Width and Serialization of Classical Planning Problems

Nir Lipovetzky[1] Héctor Geffner[1,2]

DTIC Universitat Pompeu Fabra[1]
Barcelona, Spain

ICREA[2]
Barcelona, Spain

ECAI-2012; Montpellier

Complexity of Classical Benchmarks

Planning is **NP-hard** but current planners can **solve most of benchmarks in a few seconds**

Why?

- **Tractable fragments** (Bylander, Bäckström, ...)
- **Width** notion from graphical models (Freuder, Pearl, Dechter; Amir & Engelhardt, Brafman & Domshlak, Chen & Giménez)
- **Properties of h^+** over benchmarks (Hoffmann)

Accounts however don't appear to explain well simplicity of benchmarks . . .

Our Approach

A new **width** notion and a planning algorithm **exponential in problem width**:

- Benchmark domains have **small width** when **goals** restricted to **single atoms**
- Joint goals **easy to serialize**

Suggests recipe for **hard problems:**

- **single goal** problems with **high width** (apparently no benchmark in this class)
- **multiple goal** problems that are **not easy to serialize** (e.g. Sokoban)

Contributions of Paper: Theoretical and Practical

1. A **new width** notion for planning problems and domains
2. A **proof** that many domains have **low width** when goals are **single atoms**
3. A **simple planning algorithm**, *IW*, **exponential in problem width**
4. A **blind-search planner** that combines ***IW*** and goal **serialization**, competitive with GBFS planner with h_{add}
5. A planner that integrates **new ideas** into a **best-first planner** competitive with state-of-the-art

A Simple Pruned Breadth-First Search Algorithm

Definition (novelty)

The **novelty** of a newly generated state s during a search is the **size of the smallest tuple of atoms** t that is **true** in s and **false** in all previously generated states s'. If no such tuple, the novelty of s is $n + 1$ where n is number of problem vars.

- $IW(i)$ = **breadth-first** search that **prunes** newly generated states whose $novelty(s) > i$.
- IW is a **sequence of calls** $IW(i)$ for $i = 0, 1, 2, \ldots$ over problem P until problem solved or i exceeds number of vars in problem

Iterative Width (*IW*) Algorithm: Properties

Key theoretical properties of *IW* in terms of “width” (to be defined):

- $IW(i)$ solves P **optimally** in time $O(n^i)$ if $\boldsymbol{width(P) = i}$
- *IW* solves P in time $O(n^i)$ if $width(P) = i$ but **not necessarily optimally**
  - $IW(k)$ **may solve** P as well for $k < width(P)$, with **no optimality** guarantees

n = number of problem variables

Iterative Width (*IW*) Algorithm: Experiments

- *IW*, while simple and blind, is a pretty **good algorithm** over benchmarks when goals restricted to **single atoms**
- This is no accident, **width** of benchmarks domains is **small** for such goals

We tested domains from previous IPCs. For **each instance** with *N* goal atoms, we **created** *N* **instances** with a **single goal**

- Results quite remarkable: *IW* is much better than **blind-search**, and as good as **GBFS** with h_{add}

# Instances	*IW*	*ID*	*BrFS*	$GBFS + h_{add}$
37921	91%	24%	23%	91%

What about **conjunctive** goals?

Decomposition: Serialized Iterated Width (SIW)

- Simple way to **use *IW*** for solving real benchmarks P with **joint goals** is by simple form of **"hill climbing"** over goal set G with $|G| = n$

Starting with $G_0 = \emptyset$, $s = s_0$ and $\pi_0 = \emptyset$

For $i = 1, .., n - 1$ do

1 - **Run *IW*** from $\boldsymbol{s_{i-1}}$ until a state $\boldsymbol{s_i}$ is reached such that $\boldsymbol{G_i \subseteq s_i}$ and $\boldsymbol{G_{i-1} \subseteq G_i \subseteq G}$

2 - **If** this fails, return FAILURE

3 - **Else** keep action sequence in $\boldsymbol{\pi_{i-1}}$

End For

If ***SIW*** doesn't return FAILURE, $\boldsymbol{\pi_0, \pi_1, .., \pi_{n-1}}$ is a **plan** that solves P

Serialized Iterated Width (SIW)

- ***SIW* uses *IW*** for both **decomposing** a problem into subproblems and for **solving** subproblems
- It's a **blind search** procedure, **no heuristic** of any sort, *IW* **does not even know next goal** G_i "to achieve"
- **Boolean polynomial consistency** test to check if G_i is "**consistent**" in s_i (needs to be undone later on) in step 1, else s_i skipped

More remarkable news: Blind *SIW* better than GBFS with h_{add}

Testing *SIW* Experimentally

		Serialized *IW* (*SIW*)				GBFS + h_{add}		
Domain	I	S	Q	T	M/Aw_e	S	Q	T
8puzzle	50	50	42.34	0.64	4/1.75	50	55.94	0.07
Blocks World	50	50	48.32	5.05	3/1.22	50	122.96	3.50
Depots	22	21	34.55	22.32	3/1.74	11	104.55	121.24
Driver	20	16	28.21	2.76	3/1.31	14	26.86	0.30
Elevators	30	27	55.00	13.90	2/2.00	16	101.50	210.50
Freecell	20	19	47.50	7.53	2/1.62	17	62.88	68.25
Grid	5	5	36.00	22.66	3/2.12	3	195.67	320.65
OpenStacksIPC6	30	26	29.43	108.27	4/1.48	30	32.14	23.86
ParcPrinter	30	9	16.00	0.06	3/1.28	30	15.67	0.01
Parking	20	17	39.50	38.84	2/1.14	2	68.00	686.72
Pegsol	30	6	16.00	1.71	4/1.09	30	16.17	0.06
Pipes-NonTan	50	45	26.36	3.23	3/1.62	25	113.84	68.42
Rovers	40	27	38.47	108.59	2/1.39	20	67.63	148.34
Sokoban	30	3	80.67	7.83	3/2.58	23	166.67	14.30
Storage	30	25	12.62	0.06	2/1.48	16	29.56	8.52
Tidybot	20	7	42.00	532.27	3/1.81	16	70.29	184.77
Transport	30	21	54.53	94.61	2/2.00	17	70.82	70.05
Visitall	20	19	199.00	0.91	1/1.00	3	2485.00	174.87
Woodworking	30	30	21.50	6.26	2/1.07	12	42.50	81.02
Summary	1150	819	44.4	55.01	2.5/1.6	789	137.0	91.05

Theory: Width

- *IW* is a **blind search** algorithm that manages to **exploit the structure** of existing benchmarks
- We **characterize** this structure in terms of a **new width** which we now define ...

Width: Definition

- Consider a **chain** $t_0 \rightarrow t_1 \rightarrow \ldots \rightarrow t_n$ where each t_i is a **set of atoms** from P
- A chain is **valid** if t_0 is true in Init and **all optimal plans** for t_i can be **extended into optimal plans** for t_{i+1} by adding a **single** action
- A valid chain $t_0 \rightarrow t_1 \rightarrow \ldots \rightarrow t_n$ **implies** G if all **optimal plans** for t_n are also **optimal plans** for G
- The **size** of the chain is the **size of largest** t_i in the chain

Definition (Width)

Width of P is **size of smallest** chain that **implies** goal G of P

Width: Properties

Theorem

*Blocks, Logistics, Gripper, and n-puzzle have a **bounded width** independent of problem **size** and **initial situation**, provided that goals are **single atoms**.*

- Establishing widths of benchmark **domains** for single goals possible, but tedious
- Establishing widths of **problems** automatically, as hard as optimal planning
- Yet finding **effective width** $w_e(P)$ = min i for which $IW(i)$ solves P, exponential in width(P)
- $w_e(P) \leq w(P)$

Effective Width: Experiments (Atomic Goals)

$\boldsymbol{w_e(P)}$ = min i for which $IW(i)$ solves P

Domain	I	$w_e = 1$	$w_e = 2$	$w_e > 2$
8puzzle	400	55%	45%	0%
Barman	232	9%	0%	91%
Blocks	598	26%	74%	0%
Cybersec	86	65%	0%	35%
Depots	189	11%	66%	23%
Driver	259	45%	55%	0%
Elevators	510	0%	100%	0%
Ferry	650	36%	64%	0%
Floortile	538	96%	4%	0%
Freecell	76	8%	92%	0%
Grid	19	5%	84%	11%
Gripper	1275	0%	100%	0%
Logistics	249	18%	82%	0%
Miconic	650	0%	100%	0%
Mprime	43	5%	95%	0%
Mystery	30	7%	93%	0%
NoMystery	210	0%	100%	0%
OpenSt	630	0%	0%	100%
OpenStIPC6	1230	5%	16%	79%

Domain	I	$w_e = 1$	$w_e = 2$	$w_e > 2$
ParcPrinter	975	85%	15%	0%
Parking	540	77%	23%	0%
Pegsol	964	92%	8%	0%
Pipes-NT	259	44%	56%	0%
Pipes-T	369	59%	37%	3%
PSRsmall	316	92%	0%	8%
Rovers	488	47%	53%	0%
Satellite	308	11%	89%	0%
Scanalyzer	624	100%	0%	0%
Sokoban	153	37%	36%	27%
Storage	240	100%	0%	0%
Tidybot	84	12%	39%	49%
Tpp	315	0%	92%	8%
Transport	330	0%	100%	0%
Trucks	345	0%	100%	0%
Visitall	21859	100%	0%	0%
Woodwork	1659	100%	0%	0%
Zeno	219	21%	79%	0%
Summary	**37921**	37.0%	51.3%	11.7%

Summary (so far)

***IW*:** sequence of novelty-based pruned breadth-first searches

- **Experiments:** excellent when goals restricted to atomic goals
- **Theory:** such problems have low width w and *IW* runs in time $O(n^w)$

***SIW*:** *IW* serialized, used to attain top goals one by one

- **Experiments:** *SIW* faster and better coverage and plans than GBFS planner with h_{add}

Summary (so far)

Last question: can these ideas be used to yield state-of-the-art performance; e.g., comparable with LAMA-2011?

Last: BFS(f)

Pure **best-first planner** with evaluation function:

$$f(s) = 2[novel(s) - 1] + help(s)$$

- Function combines **novelty** of s and whether action leading to s is **helpful**: $novel(s)$ ranges over $[1, 2, 3]$, $help(s)$ over $[1, 2]$, and hence $f(s)$ over $[1, \ldots, 6]$
- Ties broken by ***number of unachieved landmarks*** and $\boldsymbol{h_{add}}$ in that order
- **Novelty** of s computed by considering previously generated states s' on **same "subproblem"** (same number of unachieved landmarks)

Experimental Results for BFS(f)

		BFS(f)			PROBE			LAMA'11			FF
Domain	I	S	Q	T	S	Q	T	S	Q	T	S
Barman	20	20	174.45	281.28	20	169.30	12.93	20	203.85	**8.39**	–
Blocks	50	50	54.24	2.40	50	**43.88**	**0.23**	50	88.92	0.41	44
Cyber	30	28	39.23	70.14	24	52.85	69.22	30	37.54	576.69	4
Floortile	20	**7**	43.50	**29.52**	5	45.25	71.33	5	49.75	95.54	5
Freecell	20	20	64.39	**13.00**	20	62.44	41.26	19	68.94	27.34	20
NoMystery	20	**19**	24.33	**1.09**	5	25.17	5.47	11	24.67	2.66	4
OpenSt	30	30	125.89	40.19	30	134.14	48.89	30	130.18	**4.91**	30
ParcPrinter	30	27	35.92	6.48	28	36.40	0.26	30	37.72	0.28	30
Parking	20	17	90.46	577.30	17	146.08	693.12	**19**	87.23	**363.89**	3
Pegsol	30	30	24.20	**1.17**	30	25.17	8.60	30	25.90	2.76	30
Scanalyzer	30	27	29.37	7.40	28	25.15	**5.59**	28	27.52	8.14	30
Sokoban	30	23	220.57	125.12	25	233.48	39.63	**28**	213.00	58.24	26
Tidybot	20	18	62.94	198.22	**19**	**53.50**	35.33	16	62.31	113.00	15
Transport	30	30	107.70	55.04	30	137.17	44.72	30	108.03	94.11	29
Visitall	20	20	**947.67**	84.67	19	1185.67	308.42	20	1285.56	77.80	6
Wood.	30	30	41.13	19.12	30	41.13	15.93	30	51.57	**12.45**	17
...											
Summary	1150	**1070**	**87.93**	63.36	1052	98.71	49.94	1065	98.67	**44.35**	909

Summary (this one final)

1. A **new width** notion for planning problems and domains
2. A **proof** that many domains have **low width** when goals are **single atoms**
3. A simple planning algorithm, ***IW***, **exponential in problem width**
4. A **blind-search** planner ***SIW*** that combines ***IW*** and **goal serialization**, competitive with GBFS planner with h_{add}
5. A **best-first** planner that **integrate new ideas** and competes with LAMA-2011