

Lokální prohledávání

Lokální prohledávání (LS)

● Princip

- pracuje s **úplnými nekonzistentními přiřazeními proměnných**
- snaží se **lokálními opravami** snížit počet konfliktů

● Příklad: umístění n dam na šachovnici

- iniciální přiřazení umístí každou královnu do každého sloupce a řádku bez ohledu na diagonální omezení
- přesunujeme královnu v jejím sloupci do jiného řádku tak, abychom odstranili co nejvíce konfliktů
- LS vyřeší řádově větší problémy než algoritmy založené na prohledávání do hloubky

Lokální prohledávání (LS)

● Princip

- pracuje s **úplnými nekonzistentními přiřazeními proměnných**
- snaží se **lokálními opravami** snížit počet konfliktů

● Příklad: umístění n dam na šachovnici

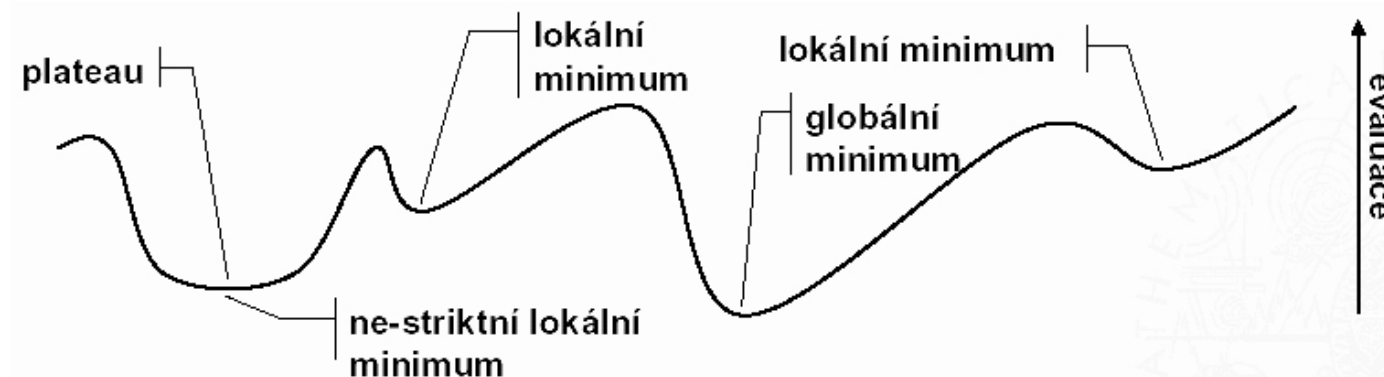
- iniciální přiřazení umístí každou královnu do každého sloupce a řádku bez ohledu na diagonální omezení
- přesunujeme královnu v jejím sloupci do jiného řádku tak, abychom odstranili co nejvíce konfliktů
- LS vyřeší řádově větší problémy než algoritmy založené na prohledávání do hloubky

● Přibližná metoda prohledávání

- neúplná metoda
- nezaručuje nalezení (vyloučení existence) řešení i když existuje (neexistuje)
- malé paměťové nároky

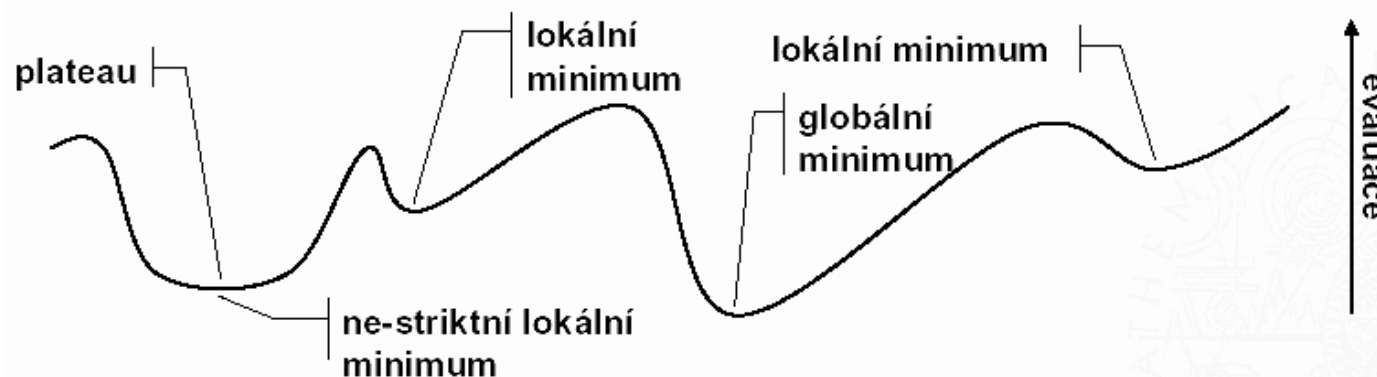
Terminologie lokálního prohledávání

- **Stav θ :** ohodnocení všech proměnných
- **Evaluace E :** hodnota objektivní funkce (počet nesplněných podmínek=**počet konfliktů**)
- **Globální optimum:** stav s nejlepší evaluací



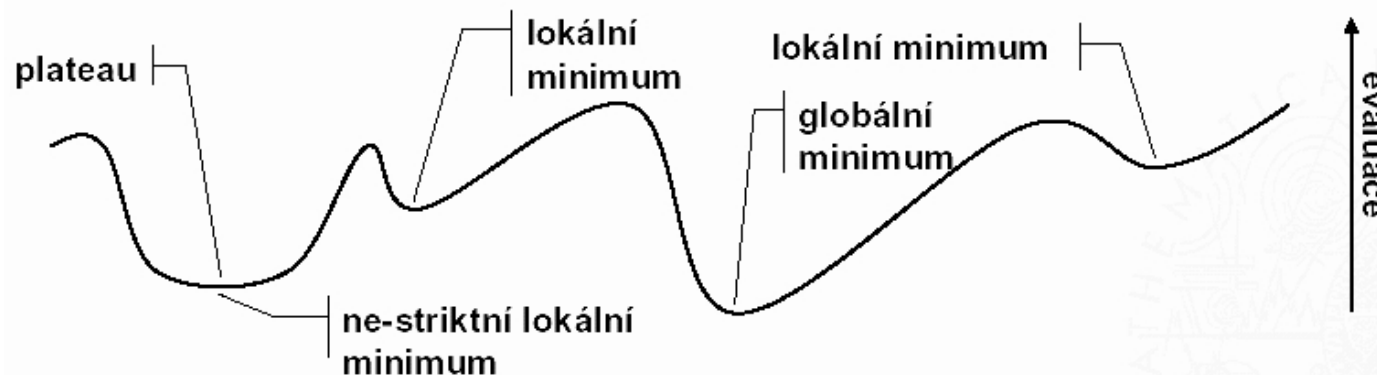
Terminologie lokálního prohledávání

- **Stav θ :** ohodnocení všech proměnných
- **Evaluace E :** hodnota objektivní funkce (počet nesplněných podmínek=**počet konfliktů**)
- **Globální optimum:** stav s nejlepší evaluací
- **Lokální změna:** změna hodnoty (jedné) proměnné
- **Okolí o :** množina stavů lišící se od daného stavu o jednu lokální změnu
- **Lokální optimum:** stav, v jehož okolí jsou stavy s horší evaluací; není globálním optimumm



Terminologie lokálního prohledávání

- **Stav θ :** ohodnocení všech proměnných
- **Evaluace E :** hodnota objektivní funkce (počet nesplněných podmínek=**počet konfliktů**)
- **Globální optimum:** stav s nejlepší evaluací
- **Lokální změna:** změna hodnoty (jedné) proměnné
- **Okolí o :** množina stavů lišící se od daného stavu o jednu lokální změnu
- **Lokální optimum:** stav, v jehož okolí jsou stavy s horší evaluací; není globálním optimum
- **Striktní lokální optimum:** stav, v jehož okolí jsou pouze stavy s horší evaluací; není globálním optimum
- **Ne-striktní lokální optimum:** stav, v jehož okolí existují stavy se stejnou evaluací; není globálním optimum
- **Plateau:** množina stavů se stejnou evaluací



Algoritmus lokálního prohledávání

- Algoritmy lokálního prohledávání mají společnou kostru

```
procedure LS(MaxPokusu,MaxZmen)
```

parametry algoritmu

```
 $\theta :=$  náhodné ohodnocení proměnných
```

```
for i := 1 to MaxPokusu while GPodminka do
```

```
  for j := 1 to MaxZmen while LPodminka do
```

```
    if E( $\theta$ )=0 then
```

```
      return  $\theta$ 
```

```
    vyber  $\delta \in \underline{o}(\theta)$ 
```

```
    if akceptovatelné( $\delta$ ) then
```

```
       $\theta := \delta$ 
```

```
   $\theta :=$  novyStav( $\theta$ )
```

```
return nejlepší  $\theta$ 
```

Algoritmus lokálního prohledávání

- Algoritmy lokálního prohledávání mají společnou kostru

```
procedure LS(MaxPokusu,MaxZmen)
```

parametry algoritmu

```
 $\theta :=$  náhodné ohodnocení proměnných
```

```
for i := 1 to MaxPokusu while GPodminka do
```

```
  for j := 1 to MaxZmen while LPodminka do
```

```
    if E( $\theta$ )=0 then
```

```
      return  $\theta$ 
```

```
      vyber  $\delta \in \underline{o}(\theta)$ 
```

```
      if akceptovatelné( $\delta$ ) then
```

```
         $\theta := \delta$ 
```

```
     $\theta :=$  novyStav( $\theta$ )
```

```
  return nejlepší  $\theta$ 
```

- Jak stanovit MaxPokusu ,MaxZmen?

- pokračovat dokud existuje přiřazení s lepší evaluací
- restart (MaxPokusu>1) v některých případech diskutabilní

Metoda největšího stoupání (*hill climbing*) HC

- Začíná v náhodně vybraném stavu
- Hledá vždy nejlepší stav v okolí
- Okolí = hodnota libovolné proměnné je změněna, velikost okolí $n \times (d - 1)$
- Útěk ze striktního lokálního minima pomocí restartu

```
procedure HC(MaxZmen)
```

```
  restart:  $\theta$  := náhodné ohodnocení proměnných
```

```
    for j := 1 to MaxZmen do
```

```
      if  $E(\theta) = 0$  then
```

```
        return  $\theta$ 
```

```
      if  $\theta$  je striktní lokální optimum then
```

```
        goto restart
```

```
      else  $\theta$  := stav z  $o(\theta)$  s nejlepší evaluací
```

```
    goto restart
```

```
end HC
```

Metoda minimalizace konfliktů (MC)

- Okolí u HC je poměrně velké: $n \times (d - 1)$
 - ale pouze změna konfliktní proměnné může přinést zlepšení
 - konfliktní proměnná = vystupuje v některých nesplněných podmínkách
- MC mění pouze konfliktní proměnné
 - okolí = hodnota zvolené proměnné je změněna, velikost okolí $d - 1$

procedure MC(MaxZmen)

θ := náhodné ohodnocení proměnných

PocetZmen := 0

while $E(\theta) > 0 \wedge$ PocetZmen < MaxZmen do

vyber náhodně konfliktní proměnnou V

vyber hodnotu a, která minimalizuje počet konfliktů pro V

if $a \neq$ současná hodnota V then

přiřad' a do V

PocetZmen := PocetZmen+1

return θ

neřešíme únik z lokálního optima

Náhodná procházka (*Random Walk*) RW

- Jak **uniknout z lokálního optima bez restartu?**
 - přidáním „šumu“ do algoritmu
- Pokud se dostaneme do lokálního optima
 - náhodně zvolíme stav z okolí a pokračujeme jím
 - tato metoda samostatně k řešení nepovede
 - potřebuje další směřování v prohledávacím prostoru
 - lze kombinovat s HC i MC

Náhodná procházka (*Random Walk*) RW

- Jak **uniknout z lokálního optima bez restartu?**
 - přidáním „šumu“ do algoritmu
- Pokud se dostaneme do lokálního optima
 - náhodně zvolíme stav z okolí a pokračujeme jím
 - tato metoda samostatně k řešení nepovede
 - potřebuje další směřování v prohledávacím prostoru
 - lze kombinovat s HC i MC
- **RW je kombinováno s heuristikou pomocí pravděpodobnostního rozložení**
 - pravděpodobnost náhodného kroku je p
 - pravděpodobnost použití směrové heuristiky je $1 - p$
 - náhodná procházka tedy nahrazuje restart
 - únik z lokálního minima prostřednictvím náhodného výběru

Minimalizace konfliktů s náhodnou procházkou

procedure MCRW(MaxZmen,p)

Rozdíly od MC

θ := náhodné ohodnocení proměnných

PocetZmen := 0

while $E(\theta) > 0 \wedge$ PocetZmen < MaxZmen do

vyber náhodně konfliktní proměnnou V

generuj náhodné číslo Pravdepodobnost $\in [0, 1]$

if Pravdepodobnost $\geq (1 - p)$

$0.02 \leq p \leq 0.1$

vyber náhodně hodnotu a pro V

else vyber hodnotu a, která minimalizuje počet konfliktů pro V

if a \neq současná hodnota V then

přiřad' a do V

PocetZmen := PocetZmen+1

return θ

Největší stoupání s náhodnou procházkou

```
procedure HCRW(MaxZmen,p)
 $\theta$  := náhodné ohodnocení proměnných
PocetZmen := 0
while  $E(\theta) > 0 \wedge$  PocetZmen < MaxZmen do
    generuj náhodné číslo Pravdepodobnost  $\in [0,1]$ 
    if Pravdepodobnost  $\geq (1 - p)$ 
        vyber náhodně konfliktní proměnnou V
        vyber náhodně hodnotu a pro V
    else vyber  $\langle V,a \rangle$  s nejlepší evaluací
        if a  $\neq$  současná hodnota V then
            přiřad' a do V
            PocetZmen := PocetZmen+1
return  $\theta$ 
```

Rozdíly od MCRW

Tabu seznam

- Setrvání v lokálním optimu je speciálním případem cyklu
- Jak se obecně zbavit cyklů?
 - stačí si pamatovat předchozí stavy a zakázat opakování stavu
 - paměťově příliš náročné (mnoho stavů)
 - můžeme si zapamatovat pouze několik posledních stavů (zabrání krátkým cyklům)

Tabu seznam

- Setrvání v lokálním optimu je speciálním případem cyklu
- Jak se obecně zbavit cyklů?
 - stačí si pamatovat předchozí stavy a zakázat opakování stavu
 - paměťově příliš náročné (mnoho stavů)
 - můžeme si zapamatovat pouze několik posledních stavů (zabrání krátkým cyklům)
- **Tabu seznam** = seznam **tabu (zakázaných) stavů**
 - stav lze popsat význačným atributem (není nutné uchovávat celý stav)
 - (proměnná, hodnota): zachycuje změnu stavu (uložíme původní hodnoty)
 - tabu seznam má **fixní délku (*tabu tenure*)**
 - „staré“ stavy ze seznamu vypadávají s přicházejícími novými stavy

Tabu seznam

- Setrvání v lokálním optimu je speciálním případem cyklu
- Jak se obecně zbavit cyklů?
 - stačí si pamatovat předchozí stavy a zakázat opakování stavu
 - paměťově příliš náročné (mnoho stavů)
 - můžeme si zapamatovat pouze několik posledních stavů (zabrání krátkým cyklům)
- **Tabu seznam** = seznam **tabu (zakázaných) stavů**
 - stav lze popsat význačným atributem (není nutné uchovávat celý stav)
 - (proměnná, hodnota): zachycuje změnu stavu (uložíme původní hodnoty)
 - tabu seznam má **fixní délku (*tabu tenure*)**
 - „staré“ stavy ze seznamu vypadávají s přicházejícími novými stavy
- **Aspirační kritérium** = odtabuizování stavu
 - do stavu lze přejít, i když je v tabu seznamu (např. krok vede k celkově lepšímu stavu)

Tabu seznam

- Setrvání v lokálním optimu je speciálním případem cyklu
- Jak se obecně zbavit cyklů?
 - stačí si pamatovat předchozí stavy a zakázat opakování stavu
 - paměťově příliš náročné (mnoho stavů)
 - můžeme si zapamatovat pouze několik posledních stavů (zabrání krátkým cyklům)
- **Tabu seznam** = seznam **tabu (zakázaných) stavů**
 - stav lze popsat význačným atributem (není nutné uchovávat celý stav)
 - (proměnná, hodnota): zachycuje změnu stavu (uložíme původní hodnoty)
 - tabu seznam má **fixní délku (*tabu tenure*)**
 - „staré“ stavy ze seznamu vypadávají s přicházejícími novými stavy
- **Aspirační kritérium** = odtabuizování stavu
 - do stavu lze přejít, i když je v tabu seznamu (např. krok vede k celkově lepšímu stavu)
- Tabu seznam je používán samostatně i v kombinaci s jinými metodami LS

Algoritmus prohledávání s tabu seznamem

- Tabu seznam zabraňuje krátkému cyklení
- Povoleny jsou pouze tahy mimo tabu seznam a tahy splňující aspirační kritérium
- **Tabu seznam (TS) v kombinaci s metodou stoupání (HC):**

procedure TSHC(MaxZmen)

θ := náhodné ohodnocení proměnných

PocetZmen := 0

while $E(\theta) > 0 \wedge$ PocetZmen < MaxZmen do

vyber $\langle V, a \rangle$ s nejlepší evaluací tak, že

není v tabu seznamu a nebo splňuje aspirační kritérium

přidej $\langle V, c \rangle$ do tabu seznamu, kde c je současná hodnota V

smaž nejstarší položku v tabu seznamu

přiřad' a do V

PocetZmen := PocetZmen+1

return θ

Výběr souseda: přehled

● Metoda stoupání (HC)

- soused s nejlepší evaluací vybrán

● Tabu prohledávání (TS+HC)

- soused s nejlepší evaluací vybrán (metoda stoupání)
- sousedé z tabu seznamu nemohou být vybráni

Výběr souseda: přehled

● Metoda stoupání (HC)

- soused s nejlepší evaluací vybrán

● Tabu prohledávání (TS+HC)

- soused s nejlepší evaluací vybrán (metoda stoupání)
- sousedé z tabu seznamu nemohou být vybráni

● Minimální konflikt (MC)

- soused je omezen na náhodně vybranou konfliktní proměnnou
- výběr její hodnoty s nejlepší evaluací

● Náhodná procházka (RW)

- soused vybrán náhodně

Výběr souseda: přehled

● Metoda stoupání (HC)

- souseď s nejlepší evaluací vybrán

● Tabu prohledávání (TS+HC)

- souseď s nejlepší evaluací vybrán (metoda stoupání)
- souseď z tabu seznamu nemohou být vybráni

● Minimální konflikt (MC)

- souseď je omezen na náhodně vybranou konfliktní proměnnou
- výběr její hodnoty s nejlepší evaluací

● Náhodná procházka (RW)

- souseď vybrán náhodně

● Min. konflikt (metoda stoupání) s náhodnou procházkou MC+RW (HC+RW)

- s malou pravděpodobností: náhodný výběr souseda
- jinak: minimální konflikt (metoda stoupání)

Simulované žíhání (*simulated annealing*) SA

- Myšlenka: **simulace procesu ochlazování kovů**
 - na začátku při vyšší teplotě atomy více kmitají a pravděpodobnost změny krystalické mřížky je vyšší
 - postupným ochlazováním se atomy usazují co „nejlepší polohy“ s nejmenší energií a pravděpodobnost změny je menší
- ⇒ na začátku je tedy pravděpodobnost toho, že akceptujeme zhoršování řešení, vyšší

Simulované žíhání (*simulated annealing*) SA

● Myšlenka: **simulace procesu ochlazování kovů**

- na začátku při vyšší teplotě atomy více kmitají a pravděpodobnost změny krystalické mřížky je vyšší
 - postupným ochlazováním se atomy usazují co „nejlepší polohy“ s nejmenší energií a pravděpodobnost změny je menší
- ⇒ na začátku je tedy pravděpodobnost toho, že akceptujeme zhoršování řešení, vyšší

● **Akceptování nového stavu**

- vždy při zlepšení
- při zhoršení pouze za dané pravděpodobnosti, která klesá se snížením teploty

● **Cykly algoritmu**

- vnější: simulace procesu ochlazování snižováním **teploty T**
čím nižší bude teplota, tím nižší bude pravděpodobnost akceptování zhoršení
- vnitřní: počítáme, kolikrát jsme neakceptovali zhoršení (dán limit MaxIter)

Metropolisovo kritérium

Rozdíl mezi kvalitou nového (δ) a existujícího (θ) řešení

- $\Delta E = E(\delta) - E(\theta)$

- E (chybovost) musí být minimalizováno

Metropolisovo kritérium

- lepší (někdy případně i stejně kvalitní) řešení akceptováno: $\Delta E < 0$

- **horší řešení ($\Delta E > 0$) akceptováno pokud**

$$U < e^{-\Delta E/T}$$

- U náhodné číslo z intervalu $(0, 1)$

Metropolisovo kritérium

Rozdíl mezi kvalitou nového (δ) a existujícího (θ) řešení

- $\Delta E = E(\delta) - E(\theta)$

- E (chybovost) musí být minimalizováno

Metropolisovo kritérium

- lepší (někdy případně i stejně kvalitní) řešení akceptováno: $\Delta E < 0$

- **horší řešení ($\Delta E > 0$) akceptováno pokud**

$$U < e^{-\Delta E/T}$$

- U náhodné číslo z intervalu $(0, 1)$

- pomůcka: porovnej $e^{-10/100}$ vs. $e^{-100/100}$

Metropolisovo kritérium

Rozdíl mezi kvalitou nového (δ) a existujícího (θ) řešení

- $\Delta E = E(\delta) - E(\theta)$

- E (chybovost) musí být minimalizováno

Metropolisovo kritérium

- lepší (někdy případně i stejně kvalitní) řešení akceptováno: $\Delta E < 0$

- **horší řešení ($\Delta E > 0$) akceptováno pokud**

$$U < e^{-\Delta E/T}$$

- U náhodné číslo z intervalu $(0, 1)$

- pomůcka: porovnej $e^{-10/100}$ vs. $e^{-100/100}$ a $e^{-10/100}$ vs. $e^{-10/1}$

Algoritmus simulovaného žíhání

procedure SA(TInit, TEnd, MaxIter)

θ := náhodné ohodnocení proměnných

α := θ (dosud nejlepší nalezené přiřazení)

for T := TInit to TEnd

PocetChyb := 0

while PocetChyb < MaxIter

vyber lokální změnu z θ do δ

if $E(\delta) < E(\theta)$ then (akceptuj přiřazení)

θ := δ

if $E(\theta) < E(\alpha)$ then

α := θ (nové optimum)

else generuj náhodné číslo $p \in [0,1)$

if $p < e^{(E(\theta) - E(\delta))/T}$ then (akceptuj přiřazení)

θ := δ

else PocetChyb := PocetChyb + 1 (neakceptuj přiřazení)

T := max(round($0.8 \times T$), TEnd) (sniž teplotu)

end SA

Lokální prohledávání pro SAT problém

- **SAT problém**: splnitelnost logické formule v konjunktivní normální formě
 - **CNF** = konjunkce klauzulí
 - **klauzule** = disjunkce literálů (podmínka)
 - **literál** = atom nebo negace atomu
 - příklad: $(A \vee B) \wedge (\neg B \vee C) \wedge (\neg C \vee \neg A)$
 \Rightarrow CSP nad disjunkcemi boolean proměnných
- SAT je NP-úplný

Lokální prohledávání pro SAT problém

- **SAT problém**: splnitelnost logické formule v konjunktivní normální formě
 - **CNF** = konjunkce klauzulí
 - **klauzule** = disjunkce literálů (podmínka)
 - **literál** = atom nebo negace atomu
 - příklad: $(A \vee B) \wedge (\neg B \vee C) \wedge (\neg C \vee \neg A)$

⇒ CSP nad disjunkcemi boolean proměnných
- SAT je NP-úplný
- LS řeší poměrně velké formule
 - formulace LS je velice přirozená a jeho použití je velice populární
 - lokální změna je **překlápěním (*flipping*)** hodnoty proměnné
- Algoritmus **GSAT** (*greedy SAT*)
 - postupné překlápění proměnných
 - evaluace udává, jaký je (vážený) počet nesplněných klauzulí

Algoritmus GSAT

```
procedure GSAT(A,MaxPokusu,MaxZmen)  (A je CNF formule)
for i := 1 to MaxPokusu do
   $\theta$  := náhodné ohodnocení proměnných
  for j := 1 to MaxZmen do
    if A je splnitelná pomocí  $\theta$  then
      return  $\theta$ 
  V := proměnná, jejíž změna hodnoty
    nejvíce sníží počet nesplněných klauzulí
   $\theta$  := přiřazení, které se liší od  $\theta$  změnou hodnoty V
return nejlepší  $\theta$ 
```

Algoritmus GSAT

```
procedure GSAT(A,MaxPokusu,MaxZmen)  (A je CNF formule)
for i := 1 to MaxPokusu do
   $\theta$  := náhodné ohodnocení proměnných
  for j := 1 to MaxZmen do
    if A je splnitelná pomocí  $\theta$  then
      return  $\theta$ 
  V := proměnná, jejíž změna hodnoty
    nejvíce sníží počet nesplněných klauzulí
   $\theta$  := přiřazení, které se liší od  $\theta$  změnou hodnoty V
return nejlepší  $\theta$ 
```

● Příklad: $\{\neg C, \neg A \vee \neg B \vee C, \neg A \vee D \vee E, \neg B \vee \neg C\}$

- iniciální přiřazení (všechny proměnné mají hodnotu 1) nespĺňuje první a poslední klauzuli
- změna A, D, E

Algoritmus GSAT

```
procedure GSAT(A,MaxPokusu,MaxZmen)  (A je CNF formule)
for i := 1 to MaxPokusu do
   $\theta$  := náhodné ohodnocení proměnných
  for j := 1 to MaxZmen do
    if A je splnitelná pomocí  $\theta$  then
      return  $\theta$ 
  V := proměnná, jejíž změna hodnoty
    nejvíce sníží počet nesplněných klauzulí
   $\theta$  := přiřazení, které se liší od  $\theta$  změnou hodnoty V
return nejlepší  $\theta$ 
```

● Příklad: $\{\neg C, \neg A \vee \neg B \vee C, \neg A \vee D \vee E, \neg B \vee \neg C\}$

- iniciální přiřazení (všechny proměnné mají hodnotu 1) nesplňuje první a poslední klauzuli
- změna A, D, E nemá vliv na počet nesplněných klauzulí
- změna C na 0

Algoritmus GSAT

```
procedure GSAT(A,MaxPokusu,MaxZmen)  (A je CNF formule)
for i := 1 to MaxPokusu do
   $\theta$  := náhodné ohodnocení proměnných
  for j := 1 to MaxZmen do
    if A je splnitelná pomocí  $\theta$  then
      return  $\theta$ 
  V := proměnná, jejíž změna hodnoty
    nejvíce sníží počet nesplněných klauzulí
   $\theta$  := přiřazení, které se liší od  $\theta$  změnou hodnoty V
return nejlepší  $\theta$ 
```

● Příklad: $\{\neg C, \neg A \vee \neg B \vee C, \neg A \vee D \vee E, \neg B \vee \neg C\}$

- iniciální přiřazení (všechny proměnné mají hodnotu 1) nespĺňuje první a poslední klauzuli
- změna A, D, E nemá vliv na počet nespĺněných klauzulí
- změna C na 0 splní první i poslední klauzuli ale nespĺní $\neg A \vee \neg B \vee C$ (evaluace=1)
- změna B

Algoritmus GSAT

```
procedure GSAT(A,MaxPokusu,MaxZmen)  (A je CNF formule)
for i := 1 to MaxPokusu do
   $\theta$  := náhodné ohodnocení proměnných
  for j := 1 to MaxZmen do
    if A je splnitelná pomocí  $\theta$  then
      return  $\theta$ 
  V := proměnná, jejíž změna hodnoty
    nejvíce sníží počet nesplněných klauzulí
   $\theta$  := přiřazení, které se liší od  $\theta$  změnou hodnoty V
return nejlepší  $\theta$ 
```

● Příklad: $\{\neg C, \neg A \vee \neg B \vee C, \neg A \vee D \vee E, \neg B \vee \neg C\}$

- iniciální přiřazení (všechny proměnné mají hodnotu 1) nespĺňuje první a poslední klauzuli
- změna A, D, E nemá vliv na počet nespĺněných klauzulí
- změna C na 0 splní první i poslední klauzuli ale nespĺní $\neg A \vee \neg B \vee C$ (evaluace=1)
- změna B má evaluaci 1 také, pokud ale změníme C a pak B , pak dostáváme evaluaci 0

Heuristiky pro GSAT

- GSAT lze kombinovat s různými heuristikami, které zvyšují jeho efektivitu
 - obzvláště při řešení strukturovaných problémů
- Použití **náhodné procházky spolu s minimalizací konfliktů**
- **Vážení klauzulí**
 - některé klauzule zůstávají po řadu iterací nesplněné, klauzule tedy mají různou důležitost
 - splnění „těžké“ klauzule lze preferovat přidáním váhy
 - váhu může systém odvodit
 - na začátku mají všechny klauzule stejnou váhu
 - po každém pokusu zvyšujeme váhu u nesplněných klauzulí

Heuristiky pro GSAT

- GSAT lze kombinovat s různými heuristikami, které zvyšují jeho efektivitu
 - obzvláště při řešení strukturovaných problémů
- Použití **náhodné procházky spolu s minimalizací konfliktů**
- **Vážení klauzulí**
 - některé klauzule zůstávají po řadu iterací nesplněné, klauzule tedy mají různou důležitost
 - splnění „těžké“ klauzule lze preferovat přidáním váhy
 - váhu může systém odvodit
 - na začátku mají všechny klauzule stejnou váhu
 - po každém pokusu zvyšujeme váhu u nesplněných klauzulí
- **Průměrování řešení**
 - standardně každý pokus začíná z náhodného řešení
 - společné části předchozích řešení lze zachovat
 - restartovací stav se vypočte ze dvou posledních výsledků bitovým srovnáním
stejně bity zachovány, ostatní nastaveny náhodně

Hybridní prohledávání

- Příklady kombinace lokálního prohledávání
 - prohledává úplná nekonzistentní přiřazení
a stromového prohledávání
 - rozšiřuje částečné konzistentní přiřazení

Hybridní prohledávání

● Příklady kombinace lokálního prohledávání

- prohledává úplná nekonzistentní přiřazení

a stromového prohledávání

- rozšiřuje částečné konzistentní přiřazení

1. Lokální prohledávání **před nebo po** stromovém prohledávání

např:

- před:

Hybridní prohledávání

● Příklady kombinace lokálního prohledávání

- prohledává úplná nekonzistentní přiřazení

a stromového prohledávání

- rozšiřuje částečné konzistentní přiřazení

1. Lokální prohledávání **před nebo po** stromovém prohledávání

např:

- před: lokální prohledávání nám poskytne heuristiku na uspořádání hodnot

- po:

Hybridní prohledávání

● Příklady kombinace lokálního prohledávání

- prohledává úplná nekonzistentní přiřazení

a stromového prohledávání

- rozšiřuje částečné konzistentní přiřazení

1. Lokální prohledávání **před nebo po** stromovém prohledávání např:

- před: lokální prohledávání nám poskytne heuristiku na uspořádání hodnot

- po: lokálním prohledáváním se snažíme lokálně vylepšit spočítané řešení (optimalizace)

2. **Stromové prohledávání** je doplněno lokálním prohledáváním

- v listech prohledávacího stromu i v jeho uzlech

- např. lokální prohledávání pro výběr hodnoty nebo vylepšení spočítaného řešení

3. **Lokální prohledávání** je doplněno stromovým prohledáváním

- pro výběr souseda z okolí a nebo pro prořezávání stavového prostoru

Iterativní dopředné prohledávání

- **Iterative Forward Search (IFS)**

- **Hybridní prohledávání: konstruktivní nesystematický algoritmus**

- pracuje nad **modelem s pevnými a měkkými omezujícími podmínkami**
- pevné podmínky: musí být splněny
- měkké podmínky: reprezentují účelové funkce, jejichž vážený součet je minimalizován

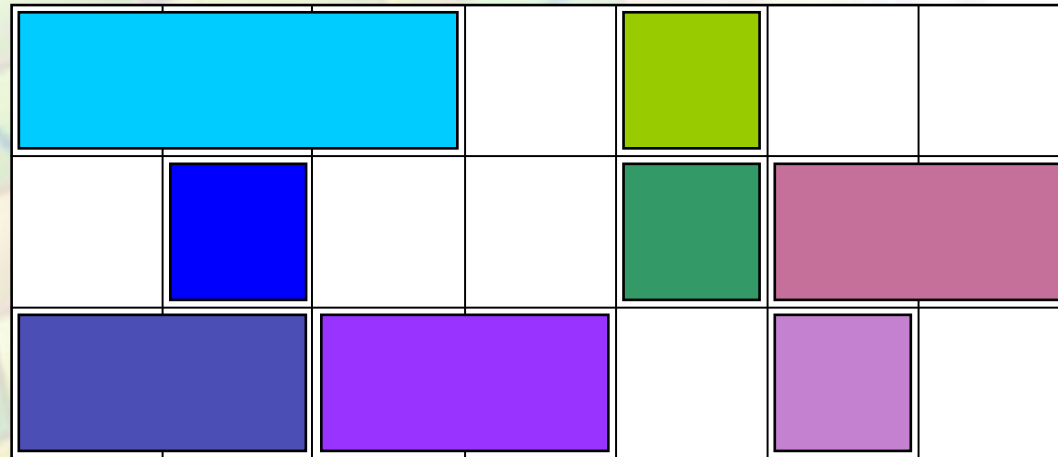
- Pracuje s **konzistentními přiřazeními**

- Základní myšlenky (blízký dynamickému backtrackingu)

- začíná s prázdným přiřazením
- vybere novou proměnnou k přiřazení
- pokud nalezne konflikt,
zruší přiřazení všech proměnných v konfliktu s vybranou proměnnou
- výběr hodnot pomocí **konfliktní statistiky**
- výběr proměnných není pro algoritmus kritický,
protože lze proměnné přiřadit opakovaně

Iterative Forward Search Algorithm

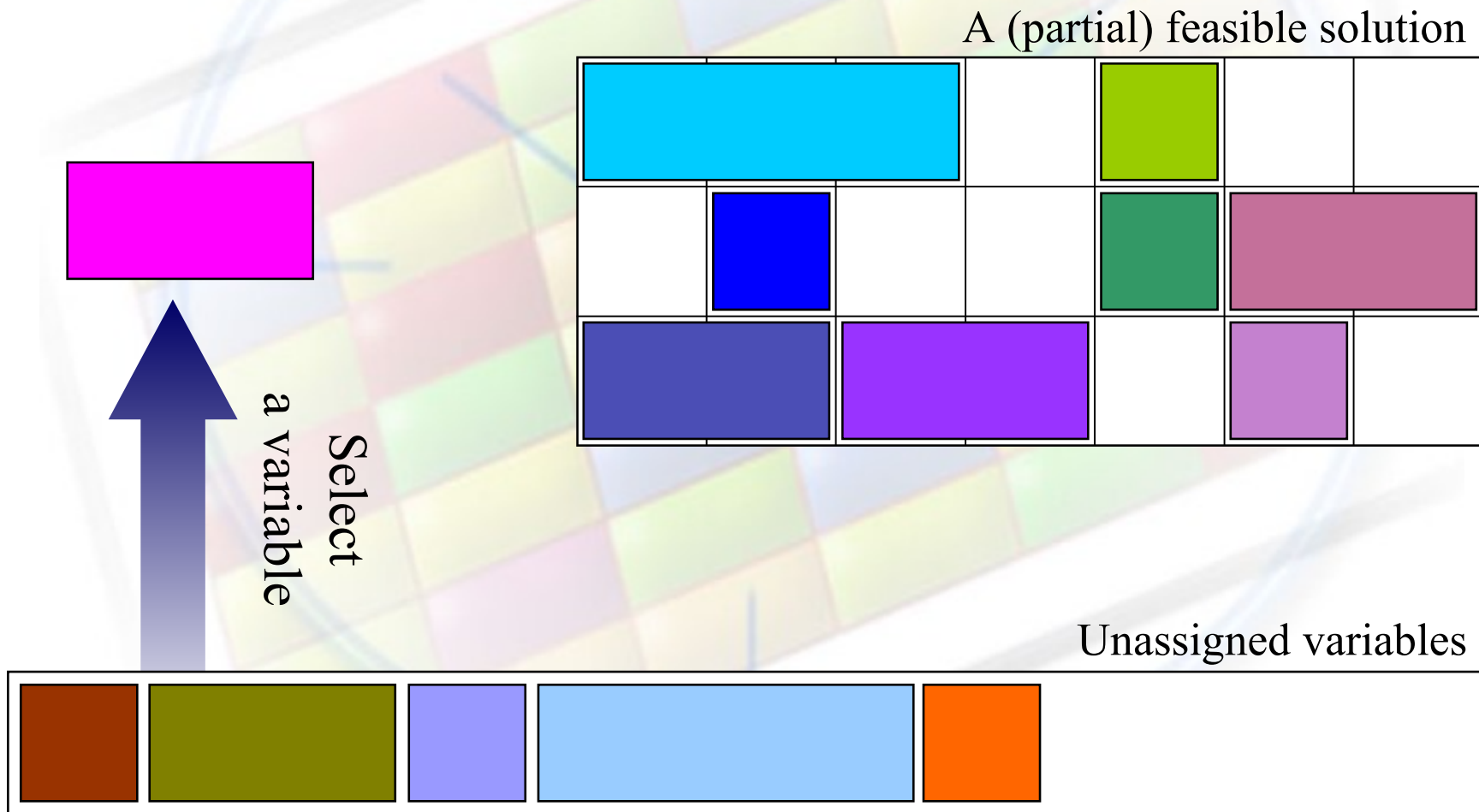
A (partial) feasible solution



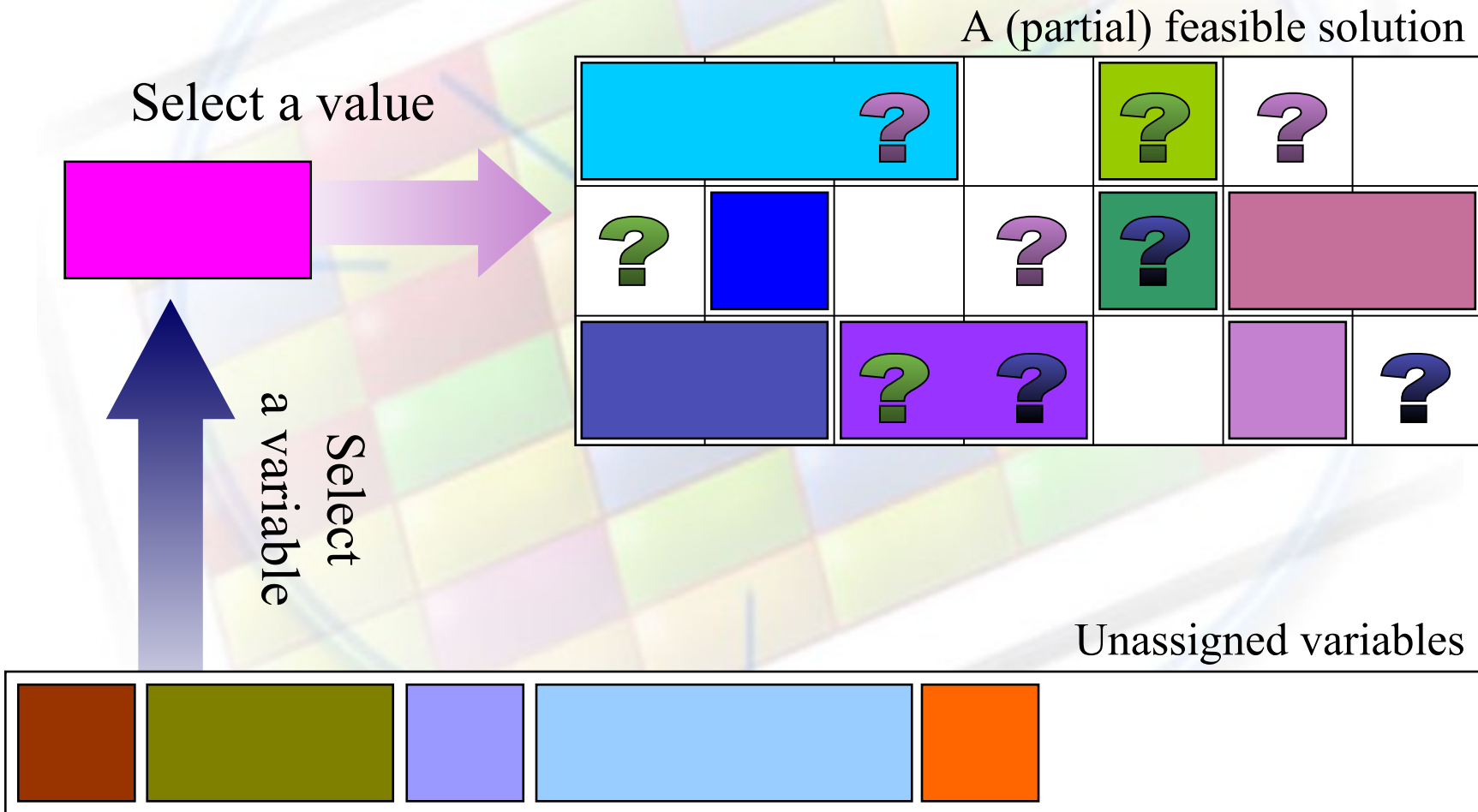
Unassigned variables



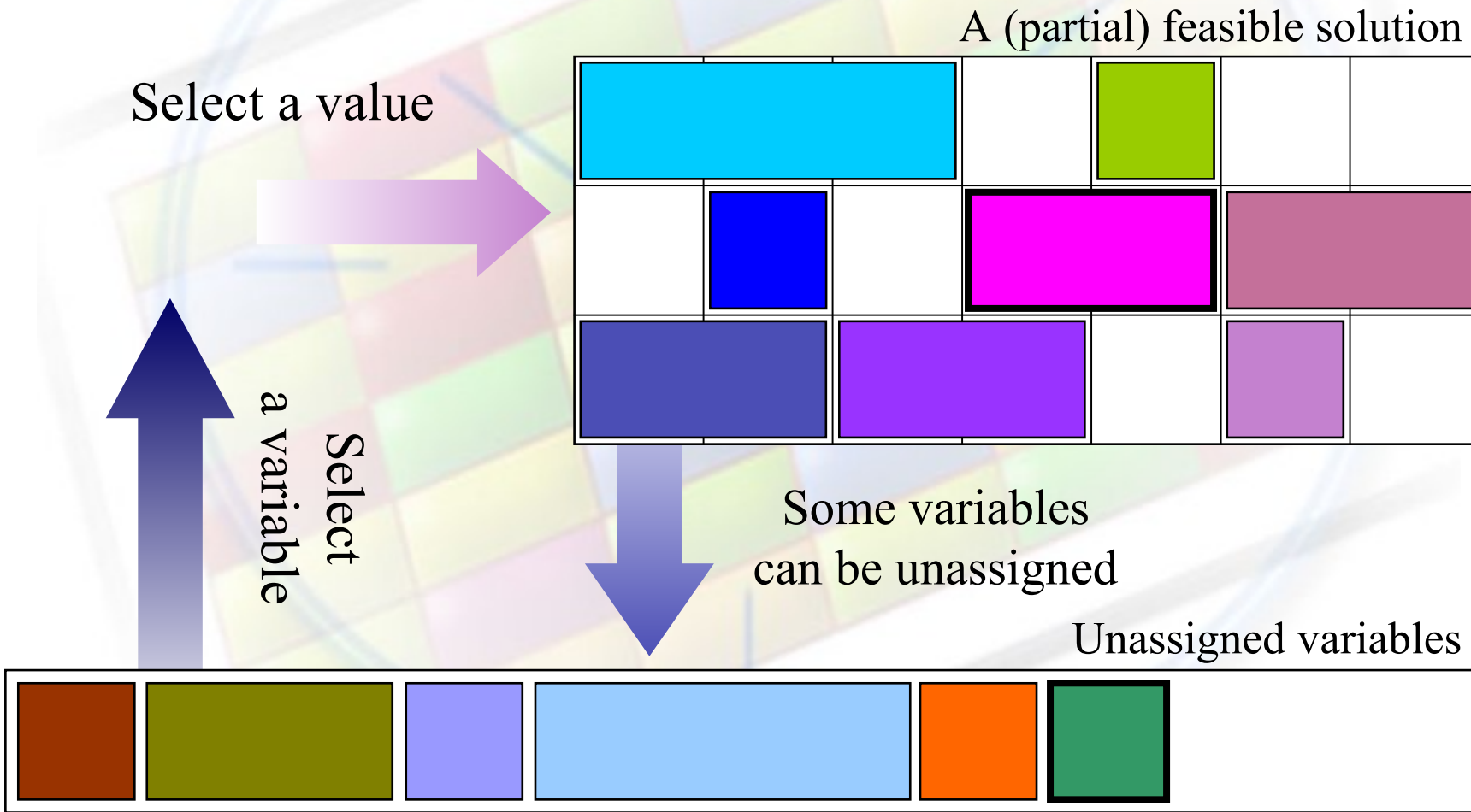
Iterative Forward Search Algorithm



Iterative Forward Search Algorithm



Iterative Forward Search Algorithm



IFS: algoritmus

```
procedure IFS()
iteration = 0;           % čítač iterací
current = ∅;           % aktuální řešení
best = ∅;              % nejlepší řešení
while canContinue (current, iteration) do
    iteration = iteration + 1;
    variable = selectVariable (current);
    value = selectValue (current, variable);
    unassign(current, conflictingVariables(current, variable, value));
    assign(current, variable, value);
    if better (current, best) then best = current
return best
end IFS procedure
```

conflictingVariables: kontroluje konzistenci tak, aby byla splněna omezení na přiřazených proměnných, a vrací konfliktní proměnné

unassign: odstraní přiřazení konfliktních proměnných

IFS: konfliktní statistika

- Předpoklad: při výběru hodnoty a proměnné A
je nutné zrušit přiřazení hodnoty b proměnné B , tj. $[A = a \rightarrow \neg B = b]$
- V průběhu výpočtu si tedy lze pamatovat:

$$A = a \Rightarrow 3 \times \neg B = b, \quad 4 \times \neg B = c, \quad 1 \times \neg C = a, \quad 120 \times \neg D = a$$

IFS: konfliktní statistika

- Předpoklad: při výběru hodnoty a proměnné A
je nutné zrušit přiřazení hodnoty b proměnné B , tj. $[A = a \rightarrow \neg B = b]$

- V průběhu výpočtu si tedy lze pamatovat:

$$A = a \Rightarrow 3 \times \neg B = b, \quad 4 \times \neg B = c, \quad 1 \times \neg C = a, \quad 120 \times \neg D = a$$

- **Při výběru hodnoty**

- vybíráme hodnoty s nejnižším počtem konfliktů vážených jejich frekvencí

- konflikt započítáme pouze tehdy, pokud to vede k odstranění přiřazení

- př. $A = a \Rightarrow 3 \times \neg B = b, \quad 4 \times \neg B = c, \quad 1 \times \neg C = a, \quad 120 \times \neg D = a$

$$A = b \Rightarrow 1 \times \neg B = a, \quad 3 \times \neg B = b, \quad 2 \times \neg C = a$$

Máme přiřazení $B = c, C = a, D = b$ a vybíráme hodnotu pro A :

- necht' A/a vede ke konfliktu s B/c :

IFS: konfliktní statistika

- Předpoklad: při výběru hodnoty a proměnné A
je nutné zrušit přiřazení hodnoty b proměnné B , tj. $[A = a \rightarrow \neg B = b]$

- V průběhu výpočtu si tedy lze pamatovat:

$$A = a \Rightarrow 3 \times \neg B = b, \quad 4 \times \neg B = c, \quad 1 \times \neg C = a, \quad 120 \times \neg D = a$$

- **Při výběru hodnoty**

- vybíráme hodnoty s nejnižším počtem konfliktů vážených jejich frekvencí

- **konflikt započítáme pouze tehdy, pokud to vede k odstranění přiřazení**

- př. $A = a \Rightarrow 3 \times \neg B = b, \quad 4 \times \neg B = c, \quad 1 \times \neg C = a, \quad 120 \times \neg D = a$

$$A = b \Rightarrow 1 \times \neg B = a, \quad 3 \times \neg B = b, \quad 2 \times \neg C = a$$

Máme přiřazení $B = c, C = a, D = b$ a vybíráme hodnotu pro A :

- necht' A/a vede ke konfliktu s B/c : vyhodnoceno jako 4

· není konflikt s C/a , tak se nezapočítává

- necht' A/b vede ke konfliktu s C/a :

IFS: konfliktní statistika

- Předpoklad: při výběru hodnoty a proměnné A
je nutné zrušit přiřazení hodnoty b proměnné B , tj. $[A = a \rightarrow \neg B = b]$

- V průběhu výpočtu si tedy lze pamatovat:

$$A = a \Rightarrow 3 \times \neg B = b, \quad 4 \times \neg B = c, \quad 1 \times \neg C = a, \quad 120 \times \neg D = a$$

- **Při výběru hodnoty**

- vybíráme hodnoty s nejnižším počtem konfliktů vážených jejich frekvencí

- **konflikt započítáme pouze tehdy, pokud to vede k odstranění přiřazení**

- př. $A = a \Rightarrow 3 \times \neg B = b, \quad 4 \times \neg B = c, \quad 1 \times \neg C = a, \quad 120 \times \neg D = a$

$$A = b \Rightarrow 1 \times \neg B = a, \quad 3 \times \neg B = b, \quad 2 \times \neg C = a$$

Máme přiřazení $B = c, C = a, D = b$ a vybíráme hodnotu pro A :

- necht' A/a vede ke konfliktu s B/c : vyhodnoceno jako 4

- není konflikt s C/a , tak se nezapočítává

- necht' A/b vede ke konfliktu s C/a : vyhodnoceno jako 2

- tj. vybereme hodnotu b pro proměnnou A