Cvičení 11

Programování
s omezujícími podmínkami

Roman Barták
Katedra teoretické informatiky a matematické logiky

roman.bartak@mff.cuni.cz
http://ktiml.mff.cuni.cz/~bartak

Základní prohledávací procedura

```
label([]).
label(Variables):-

  select_variable(Variables,V,Rest),
  !,
  choice_point(V),
  (var(V) ->
     label([V|Rest])
  ;  label(Rest)).
```

- Jednoduchá enumerace:

```
select_variable([H|T],H,T).
choice_point(V) :- indomain(V).
```


DBS

Depth-bounded Backtrack Search
- omezená hloubka, kde se prozkoumávají alternativy

```
dbs_search([],_).
dbs_search([X|RestVariables],Depth):-
  (Depth>0 ->
    NewDepth is Depth-1,
    assign_value(X)
  ;
    NewDepth = 0,
    once(assign_value(X))
  ),
  dbs_search(RestVariables,NewDepth).
```

zakázané alternativy
vrátí pouze první řešení (pokud existuje), alternativní odpovědi nejsou povoleny

BBS

Bounded Backtrack Search
- omezený počet návratů

```
bbs_search(Variables,Limit):-
  bb_put(limit,Limit),
  bb_put(stage,fw),
  bbs(Variables).

bbs([]).
bbs([X|RestVariables]):-
  (bbs_assign_value(X) ; bb_put(stage,bw),fail),
  bbs(RestVariables).

bbs_assign_value(X):-
  assign_value(X),
  bb_update(stage,Stage,fw),
  (Stage=fw -> true
  ;  bb_get(limit,L), NL is L-1, bb_put(limit,NL),
     (NL>0 -> true ; !,fail)
  ).
```

```
assign_value(X):-
  indomain(X).
```

trik
na blackboardu označíme, že jsme se začali vracet

IB

Iterative Broadening

- omezený počet alternativ v každém bodě volby

```
ib_search(Variables,Width):-
  bb_put(width,Width),
  ib(Variables,Width).

ib([],_).
ib([X|RestVariables],Width):-
  bb_update(width,TW,Width),
  (ib_assign_value(X) ; bb_put(width,TW),!,fail),
  ib(RestVariables,Width).

ib_assign_value(X):-
  assign_value(X),
  bb_get(width,RestWidth),
  (RestWidth=0 -> !,fail
  ;   NewW is RestWidth-1, bb_put(width,NewW)
  ).
```

trik
počet zbývajících hodnot předchozí proměnné si pamatujeme v TW

Optimalizace

přímočaře

- Minimalizuj/maximalizuj hodnotu dané proměnné
  - typicky máme podmínku `X#=ObjectiveFunction`
  - propagace z ObjectiveFunction do X odpovídá heuristické funkci h odhadující hodnotu objektivní funkce
- **Přímočará metoda** pro minimalizaci X:
  - zkus najít řešení s dolní mezí pro X
  - v případě neúspěchu zvětši hodnotu X o jedna

```
minimizeSimple(Vars,X):-
  fd_min(X,X),
  label(Vars),!.
minimizeSimple(Vars,X):-
  fd_min(X,Min),
  X#>Min,
  minimizeSimple(Vars,X).
```

poznámka
hledá řešení pro dolní mez proměnné X

B&B v Prologu

Enumeraci lze upravit na **branch and bound.**
Mez je uložena na blackboardu a kontrolována při každém přiřazení.

```
minimizeBB(Vars,X,InitialBound):-
  bb_put(bound,InitialBound),     % save upper bound
  minBB(Vars,Vars,X).
minimizeBB(Vars,_,_):-
  bb_get(best,Vars).              % restore best solution

minBB([],AllVars,X):-             % all variables known
  bb_put(bound,X),                % save new upper bound
  bb_put(best,AllVars),           % save best solution
  fail.                           % explore alternatives
minBB([H|Rest],AllVars,X):-
  indomain(H),                    % assign a value
  bb_get(bound,Bound),
  fd_min(X,MinX),
  MinX<Bound,                     % check bound
  minBB(Rest,AllVars,X).
```


B&B s půlením

- Pokud je doména minimalizované proměnné konečná, můžeme použít techniku půlení domény této proměnné.

```
minimizeBBsplit(Vars,X):-
  (var(X) ->
    fd_min(X,MinX),fd_max(X,MaxX),
    Middle is integer((MinX+MaxX)/2),
    ((X#=<Middle, \+ \+ label(Vars)) ->
       true
    ;  X#>Middle
    ),!,
    minimizeBBsplit(Vars,X)
  ;
    label(Vars)
  ).
```

dvojitá negace
zjistí, zda lze ohodnotit proměnné Vars, ale tyto proměnné nechá volné

poznámka
postupně vrací všechna optimální řešení, tj. řešení se stejnou hodnotou objektivní funkce