

Flow-based Heuristics for Optimal Planning: Landmarks and Merges

Blai Bonet

Universidad Simon Bolivar

Caracas, Venezuela

bonet@ldc.usd.ve



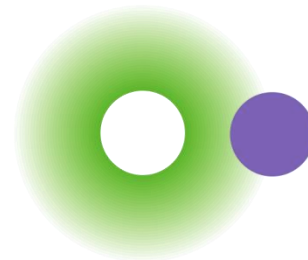
USB

Menkes van den Briel

NICTA & Australian National University

Canberra, Australia

menkes@nicta.com.au



NICTA

Outline

- Background and motivation
- Base model
- Strengthening the base model
 - Adding constraints derived from landmarks
 - Adding constraints derived from merges
- Results
- Conclusions

Background

An LP-Based Heuristic for Optimal Planning. Van den Briel, Benton, Kambhampati, and Vossen. CP2007.

An Admissible Heuristic for SAS+ Planning Obtained from the State Equation. Bonet. IJCAI2013.

An LP-Based Heuristic for Optimal Planning

Menken van den Briel¹, J. Benton², Subbarao Kambhampati³,
and Thomas Vossen¹

¹ Arizona State University, Department of Industrial Engineering,

Tempe AZ, 85287, USA

[menken,j.benton,t.vossen@asu.edu]

² Department of Computer Science and Engineering,
Florida State University, Tallahassee, FL 32306, USA

[benton@fsu.edu]

³ University of Colorado, Leeds School of Business,
Boulder CO, 80509, USA

[vossen@colorado.edu]

Abstract. One of the most successful approaches in automated planning is to use heuristic state-space search. A popular heuristic that is used by a number of state-space planners is based on relaxing the planning task by ignoring the delete effects of the actions. In several planning domains, however, this relaxation produces rather weak estimates to guide search effectively. We present a relaxation using (integer) linear programming that respects delete effects but ignores action ordering, which in a number of problems provides better distance estimates. Moreover, our approach can be used as an admissible heuristic for optimal planning.

Keywords: Automated planning, improving admissible heuristics, optimal relaxed planning

1 Introduction

Many heuristics that are used to guide heuristic state-space search planners are based on constructing a relaxation of the original planning problem that is easier to solve. The idea is to use the solution to the relaxed problem to guide search for the solution to the original problem. A popular relaxation that has been implemented by several planning systems, including UNPOP [Katz, 1992], HSP [Bonet, 1995], and FF [Helfmuth, 2006], involves using relaxed actions in which the delete effects of the original actions are ignored.

For example, FF estimates the distance between an intermediate state and the goal by creating a planning graph [Gulwani, 1995] using relaxed actions. From this graph, FF extracts in polynomial time a relaxed plan whose corresponding plan length is used as an inadmissible, but effective, distance estimate. One can transform this approach into an admissible heuristic by finding the optimal relaxed plan, also referred to as h^* [Gulwani, 1995], but computing such a plan is NP-Complete [Gulwani, 1995]. In order to extract the optimal relaxed plan one must extend the relaxed planning graph to level off [Gulwani, 1995] so that all reachable actions can be considered.

Although ignoring delete effects turns out to be quite effective for many planning domains, there are some obvious weaknesses with FF's relaxed plan heuristic. For example, in a relaxed plan no atom changes more than once, if an

© Siemens (Eds.) CP 2007, LNCS 4744, pp. 60–69, 2007.
© Springer-Verlag Berlin Heidelberg 2007

Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence

An Admissible Heuristic for SAS⁺ Planning Obtained from the State Equation

Blai Bonet

Departamento de Computación

Universidad Simón Bolívar

Caracas, Venezuela

bonet@dcc.usb.ve

Abstract

Domain-independent optimal planning has seen important breakthroughs in recent years with the development of tractable and informative admissible heuristics, suitable for planners based on forward state-space search. These heuristics allow planners to optimally solve an important number of benchmark problems, including problems that are quite involved and difficult for the layman. In this paper we present a new admissible heuristic that is obtained from the state equation associated to the Petri-net representation of the planning problem. The new heuristic, that does not fall into one of the four standard classes, can be computed in polynomial time and is competitive with the current state of the art for optimal planning, as empirically demonstrated over a large number of problems, mainly because it often shows an improved quality-to-cost ratio. The new heuristic applies to SAS⁺ planning tasks with arbitrary non-negative action costs.

1 Introduction

Domain-independent planning deals with the development of planners for solving unknown input problems that are specified in a high-level description language. Domain-independent means that the planner has not other information about the problem than the one it can infer from its description. The general interest is in building “satisficing” planners whose task is to compute a valid solution, while “optimal” planners are required to output solutions of minimum cost.

Recent years have witnessed a remarkably progress in optimal planning in terms of the type and size of problems that can be dealt with. Current state-of-the-art optimal planners perform forward search in state space using A^* with an admissible heuristic in order to meet the optimality requirement, and thus the basic difference between optimal planners is the heuristics that are used to guide the search.

More recently, best optimal planners correspond to systems that use a “portfolio” of heuristics that are scheduled according to features of the input problem. However, a portfolio is a collection of base heuristics and hence its intrinsic limitations is a function of the heuristics in the portfolio.

Helfmuth and Domschik [2009] observed that most of the well-known heuristics for optimal (and also for satisficing) planning fall in one of four categories: delete-relaxation heuristics that try to estimate the optimal cost h^* of the delete-relaxed problem [Bonet and Geffner, 2001; Hoffmann and Nebel, 2001; Coles et al., 2008], abstraction heuristics that correspond to the optimal costs of a simplified yet informative abstraction of the problem [Helmert, 2001; Haslum et al., 2007; Helmert et al., 2007; Katz and Domschik, 2008], heuristics based on critical paths such as the family h^m [Haslum and Geffner, 2000], and landmark heuristics that compute sets of facts or actions that every plan must achieve or execute from which the cost of an optimal plan can be estimated [Hoffmann et al., 2004; Richter and Westphal, 2010; Karas and Domschik, 2009; Helmert and Domschik, 2009; Bonet and Helmert, 2010]. Currently, the most successful heuristic is the LM-cut heuristic [Helmert and Domschik, 2009] that approximates h^* quite well on some domains, but that can also be thought as a cost-partitioning heuristic or as a landmark heuristic. LM-cut is always bounded by h^* and hence, ineffective at assessing the need to apply a fixed action multiple times for reaching the goal from a given state. On the other hand, abstraction heuristics are not delete-relaxation heuristics and have the potential to overcome this and other limitations [Helmert and Domschik, 2009], yet to this date, the full potential of such heuristics have not been realized in a domain-independent manner and thus do not stand out as the best general heuristics.

In this paper we define a new heuristic for optimal planning that targets problems specified in SAS⁺ involving multi-valued variables, arbitrary action costs, but without conditional effects. This is a class of problems that subsumes STRIPS and is as general as the classes of problems handled by other state-of-the-art heuristics. The heuristic is simple to formulate and is related to the so-called state equation for the Petri-net that is obtained from the planning problem in a standard way. For computing it, though, a small and simple linear programming problem needs to be solved. Despite the overhead involved in this computation, the new heuristic is competitive with the best current heuristics, and in some domains, it is able to solve more problems than any other heuristic. This is due to an improved quality-to-cost ratio on some domains, as observed in our experiments. We also mention how the new heuristic can be improved in three different

Flow-based linear programming model

- **SAS+ planning** task with action cost $P = \langle V, A, s_0, s_*, c \rangle$
 - V set of variables each with a finite domain D_X
 - A set of actions $\langle Pre, Post, Prev \rangle$
 - s_0 initial state
 - s_* goal state (partial valuation)
 - c non-negative action cost
- **Domain transition graph** (DTG)
 - Node for each value in D_X
 - Labeled arc for each transition T_X
$$T_X = \{(x, a, x') : X = x \in Pre, X = x' \in Post\} \cup \{(\perp, a, x') : X \notin Var(Pre), X = x' \in Post\} \cup \{(x, a, x) : X = x \in Prev\}$$

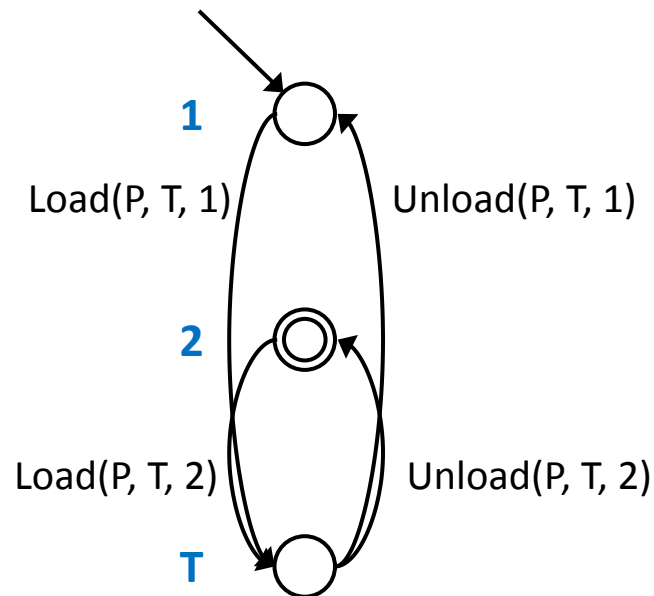
Example



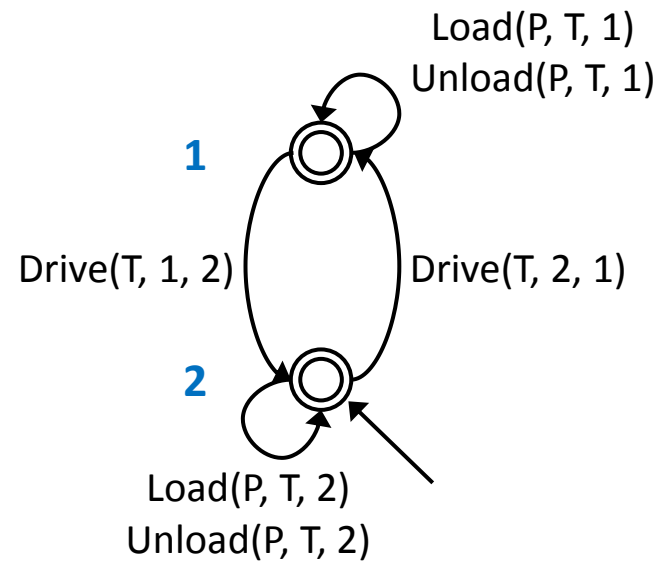
1



2



DTG(P)



DTG(T)

Flow-based linear programming model

- A **flow** for a planning task $P = \langle V, A, s_0, s_*, c \rangle$ is a function $f: A \rightarrow \mathbb{R}^+$ mapping action labels into non-negative real numbers

- Minimize

$$\sum_{a \in A} c(a) f(a)$$

- Subject to

$$\sum_{(x', a, x) \in T_X} f(a) - \sum_{(x, a, x') \in T_X} f(a) = 0$$

Flow-based linear programming model

- Minimize

$$\sum_{a \in A} c(a)f(a)$$

- Subject to

$$\sum_{(x',a,x) \in T_X} f(a) - \sum_{(x,a,x') \in T_X} f(a) = \begin{cases} 1 & \text{if } x \notin s_0, x \in s_* \\ -1 & \text{if } x \in s_0, x \notin s_* \\ 0 & \text{otherwise} \end{cases}$$

Flow-based linear programming model

- Minimize

$$\sum_{a \in A} c(a) f(a)$$

- Subject to

$$\sum_{(x', a, x) \in T_X} f(a) - \sum_{(x, a, x') \in T_X} f(a) \geq \begin{cases} 1 & \text{if } x \notin s_0, x \in s_* \\ -1 & \text{if } x \in s_0, x \notin s_* \\ 0 & \text{otherwise} \end{cases}$$

Due to incomplete actions
(see paper for details)

Theorem 1. The solution to the base model h_{base} can be used as an admissible heuristic

Flow-based heuristic

- Base heuristic is effective and efficient

– Bonet. IJCAI2013

Domain	LM-cut	base
Airport(50)	28	20
Barman-opt11(20)	4	4
Blocks(35)	28	28
Depot(22)	7	7
Driverlog(20)	13	11
Elevators-opt08(30)	22	9
Elevators-opt11(20)	17	7
Floortile-opt11(20)	6	4
Freecell(80)	15	35
Grid(5)	2	1
Gripper(20)	7	6
Logistics00(28)	20	15
Logistics98(35)	6	2
Miconic(150)	141	50
Mprime(35)	22	18
Mystery(30)	19	15
Nomystery-opt11(20)	14	10
Openstacks-opt08(30)	20	15
Openstacks-opt11(20)	15	7
Openstacks(30)	7	7
Parcprinter-08(30)	18	28
Parcprinter-opt11(20)	13	20

Domain	LM-cut	base
Parking-opt11(20)	2	1
Pathways-noneg(30)	5	4
Pegsol-08(30)	27	28
Pegsol-opt11(30)	17	18
Pipesworld-notank(50)	17	15
Pipesworld-tank(50)	11	10
Psr-small(50)	49	50
Rovers(40)	7	6
Satellite(36)	7	6
Scanalyzer-08(30)	15	13
Scanalyzere-opt11(20)	12	10
Sokoban-opt08(30)	28	16
Sokoban-opt11(20)	20	15
Tidybot-opt11(20)	13	5
Tpp(30)	6	8
Transport-opt08(30)	11	10
Transport-opt11(20)	6	6
Trucks(30)	10	9
Visitall-opt11(20)	10	17
Woodworking-opt08(30)	16	12
Woodworking-opt11(20)	11	7
Zenotravel(20)	12	9
Total(1396)	756	594

#Problems solved in 30mins

Flow-based heuristic

– Van den Briel, Benton, Kambhampati, and Vossen. CP2007

Problem	Opt	$h+$	LP–
logitstics4-0	20	19	16
logitstics4-1	19	17	14
logitstics4-2	15	13	10
logitstics5-1	17	15	12
logitstics5-2	8	8	6
logistics6-1	14	13	10
logistics6-9	24	21	18
logistics12-0	42	39	32
logistics15-1		66	54
driverlog1	7	6	3
driverlog2	19	14	12
driverlog3	12	11	8
driverlog4	16	12	11
driverlog6	11	10	8
driverlog7	13	12	11
driverlog13	26	21	15
driverlog19		89	60
driverlog20		84	60

Problem	Opt	$h+$	LP–
zenotravel1	1	1	1
zenotravel2	6	4	3
zenotravel3	6	5	4
zenotravel4	8	6	5
zenotravel5	11	11	8
zenotravel6	11	11	8
zenotravel13	26	23	18
zenotravel19		62	46
zenotravel20			50
tpp1	5	4	3
tpp2	8	7	6
tpp3	11	10	9
tpp4	14	13	12
tpp5	19	17	15
tpp6	25	21	21
tpp28			150
tpp29			
tpp30			174

Heuristic value at initial state

Flow-based heuristic

- Base heuristic is weak and can be improved significantly
 - **Van den Briel, Benton, Kambhampati, and Vossen. CP2007**

Problem	Opt	$h+$	LP–	LP
logitstics4-0	20	19	16	20
logitstics4-1	19	17	14	19
logitstics4-2	15	13	10	15
logitstics5-1	17	15	12	17
logitstics5-2	8	8	6	8
logistics6-1	14	13	10	14
logistics6-9	24	21	18	24
logistics12-0	42	39	32	42
logistics15-1		66	54	67
driverlog1	7	6	3	7
driverlog2	19	14	12	19
driverlog3	12	11	8	11
driverlog4	16	12	11	16
driverlog6	11	10	8	11
driverlog7	13	12	11	13
driverlog13	26	21	15	24
driverlog19		89	60	97
driverlog20		84	60	90

Problem	Opt	$h+$	LP–	LP
zenotravel1	1	1	1	1
zenotravel2	6	4	3	6
zenotravel3	6	5	4	6
zenotravel4	8	6	5	8
zenotravel5	11	11	8	11
zenotravel6	11	11	8	11
zenotravel13	26	23	18	24
zenotravel19		62	46	67
zenotravel20			50	69
tpp1	5	4	3	5
tpp2	8	7	6	8
tpp3	11	10	9	11
tpp4	14	13	12	14
tpp5	19	17	15	19
tpp6	25	21	21	25
tpp28			150	
tpp29				
tpp30			174	

Heuristic value at initial state

Motivation

- Base heuristic is effective and efficient yet can be improved significantly
 - Adding constraints derived from **landmarks**
Bonet. IJCAI2013
 - Adding constraints derived from **merges**
Van den Briel, Benton, Kambhampati, and Vossen. CP2007

Landmarks

- A landmark $L \subseteq A$ for a planning task $P = \langle V, A, s_0, s_*, c \rangle$ is a subset of actions such that every plan P contains at least one action in L
- A set of landmarks \mathcal{L} (**state dependent**) introduces the following constraints

$$\sum_{a \in L} f(a) \geq 1 \quad \forall L \in \mathcal{L}$$

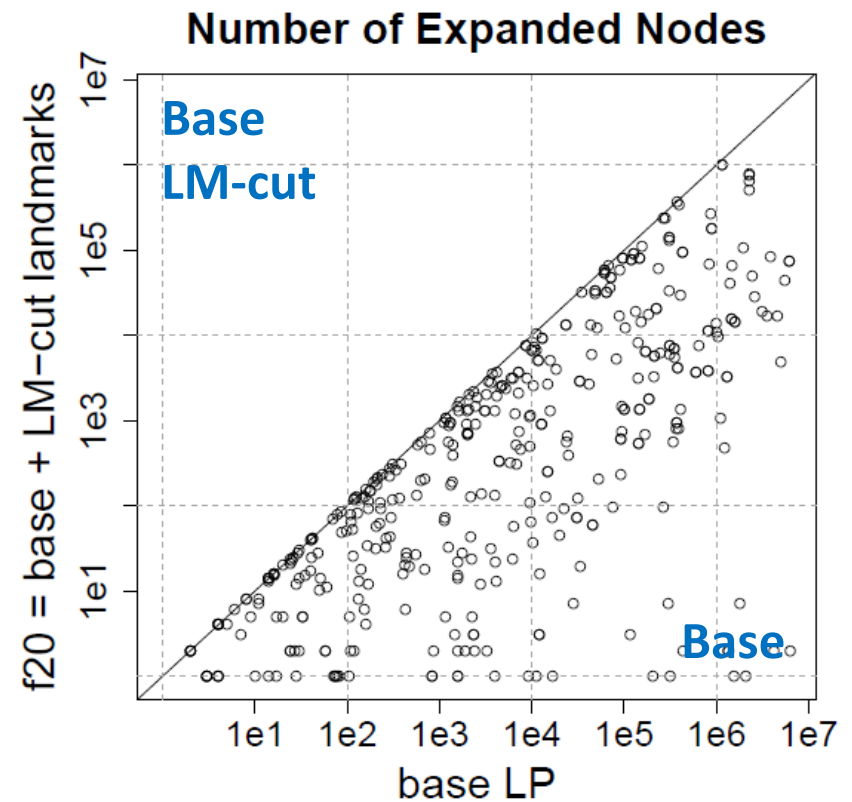
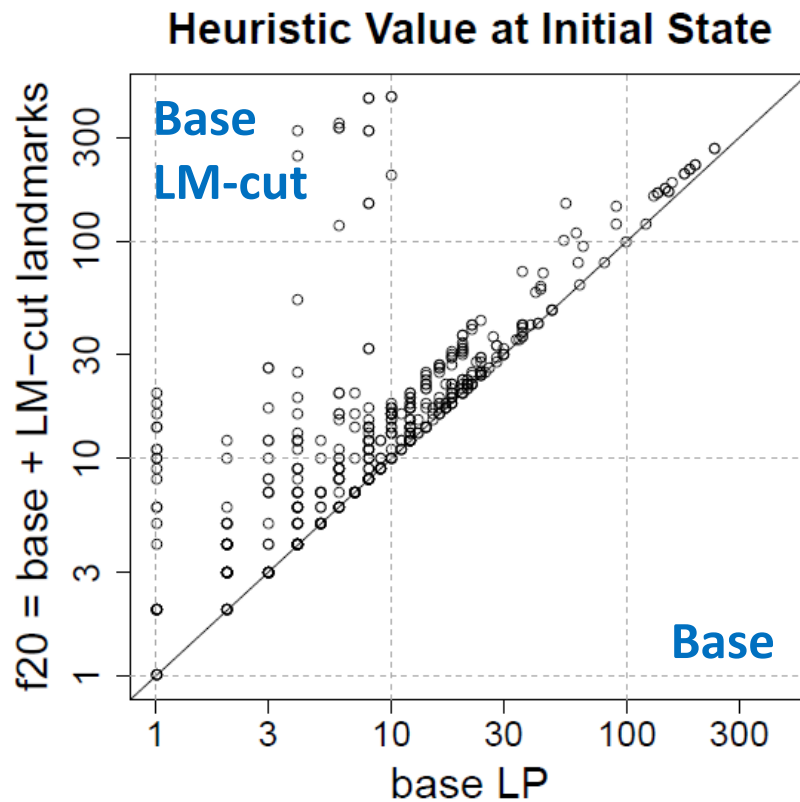
Theorem 2. The set of landmark constraints is admissible. As a result, the solution to the base model with landmark constraints $h_{base}^{\mathcal{L}}$ can be used as an admissible heuristic.

Further, $h_{base}^{\mathcal{L}}$ dominates $h_{\mathcal{L}}^*$. In particular,

$$h_{base}^{LM-cut} \geq h_{LM-cut}^* \geq h_{LM-cut}$$

Results

- Heuristic performance across common solved tasks
 - Base model versus Base model with LM-cut landmarks



Heuristic comparison

Results

- Coverage results
 - f10 **Zhu-Givan** method, f20 **LM-cut** method

Domain	base	f 10	f 20
Airport(50)	20	26	25
Barman-opt11(20)	4	4	4
Blocks(35)	28	28	29
Depot(22)	7	7	7
Driverlog(20)	11	11	13
Elevators-opt08(30)	9	9	18
Elevators-opt11(20)	7	7	15
Floortile-opt11(20)	4	4	6
Freecell(80)	35	47	27
Grid(5)	1	2	1
Gripper(20)	6	7	5
Logistics00(28)	15	14	20
Logistics98(35)	2	3	6
Miconic(150)	50	57	140
Mprime(35)	18	20	20
Mystery(30)	15	16	16
Nomystery-opt11(20)	10	8	12
Openstacks-opt08(30)	15	12	15
Openstacks-opt11(20)	7	7	7
Openstacks(30)	7	8	7
Parcprinter-08(30)	28	28	29
Parcprinter-opt11(20)	20	20	20

Domain	base	f 10	f 20
Parking-opt11(20)	1	1	1
Pathways-noneg(30)	4	4	5
Pegsol-08(30)	28	26	27
Pegsol-opt11(30)	18	16	17
Pipesworld-notank(50)	15	16	11
Pipesworld-tank(50)	10	10	9
Psr-small(50)	50	50	50
Rovers(40)	6	6	7
Satellite(36)	6	6	7
Scanalyzer-08(30)	13	13	11
Scanalyzer-opt11(20)	10	9	8
Sokoban-opt08(30)	16	20	27
Sokoban-opt11(20)	15	16	20
Tidybot-opt11(20)	5	11	8
Tpp(30)	8	8	8
Transport-opt08(30)	10	10	11
Transport-opt11(20)	6	5	6
Trucks(30)	9	9	9
Visitall-opt11(20)	17	17	19
Woodworking-opt08(30)	12	14	20
Woodworking-opt11(20)	7	9	15
Zenotravel(20)	9	9	11
Total(1396)	594	630	749

#Problems solved in 30mins

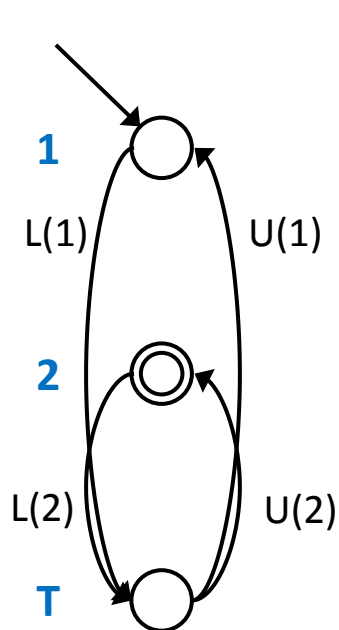
Example



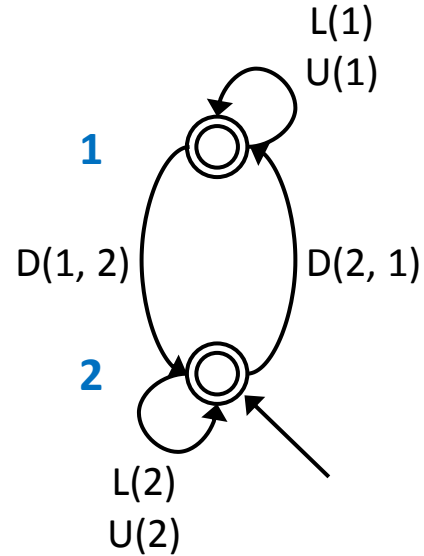
1



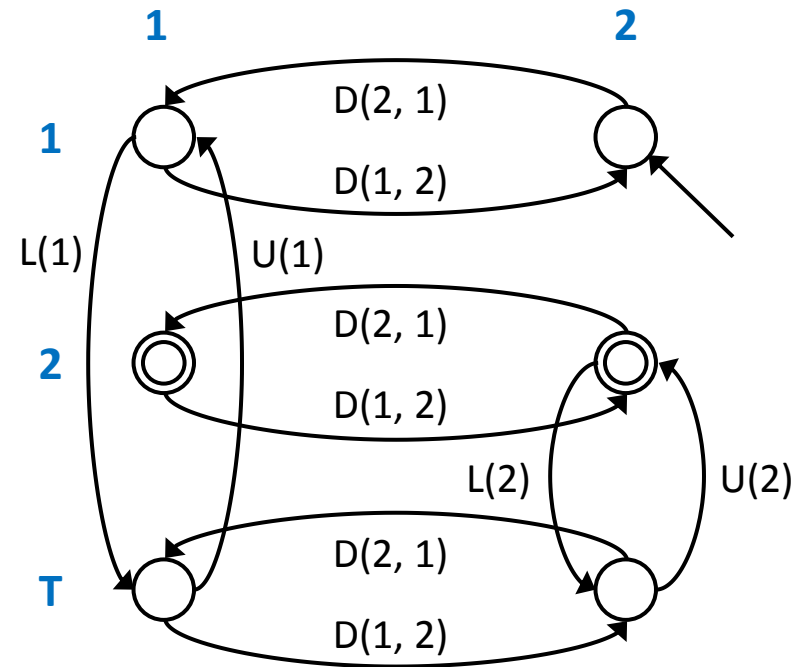
2



DTG(P)



DTG(T)



DTG(P, T)

Merges

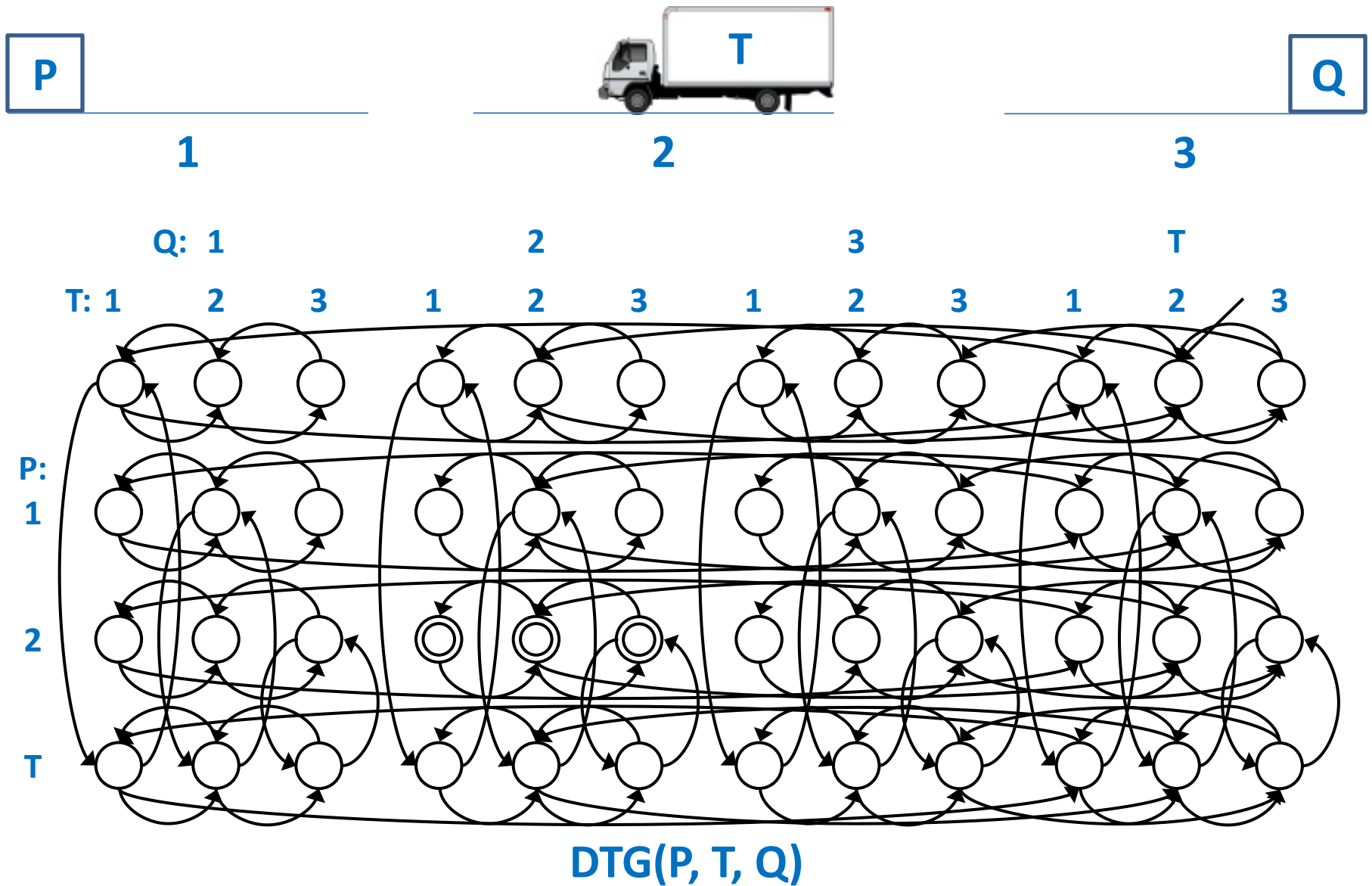
- A merge combines two or more variables into one “super” variable.
- A merged variable introduces the following constraints

$$\sum_{(x',a,x) \in T_X} f(a) - \sum_{(x,a,x') \in T_X} f(a) \geq \begin{cases} 1 & \text{if } x \notin s_0, x \in s_* \\ -1 & \text{if } x \in s_0, x \notin s_* \\ 0 & \text{otherwise} \end{cases}$$

$$f(a) \geq \sum_{a' \in \text{Copies}(a,Z)} f(a')$$

Theorem 3. The set of merge constraints is admissible. As a result, the solution to the base model with merge constraints can be used as an admissible heuristic

Merges



Dynamic merging

- **Dynamic merging** is dynamic in all ways
 - Incrementally merge more variables
 - Incrementally merge variables
- Very simple merge strategy
 - Merge prevail conditions with the preconditions of an action
 $merge(p, q)$ for all atoms $p \in Prev(a)$ and $q \in Pre(a)$

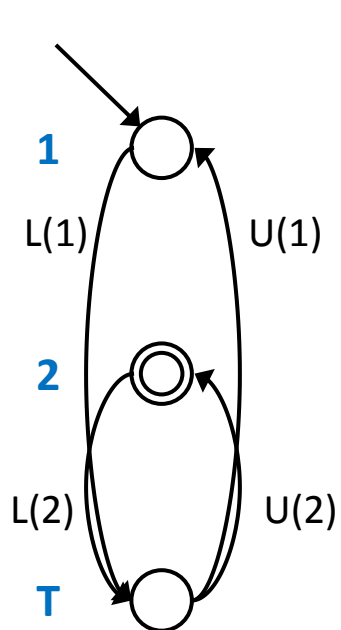
Example



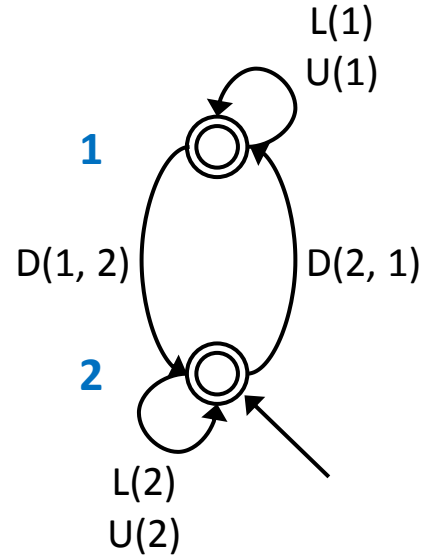
1



2



DTG(P)



DTG(T)

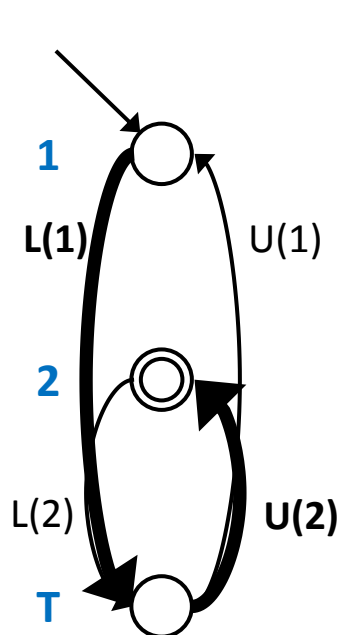
Example



1

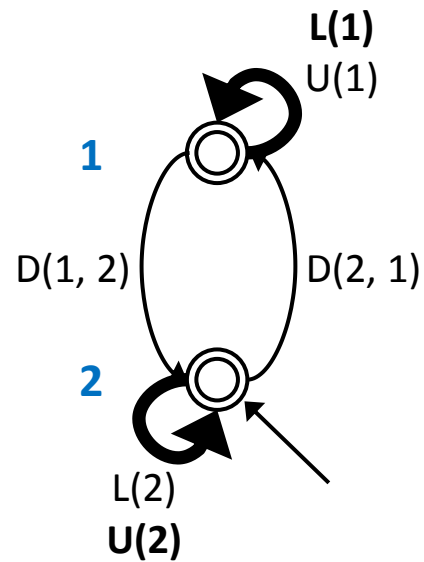


2



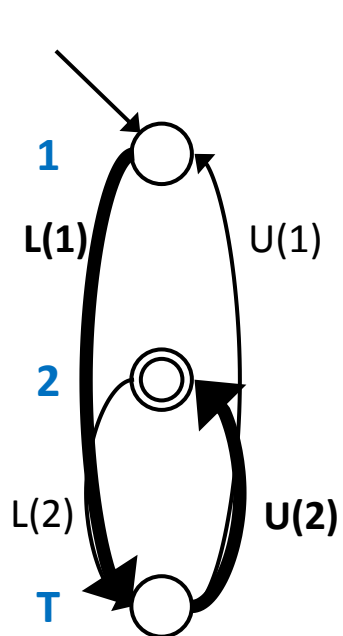
$h = 2$

DTG(P)

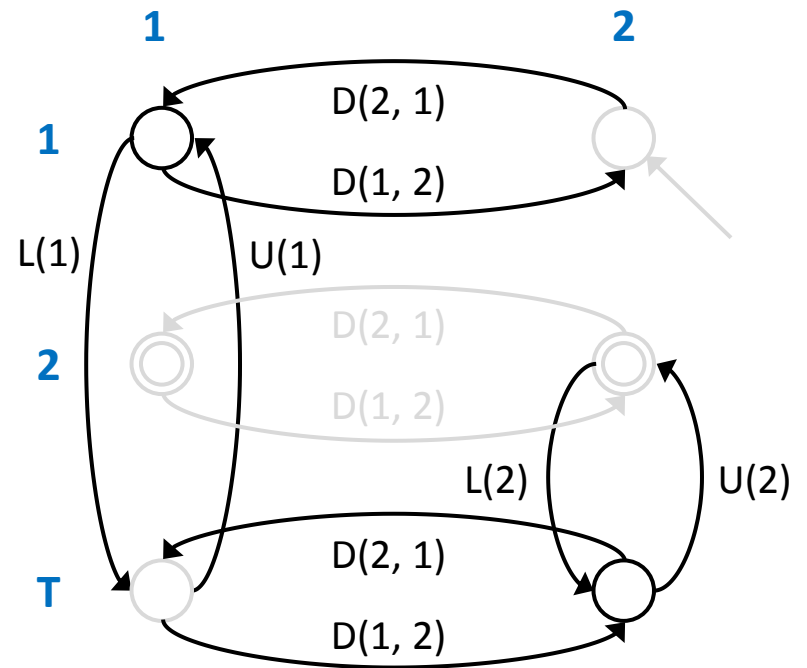
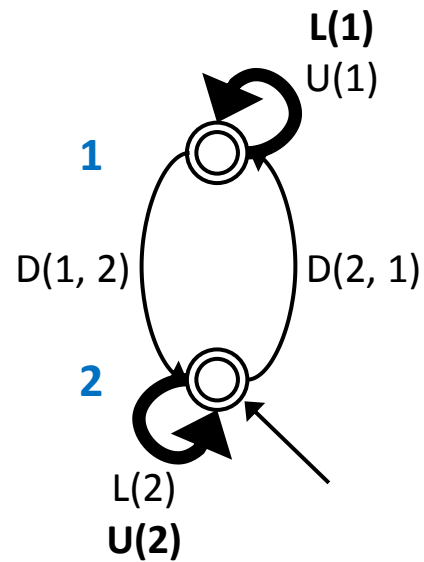


DTG(T)

Example



$h = 2$

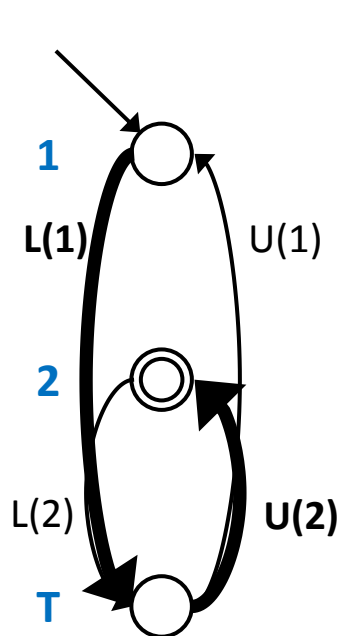


DTG(P)

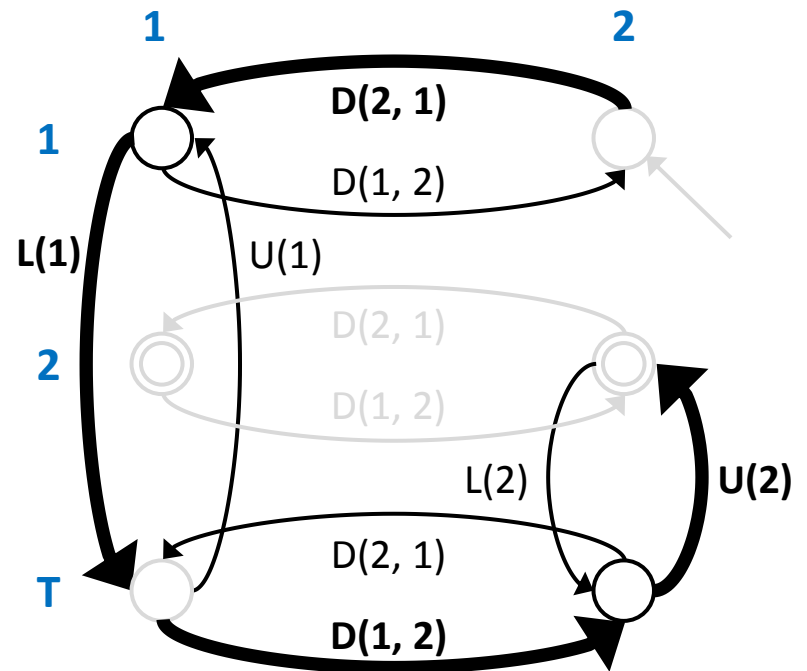
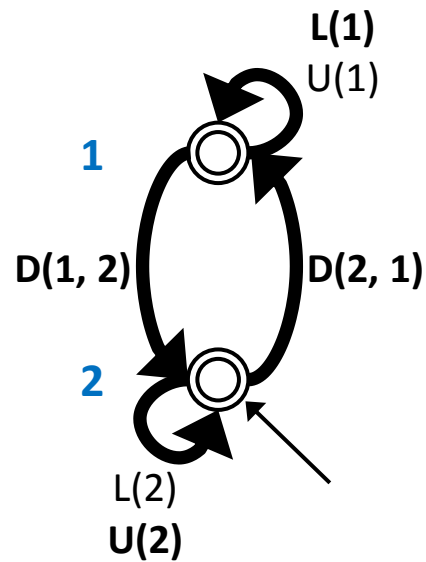
DTG(T)

DTG(P, T)

Example



$h = 4$



DTG(P)

DTG(T)

DTG(P, T)

Results

- All experiments run with **Fast Downward**
 - **Helmert, Journal of Artificial Intelligence Research, 2006**
- Setup
 - AMD Opteron 6378 CPUs running at 2.4GHz
 - 2Gb memory limit
 - 1800 seconds timeout
 - Using IBM CPLEX 12.5.1 as the LP solver
- Heuristics
 - LM-cut
 - $h^*_{\text{LM-cut}}$
 - base
 - fXY
 - X = 0, without landmarks, X = 1, with Zhu-Givan, X = 2, with LM-cut
 - Y = 0, without merge, Y = 1, with merge

Results

- Coverage results

	LM-cut		Base Merge		Base LM-cut		Base LM-cut Merge	
Domain	LM-cut	h^*_{LM-cut}	base	f 01	f 10	f 11	f 20	f 21
Airport(50)	28	28	20	22	26	22	25	22
Barman-opt11(20)	4	4	4	0	4	0	4	0
Blocks(35)	28	28	28	28	28	28	29	29
Depot(22)	7	7	7	6	7	5	7	5
Driverlog(20)	13	13	11	15	11	15	13	15
Elevators-opt08(30)	22	19	9	21	9	21	18	21
Elevators-opt11(20)	17	16	7	17	7	17	15	17
Floortile-opt11(20)	6	6	4	2	4	2	6	5
Freecell(80)	15	15	35	33	47	28	27	29
Grid(5)	2	2	1	2	2	2	1	2
Gripper(20)	7	6	6	20	7	20	5	20
Logistics00(28)	20	20	15	22	14	22	20	22
Logistics98(35)	6	6	2	7	3	7	6	10
Miconic(150)	141	141	50	58	57	140	140	141
Mprime(35)	22	22	18	24	20	24	20	25
Mystery(30)	19	18	15	20	16	20	16	17
Nomystery-opt11(20)	14	14	10	14	8	14	12	14
Openstacks-opt08(30)	20	17	15	10	12	10	15	10
Openstacks-opt11(20)	15	12	7	5	7	5	7	5
Openstacks(30)	7	7	7	7	8	7	7	5
Parcprinter-08(30)	18	18	28	30	28	30	29	30
Parcprinter-opt11(20)	13	13	20	20	20	20	20	20

#Problems solved in 30mins

Results

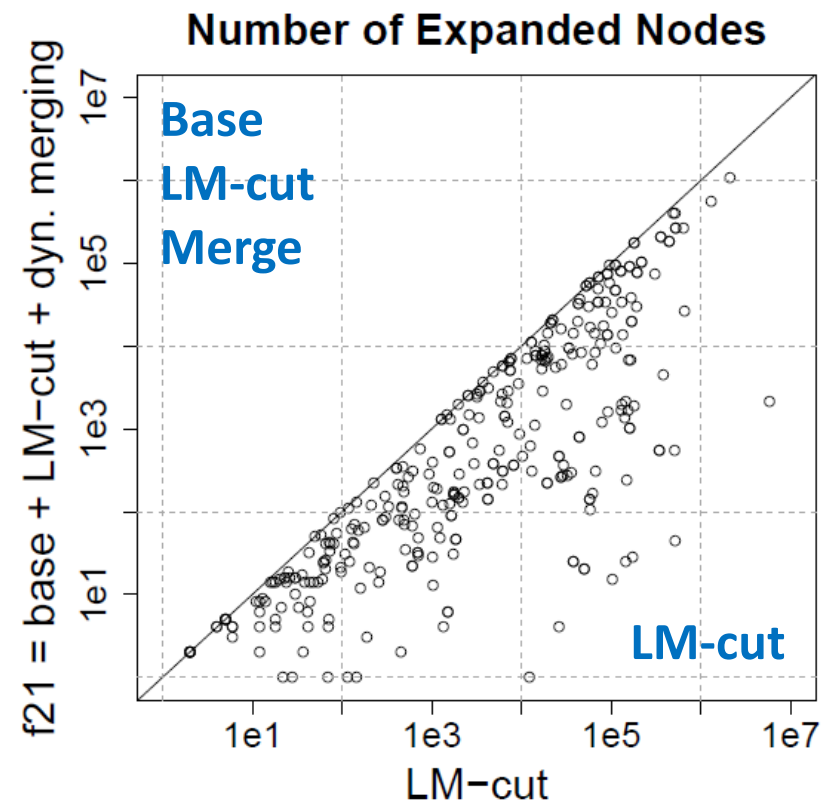
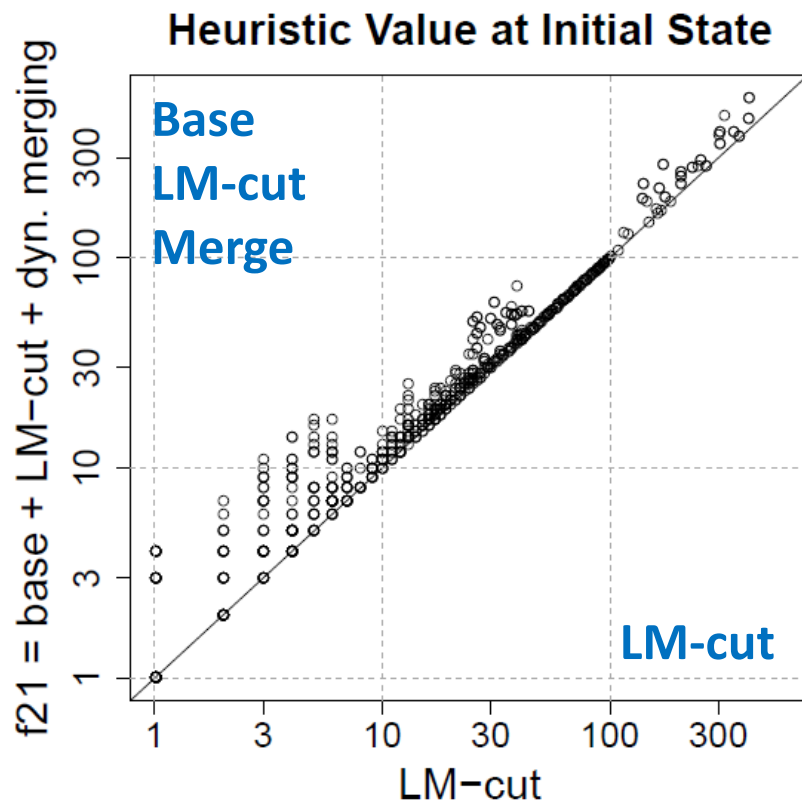
$$h_{base}^{LM-cut} \geq h_{LM-cut}^* \geq h_{LM-cut}$$


- Coverage results

Domain	LM-cut		Base Merge		Base LM-cut		Base LM-cut Merge	
	LM-cut	h_{LM-cut}^*	base	f 01	f 10	f 11	f 20	f 21
Parking-opt11(20)	2	1	1	1	1	1	1	1
Pathways-noneg(30)	5	5	4	4	4	4	5	5
Pegsol-08(30)	27	27	28	28	26	26	27	27
Pegsol-opt11(30)	17	17	18	18	16	16	17	17
Pipesworld-notank(50)	17	16	15	13	16	15	11	13
Pipesworld-tank(50)	11	9	10	10	10	10	9	10
Psr-small(50)	49	49	50	50	50	50	50	50
Rovers(40)	7	7	6	6	6	7	7	7
Satellite(36)	7	7	6	6	6	6	7	7
Scanalyzer-08(30)	15	13	13	12	13	11	11	11
Scanalyzer-opt11(20)	12	10	10	9	9	8	8	8
Sokoban-opt08(30)	28	28	16	18	20	20	27	27
Sokoban-opt11(20)	20	20	15	15	16	16	20	19
Tidybot-opt11(20)	13	13	5	1	11	1	8	1
Tpp(30)	6	6	8	11	8	11	8	10
Transport-opt08(30)	11	11	10	10	10	10	11	11
Transport-opt11(20)	6	6	6	5	5	5	6	6
Trucks(30)	10	10	9	10	9	12	9	11
Visitall-opt11(20)	10	10	17	17	17	17	19	19
Woodworking-opt08(30)	16	16	12	25	14	28	20	28
Woodworking-opt11(20)	11	11	7	18	9	20	15	20
Zenotravel(20)	12	12	9	13	9	13	11	13
Total(1396)	756	736	594	683	630	766	749	785

Results

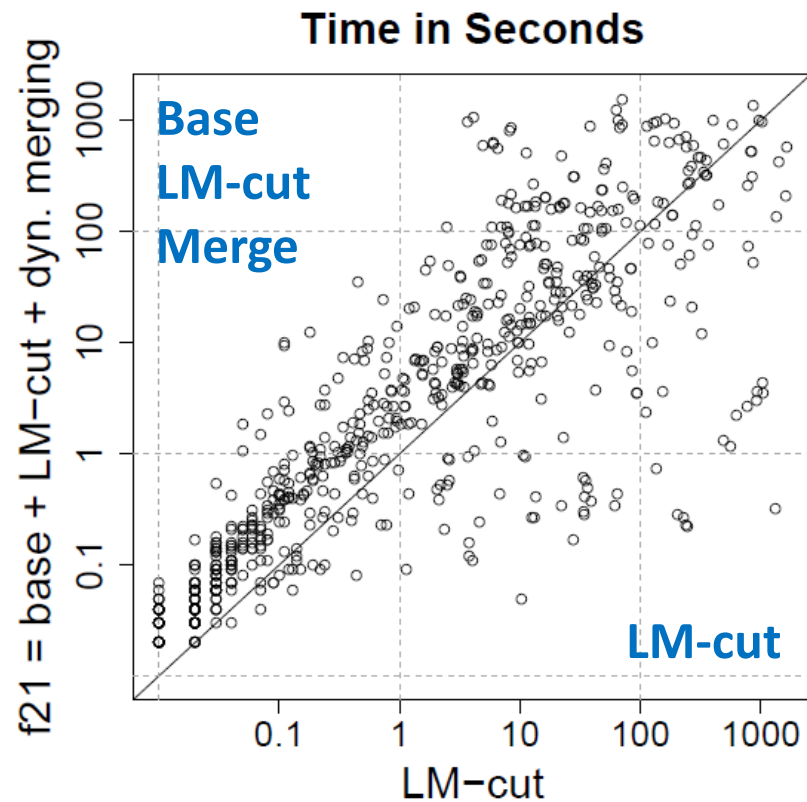
- Heuristic performance across common solved tasks
 - LM-cut versus Base model with landmarks and merges



Heuristic comparison

Results

- Time spent by A^* search for finding a solution
 - LM-cut versus Base model with LM-cut landmarks and merges



Time comparison

Conclusions

- **Flow-based heuristics** are effective and efficient yet can be improved significantly
 - Adding constraints derived from landmarks
 - Adding constraints derived from merges
- **Dynamic merging** incrementally merges more and more variables
 - So far, only incorporated a very simple merge strategy
- Flow-based heuristics provide a **rich area for future research**

Blai Bonet

Universidad Simon Bolivar
Caracas, Venezuela
bonet@ldc.usd.ve



USB

Menkes van den Briel

NICTA & Australian National University
Canberra, Australia
menkes@nicta.com.au



NICTA