Improved Pattern Selection for PDB Heuristics in Classical Planning (Extended Abstract)

Sascha Scherrer Florian Pommerening Martin Wehrle
University of Basel, Switzerland

iPDB (Haslum et al. 2007)

- State-of-the-art pattern selection algorithm
- Selects patterns (sets of variables) using hill-climbing search in the space of pattern collections
- *Canonical heuristic* of resulting pattern collection used in A*-search

Local Maxima in iPDB

- Hill-climbing can terminate early in local maximum
  - No *extension with one variable* has sufficient improvement
  - Well-known but unaddressed problem (already pointed out by Haslum et al. (2007))

iPDB with Variable Neighborhood Search

- Addresses the problem of local maxima
- Based on variable neighborhood search (Mladenovic and Hansen 1997)
- Looks for successor collections of increasing size
- Extends existing candidate patterns by further causally related variables
- Resets candidate collection once an improving candidate is found
- Anytime algorithm: runs as long as resources are available
  - We limit resources to stop the hill-climbing

Pseudocode (iPDB)

```
function generate-candidates(𝒫) :
  Candidates := ∅
  for P ∈ 𝒫 :
    for V ∈ P :
      for V′ ∈ causally-related(V) \ P :
        Candidates := Candidates ∪ {P ∪ {V′}}
  return Candidates

function iPDB() :
  𝒫 := {{V_g} | V_g is a goal variable}
  Candidates := generate-candidates(𝒫)
  while True :
    S := generate-samples(1000)
    for P_C ∈ Candidates :
      improvement[P_C] := |{s ∈ S | h^{𝒫∪{P_C}}(s) > h^𝒫(s)}|
    P_best := Candidate with highest improvement
    if improvement[P_best] > threshold :
      𝒫 := 𝒫 ∪ {P_best}
      Candidates := generate-candidates(𝒫)
    else :
      return h^𝒫
```

Pseudocode (iPDB-VNS)

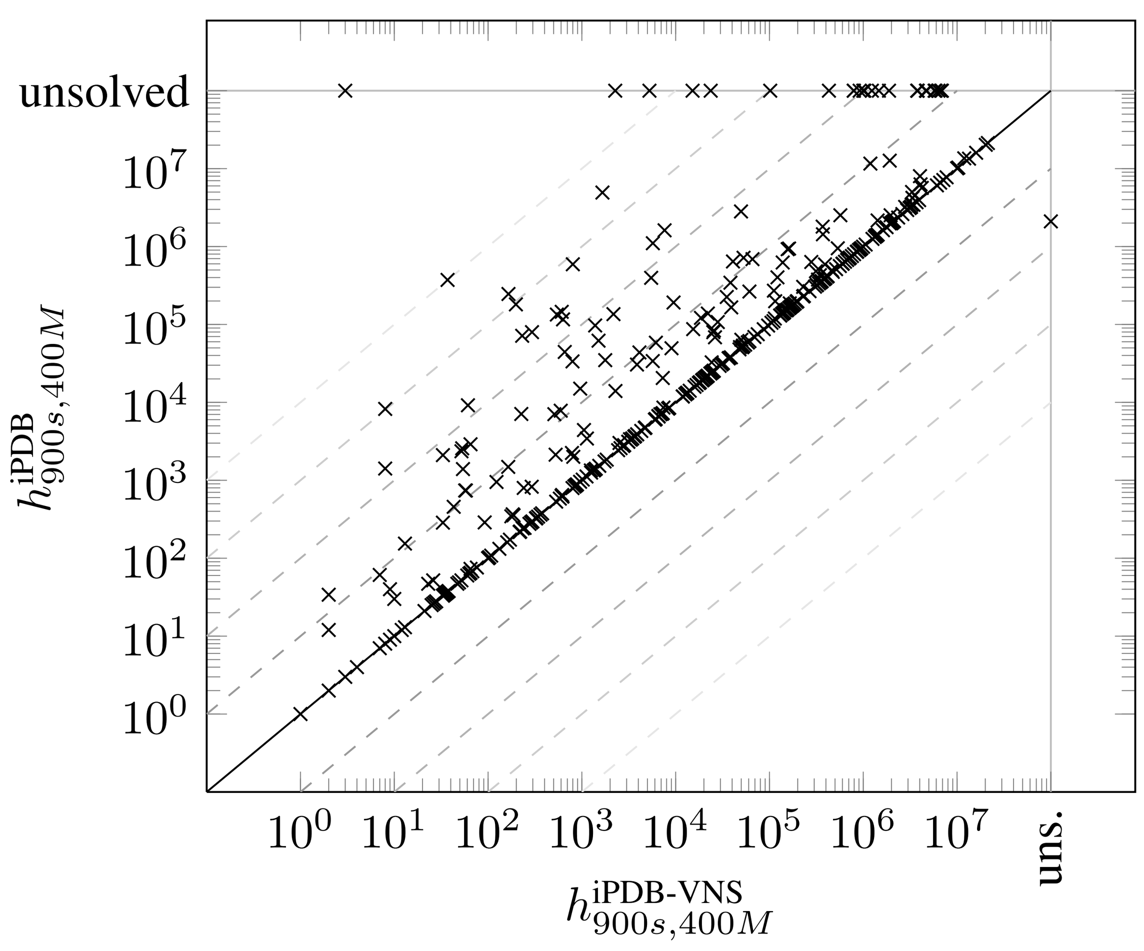
```
function generate-candidates(𝒫) :
  Candidates := ∅
  for P ∈ 𝒫 :
    for V ∈ P :
      for V′ ∈ causally-related(V) \ P :
        Candidates := Candidates ∪ {P ∪ {V′}}
  return Candidates

function iPDB-VNS() :
  𝒫 := {{V_g} | V_g is a goal variable}
  Candidates := generate-candidates(𝒫)
  while True :
    S := generate-samples(1000)
    for P_C ∈ Candidates :
      improvement[P_C] := |{s ∈ S | h^{𝒫∪{P_C}}(s) > h^𝒫(s)}|
    P_best := Candidate with highest improvement
    if improvement[P_best] > threshold :
      𝒫 := 𝒫 ∪ {P_best}
      Candidates := generate-candidates(𝒫)
    else :
      Candidates := generate-candidates(Candidates)
    if time or memory limit exceeded :
      return h^𝒫
```

Experimental Evaluation

- Evaluated on IPC tasks
  - optimal tracks 1998–2011
- Resource limits
  - Very important to limit both time and memory
  - Robust to parameter changes
- iPDB-VNS improves iPDB
  - Heuristic quality
  - Number of solved tasks
- iPDB-VNS is competitive with LM-cut

Heuristic Quality

#Expansions during A* search



Solved Tasks

	$h^{\text{iPDB}}_{\infty,\infty}$	$h^{\text{iPDB}}_{900s,400M}$	$h^{\text{iPDB-VNS}}_{900s,400M}$	$h^{\text{LM-cut}}$
Airport (50)	25	**38**	**38**	28
Depot (22)	8	8	**11**	7
Elevators (50)	36	36	**43**	40
Floortile (20)	2	2	4	**7**
Miconic (150)	55	55	55	**141**
Parcprinter (50)	22	28	28	**31**
TPP (30)	6	6	**8**	7
Transport (50)	17	17	**24**	17
Trucks (30)	8	8	**10**	**10**
Woodworking (50)	13	23	23	**29**
Sum (502)	192	221	244	**317**
Sum in other domains (894)	474	473	**481**	452
Total sum (1396)	666	694	725	**769**
Coverage score (in %)	50.72	52.68	**55.45**	53.60