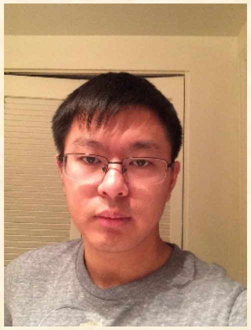
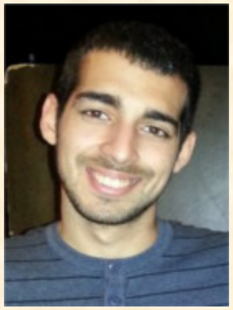
Recent Advances in Bidirectional Search

Ariel Felner
ISE Department
Ben-Gurion University
ISRAEL
felner@bgu.ac.il



Jingwei Chen
Univ. of Alberta
Canada

Eshed Shaham
HUJI Israel



Shahaf Shperberg
Ben-Gurion Univ.
Israel

Robert C. Holte
Univ. of Alberta
Canada

Ariel Felner
Ben-Gurion Univ.
Israel

Guni Sharon
Ben-Gurion Univ.
Israel

Nathan Sturtevant
Univ. of Alberta
Canada

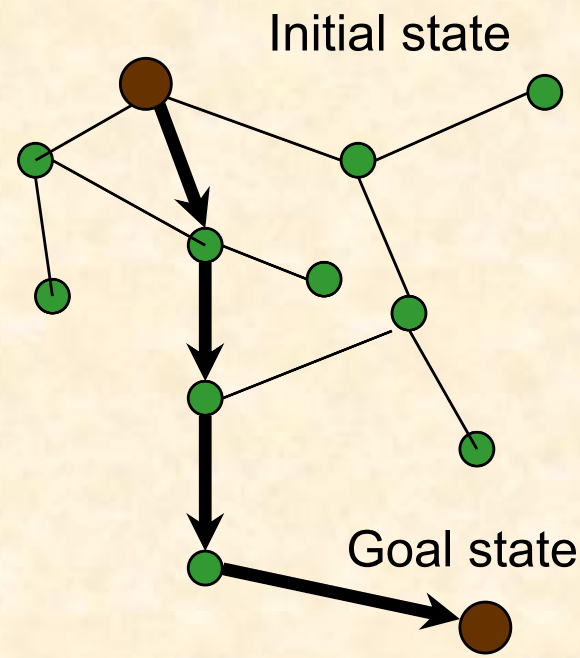
State spaces (domains)

- A set of states

- Edges between states

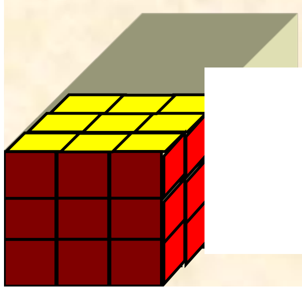
- An initial and goal state

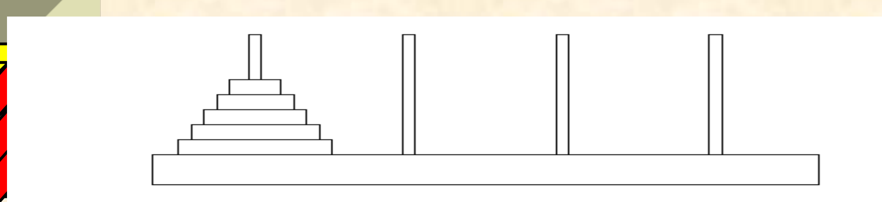
- A solution: a path from the initial state to the goal state

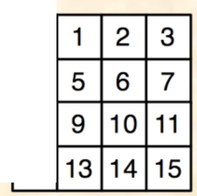


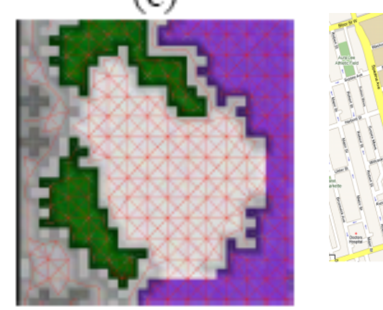
Different Domain Types

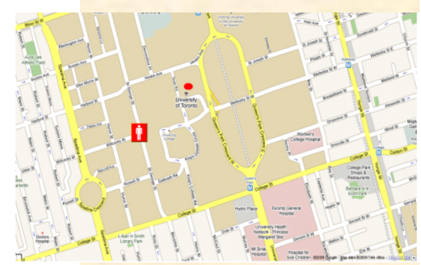
	Exponential Domains	Polynomial Domains
Space size N	$N=O(\mathbf{b^d})$ **May have cycles**	$N=O(\mathbf{d^k})$ – **May have many cycles**
Input	Implicitly given (large) Have symmetries/structure	Explicitly given May not have symmetries
Example	Permutation puzzles Planning problems	Path-finding in Maps, GPS Sequence alignment
Typical #states	10^{15}	10^6
Search time	Days (30 minutes) /offline	realtime /online
Algorithms	DFS/BFS based algorithms (IDA*/A*)	BFS based algorithms (A*)





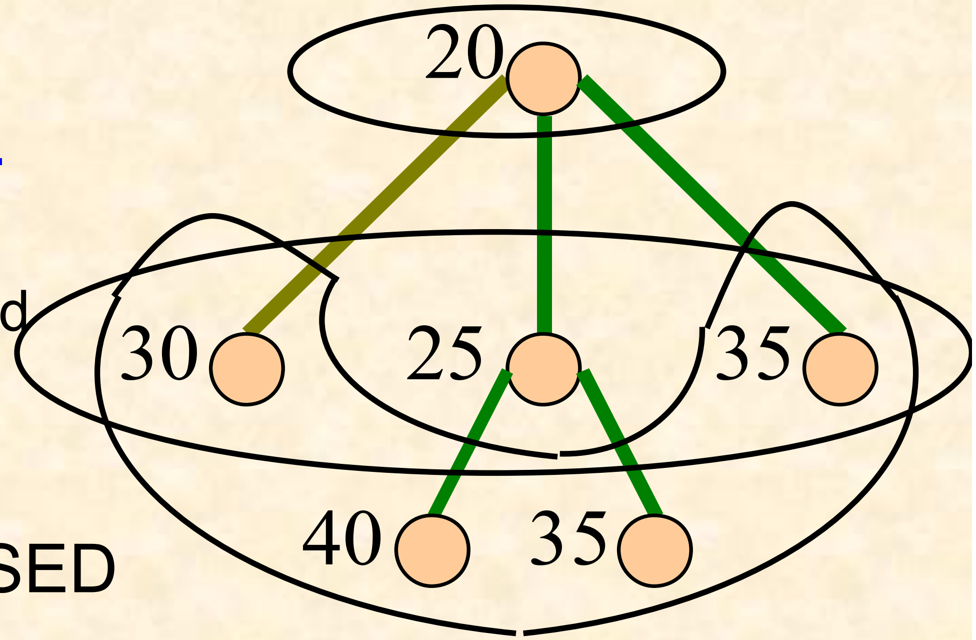



(c)



Best-first search schema

- Keeps an OPEN list of nodes.
- Expands ***best*** node from OPEN.
  - generate(x): insert x into OPEN.
  - expand(x): delete x from OPEN and generate its children.



- Expanded nodes go into a CLOSED (hash table)
- BFS depends on its cost (heuristic) function.

Closed

20	**25**

Open

30	**35**	**35**	**40**

Best-first search: Cost functions

- **g(n):** Best known distance from the initial state to n
- **h(n):** The estimated distance from n to the goal state.

- Examples:**Air distance in maps**

  Manhattan Distance in the tile puzzle

	1	2	3
4	5	6	7
8	9	10	11
12	13	14	15

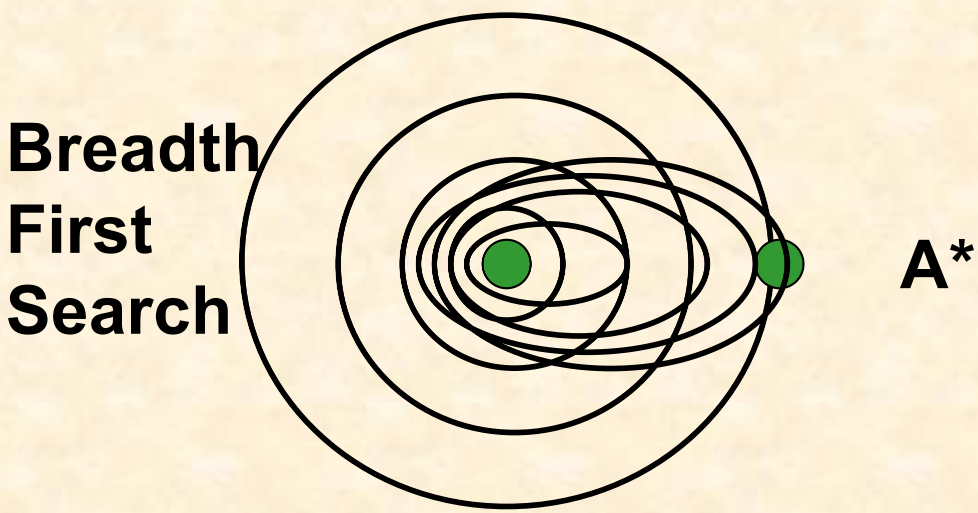
Different cost combinations of g and h

- **f(n)=level(n)** Breadth-First Search.
- **f(n)=g(n)** Uniform Cost Search

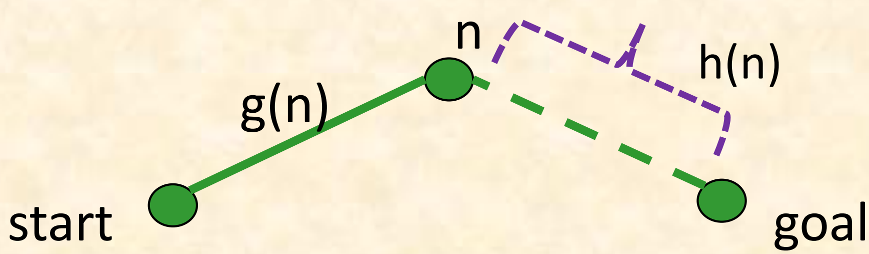
  (AKA Dijkstra's algorithms).
- **f(n)=h(n)** Pure Heuristic Search (PHS).
- **f(n)=g(n)+h(n)** The A* algorithm (1968).

A*

- ***f(n)*** in A* is an estimation of the shortest path to the goal ***via n***.
- ***h*** *is* ***admissible*** *if it is underestimating.*
- **A* theorem**: Given an admissible heuristic *h,* A* finds optimal solutions, complete and optimally effective. [Pearl 84]
- **Result: any other optimal search algorithm will expand at least all the nodes expanded by A***

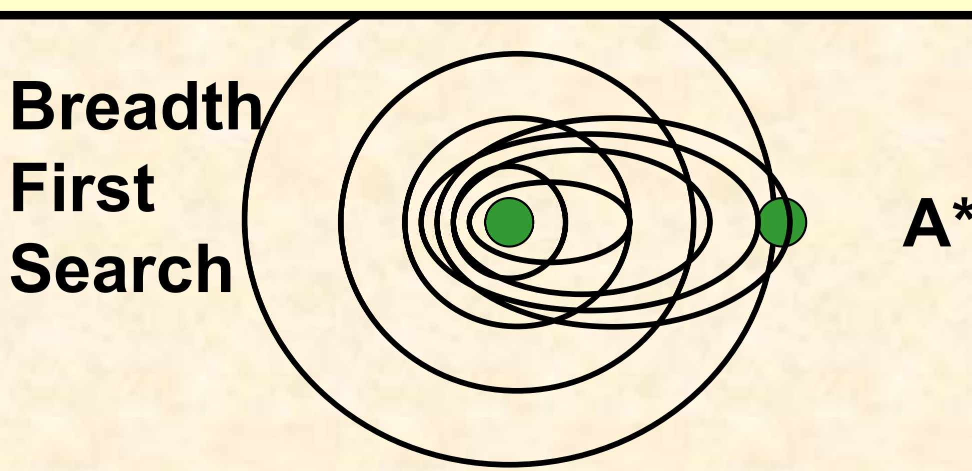


Unidirectional search



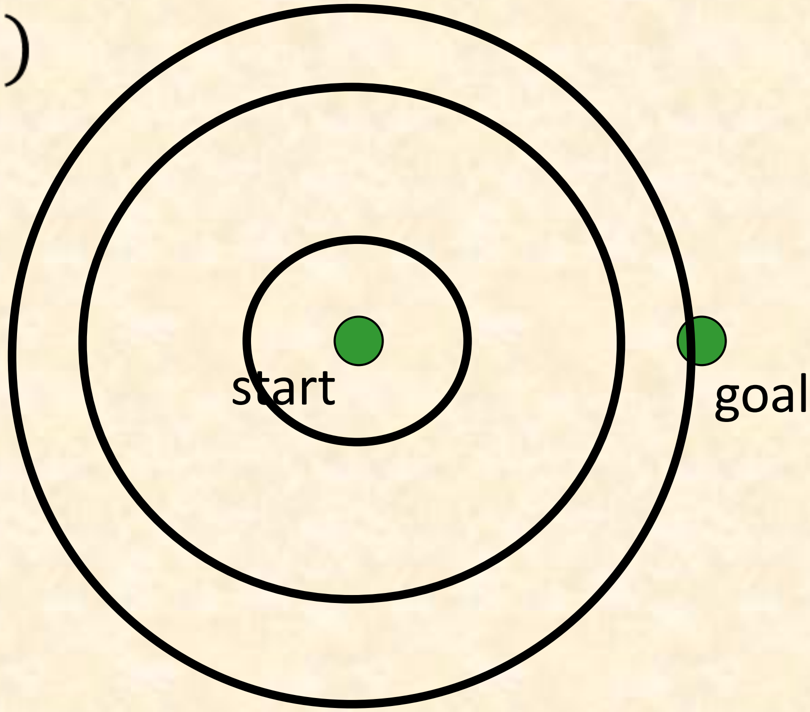
Different costs functions:

Adding heuristics to unidirectional search is very beneficial



Breadth-first search (BFS)

Unidirectional breadth first search (b^d)



Bidirectional breadth-first search (BDS)

Unidirectional breadth

Main motivation for BDS: potential exponential reduction

Improving search

1) Add heuristics

2) Run bidir

Let’s combine both direction: Bidirectional Heuristic Search

Bidirectional search algorithms

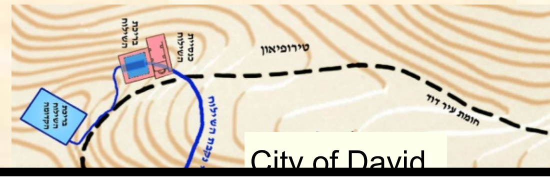
Two search frontiers: openF, openB

We select a node from either openF or openB

Once we have a match we stop with a solution

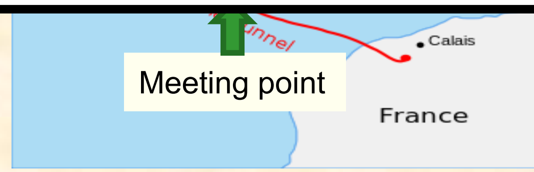
Challenge 1: The frontiers should meet

Siloam Tunnel

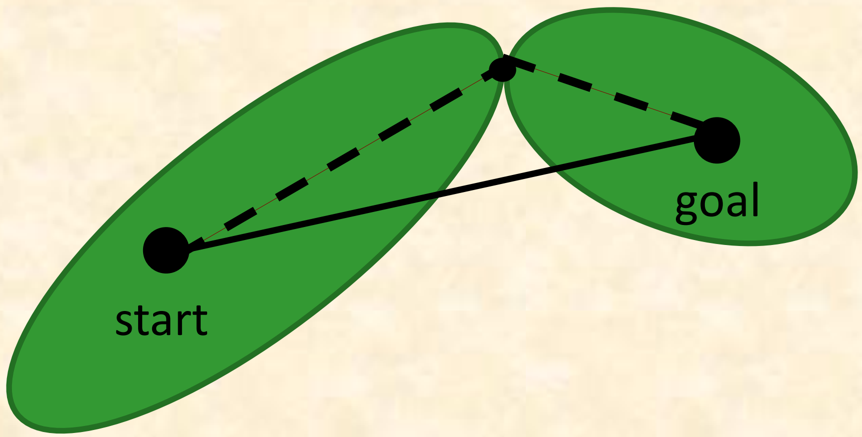


Many Bi-HS algorithms are guaranteed to meet!

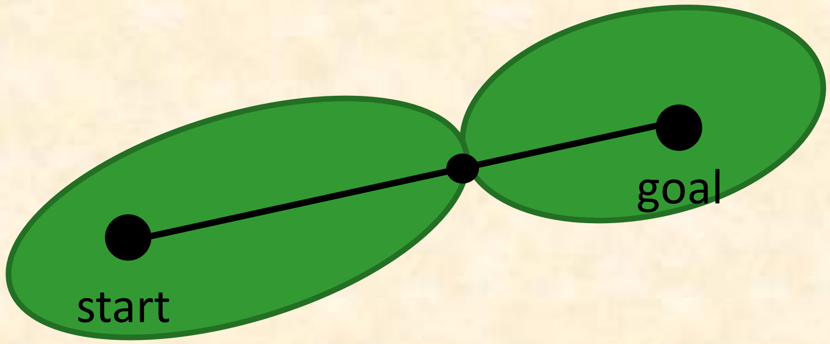
Europe 1994



Challenge 2: guaranteeing Optimality



Challenge 2: guaranteeing Optimality



Many Bi-HS algorithms guarantee optimality - no open node below U

Other challenges

1) Guarantee that the frontiers meet – they might cross each other.

2) Guarantee optimality (when applicable).

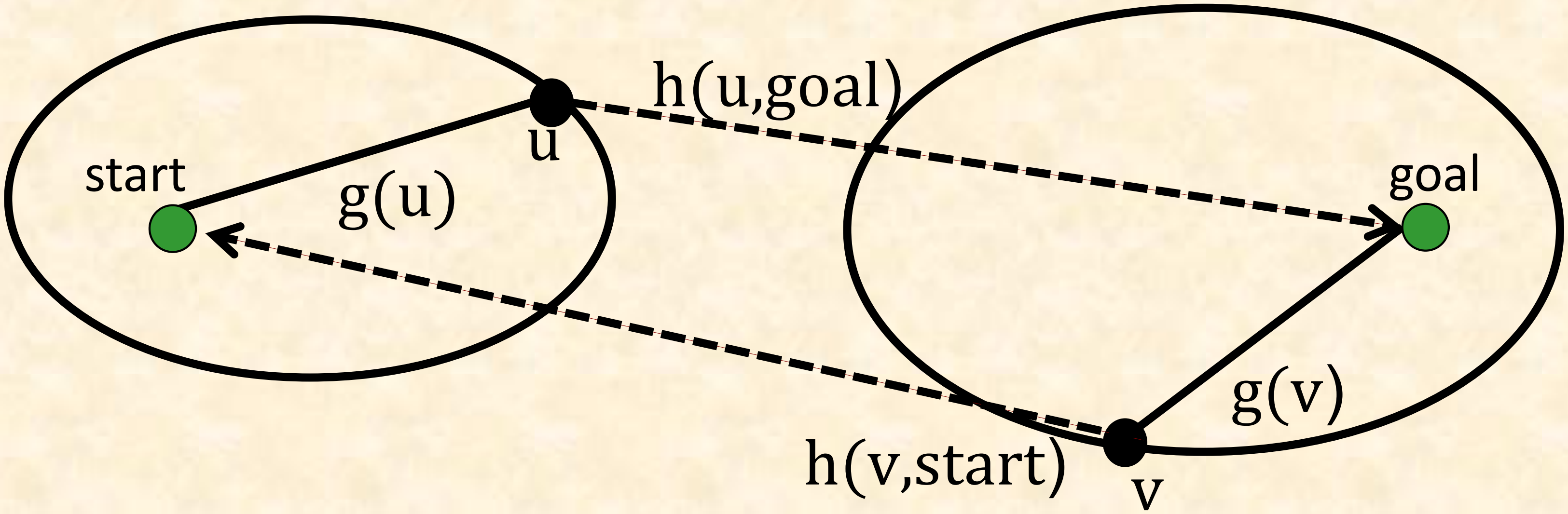
3) Which side to expand next

4) Which node within the chosen side

5) Stopping condition (when do we halt)

6) How do we add heuristics

Front-to-end Heuristics



- Each node has a heuristic towards the opposite end

Front-To-End bidirectional search:

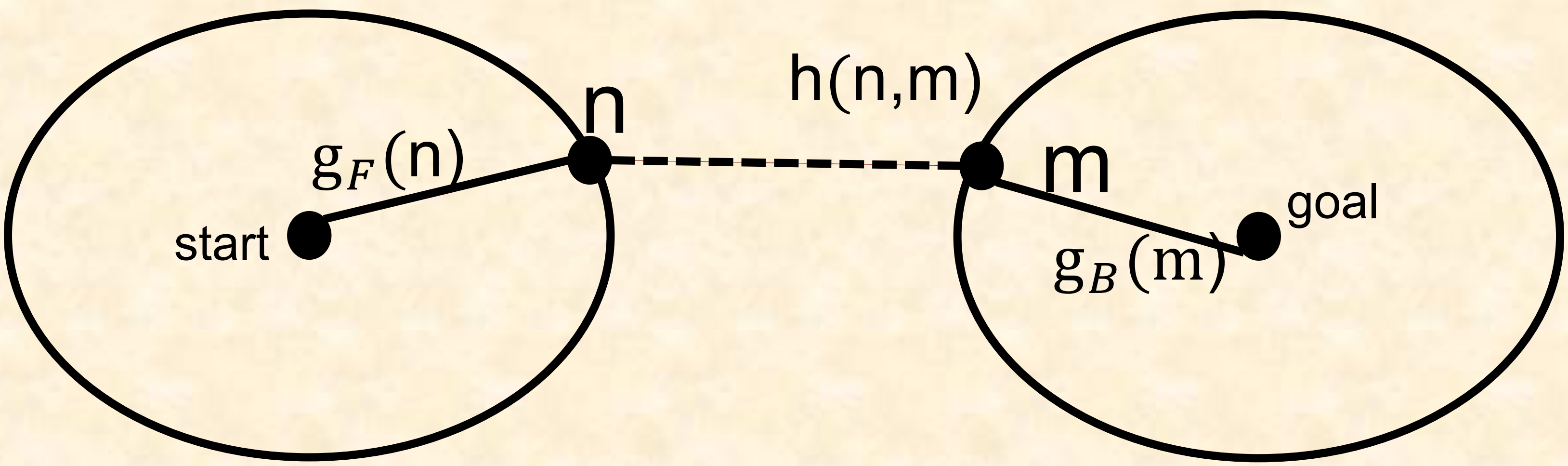
$f_F(u) = g_F(u) + h(u,goal))$

$f_B(v) = g_B(v) + h(start,v)$

Heuristics for BDS

Front-To-Front bidirectional search:

$f_F(n)=g_F(n)+ min_{m \in openB}(h(n,m)+g_B(m))$



Heuristics

Front-to-front heuristic is more accurate but takes more time to compute.

Front-to-front can be seen as a special case of front-to-end:

$f_F(n) = g_F(n) + h_F(n, goal)$

$h_F(n, goal) = min_{m \in openB}(h(n, m) + g_B(m))$

Which side/node to expand

Alternate sides

Select node within the smallest OPEN

Select side/node with smallest f(n)

Select side/node with smallest g(n)

How to break ties?

Stopping Condition

3) Stopping condition (when do we halt?)

- **Early stopping:** U: the best known path Stop when no node is smaller than U

- **Late stopping:** When a node in both sides is chosen for expansion.

50 years on Bidirectional Search

1969	Pohl	Bidirectional A*
1975	de Champeaux	
		n
		arch
2015	Wilt & Ruml	Dynamic perimeter
2015	Barker & Korf	Theoretical claim
2		

No real success &
no real understanding

New line of work in 2015

MM: The first Bidirectional Heuristic Search that is guaranteed to <u>meet in the middle</u>

[Holte et al. AAAI-2016, AIJ-2017] (#1,#2)

Robert C. Holte
Univ. of Alberta
Canada

Ariel Felner
Ben-Gurion Univ.
Israel

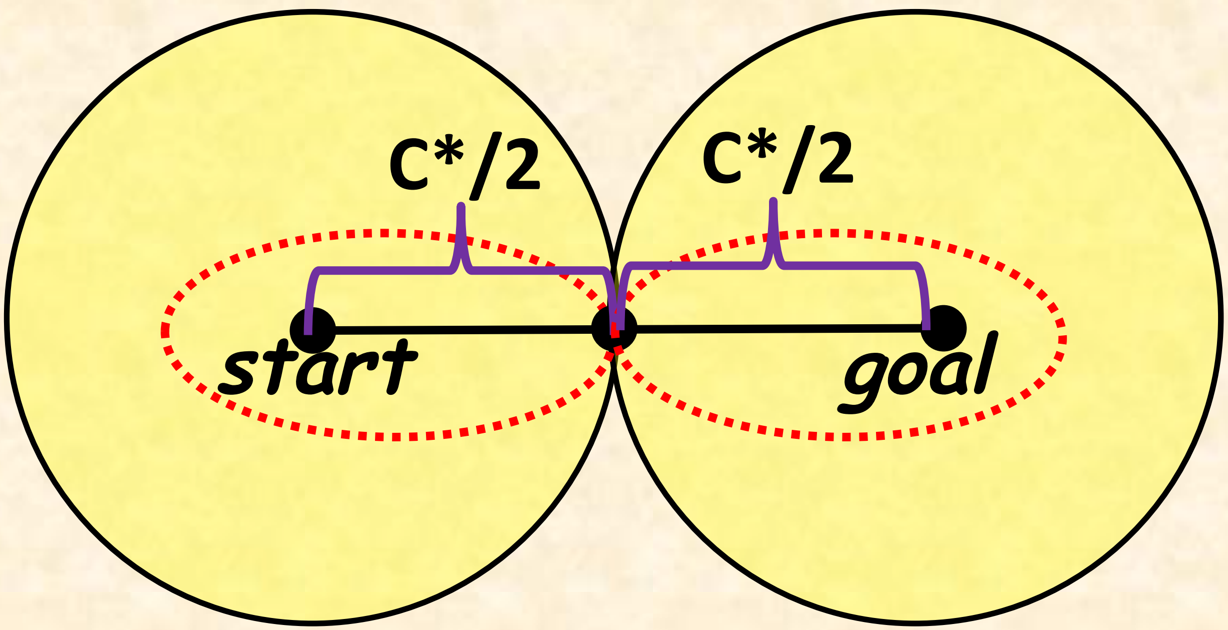
Guni Sharon
Ben-Gurion Univ.
Israel

Nathan Sturtevant
Univ. of Denver
USA

Challenge 3: Where do they meet?

We present MM, the **first** bidirectional heuristic search algorithm that is guaranteed to meet

exactly in the middle!



How MM works

Nodes are ordered by **priority**:

$$pr(n) = \max \begin{cases} g(n)+h(n) & \text{(case 1)} \\ 2\times g(n) & \text{(case 2)} \end{cases}$$

$$pr(n)=g(n)+\max\{g(n),h(n)\}$$

Expand a node (on either sides) with minimal pr(n)

When a node n is generated, check if n is in Open of the opposite side

Remember the cheapest path found **(cost = U)**.

MM stops when **U ≤ LB**

LB=max(C; fminF ; fminB; gminF +gminB +e)

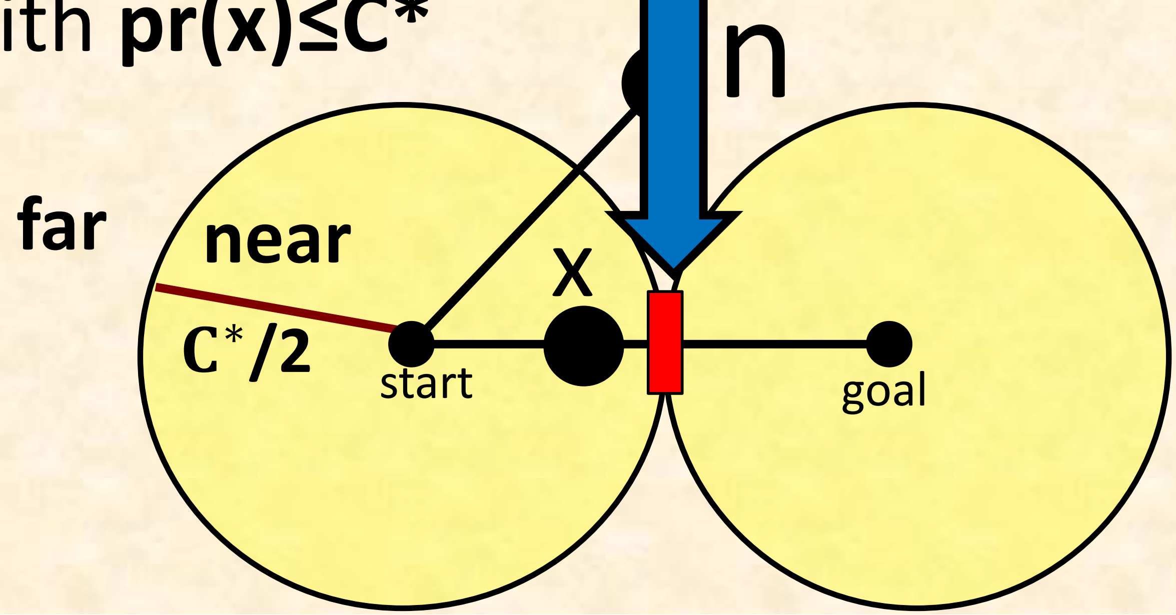
Main lemma:

MM never expands nodes with $g(n)>C^*/2$

Result: must meet in the middle

Proof:

- Let **$g(n)>C^*/2$**
  - case 1: **If $g(n)<h(n)$** then **pr(n** **$g(n)+h(n) > C^*$**
  - case 2: **If $g(n)>h(n)$** then **pr(n** **$2g(n)$ $> C^*$**
- **OPEN** always includes a node **x** on the optimal path with **$pr(x) \le C^*$**

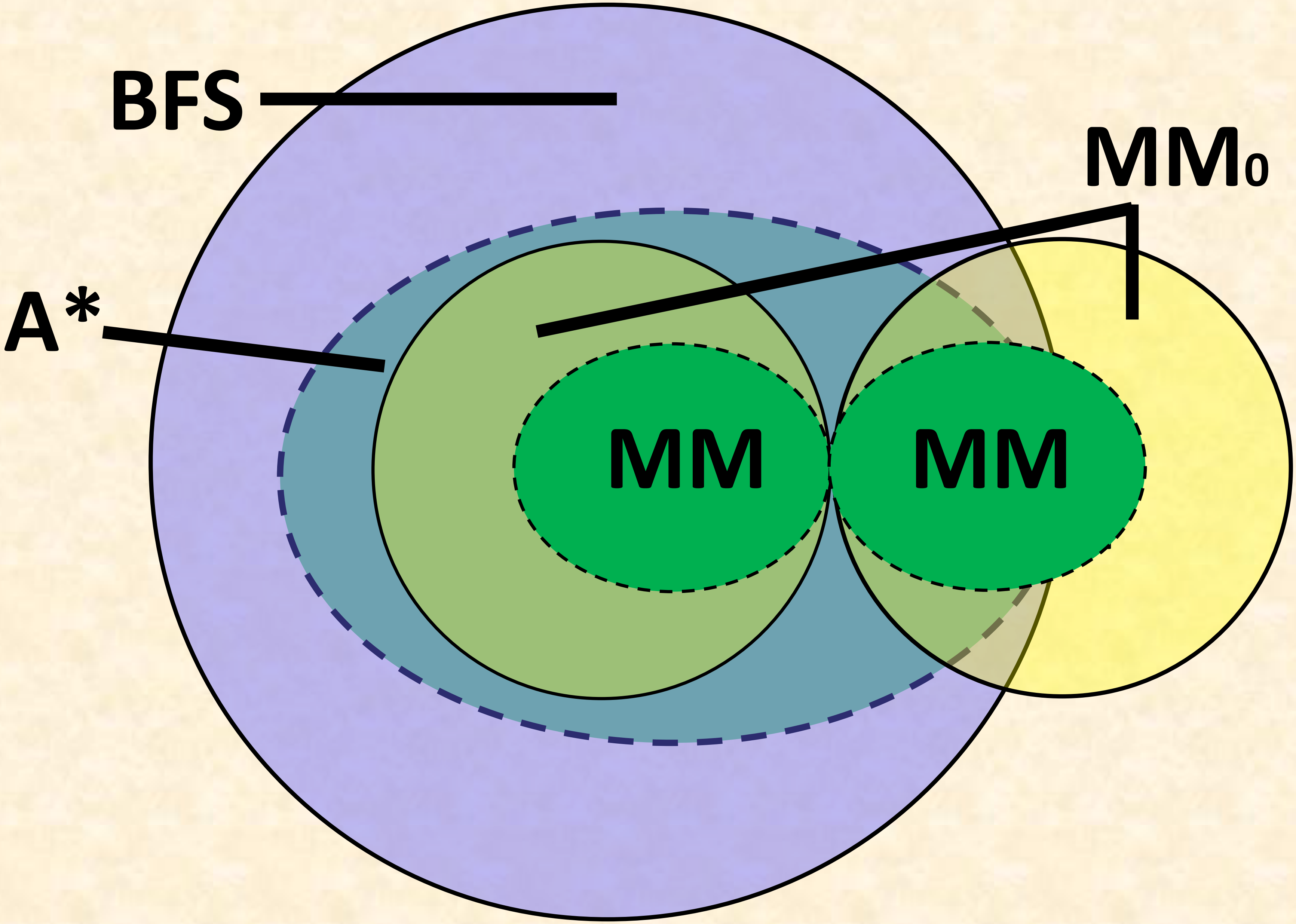


MM_0 = Brute-force MM

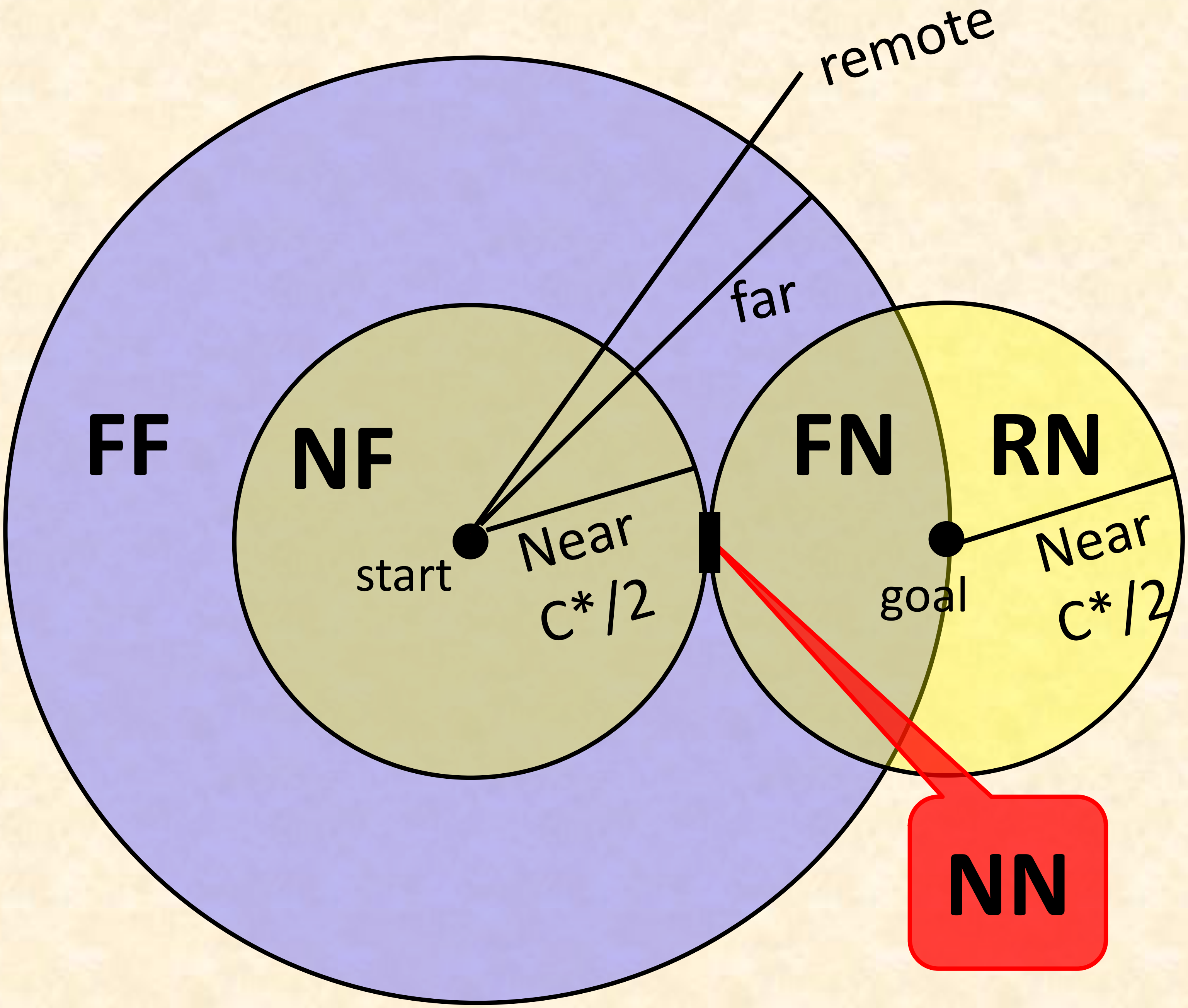
MM_0 = MM with a heuristic h(n)=0 for all n.

$$pr(n) = \max \left\{ \begin{array}{l} g(n)+0 = g(n) \\ 2 \times g(n) \end{array} \right\} \equiv g(n)$$

Intermediate Summary

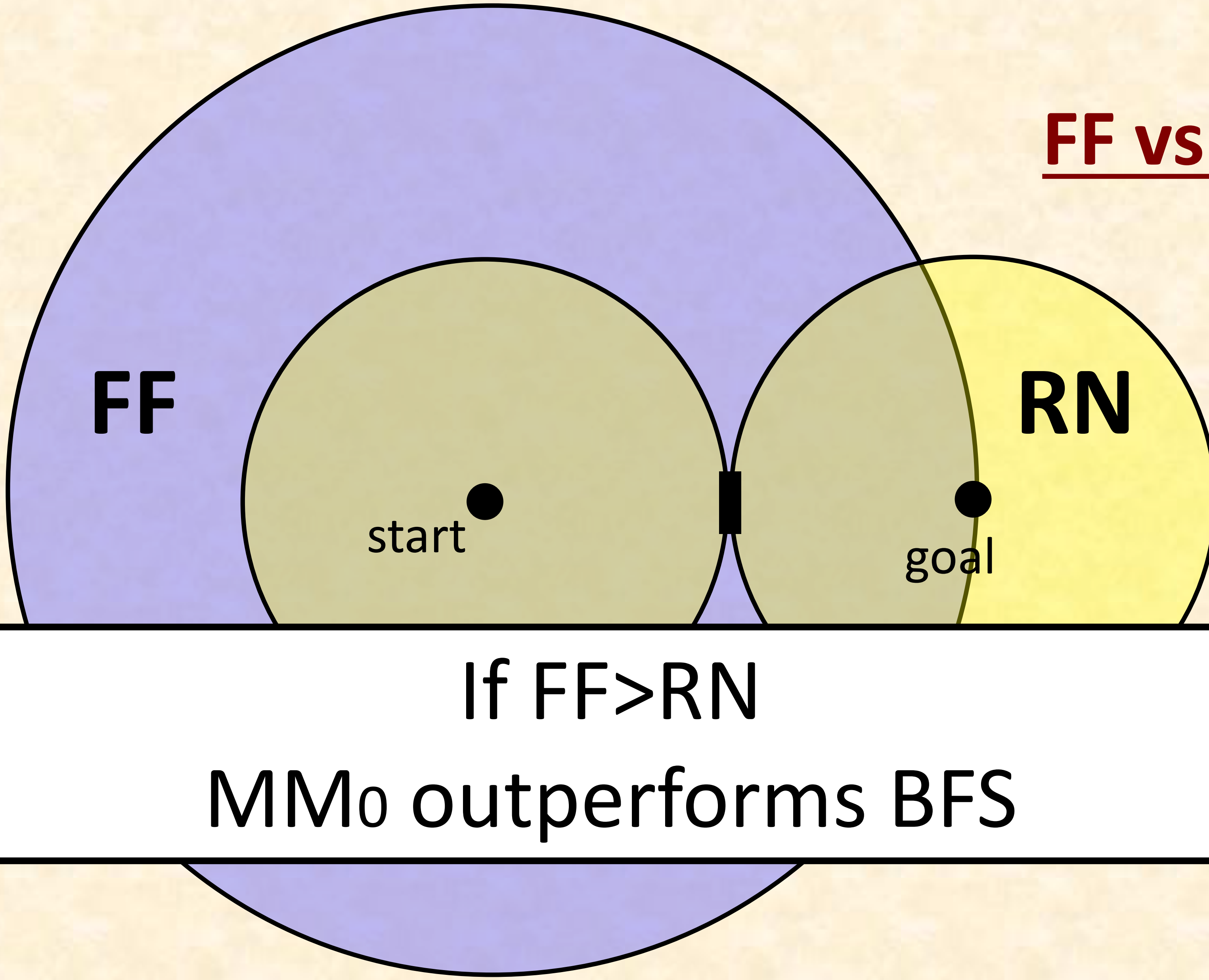


Region-Based Analysis



- **Only unidirectional search (A*) does work on FF**
- **Only MM/MM0 does work on RN**

FF vs RM



If FF>RN

MM_0 outperforms BFS

Our Conjectures

1. With a sufficiently accurate heuristic A* will expand fewer nodes than MM and MM_0.

1. With a moderately accurate heuristic, MM can expand fewer nodes than A* and MM_0 if FF > RN

2. With a sufficiently inaccurate heuristic, MM_0 will expand fewer nodes than MM and A* if FF > RN.

Experiments: 10-Pancake Puzzle, $C^*=10$

	Better Heuristic Accuracy →			
Algorithm	**GAP-3**	**GAP-2**	**GAP-1**	**GAP**
A*	97,644	27,162	4,280	117
MM	7,507	6,723	2,448	165
MM_0	5,551	5,551	5,551	5,551

#states expanded

Fractional MM – fMM(P)

[Shaham, Felner, Chen and Sturtevant . SoCS-2017}[(#3)

$0 \le P \le 1$

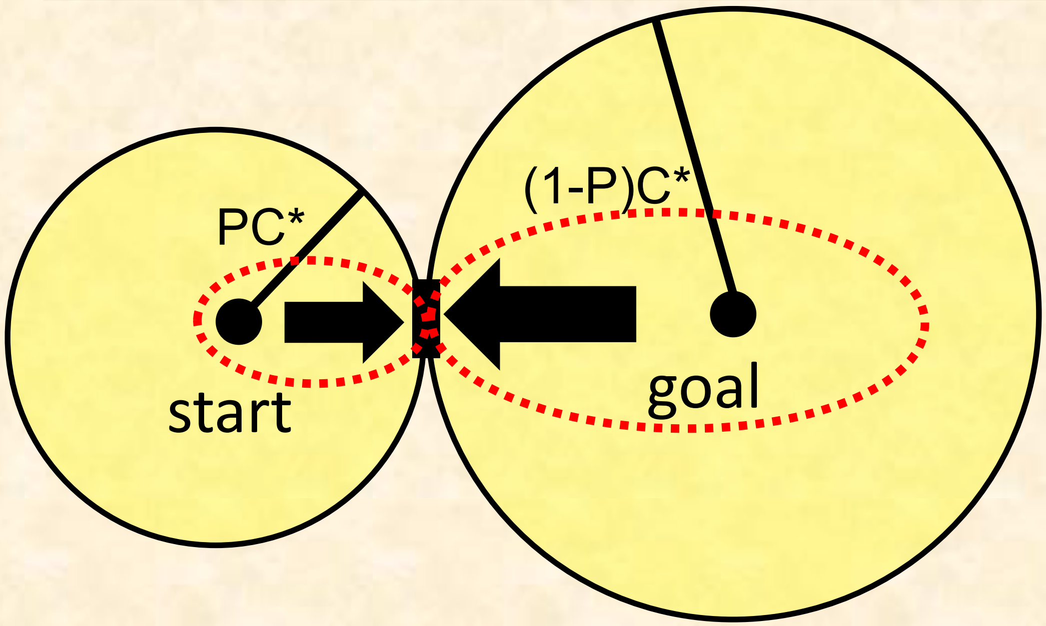
Forward side:

$$pr(n) = \max \begin{cases} g_F(n) + h_F(n) \\ g_F(n)/P \end{cases}$$

Backward side:

$$pr(n) = \max \begin{cases} g_B(n) + h_B(n) \\ g_B(n)/(1-P) \end{cases}$$

Will meet at PC*,(1-P)C*



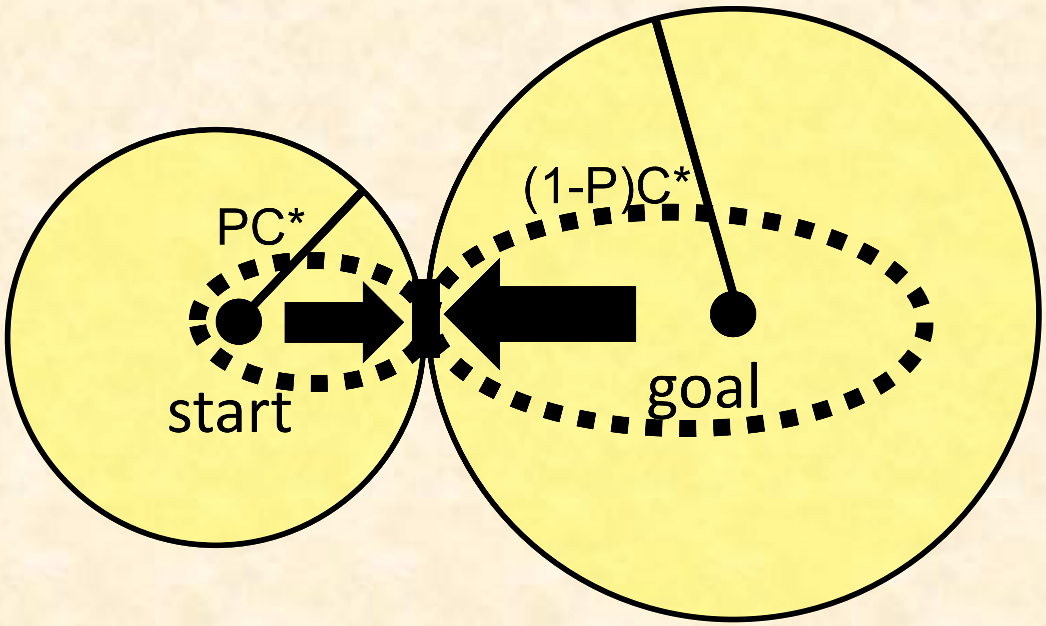
Restrained Algorithm

A Bi-HS algorithm **A** is ***restrained*** if there exist **$0 \le P \le 1$** such that:

- **A** never expands forward nodes with $g_F > PC^*$
- **A** never expands backword nodes with $g_B > (1-P)C^*$

MM and fMM are restrained

Will meet at PC^*,$(1-P)C^*$

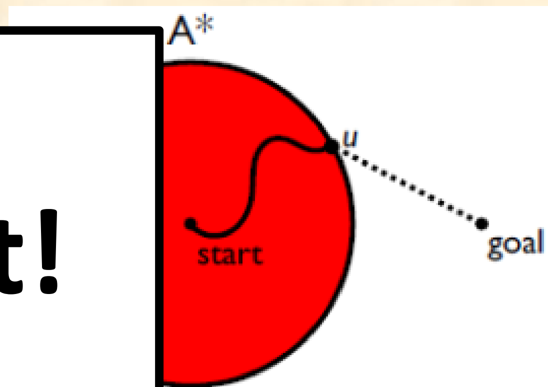


The Optimality of A*

- "Given an admissible heuristic, A* expands (up to tie breaking) the necessary and sufficient nodes to find an optimal solution and to prove that this solution is indeed optimal." [Dechter and Pearl, 1985]

All nodes with $f(u)=g(u)+h(u) < C^*$ must be expanded to prove a C^* solution

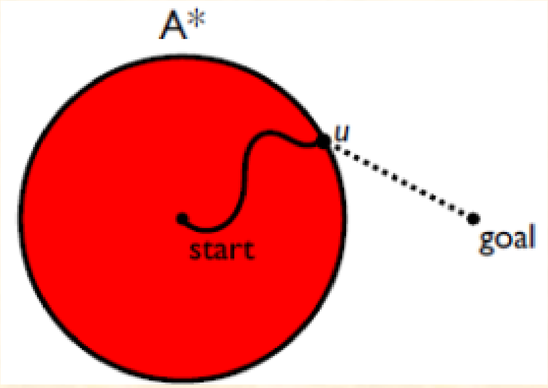
A* is optimally efficient!



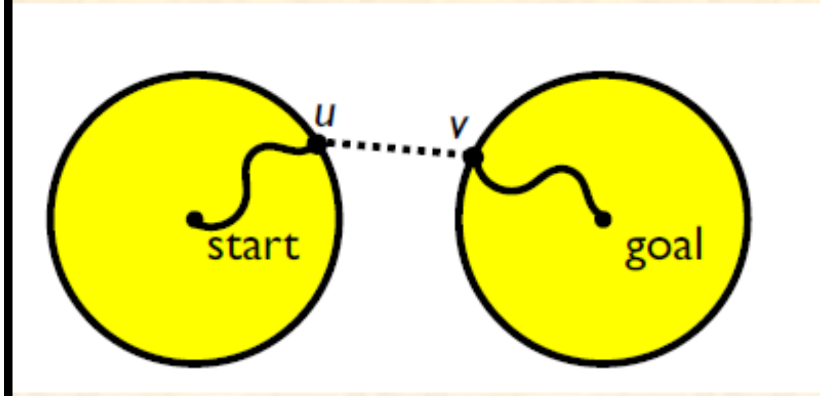
Otherwise, there might be a shorter path from n to the goal

What about bidirectional search

What are the set of states that must be expanded by a bidirectional search?



In bidirectional search we have to talk about *a pair (u,v)* of nodes



The conditions for bidirectional search

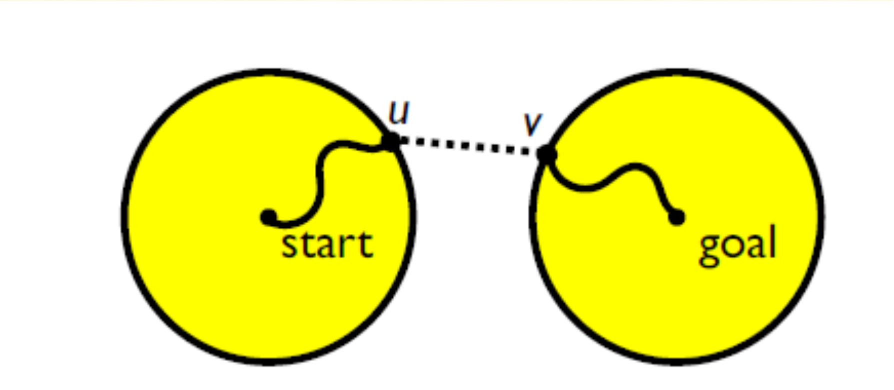
[Eckerle, Chen, Sturtevant, Zilles and Holte, ICAPS-2017](#4)

Pair of nodes **(u,v)** are a ***must-expand pair (MEP)*** if:

1) $f_F(u)=g_F(u)+h_F(u) < C^*$

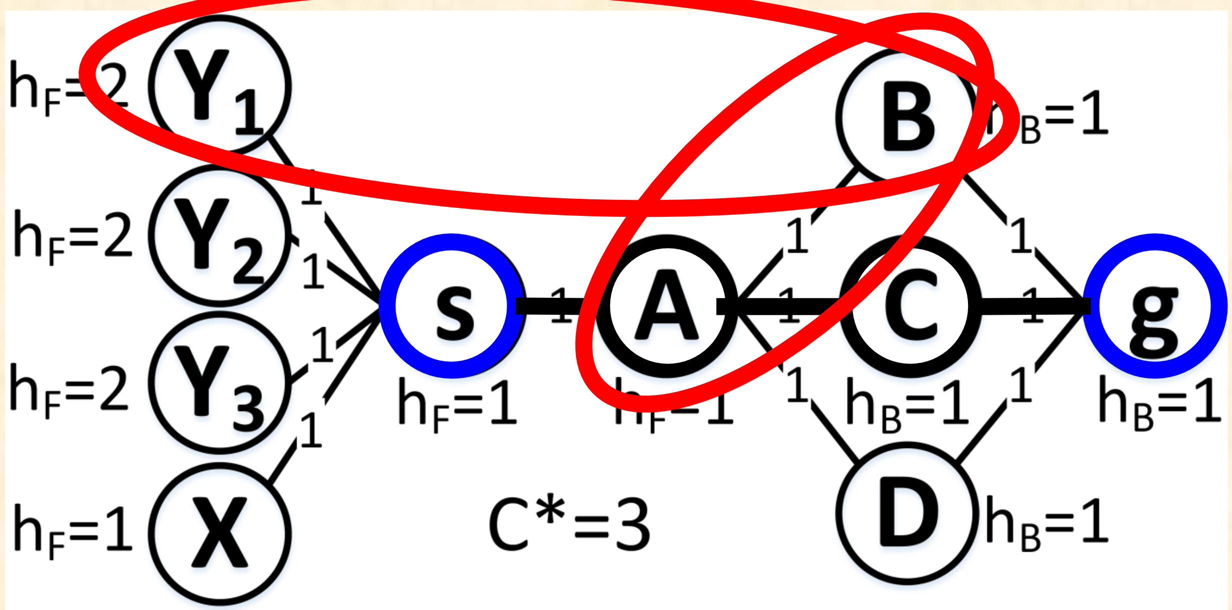
2) $f_B(v)=g_B(v)+h_B(v) < C^*$

3) $g_F(u)+g_B(v) < C^*$

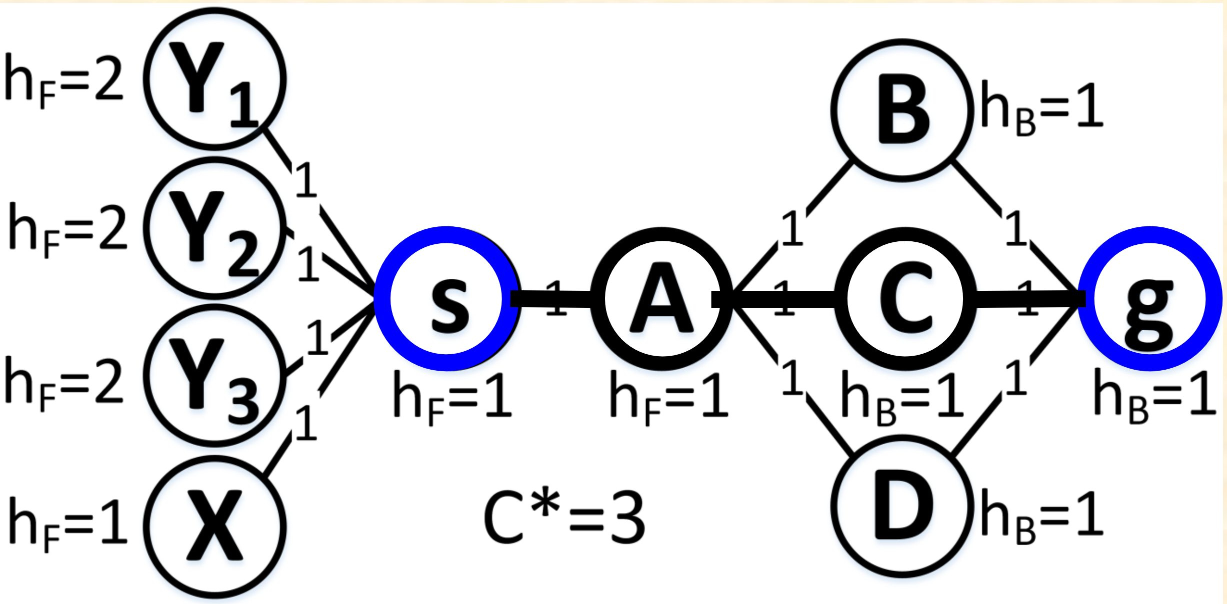


- In a MEP we must check whether there is a shorter path from **start** to **goal** via **u** and **v**
- In a MEP either ***u or v must be expanded*** to verify a C* solution

Must Expand Pairs



MEP	No MEP
A B	Y_1 B
$\mathbf{f}_F$(A)= 1+1 =2 <3	$\mathbf{f}_F$(Y_1)= 1+2 =3
$\mathbf{f}_B$(B)= 1+1 =2 <3	$\mathbf{f}_B$(B) = 1+1 =2 < 3
g_F(A)+g_B(B)= 1+1 =2<3	g_F(Y_1)+g_B(B)= 1+1 =2 <3

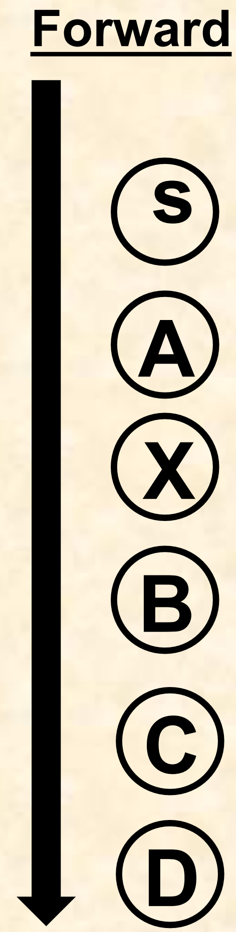


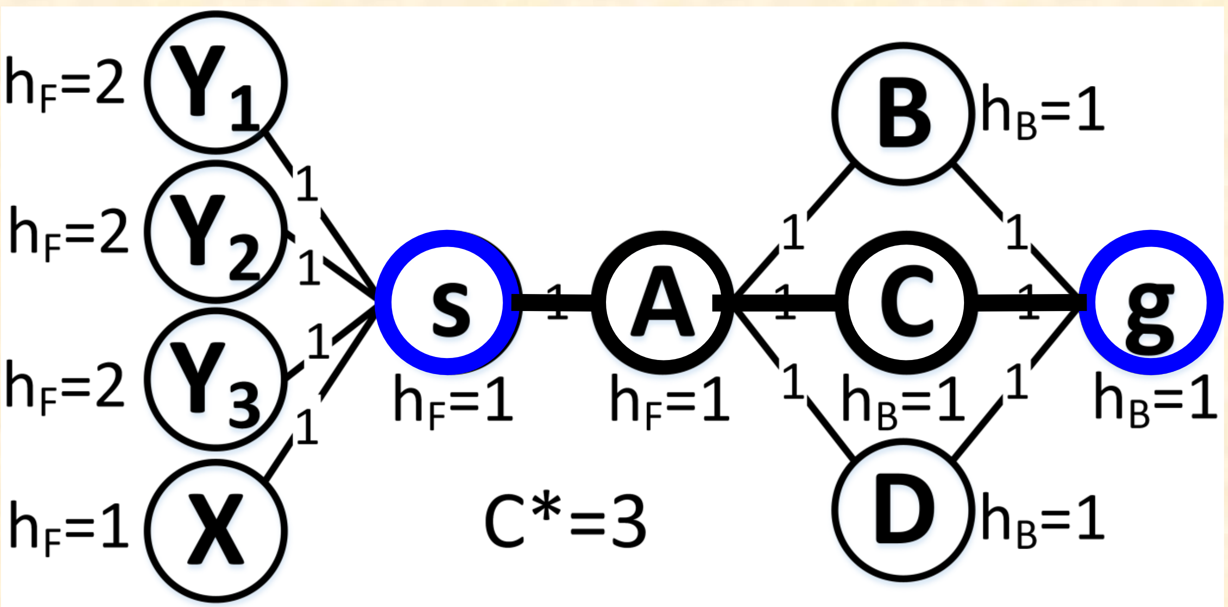
h_F=2
Y_1
h_F=2
Y_2
h_F=2
Y_3
h_F=1
X
s
h_F=1
A
h_F=1
B
h_B=1
C
h_B=1
D
h_B=1
g
h_B=1
1
C*=3

G_must-expand (GMX)

[Chen, Holte, Zilles, Sturtevant IJCAI-2017] (#5)

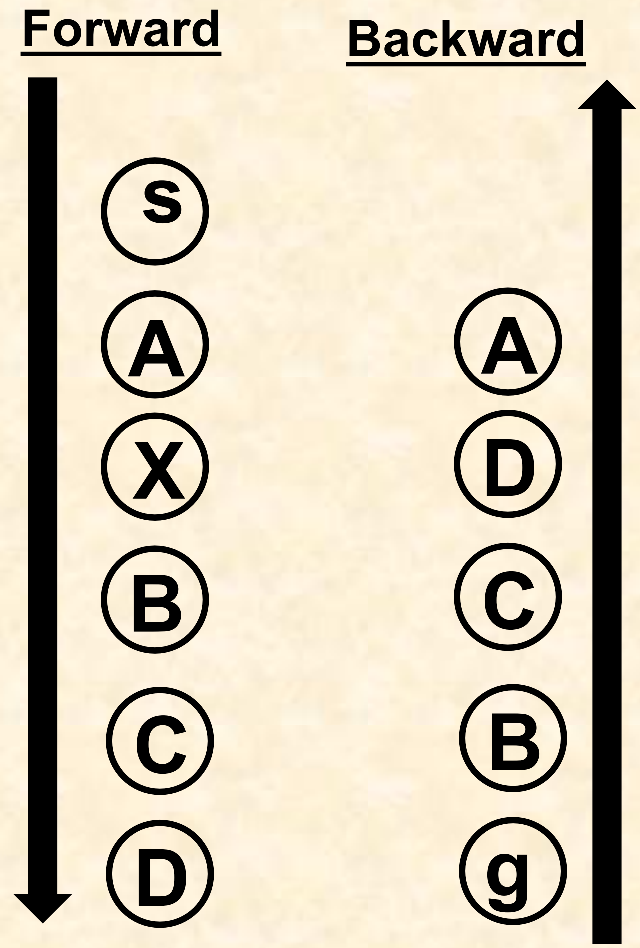
- A bipartite graph.
- Includes all forward nodes with f_F<C*

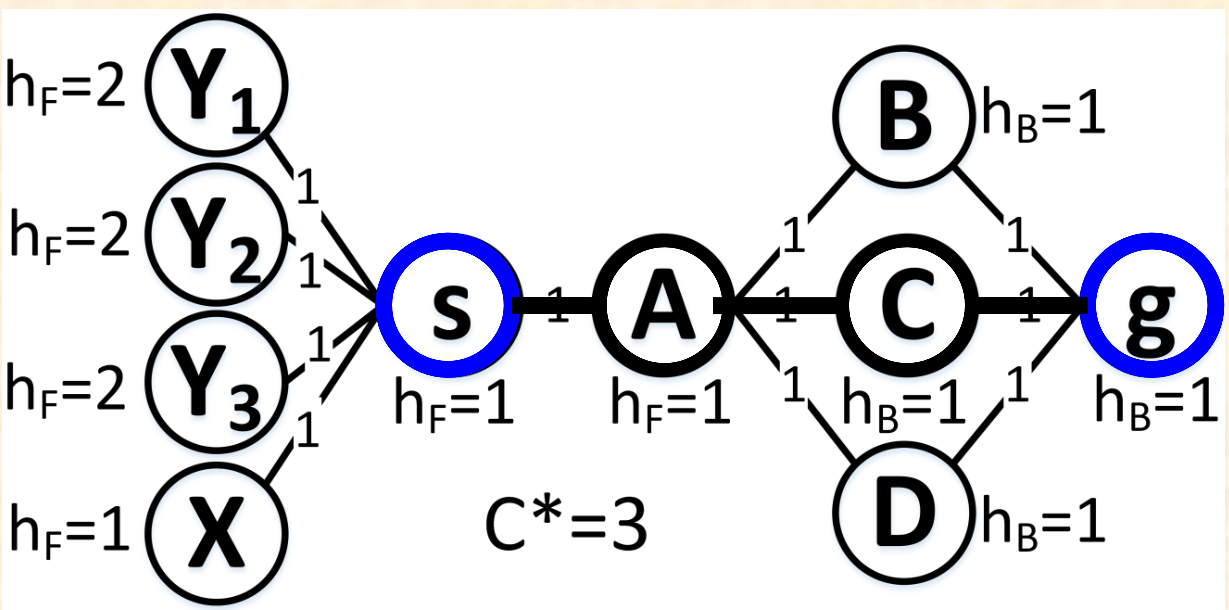




G_must-expand (GMX)

- A bipartite graph.
- Includes all forward nodes with f_F<C*
- Includes all backward nodes with f_B<C*

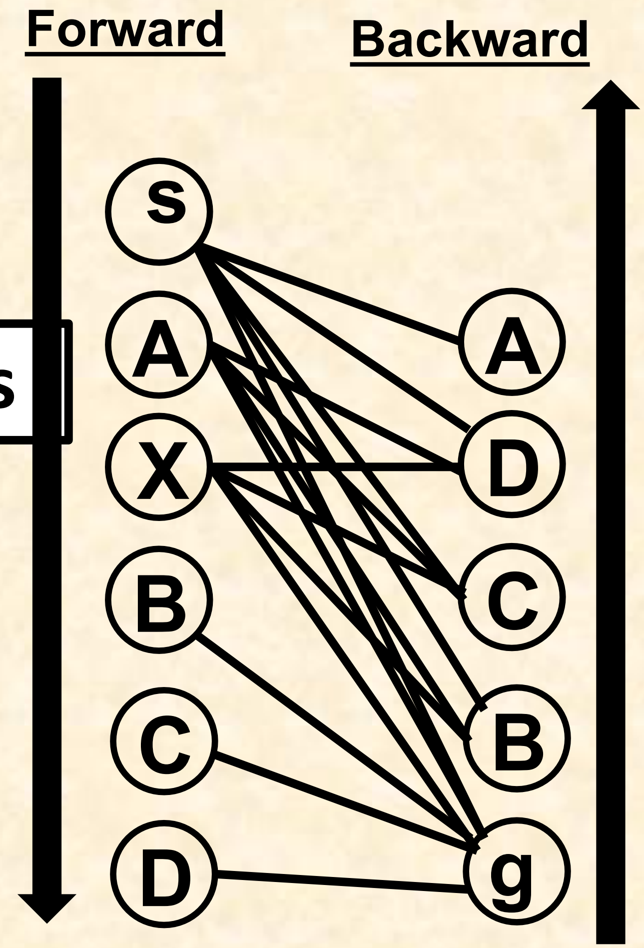


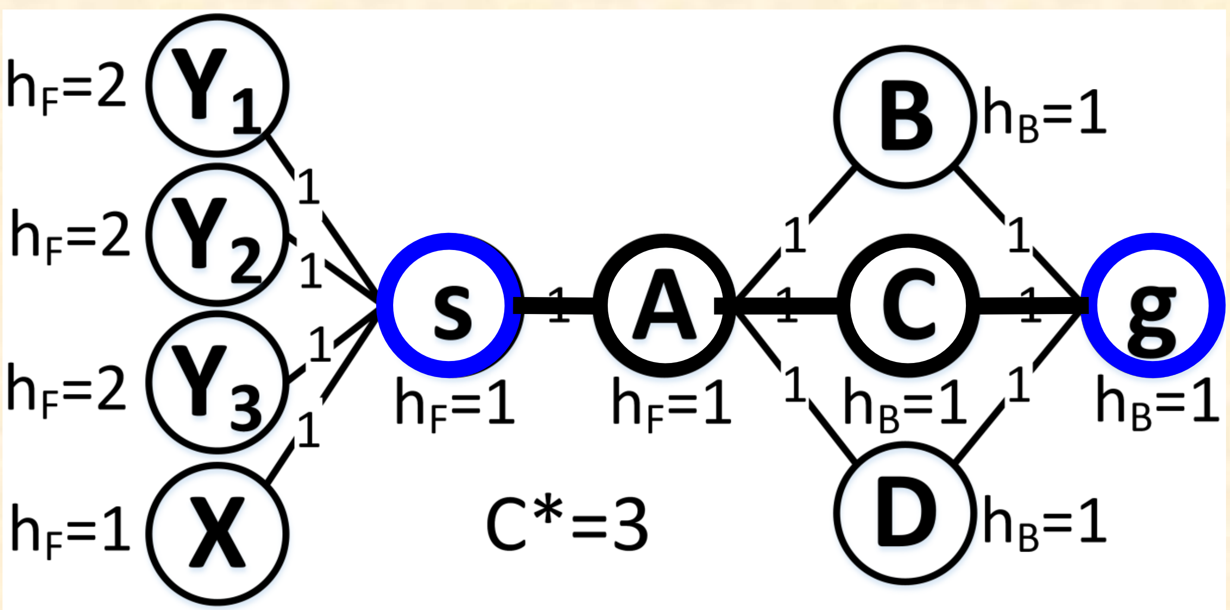


G_must-expand (GMX) [Chen, Sturtervant, Holte, Zilles, IJCAI-2017]

- A bipartite graph.
- Includes all forward nodes with f_F<C*
- Includes all backward nodes with f_B<C*
- Edges between nodes with g_F+g_B<C*

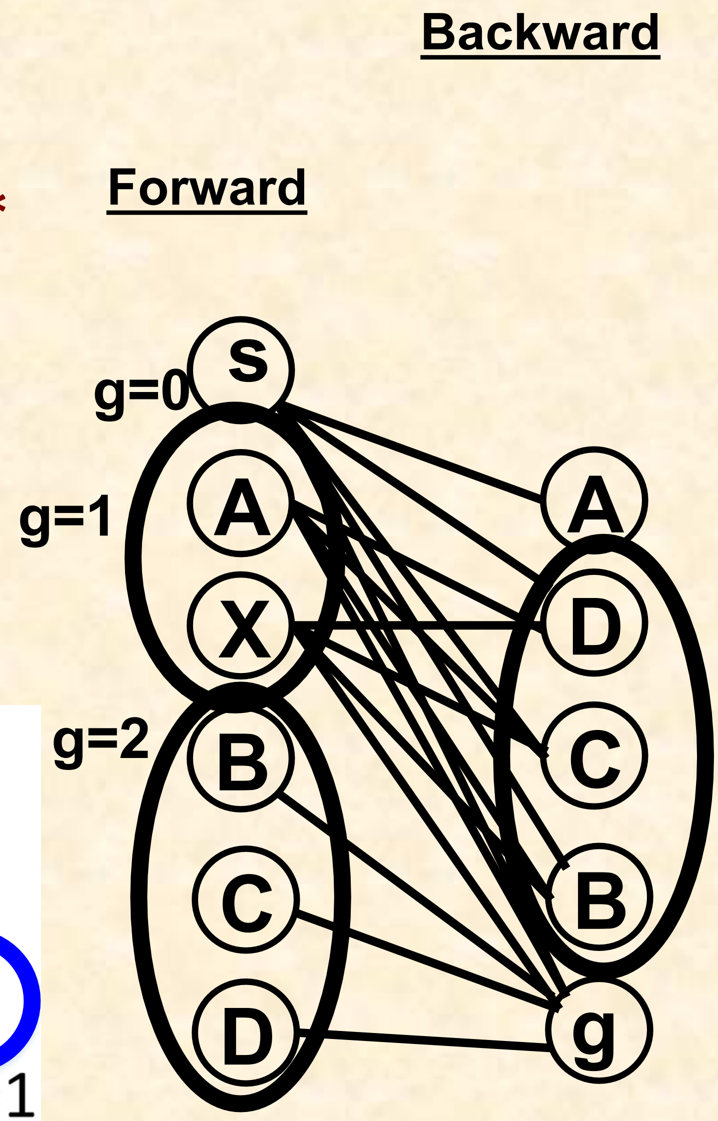
Edges exist between must-expand pairs

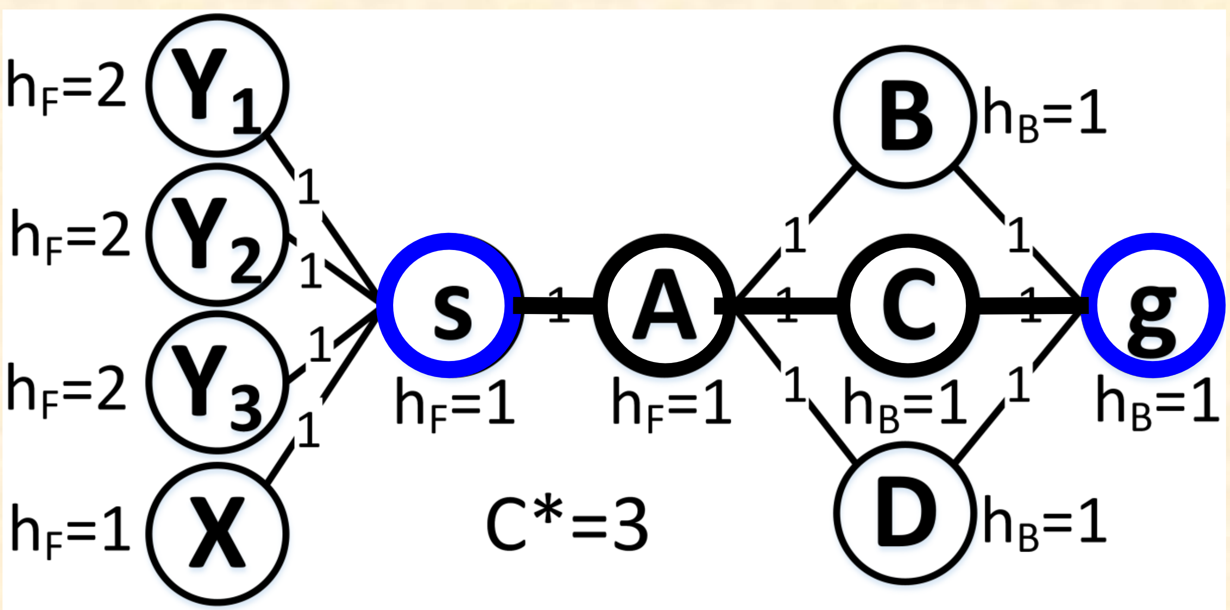




G_must-expand (GMX) [Chen, Sturtervant, Holte, Zilles, IJCAI-2017]

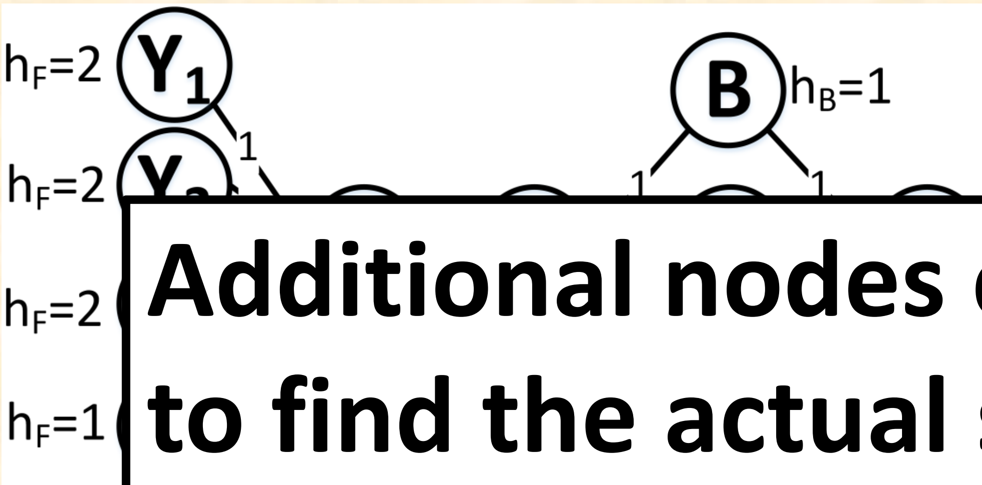
- A bipartite graph.
- Includes all forward nodes with f_F<C*
- Includes all backward nodes with f_B<C*
- Edges between nodes with g_F+g_B<C*
- Cluster nodes with the same g-value

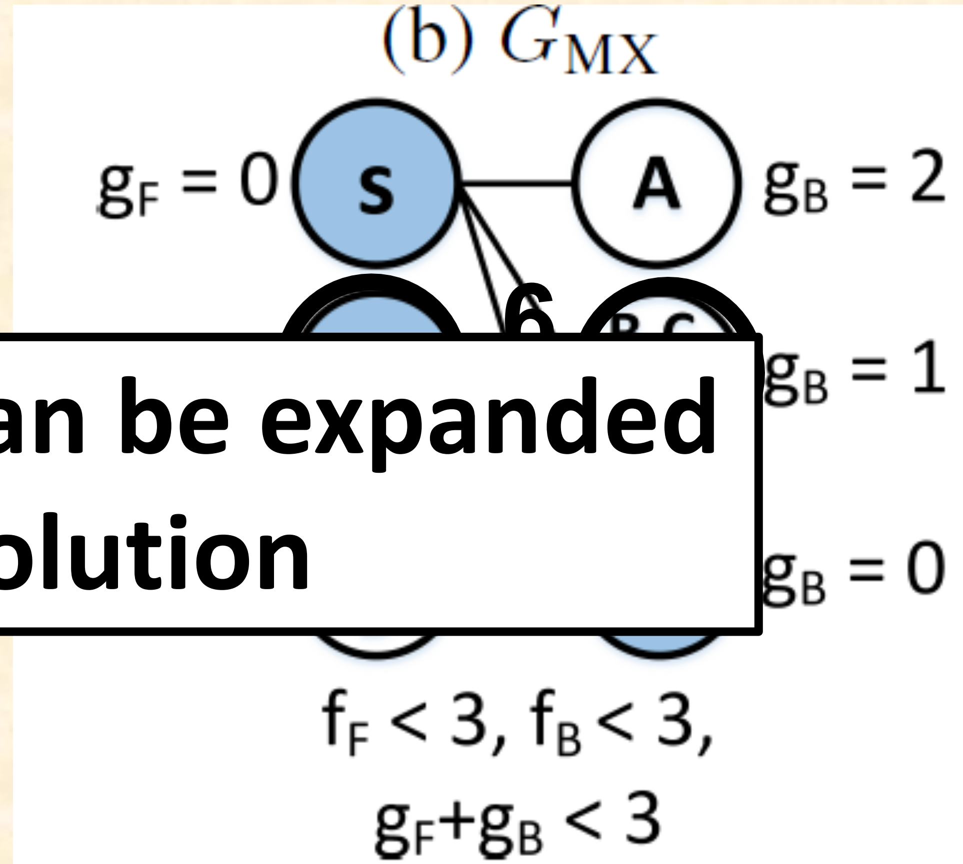




G_must-expand (GMX)

- GMX as clusters of nodes





Additional nodes can be expanded to find the actual solution

Every admissible algorithm must expand a VC of GMX

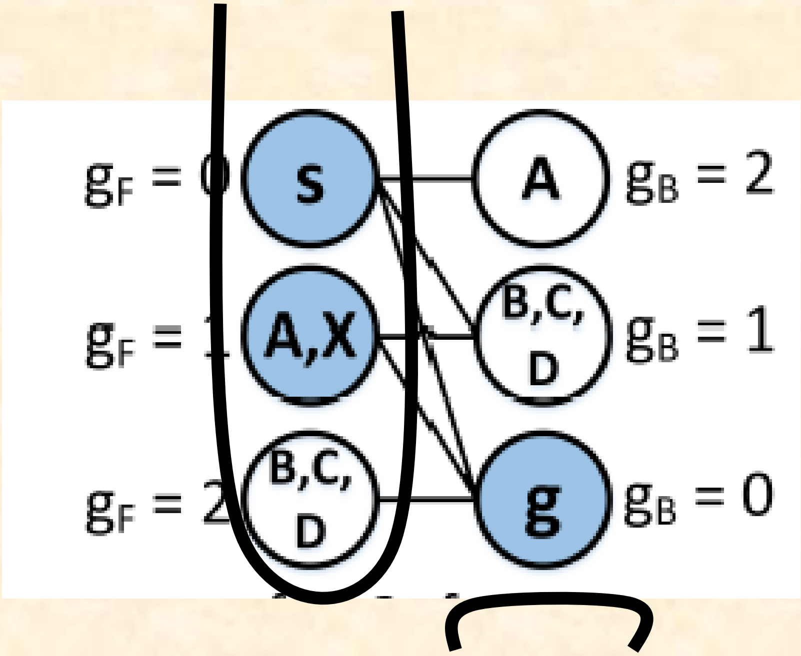
The Minimum Vertex Cover of GMX (MVC) is a lower bound

What does MVC of GMX looks like?

Properties of MVC of GMX

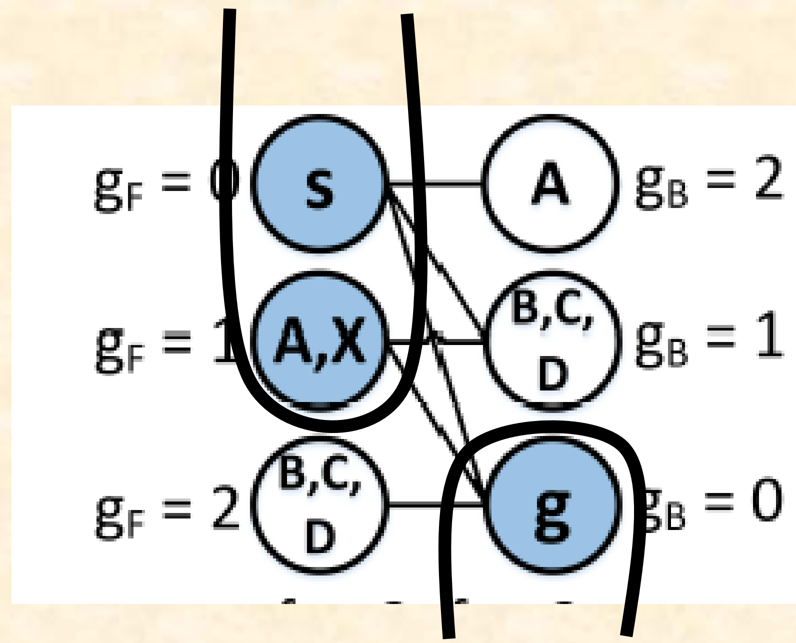
[Shaham, Felner Chen and and Sturtevant. SoCS-2017][#3]

Contiguous partition = VC



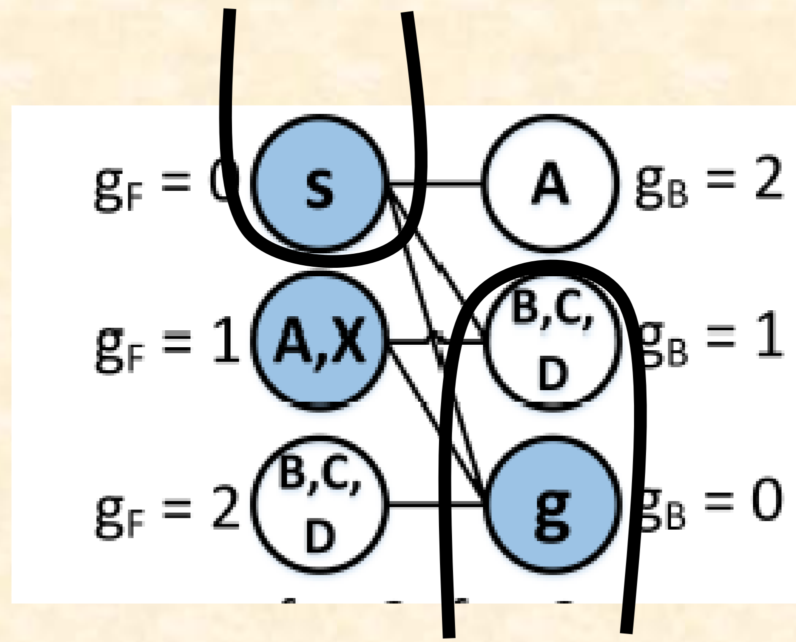
Properties of MVC [Shaham, Felner and Sturtevant. SoCS-2017]

Contiguous partition = VC



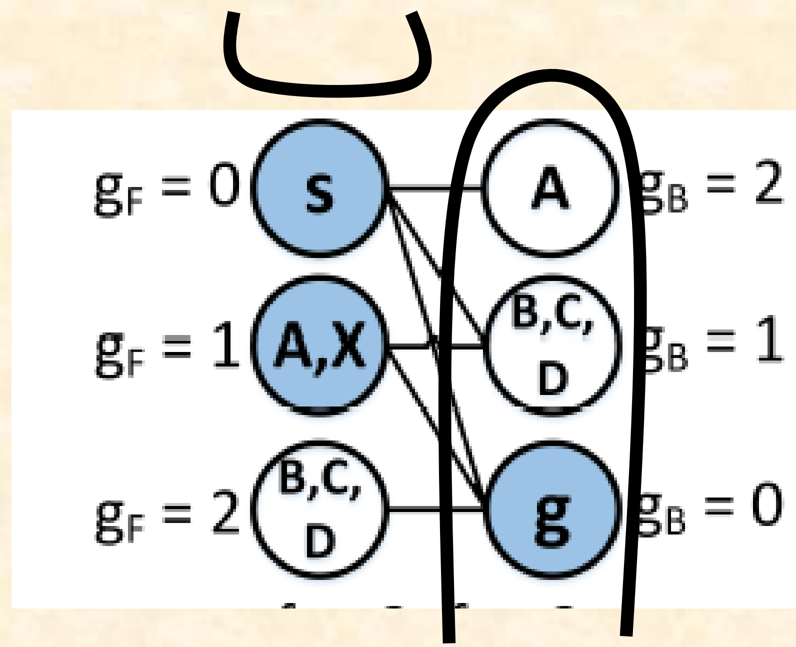
Properties of MVC [Shaham, Felner and Sturtevant. SoCS-2017]

Contiguous partition = VC



Properties of MVC [Shaham, Felner and Sturtevant. SoCS-2017]

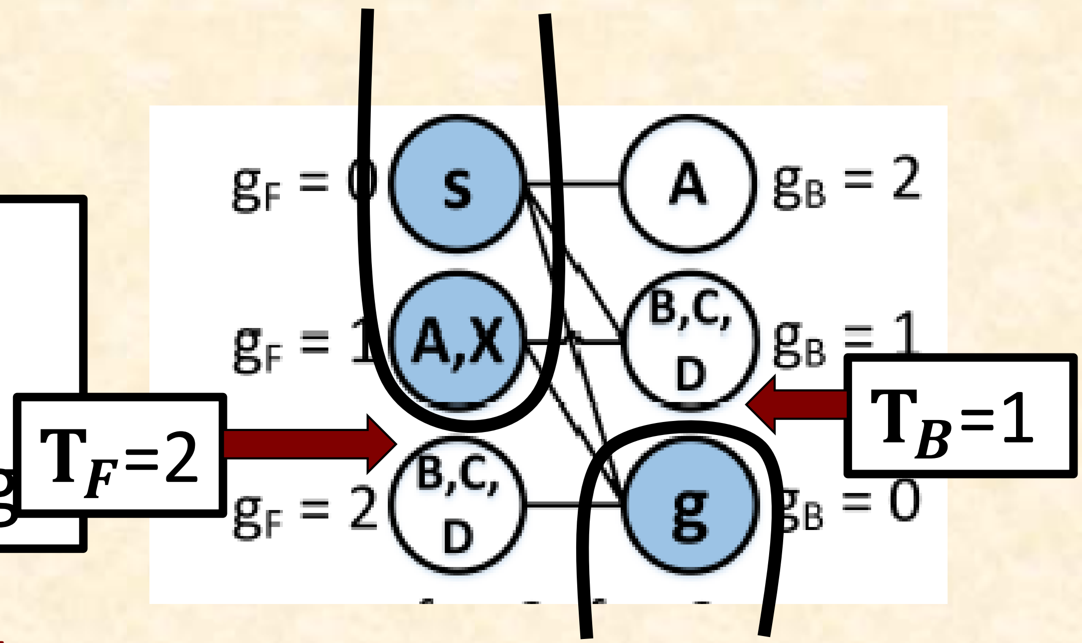
Contiguous partition = VC



Properties of MVC

[Shaham, Felner, Chen and Sturtevant. SoCS-2017]

Theorem:
MVC is one of these contiguous partitioning



There exist T_F+T_B=C* such that:

All nodes with g_F<T_F ∈ MVC

All nodes with g_B<T_B ∈ MVC

MVC of GMX is Restrained

fMM and MVC

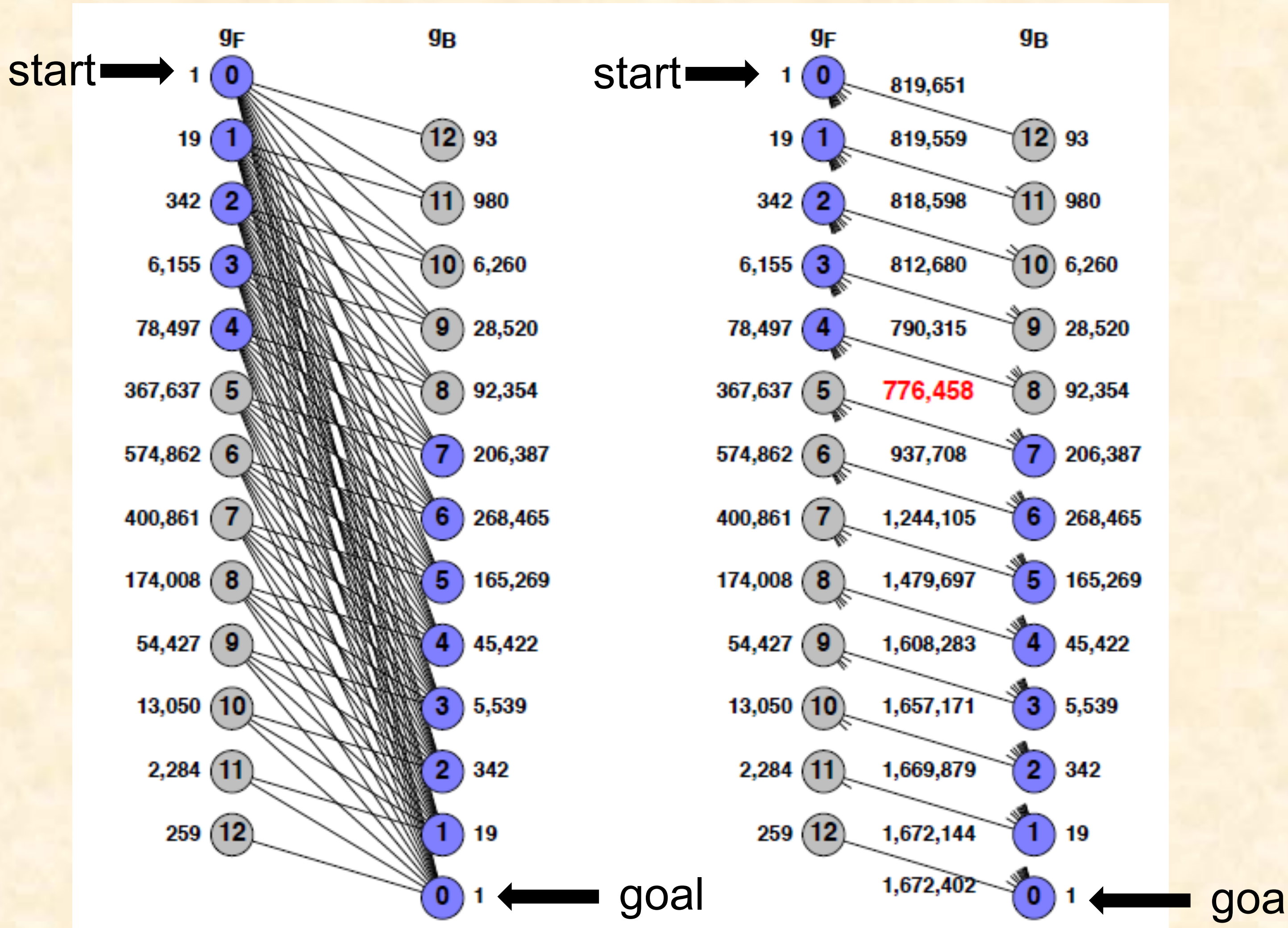
- fMM is restrained

fMM(P*) is equivalent to A*

Main result: There exists P* such that fMM(P*) is optimally efficient

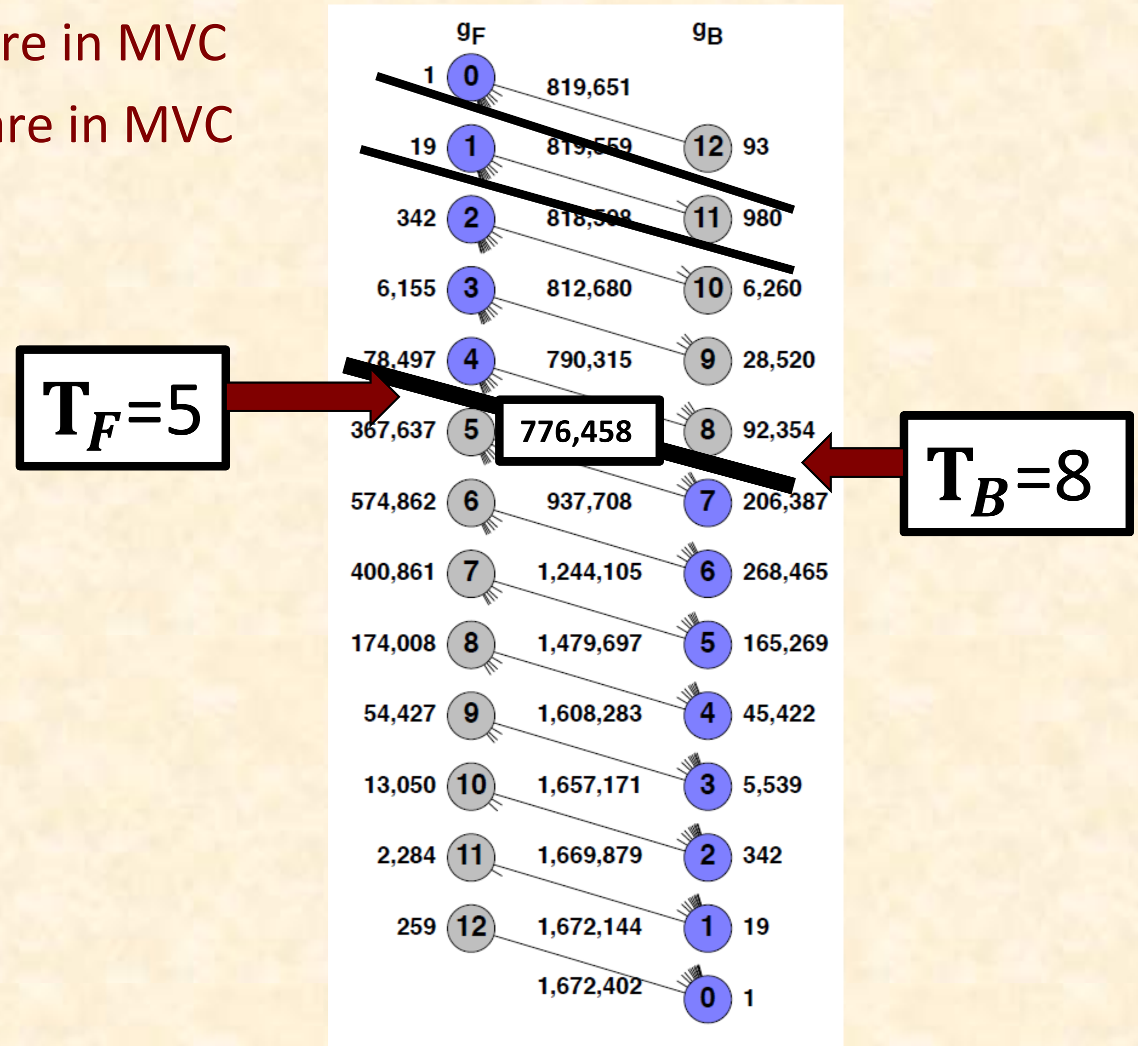
GMX for the pancake puzzle

- C*=13



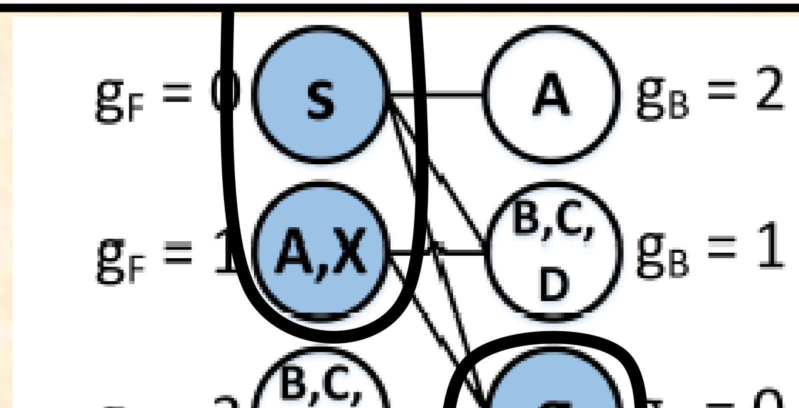
Properties of MVC [Shaham et al. 2018]

- Contiguous partitiongs
- There exist $\mathbf{T}_F+\mathbf{T}_B$=C* such that
  - All nodes with $g_F<\mathbf{T}_F$ are in MVC
  - All nodes with $g_B<\mathbf{T}_B$ are in MVC



Problem

GMX and C* are **not** known in advance →
P* cannot be known in advance either



Challenge: reason about GMX on the fly
and try to expand a VC fast

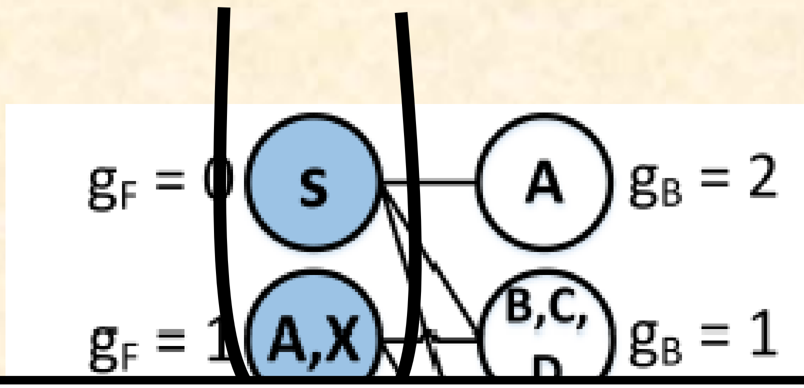
The NBS algorithm [Chen et al. 2017] and
The DVCBS algorithm [Shperberg et al. 2019]
try to expand a VC fast

Parametric Algorithms

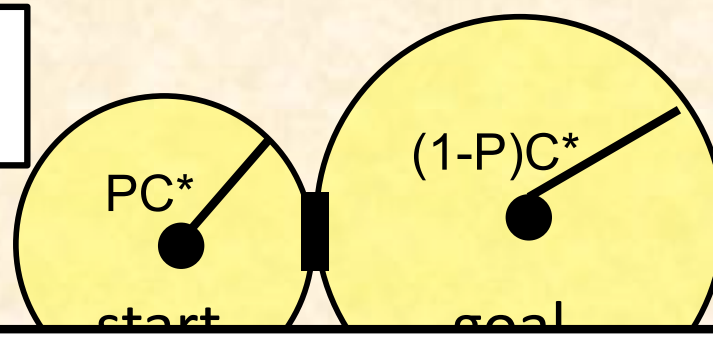
FMM and GBFSH

Two parametric algorithms which may expand exactly an MVC of GMX

1. **fMM(p)** [SoCs-2017] (fractional MM) meets at [pC*,(1-p)C*]



$$P = \frac{2}{3}$$



The optimal parameters (p*) are instance dependent and are not known in advance

2. **GBFSH [Barley et al., SoCS2018]** , requires a split function and expand nodes according to the split function.

Algorithm: GBFSH

[Barley et al. SoCS-2018] [#6]

- Define f_{lim} initialized to h(start,goal)
- f_{lim} is incremented by 1 in each iteration.

In each iteration:

- We split f_{lim}=$g_{Flim} + g_{Blim}$ (+e) according to an external split function
- In the forward side we expand all nodes n such that

  g_F(n)<g_{Flim} and f_F(n)≤f_{lim}

- In the Backward side we expand all nodes that

  g_F(n)<g_{Flim} and f_F(n)≤f_{lim}

- In each iteration one of $g_{F\mathrm{lim}}$ or $g_{B\mathrm{lim}}$ is increased.

- f_{lim}=2

  $g_{Flim} = 1$ g_{Blim}=1
- f_{lim}=3

  $g_{Flim} = 2$ g_{Blim}=1
- f_{lim}=4

  $g_{Flim} = 2$ g_{Blim}=2

- What are good split functions?
- How do we mimic MM?

GBFSH

- When f_{lim} and g_{Flim} are both increased but g_{Blim} remains the same
- In the forward side we:
  - 1) expand all old nodes ($g<$previous g_{Flim}) with f=f_{lim}
  - 2) expand new nodes with previous g_{Flim} but with f≤f_{lim}

Conjecture:

GBFSH and FMM are identical

Non-Parametric GMX-based Algorithms

The NBS algorithm [Chen et al. 2017] and
The DVCBS algorithm [Shperberg et al. 2019]
try to expand a VC fast

The NBS Algorithm [Chen, Holte, Zilles, Sturtevant, IJCAI-2017]

Near-optimal Bidirectional Search

Pair of nodes **(u,v)** is a ***must-expand pair (MEP)*** if:

$$\mathbf{f}_F(\mathbf{u})=\mathbf{g}_F(\mathbf{u})+\mathbf{h}_F(\mathbf{u}) < \mathbf{C}^*$$

$$\mathbf{f}_B(\mathbf{v})=\mathbf{g}_B(\mathbf{v})+\mathbf{h}_B(\mathbf{v}) < \mathbf{C}^*$$

$$\mathbf{g}_F(\mathbf{u})+\mathbf{g}_B(\mathbf{v}) < \mathbf{C}^*$$

The NBS Algorithm [Chen, Holte, Zilles, Sturtevant, IJCAI-2017]

Near-optimal Bidirectional Search

$$f_F(u)=g_F(u)+h_F(u)$$

$$f_B(v)=g_B(v)+h_B(v)$$

$$g_F(u)+g_B(v)$$

The NBS Algorithm [Chen, Holte, Zilles, Sturtevant, IJCAI-2017]

Near-optimal Bidirectional Search

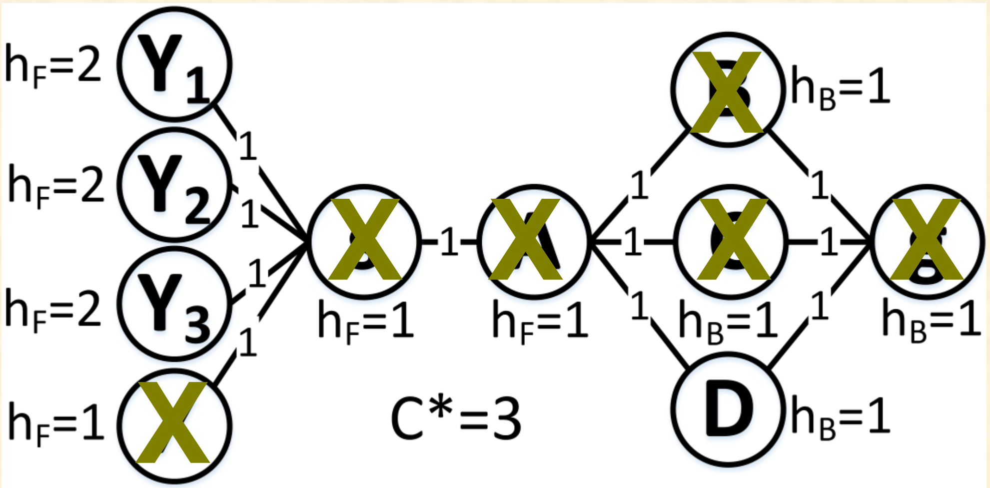
For each pair of nodes **(u,v)** we define:

.

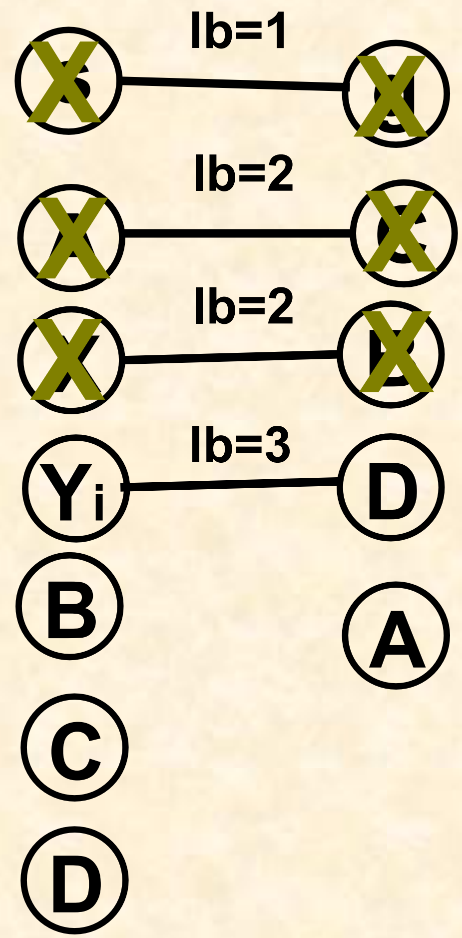
$$\mathbf{lb(u,v) = MAX} \begin{cases} \mathbf{f}_F(\mathbf{u}) = \mathbf{g}_F(\mathbf{u}) + \mathbf{h}_F(\mathbf{u}) \\ \mathbf{f}_B(\mathbf{v}) = \mathbf{g}_B(\mathbf{v}) + \mathbf{h}_B(\mathbf{v}) \\ \mathbf{g}_F(\mathbf{u}) + \mathbf{g}_B(\mathbf{v}) \end{cases}$$

- Find the pair (u,v) in open with minimal lb(u,v)
- Expand them **both**.

NBS

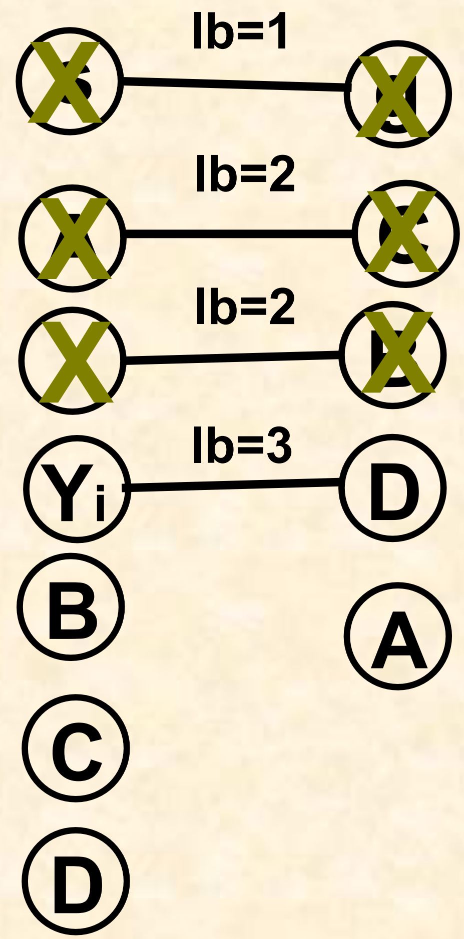


$h_F=2$ Y_1
$h_F=2$ Y_2
$h_F=2$ Y_3
$h_F=1$
1
$h_F=1$
$h_F=1$
$h_B=1$
$h_B=1$
$h_B=1$
D $h_B=1$
C*=3



lb=1
lb=2
lb=2
lb=3
Y_i
D
B
A
C
D

NBS: Main properties



1) NBS finds an optimal solution

2) NBS is at most twice than OPTIMAL

Why? Taking both vertices of disjoint edges is a VC **≤ 2 MVC**

3) No other algorithm can have a better worst-case bound

4) NBS is robust

3) New Algorithm:

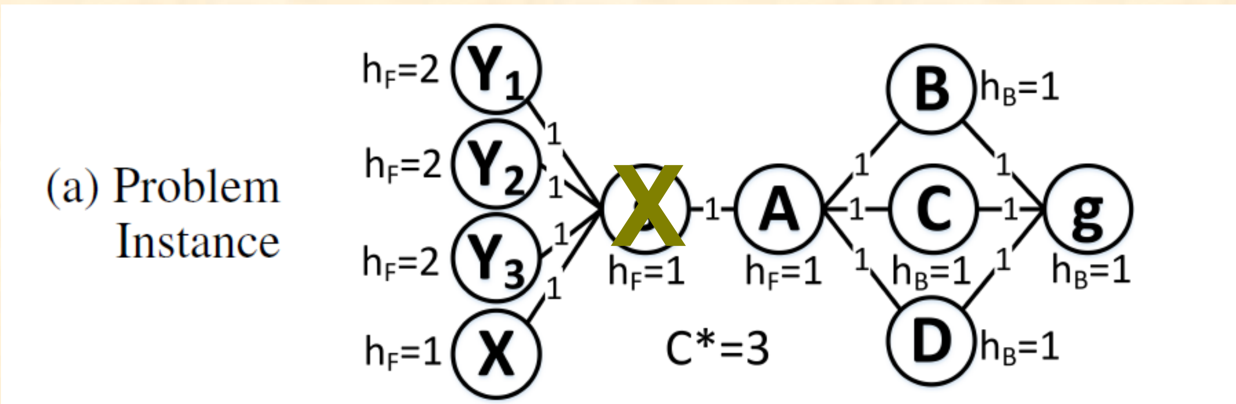
Dynamic Vertex-cover Bidirectional Search (DVCBS)

[Shperberg , Felner, Shimony and Sturtevant. AAAI 2019][#7]

- NBS expanded both nodes
- DVCBS maintains dynamic GMX (DGMX) that uses the currently known information from Open nodes
- Repeatedly find MVC of DGMX and expand it

Many variants exist

Execution of DVCBS

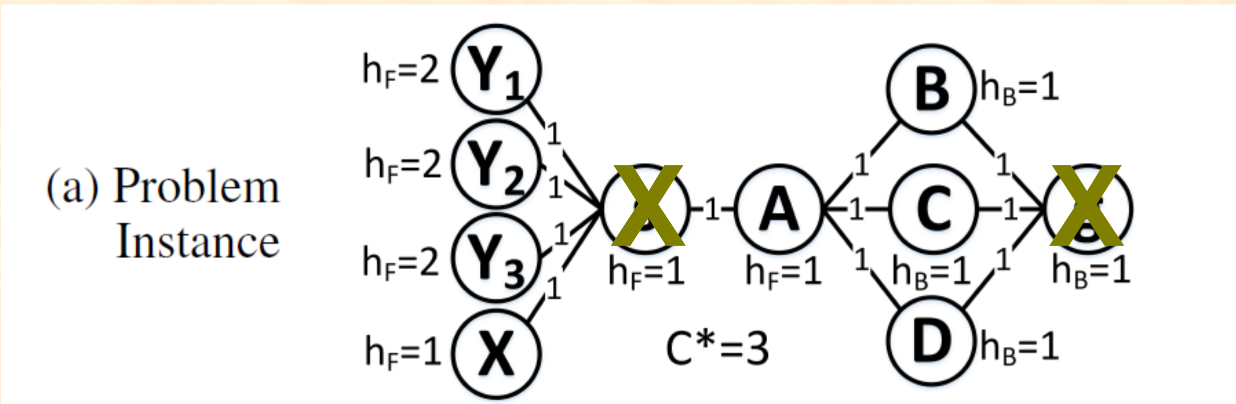


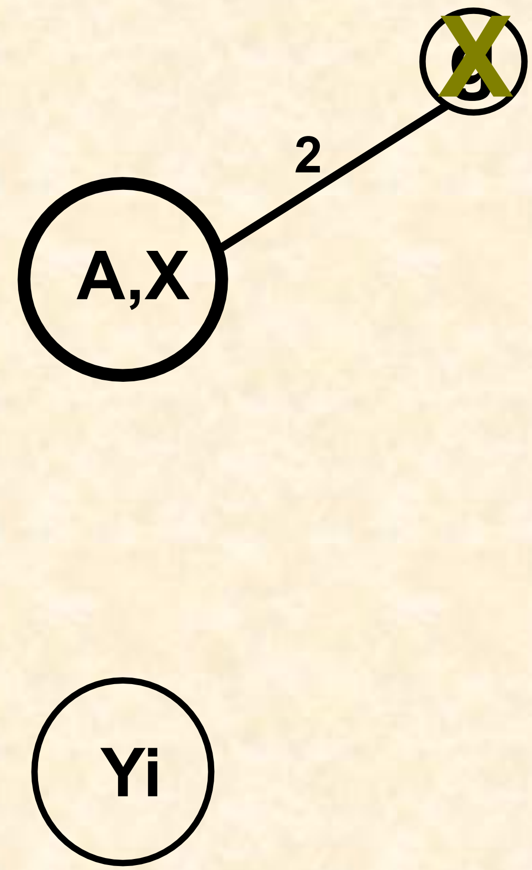
DGMX

lb=1

Execution of DVCBS

DGMX

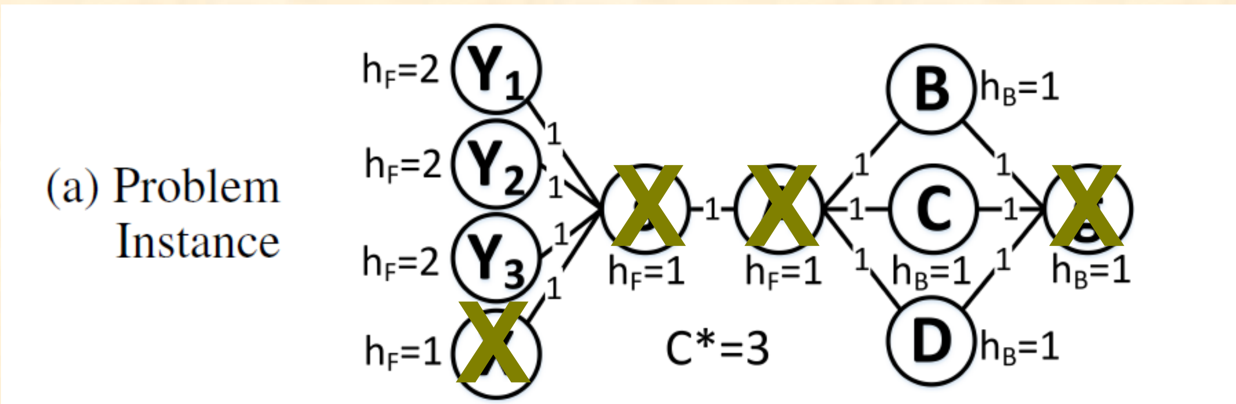


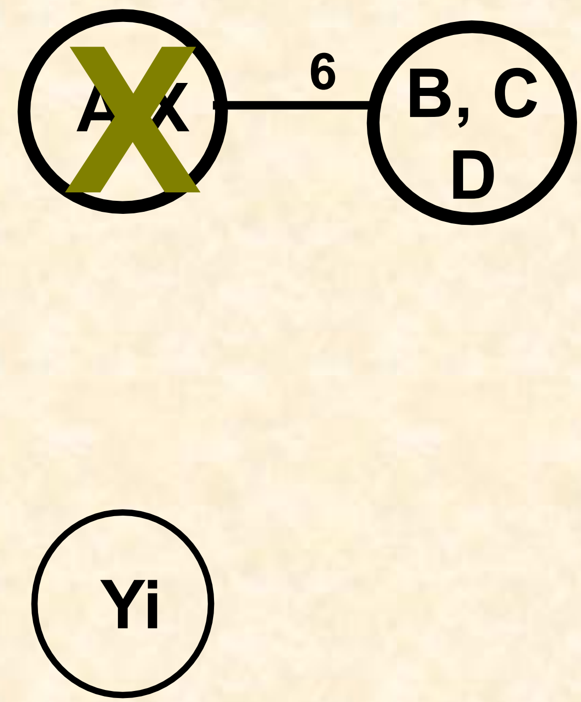


lb=2

Execution of DVCBS

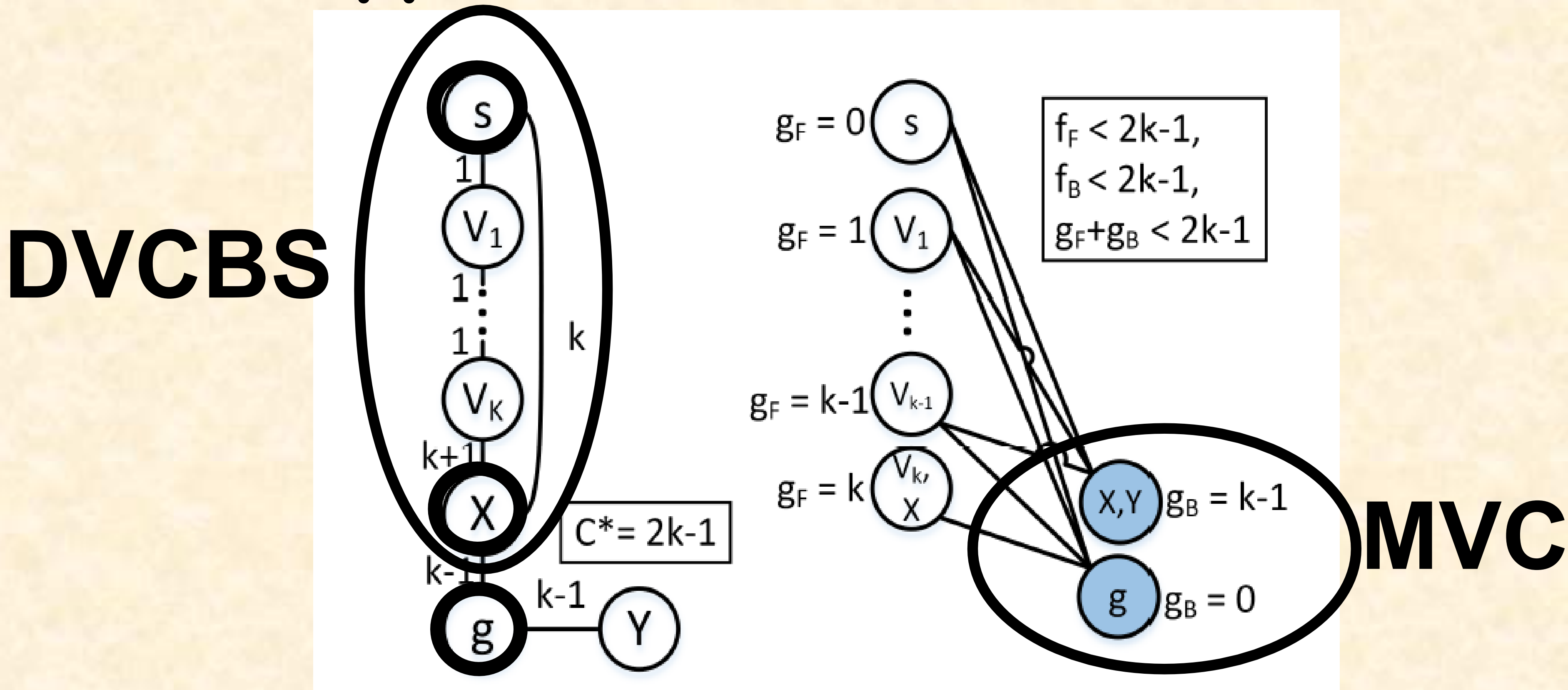
DGMX





lb=2

No upper bound for DVCBS



- Optimal path s,x, g. Cost 2K-1.
- MVC is {X,Y,g}. NBS expans 6 nodes.
- DVCBS never expands Y.
  - Generates (X,Y). This is a cluster of 2 nodes.
- It expands all the Vi nodes. K+1 nodes. Unbounded.

Experiments

All Algorithms: Nodes Expanded

		VC	Ratio VC/MVC	First solution		
20-Pancake Puzzle						
Gap-2	A*	322,299	2.65	322,378		
	NBSF	208,648	1.71	209,723		
	NBSA	151,616	1.24	152,046		
	DVSBSF	141,111	1.16	141,669		
	DVCBSA	122,054	1.00	122,587		
4-peg Towers of Hanoi						
6+6	A*	3,239,287	4.75	3,268,093		
	NBSF	234,165	1.91	234,165		
	NBSA	232,268	1.89	232,268		
	DVCBSF	704,213	1.03	707,679		
	DVCBSA	690,389	1.01	691,159		

DVCBSA is the winner in all aspects, many time is exactly MVC

Summary

- Non-parametric GMX-based algorithms

  - **NBS -** worst case guarantee (2x)

  - **DVCBS -** no guarantee but better average-case performance

Case 2

Assuming Consistent Heuristic

Assumptions [Dechter & Pearl 85]

Problem Instances

Traditionally, the analysis assumed that:

1) The algorithm can only assume admissibility
2) The actual instances are from $\mathbf{I}_{CON}$

The algorithms cannot exploit the fact that they are running on consistent heuristics

[Shaham, Felner, Sturtevant ans Rosenchein. SoCS-2017] [#8]

What happens if the algorithms have more knowledge on the instances?

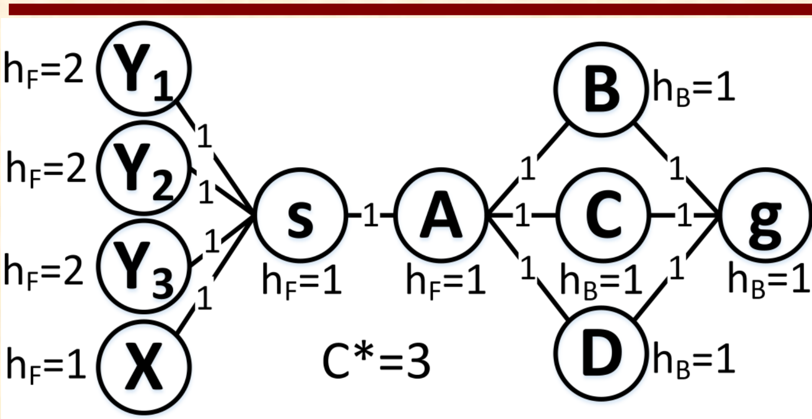
Case 1: Knowing Epsilon

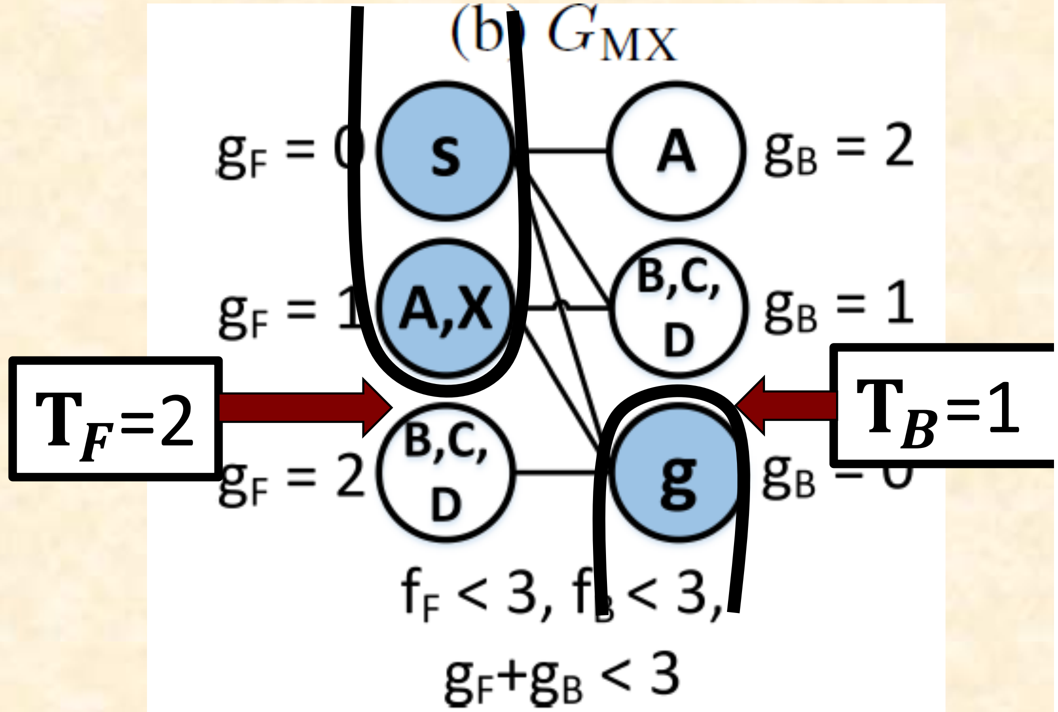
- Sometimes we have a lower bound ε on the edge costs

Pair of nodes **(u,v)** are a ***must-expand pair (MEP)*** if:

1) $\mathbf{f}_F(\mathbf{u})=\mathbf{g}_F(\mathbf{u})+\mathbf{h}_F(\mathbf{u}) < C^*$
2) $\mathbf{f}_B(\mathbf{v})=\mathbf{g}_B(\mathbf{v})+\mathbf{h}_B(\mathbf{v}) < C^*$
3) $\mathbf{g}_F(\mathbf{u})+\mathbf{g}_B(\mathbf{v})+\varepsilon < C^*$

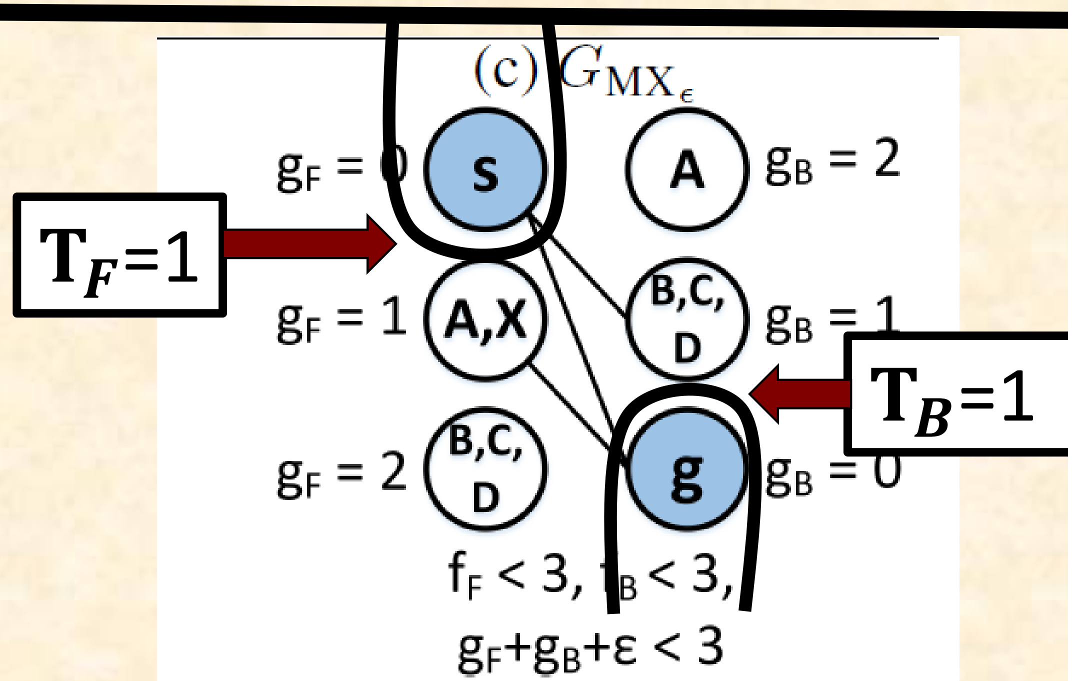
GMX vs GMXe





No knowledge on ε

Assuming ε=1



Fractional MM – fMM(P)

$0 \le P \le 1$

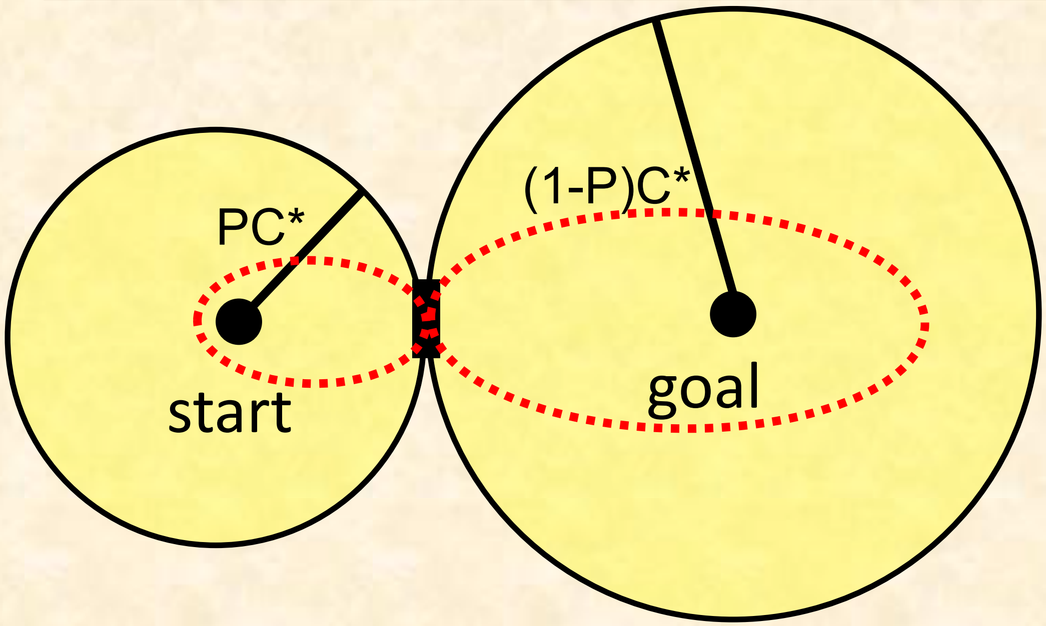
Forward side:

$$pr(n) = \max \begin{cases} g_F(n) + h_F(n) \\ g_F(n)/P + \varepsilon \end{cases}$$

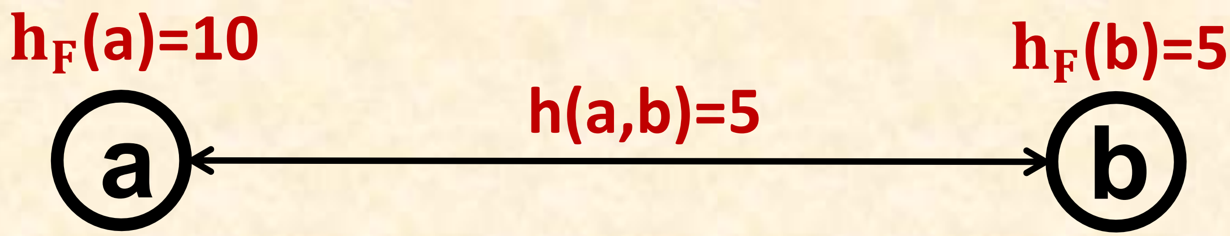
Backward side:

$$pr(n) = \max \begin{cases} g_B(n) + h_B(n) \\ g_B(n)/(1-P) + \varepsilon \end{cases}$$

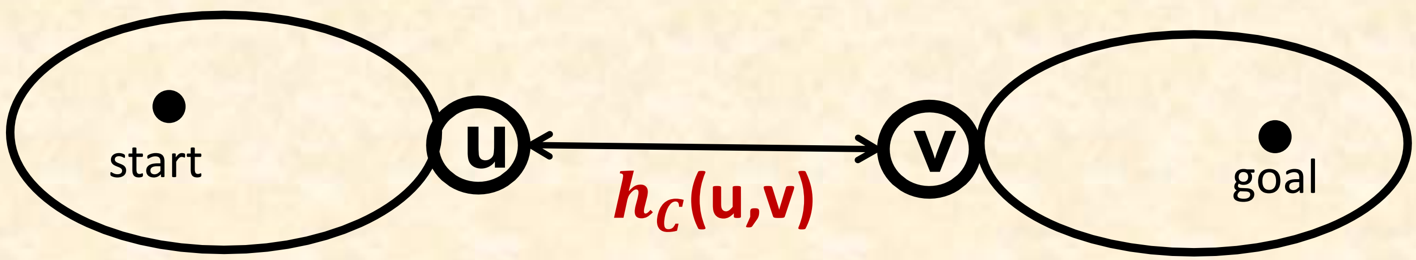
Will meet at PC*,(1-P)C*



Case 2: Assuming consistency



We can construct a front-to-front heuristic h_C



$$h_C(u,v)=\max\begin{cases}|h_F(u)-h_F(v)|\\ |h_B(u)-h_B(v)|\end{cases}$$

Case 2: assuming consistency

Pair of nodes **(u,v)** are a ***must-expand pair (MEP)*** if:

1) $f_F(u)=g_F(u)+h_F(u) < C^*$
2) $f_B(v)=g_B(v)+h_B(v) < C^*$
3) $g_F(u)+g_B(v) + h_C(u,v) < C^*$

What does MVC of GMX look like now?

It is not restrained
We have a counter example

Case 2: assuming consistency

- In GMX for each nodes we have two new dimensions:
  - **(1)** $\mathbf{h}_{\boldsymbol{F}}$-value
  - **(2)** $\mathbf{h}_{B}$-value

- In this case there isn't any one threshold T for MVC but a **matrix** of thresholds **T**, based on the $\mathbf{h}_{\boldsymbol{F}}$- and $\mathbf{h}_{B}$-values

	$\mathbf{h}_{\boldsymbol{F}}$=1	$\mathbf{h}_{\boldsymbol{F}}$=2	$\mathbf{h}_{\boldsymbol{F}}$=3
$\mathbf{h}_{B}$=1	4	5	4
$\mathbf{h}_{B}$=2	3	4	5
$\mathbf{h}_{B}$=3	2	3	4

	h_F=1	h_F=2	h_F=3
h_B=1	4	5	4
h_B=2	3	4	5
h_B=3	2	3	4

There exists a 2-dimentional function T(h_F,h_B) that provides these thresholds

$$|\mathrm{T}(x_1, y_1) - \mathrm{T}(x_2, y_2)| \leq \max\{|x_1 - x_2|, |y_1 - y_2|\}$$

Very similar to a 1-Lipschitz requirement in math

Summary

- fMM is restrained
- MVC of GMX is restrained
- fMM(P*) is optimally efficient
- fMM(P($\mathbf{h}_{\boldsymbol{F}}$(n),$\mathbf{h}_{\boldsymbol{B}}$(n)) is optimally efficient if the algorithm can exploit the fact that the heuristic is consistent

Bound propagations

Shperberg, Felner, Shimony and Stortevant, SoCS-2019] [#9]

$$\mathbf{lb(u,v)} = \mathbf{MAX}\begin{cases} \mathbf{f}_F(\mathbf{u}) = \mathbf{g}_F(\mathbf{u}) + \mathbf{h}_F(\mathbf{u}) \\ \mathbf{f}_B(\mathbf{v}) = \mathbf{g}_B(\mathbf{v}) + \mathbf{h}_B(\mathbf{v}) \\ \mathbf{g}_F(\mathbf{u}) + \mathbf{g}_B(\mathbf{v}) \end{cases}$$

lb(u)= **min** v' {lb(u,v')}

f-values are changed to their lb-values

New algorithm assuming consistency

DIBBS: Sewel and Jaconson (AIJ)

BEA*: [Alcazar, Barley and Riddle (AAAI-2020)

$\Delta(u) = g_F(u) - h_B(u, start)$

$\Delta(v) = g_B(v) - h_F(v, goal)$

start

$b(x) = 2g_F(x) + h_F(x) - h_B(x)$

$b(x) = 2g_B(x) + h_B(x) - h_F(x)$

$b(x) + b(x) = 2g_F(x) + 2g_B(x)$

$(b(x) + b(x))/2 = g_F(x) + g_B(x)$

$b(u) = f_F(u) + \Delta$ …

$b(u) = g_F(u) +$ …

$b(u) = 2g_F(u) + h_F(u) - h_B(u)$

… $) - h_F(v)$

Case 2: assuming consistency

Pair of nodes **(u,v)** are a ***must-expand pair (MEP)*** if:

1) $g_F(u)+h_F(u)+\Delta(v) < C^*$

2) $g_B(v)+h_B(v)+\Delta(u) < C^*$

3) $g_F(u)+g_B(v)+h_C(u,v) < C^*$

$$h_C(u,v)=\max\begin{cases}|h_F(u)-h_F(v)|\\ |h_B(u)-h_B(v)|\end{cases}$$

Case 2: assuming consistency

Pair of nodes **(u,v)** are a ***must-expand pair (MEP)*** if:

1) $g_F(u)+h_F(u)+\ g_B(v)-h_F(v) < C^*$

$g_F(u)+g_B(v)+h_F(u)-h_F(v) < C^*$

2) $g_B(v)+h_B(v)+\ g_F(u)-h_B(u) < C^*$

$g_B(v)+g_F(u)+h_B(v)-h_B(u) < C^*$

3) $g_F(u)+g_B(v)+h_C(u,v) < C^*$