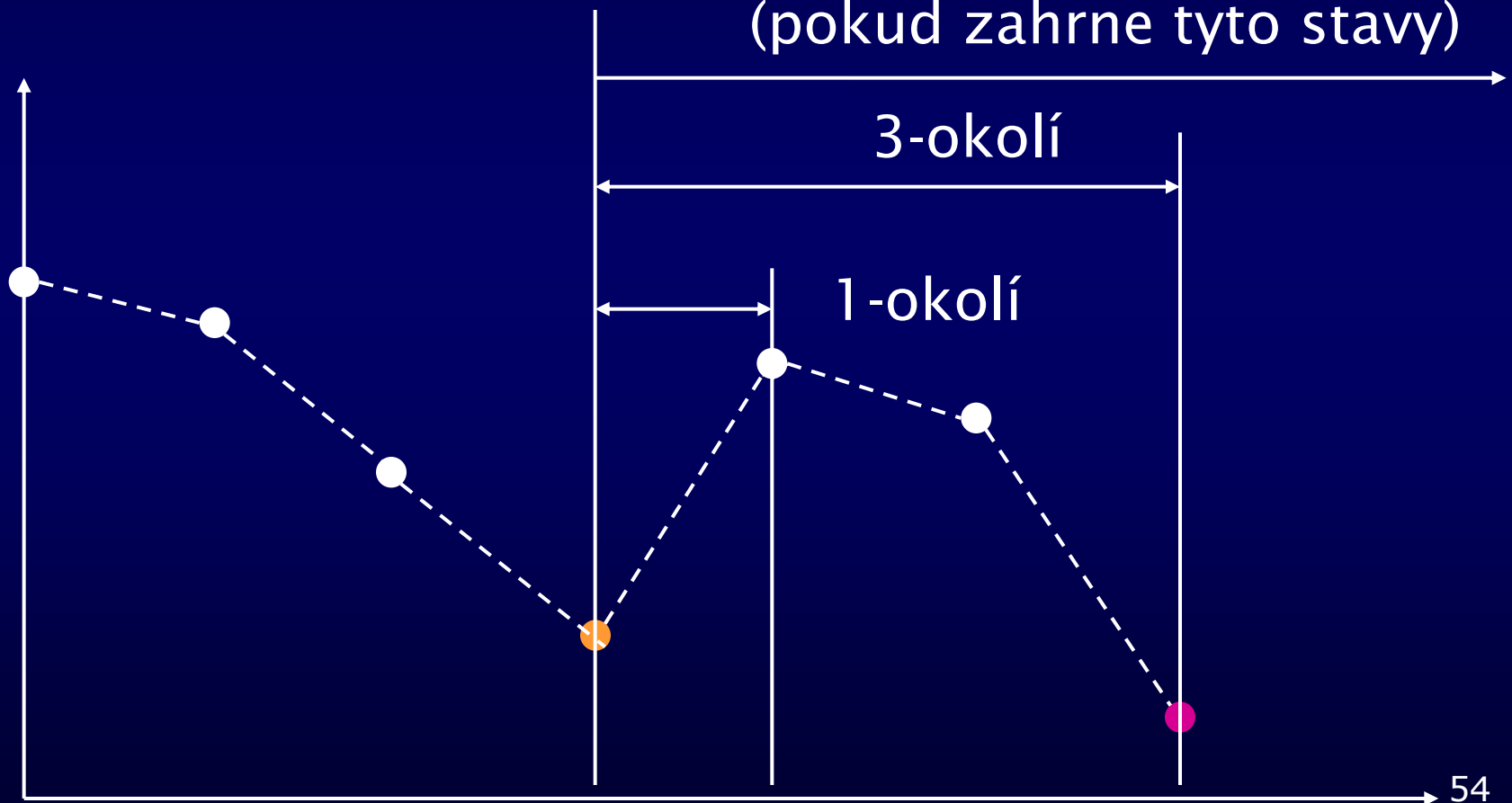
Řešení

- Zvětšení okolí
- Více náhodných počátečních řešení (GSAT)
 - odstranění následků, ne příčiny
 - závisí na možnosti generovat náhodná řešení
- Návraty
 - odvolání rozhodnutí, které vedly do slepé uličky
- Únik
 - kroky, které připustí zhoršení aktuálního stavu
 - nutnost dokonalejšího řízení algoritmu (bloudění)
 - nutnost řízení ochoty k horšímu řešení

Zvětšení okolí

k -okolí o m operátorech má m^k stavů!

Kernighan-Lin
(pokud zahrne tyto stavy)



Čemu teď rozumíme

Na čem jsou založeny lokální a globální metody

Z čeho se konstruuje stavový prostor

Jaké jeho vlastnosti jsou významné

Rozdíl mezi stavem algoritmu a stavem např.
manipulované scény, rozdíly v pojetí termínu stav

Kostra systematických strategií pohybu stavovým
prostorem

Rozdíl mezi stavovým prostorem a prostorem
prohledávání

Čemu teď rozumíme

Jak ze systematické strategie můžeme odvodit heuristiku

Jak fungují metody „pouze nejlepší“ a „prvé zlepšení“

Jak funguje heuristika Kernighan-Lin

Proč je problém uváznutí v lokálních minimech důležitý pro lokální heuristiky

Jak se tento problém projeví v práci heuristiky

Jaké jsou základní způsoby jeho potlačení

Jaké pojmy k tomu potřebujeme

stav

operátor, akce

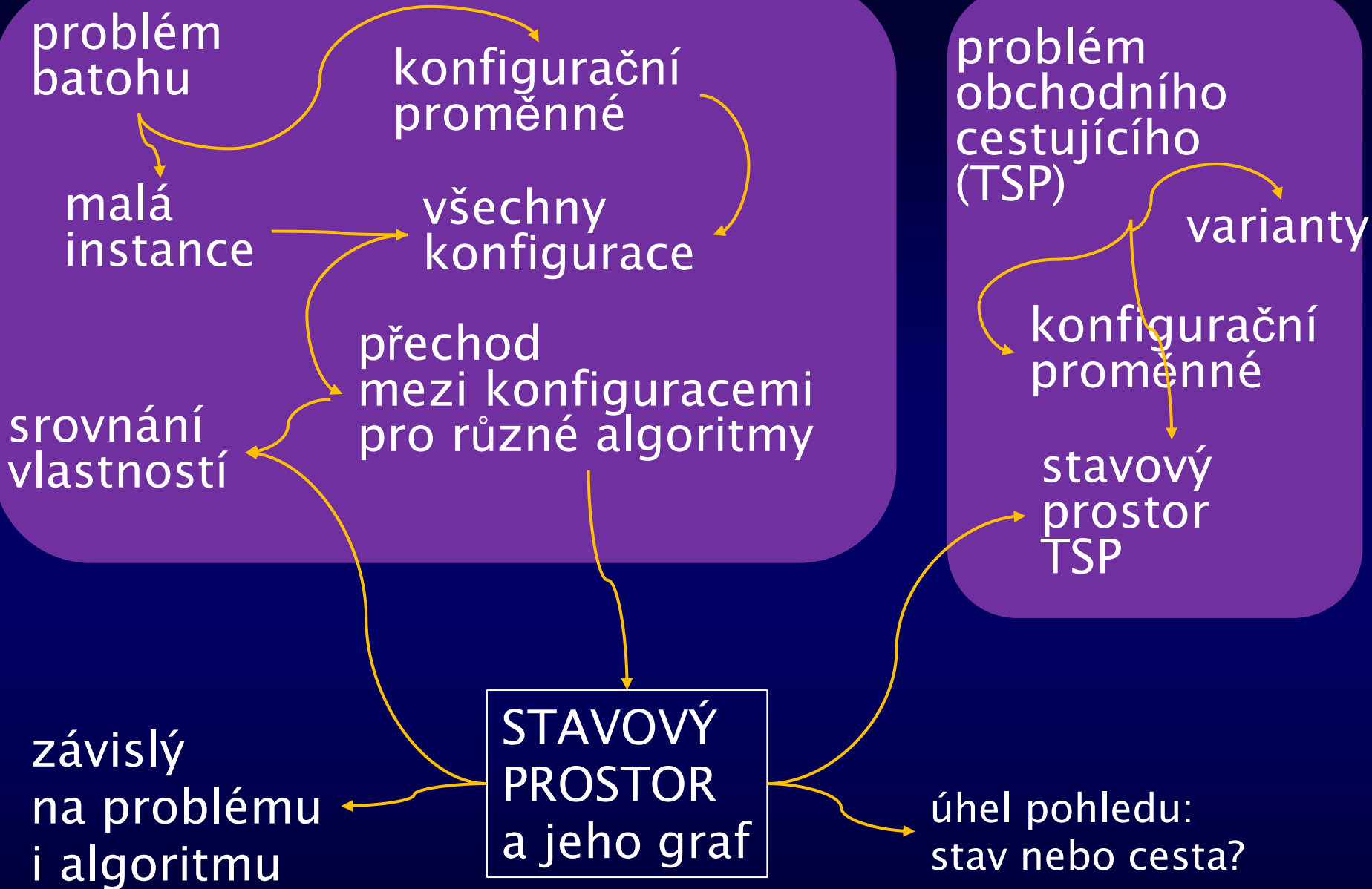
stavový prostor

strategie prohledávání, prořezávání, ukončení

strategie úplná, systematická

prořezávání

prostor prohledávání



STAVOVÝ
PROSTOR

princip

strategie
prořezávání

strategie
ukončení

strategie
prohledávání

nesystematická

úplná

systematická

typický
algoritmus

POHYB
STAVOVÝM
PROSTOREM

obměny

vztah k A^*

STAVOVÝ
PROSTOR

neúplné
ohodnocení
konfiguračních
proměnných

PROSTOR
PROHLEDÁVÁNÍ

problém
diskrétního
rozmístění

formulace
prostoru
prohledávání

aplikace
prořezávání