BREN:ICS INFORMATION AND COMPUTER SCIENCES UNIVERSITY of CALIFORNIA IRVINE

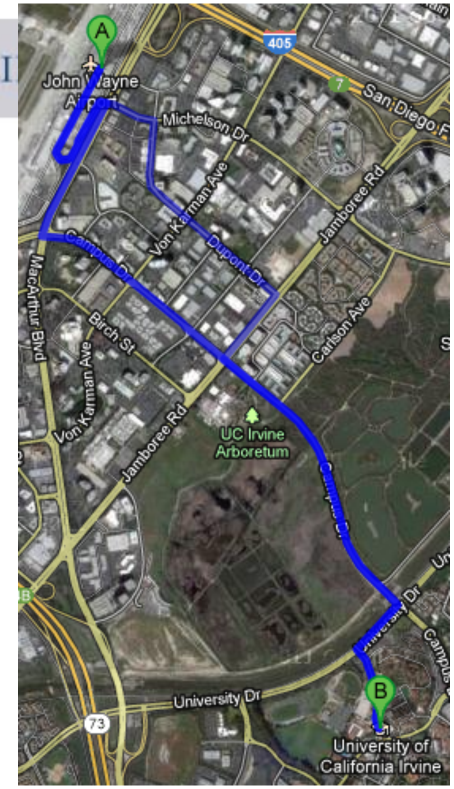
Searching for *m* Best Solutions for Graphical Models

Rina Dechter and Natalia Flerova

University of California Irvine
USA

Radu Marinescu

IBM Research
Ireland



Motivation

- Optimization problems:
  - Finding the best solution
  - Finding the **m-best** solutions

- Applications of the m-best solutions:
  - Set of diverse solutions desired (e.g., haplotyping)
  - Informal preferences (e.g., navigation)
  - Sensitivity analysis (e.g., bio sequence alignment)

Related work

- Find solutions **iteratively** by **restating** the problem
  - **Lawler, 1972;** Nilsson, 1998; Yanover and Weiss, 2004; Fromer and Globerson, 2009;
- Find **entire set** of solutions using **message-passing**
  - Seroussi and Golmard, 1994 ; Elliot, 2007; Rollon, Flerova, Dechter 2011;
- Solve related **k shortest path** task
  - Eppstein 1997; Aljazzar and Leue 2011;

Our contribution:

- Extending **search algorithms** for the **m-best task**

  Specifically:

  - Best First search (specifically, A*)
  - Depth First Branch and Bound search

- Solving **combinatorial optimization** problems over **graphical models**

  In particular, extending **AND/OR search:**

  - AND/OR Best First search
  - AND/OR Branch and Bound

[Marinescu, 2009]

Outline:

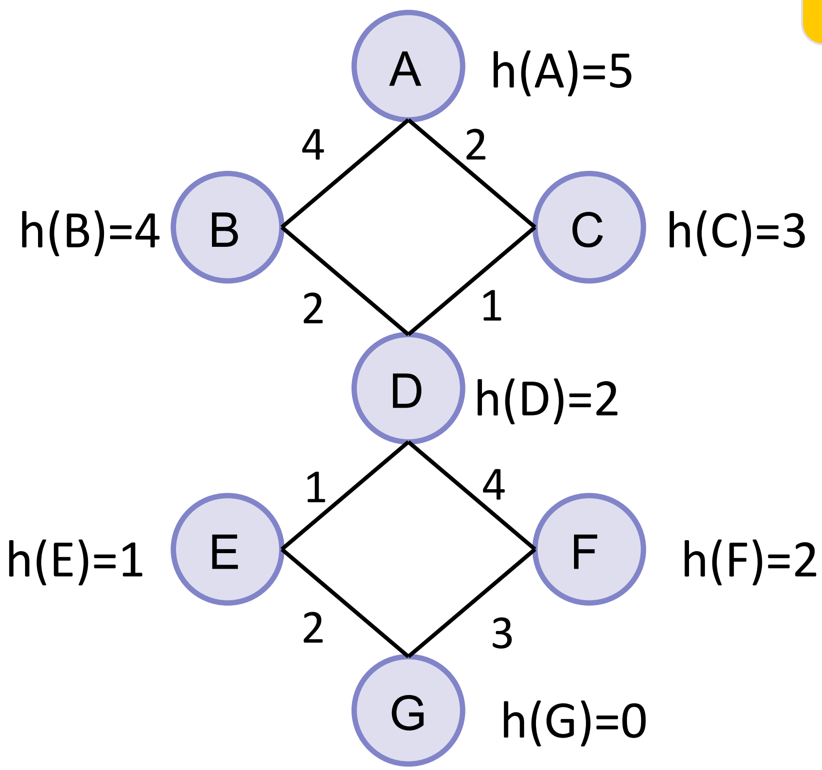
- Introduction
- **Best First search for m-best**
  - Background
  - m-A* algorithm
  - Properties of m-A*
- Branch and Bound for m-best
  - Background
  - m-BB algorithm
- Application to graphical models
- Empirical evaluation

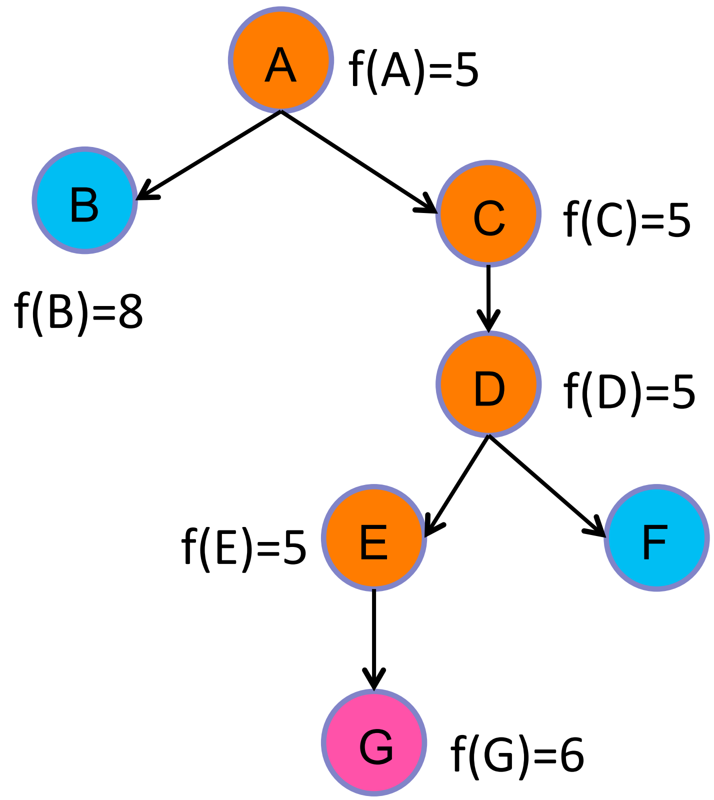
Background: Best First search

- **Evaluation function**: *f(n)*
- **A***: f(n)=g(n)+h(n)

cost from start to n (g(n))

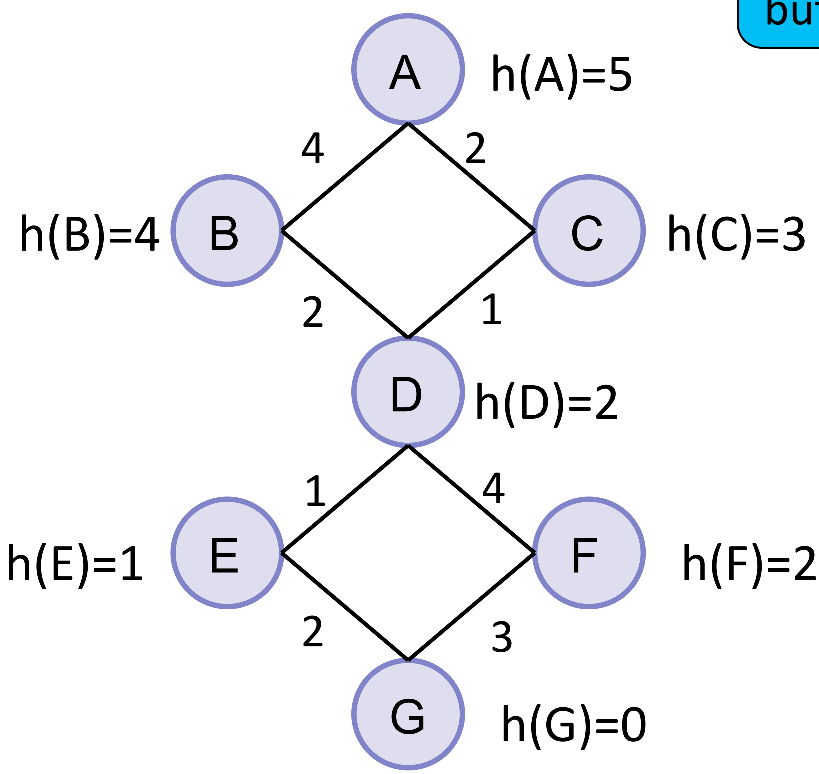
estimated cost from n to goal (h(n))



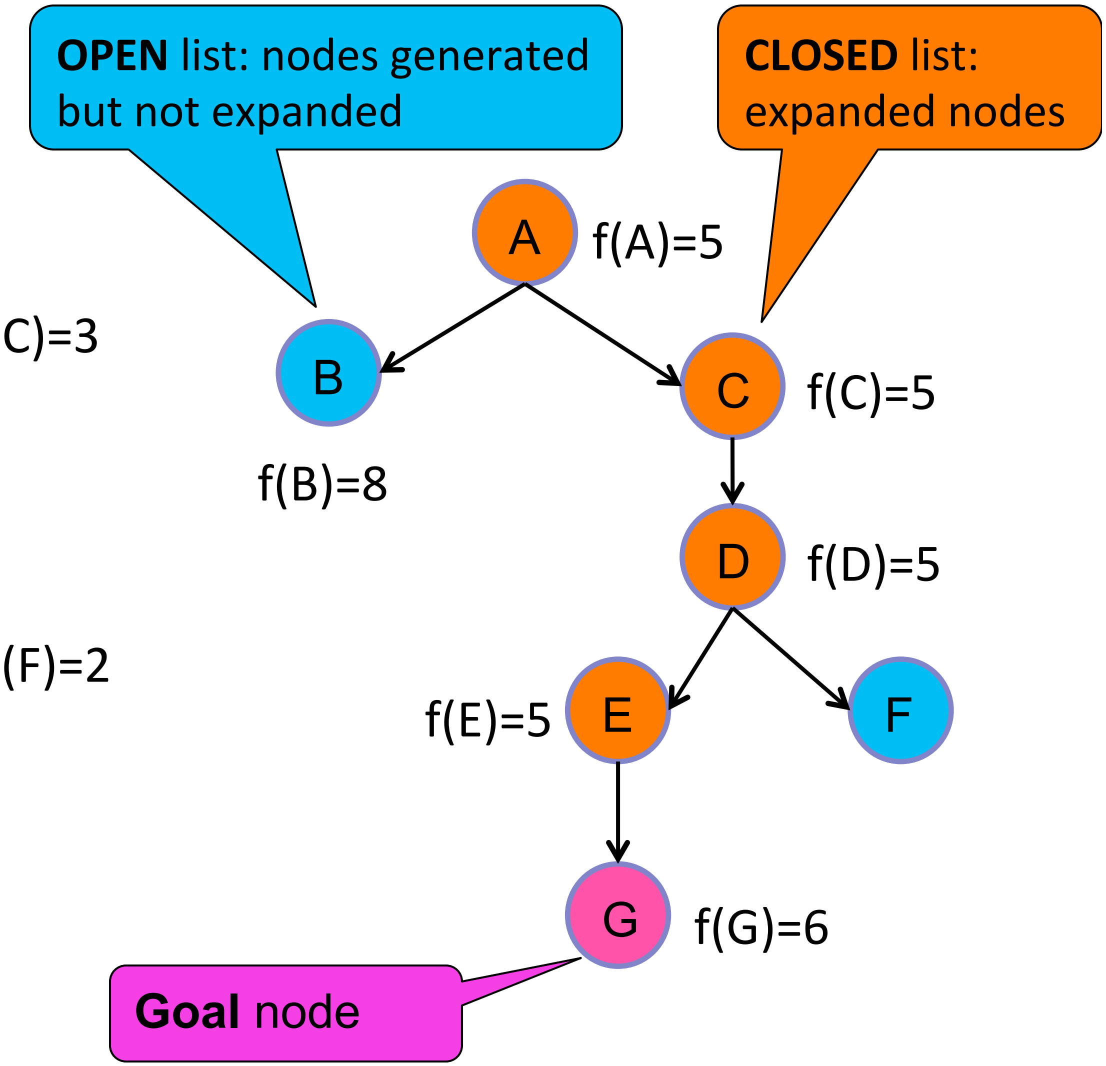


Background: Best First search

- **Evaluation function**: *f(n)*



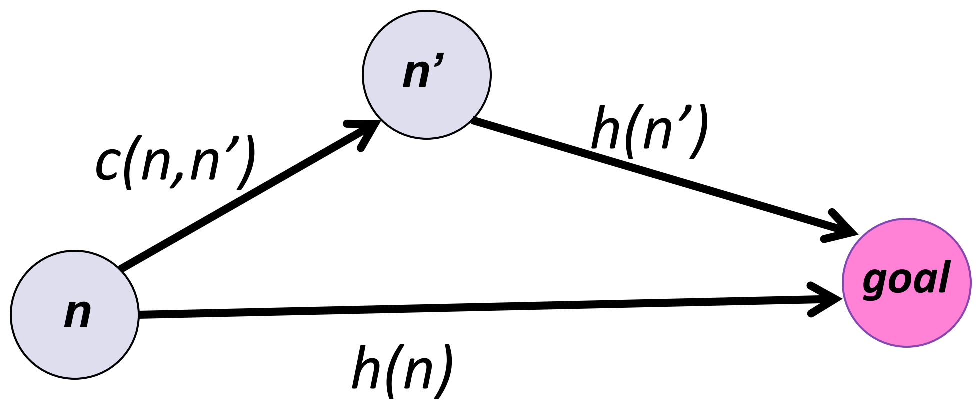
- **A***: f(n)=g(n)+h(n)



Background: heuristic evaluation functions

- **Admissible** heuristic: $h(n) \le h^*(n)$
- **Consistent** (or monotone) heuristic:

$$h(n) \le c(n,n') + h(n')$$



Algorithm m-A*

Main idea:

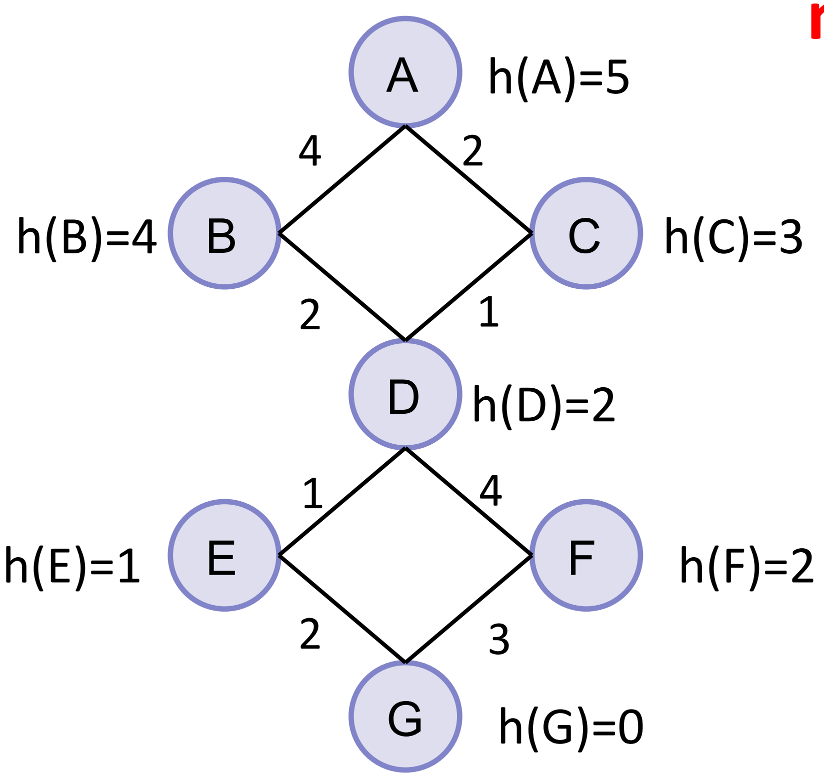
- explore nodes **in Best First manner**
- **do not stop** after finding the best solution

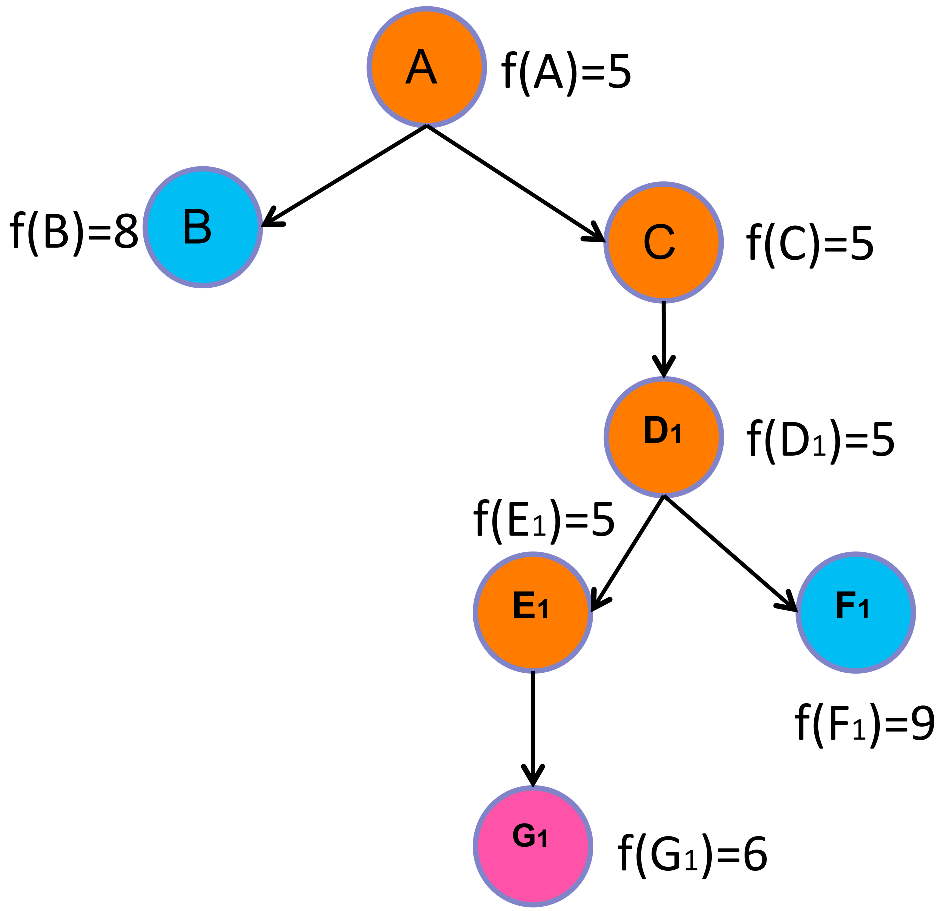
Some properties:

- explore **search tree** (not graph):
  - duplicate nodes, no caching
- bound number of duplicated nodes:
  - keep **no more than m copies** of each node
  - if h(n) **not consistent,** check if **new path is better**

m-A* example

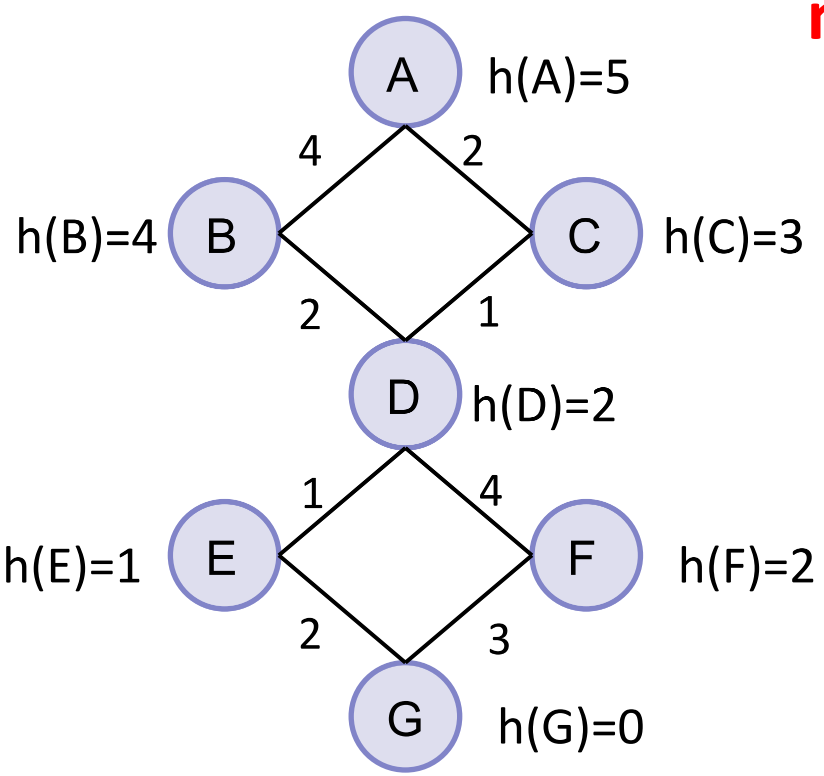
m=2

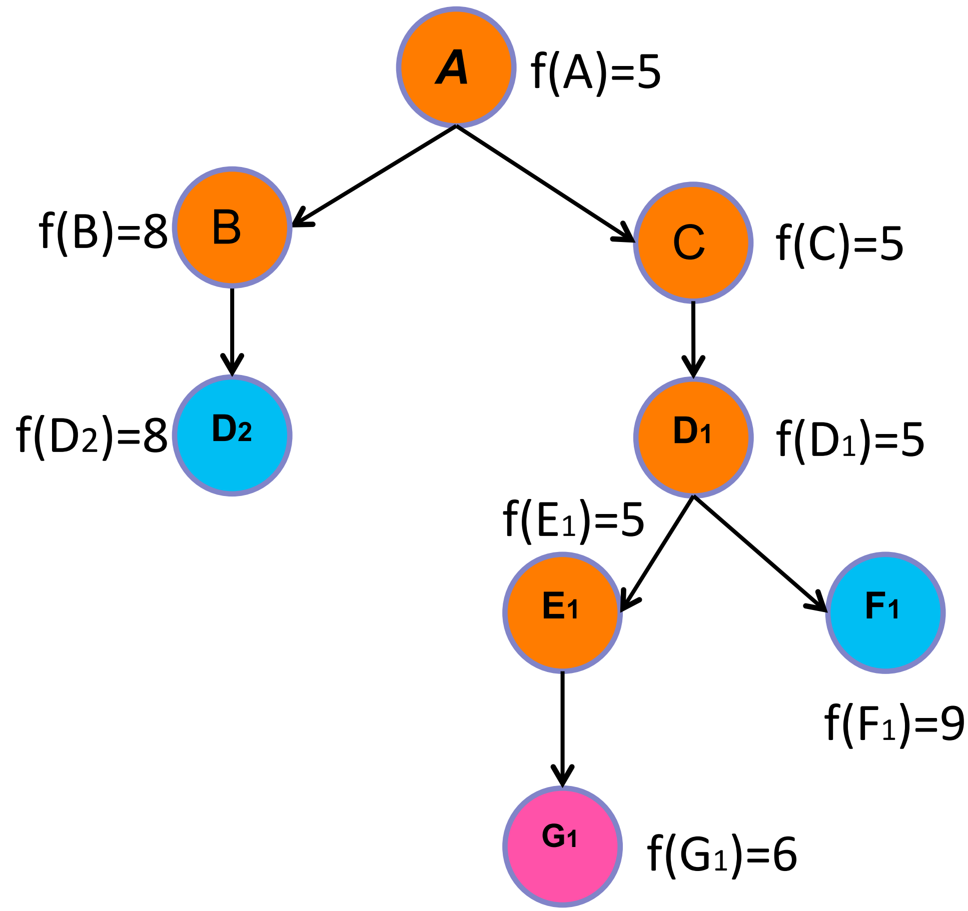




m-A* example

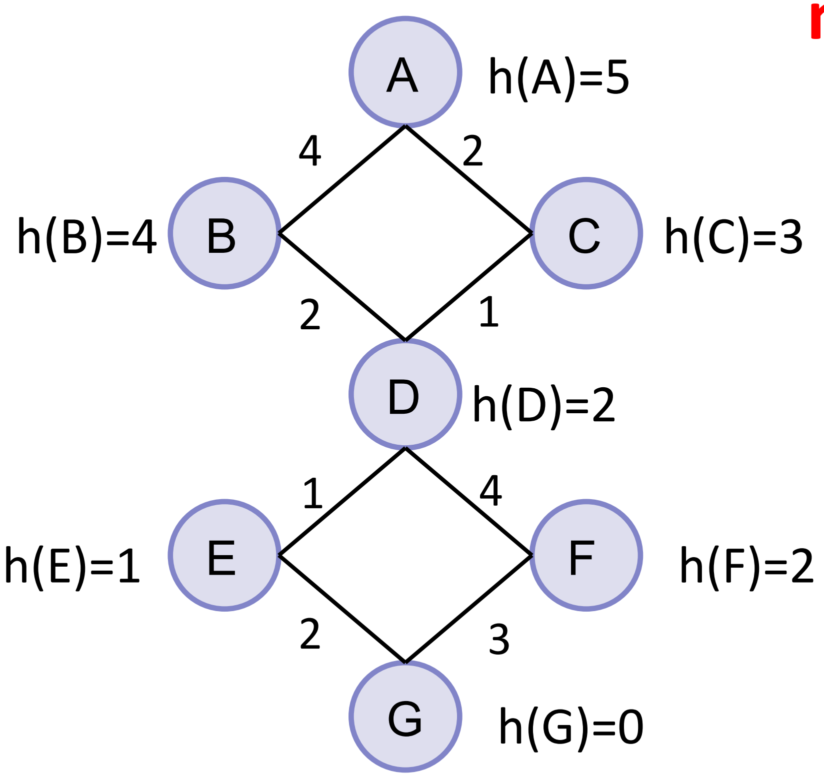
m=2

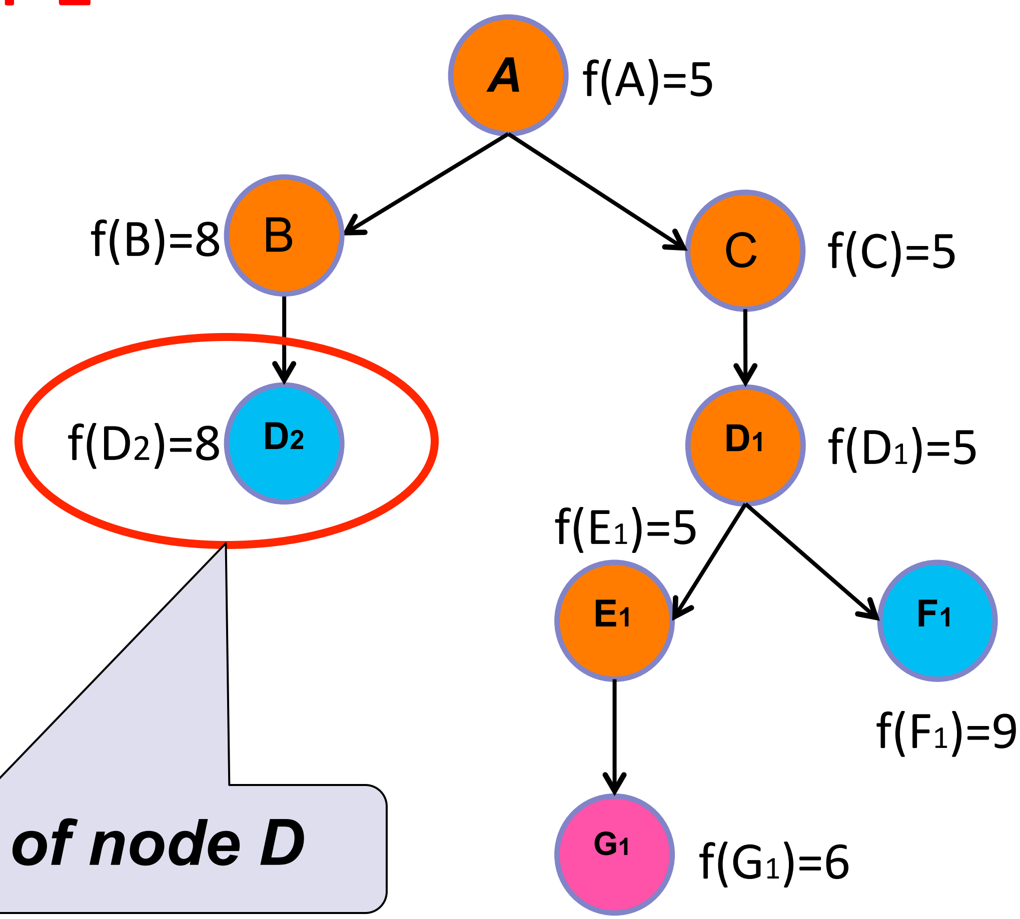




m-A* example

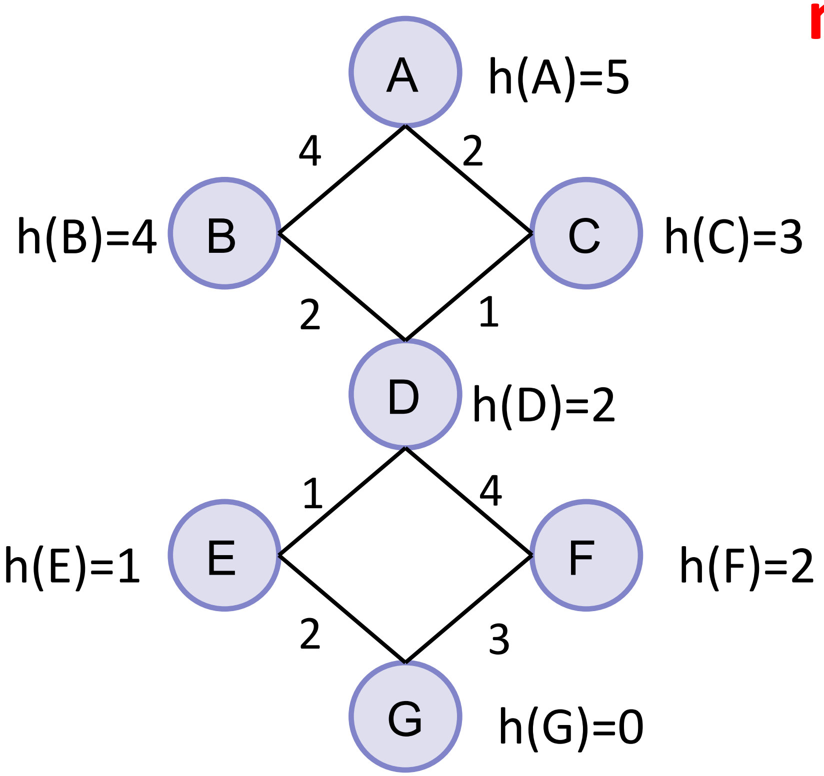
m=2

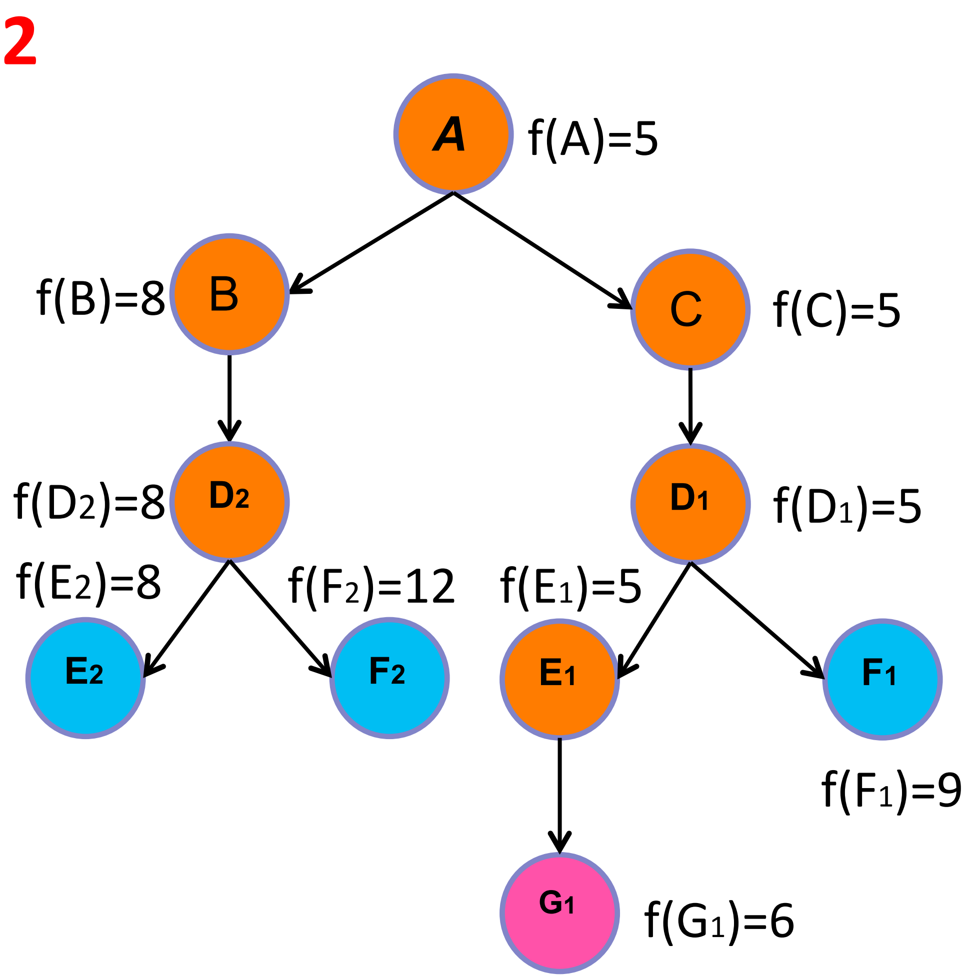




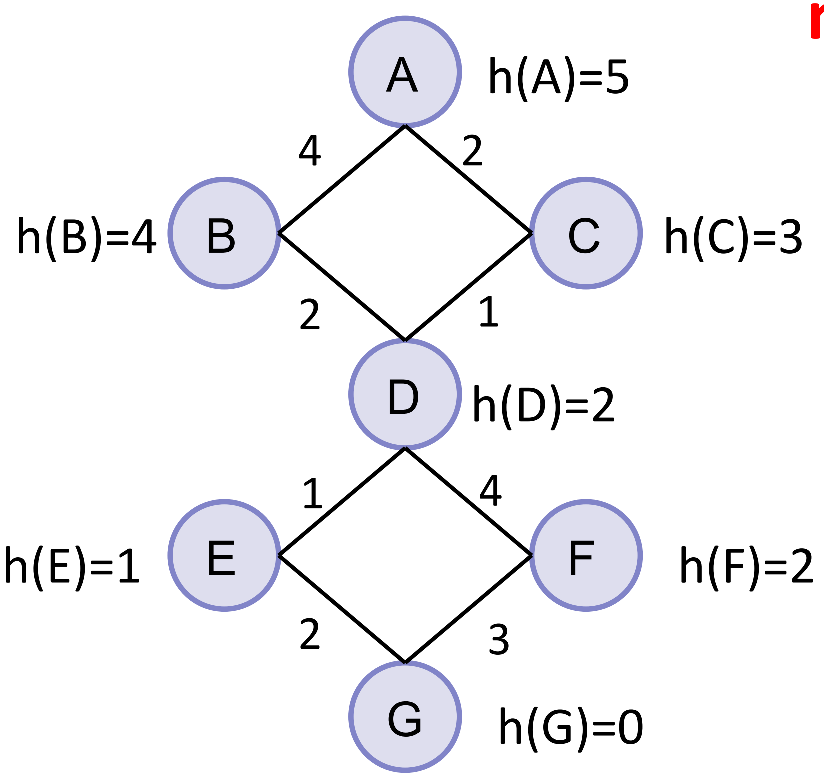
Make a new copy of node D

m-A* example

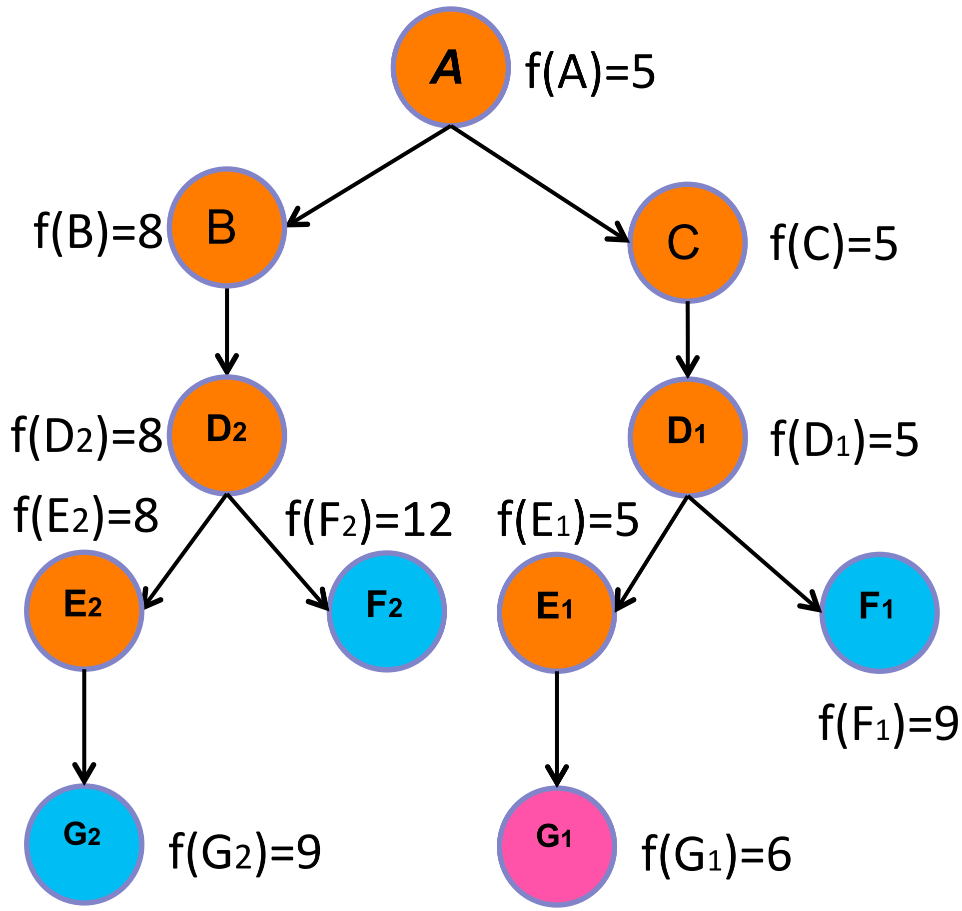

A
h(A)=5
4
2
h(B)=4
B
C
h(C)=3
2
1
D
h(D)=2
1
4
h(E)=1
E
F
h(F)=2
2
3
G
h(G)=0


m=2
A
f(A)=5
f(B)=8
B
C
f(C)=5
f(D2)=8
D2
D1
f(D1)=5
f(E2)=8
f(F2)=12
f(E1)=5
E2
F2
E1
F1
f(F1)=9
G1
f(G1)=6

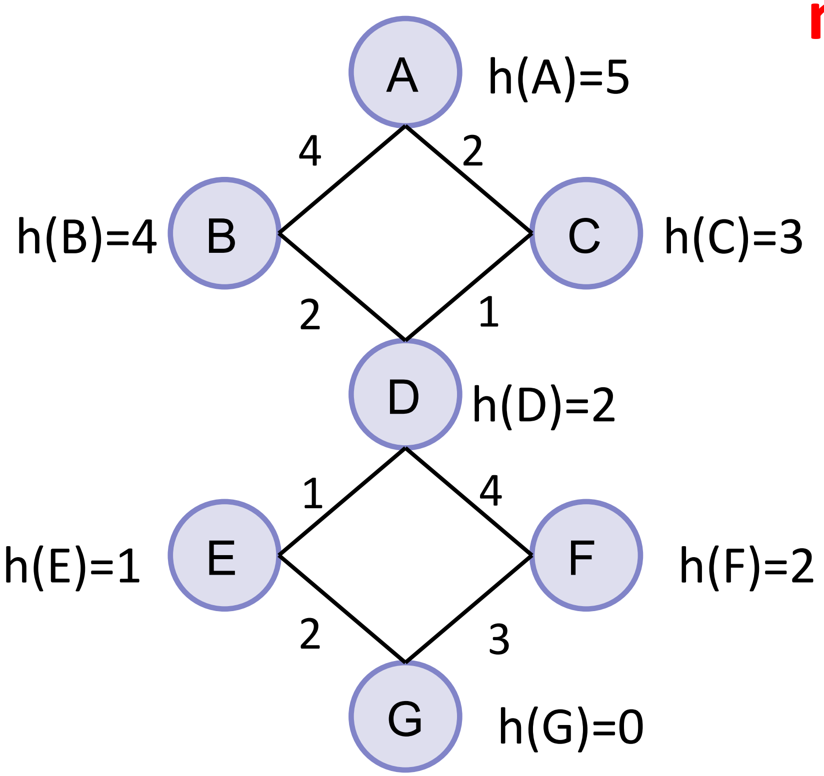
m-A* example



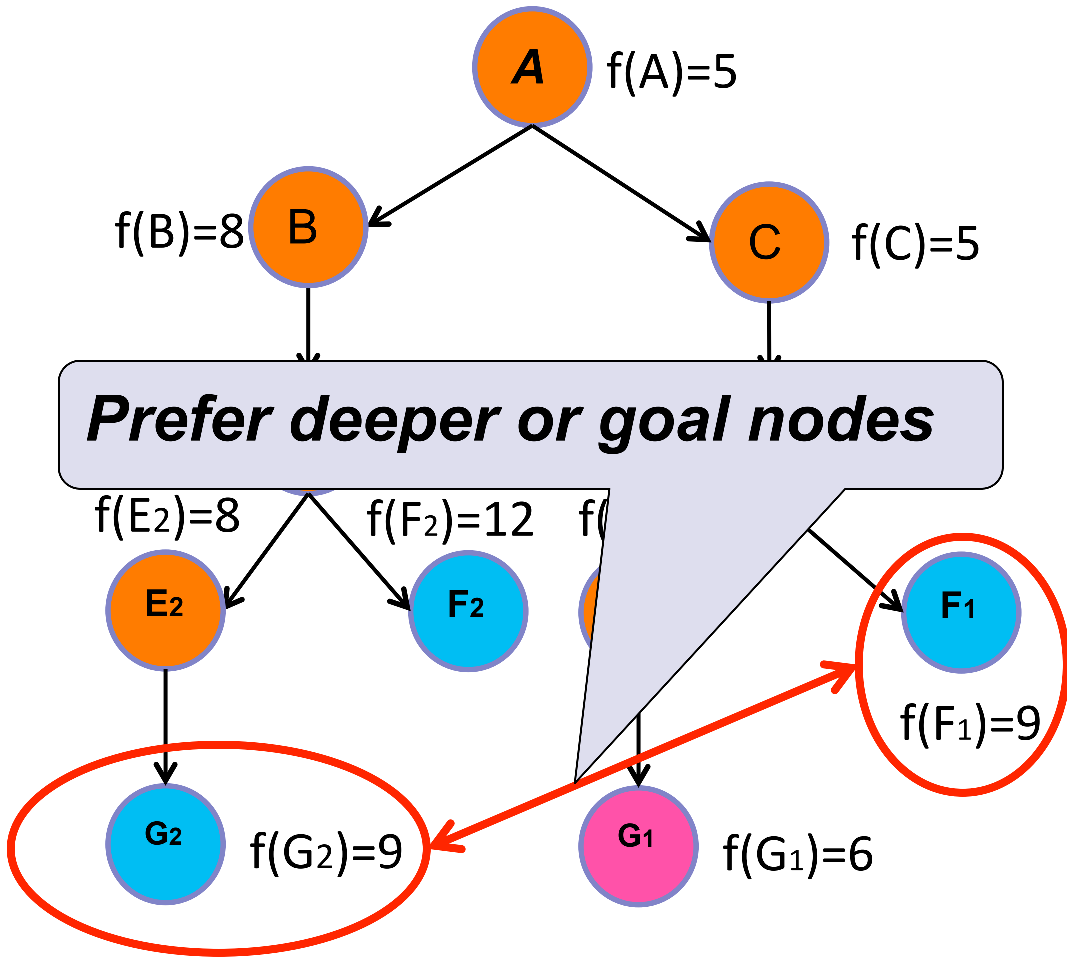
m=2



m-A* example

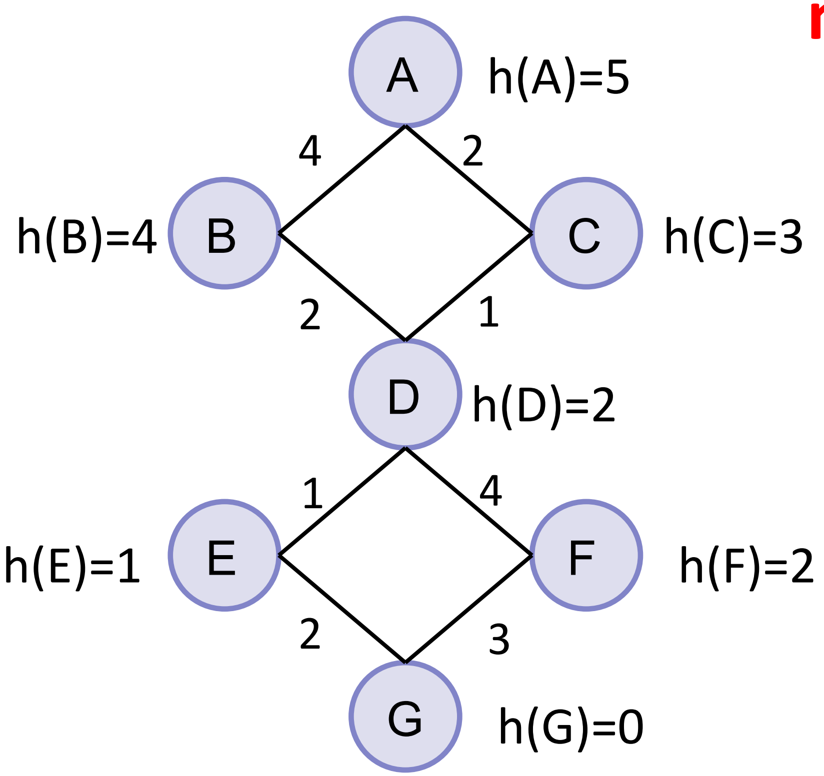


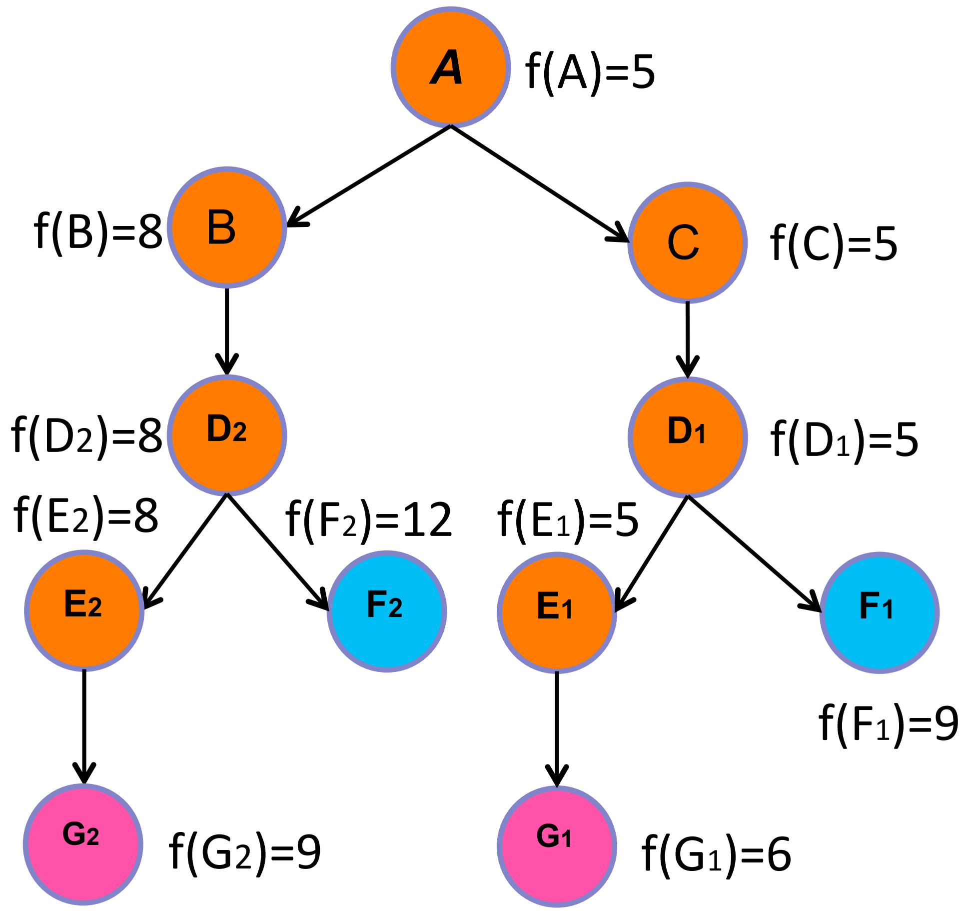
m=2



m-A* example

m=2





Properties of A* (inherited by m-A*)

- Soundness and completeness
- Optimal efficiency in term of nodes expanded
- Optimal efficiency for consistent heuristic
- Domination

Properties m-A*:

- **Soundness and completeness**
- **Optimal efficiency in term of nodes expanded:**

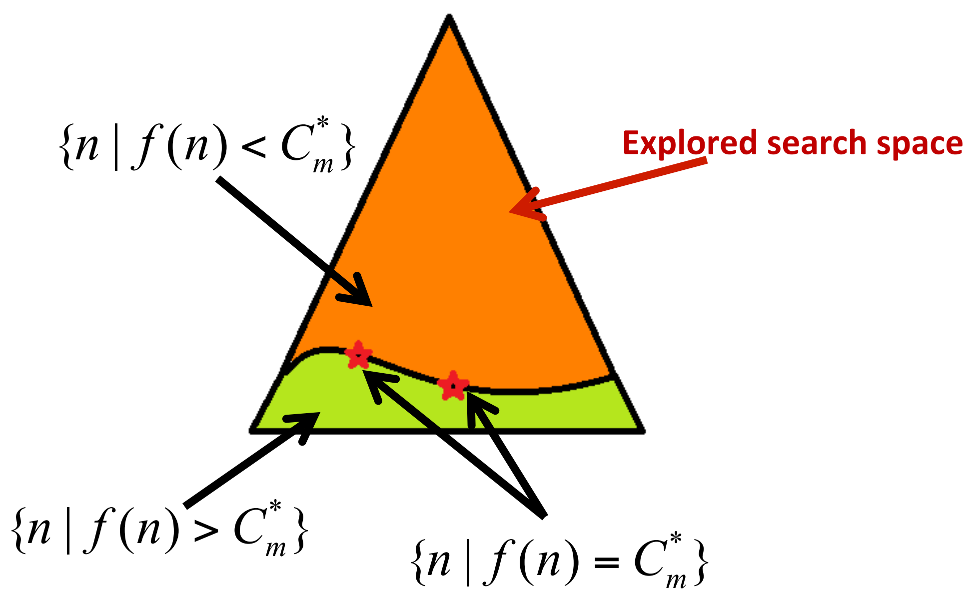
  every node *surely* expanded by m-A* *must* be expanded by any sound and complete search algorithm that uses the same heuristic information as m-A* and explores the same search graph.

- **Optimal efficiency for consisten ic**

We can show:
- any other search algorithm that doesn't expand some node n' *surely expanded by m-A*, can miss one of the m-best solutions when applied to a slightly modified problem and thus is incomplete*

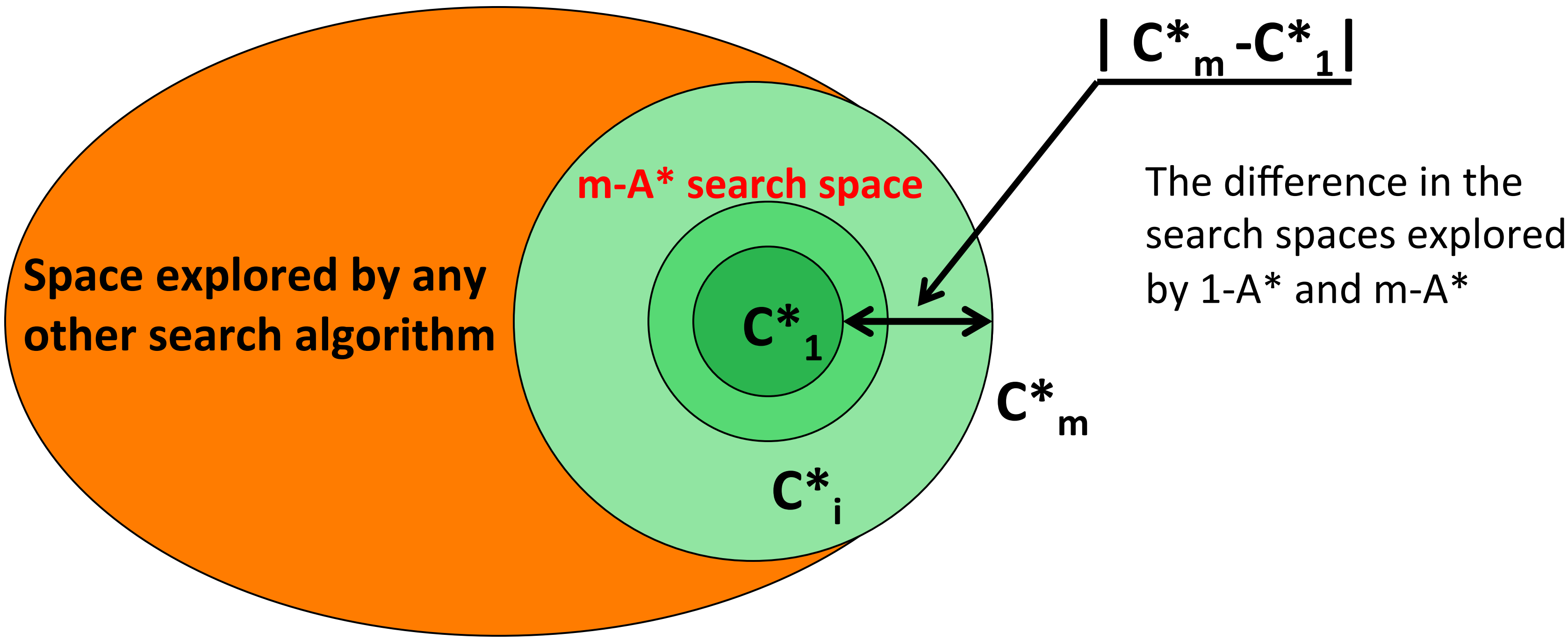
Properties m-A*

- m-A* with a consistent heuristic:
  - any node *n* will be expanded at most *m* times
  - the set $\{n \mid f(n) < C_m^*\}$ will surely be expanded
  - Some nodes with $\{n \mid f(n) = C_m^*\}$ are also expanded, depending on the tie breaking rule



Properties m-A*

- Impact of *m* on the search space size



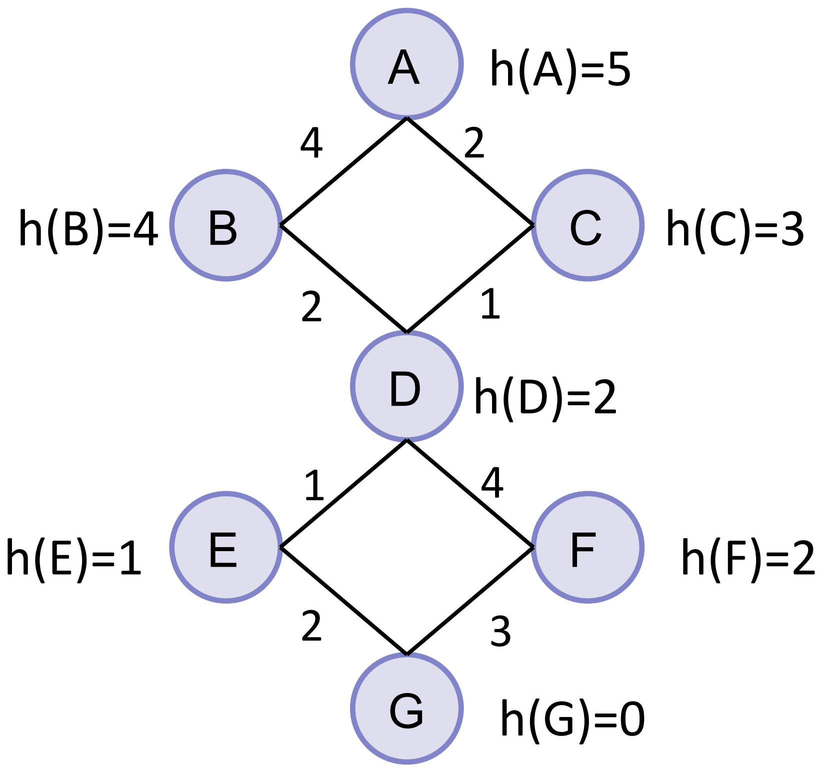
Outline:

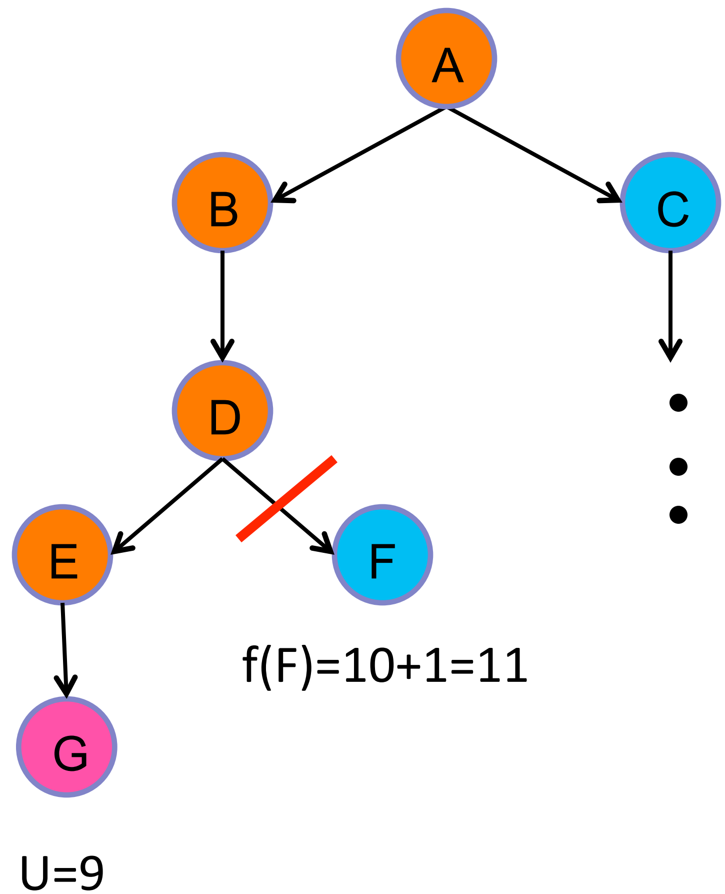
- Introduction
- Best First search for m-best
  - Background
  - m-A* algorithm
  - Properties of m-A*
- **Branch and Bound for m-best**
  - Background
  - m-BB algorithm
- Application to graphical models
- Empirical evaluation

Background: Depth First Branch and Bound

U=best cost so far

If ***f(n)<=U***, *n* is pruned



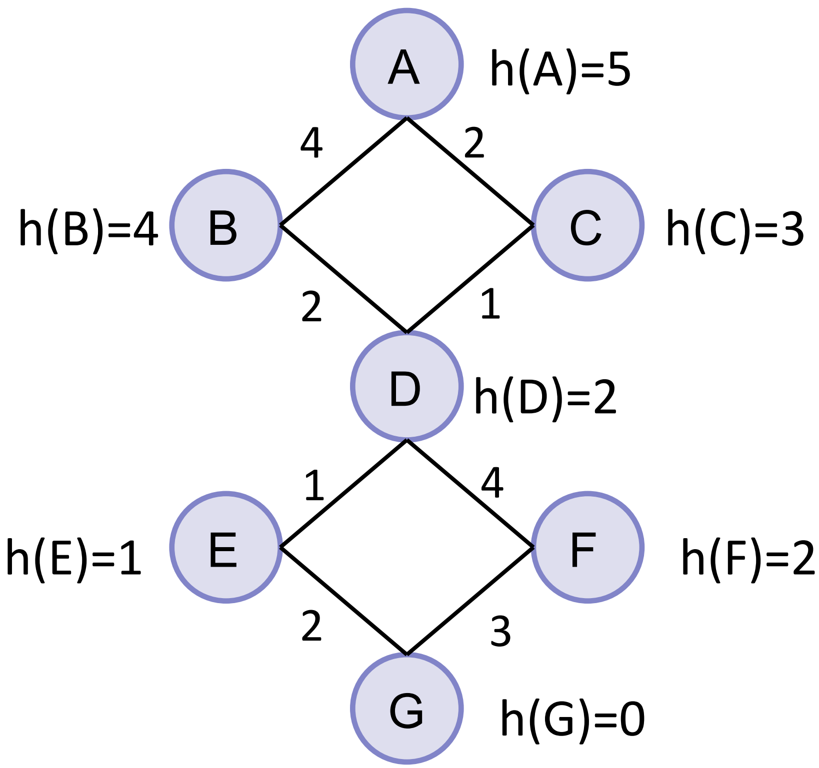


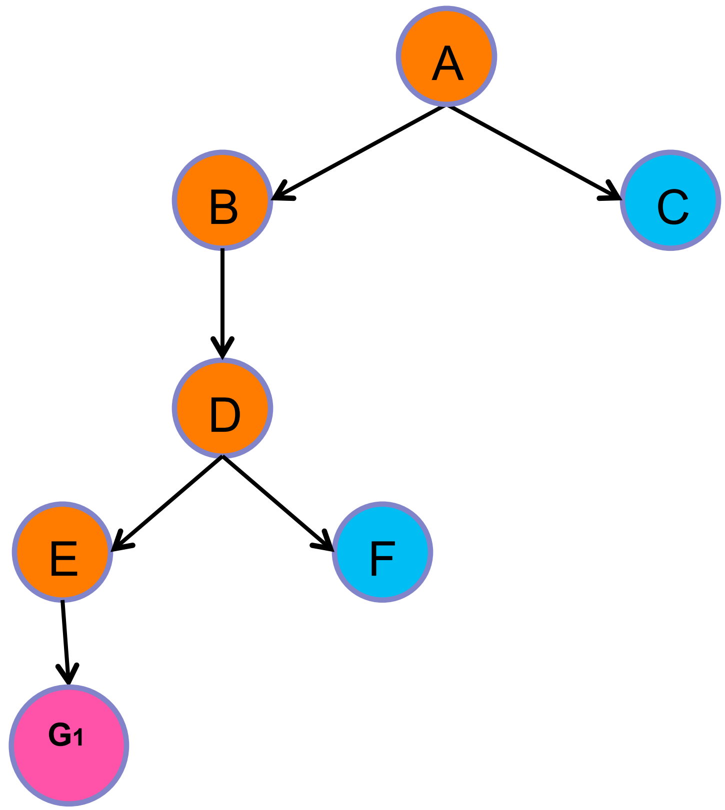
m-BB algorithm

U_m=m-best cost so far

$$U_1 = 9; U_2 = \infty$$

m=2



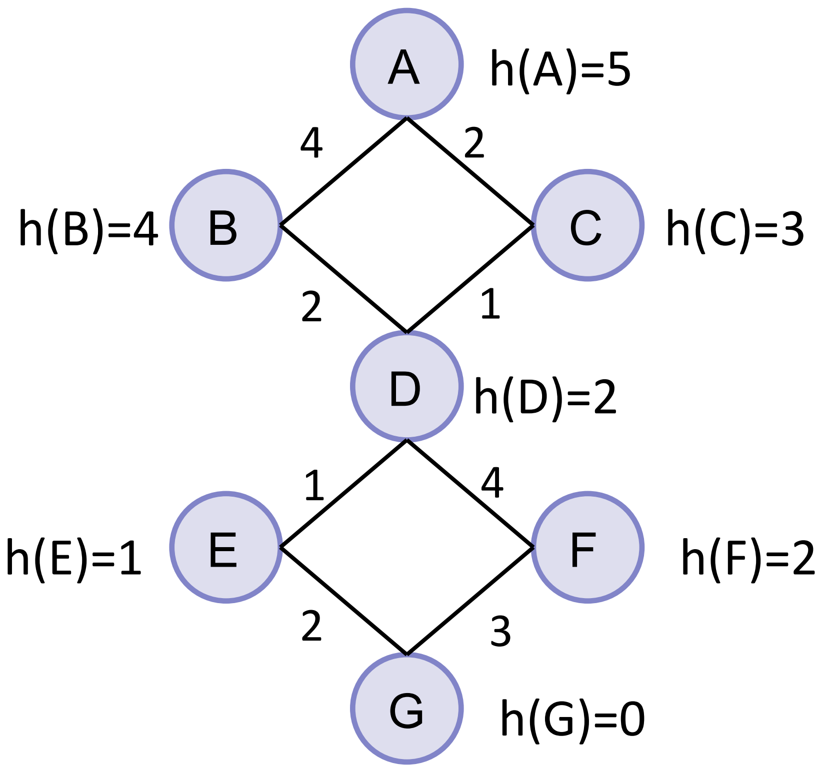


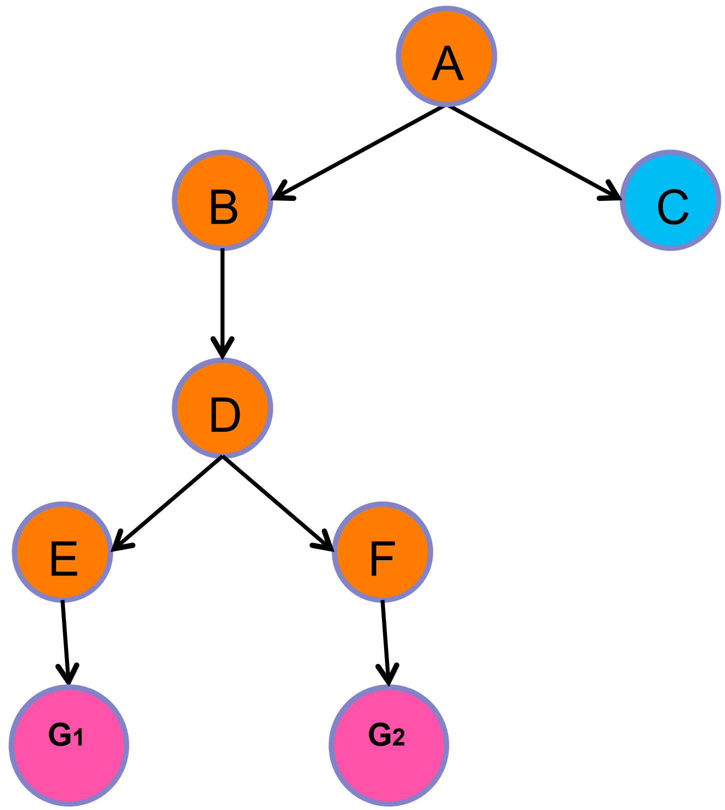
m-BB algorithm

U_m=m-best cost so far

$$U_1 = 9; U_2 = 13$$

m=2





Outline:

- Introduction
- Best First search for m-best
  - Background
  - m-A* algorithm
  - Properties of m-A*
- Branch and Bound for m-best
  - Background
  - m-BB algorithm
- **Application to graphical models**
- Empirical evaluation

AND/OR search spaces [Mateescu 2007]

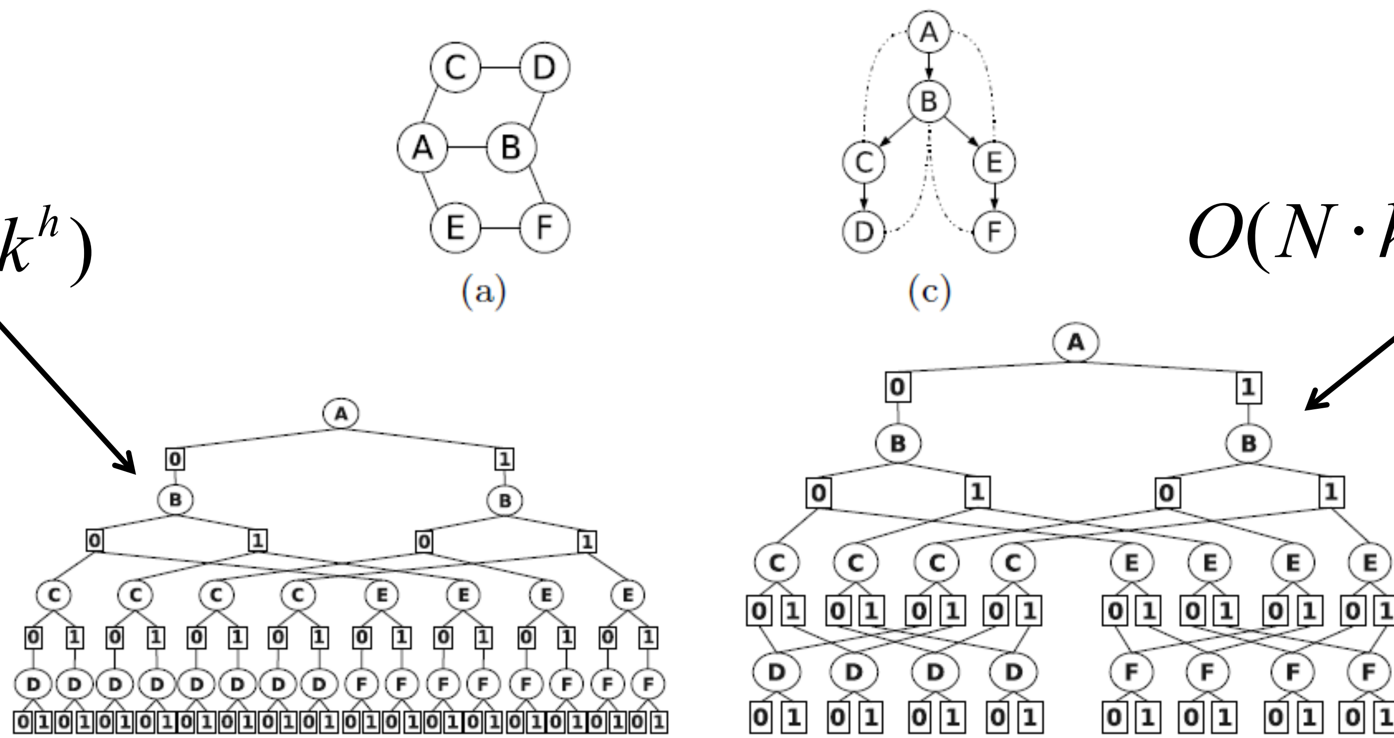
Graphical model: <**X**, **D**, **F**>

functions

variables ***domains***

$O(N \cdot k^h)$

$O(N \cdot k^{w^*+1})$



(a) (c)

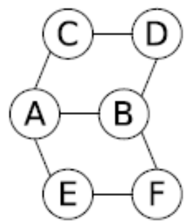
(d) (e)

AND/OR search spaces [Mateescu 2007]

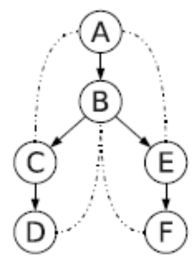
Graphical model: <**X**, **D**, **F**>

MPE task: $\max_X \prod_i F_i$

= finding max path in AND/OR search space

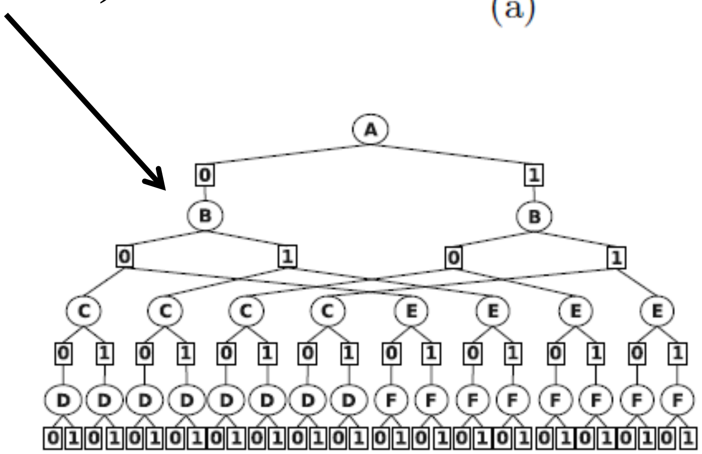


(a)



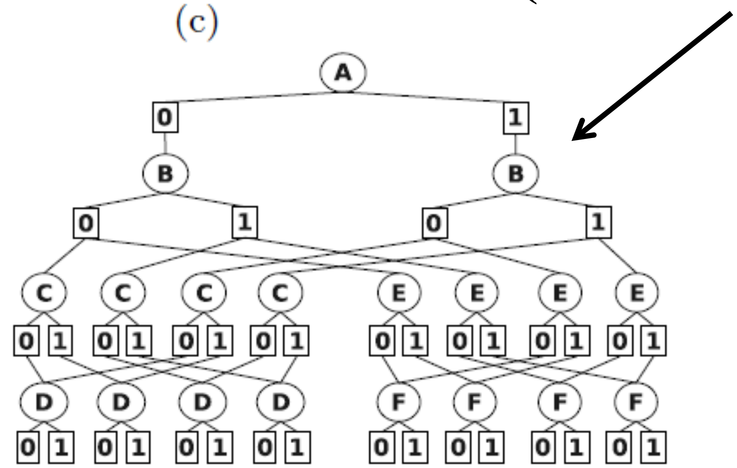
(c)

$O(N \cdot k^h)$



(d)

$O(N \cdot k^{w^*+1})$



(e)

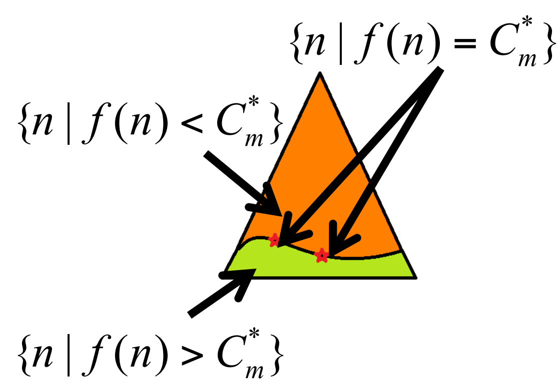
Worst time complexity for AND/OR search spaces

AND/OR tree	AND/OR graph
m-AOBF	
time, space: $O(N \cdot k^h)$	time, space: $O(N \cdot m \cdot k^{w^*})$
m-AOBB	
time: $O(m \cdot N \cdot k^h \cdot \deg \cdot \log m)$ space: $O(N \cdot m)$	time, space: $O(m \cdot N \cdot k^{w^*+1} \cdot \deg \cdot \log m)$

Worst time complexity for AND/OR search spaces

AND/OR tree	AND/OR graph
m-AOBF	
time, space: $O(N \cdot k^h)$	time, space: $O(N \cdot m \cdot k^{w^*})$
m-AOBB	
time: $O(m \cdot N \cdot k^h \cdot \deg \cdot \log m)$ space: $O(N \cdot m)$	time, space: $O(m \cdot N \cdot k^{w^*+1} \cdot \deg \cdot \log m)$

The true size of the explored search space depends on the cost of the m[th] best solution C^*_m



Heuristics for AND/OR search

- Previously used for AND/OR search:
  mini-bucket heuristics.
  [Kask 1999, Marinescu 2009]
- Parameter **i-bound** flexibly controls **accuracy**
- Extreme case:
  Bucket Elimination produces **exact heuristic**
  [Dechter 1999]

Algorithm **BE-Greedy-m-BF** (aka **BE+m-BF**)

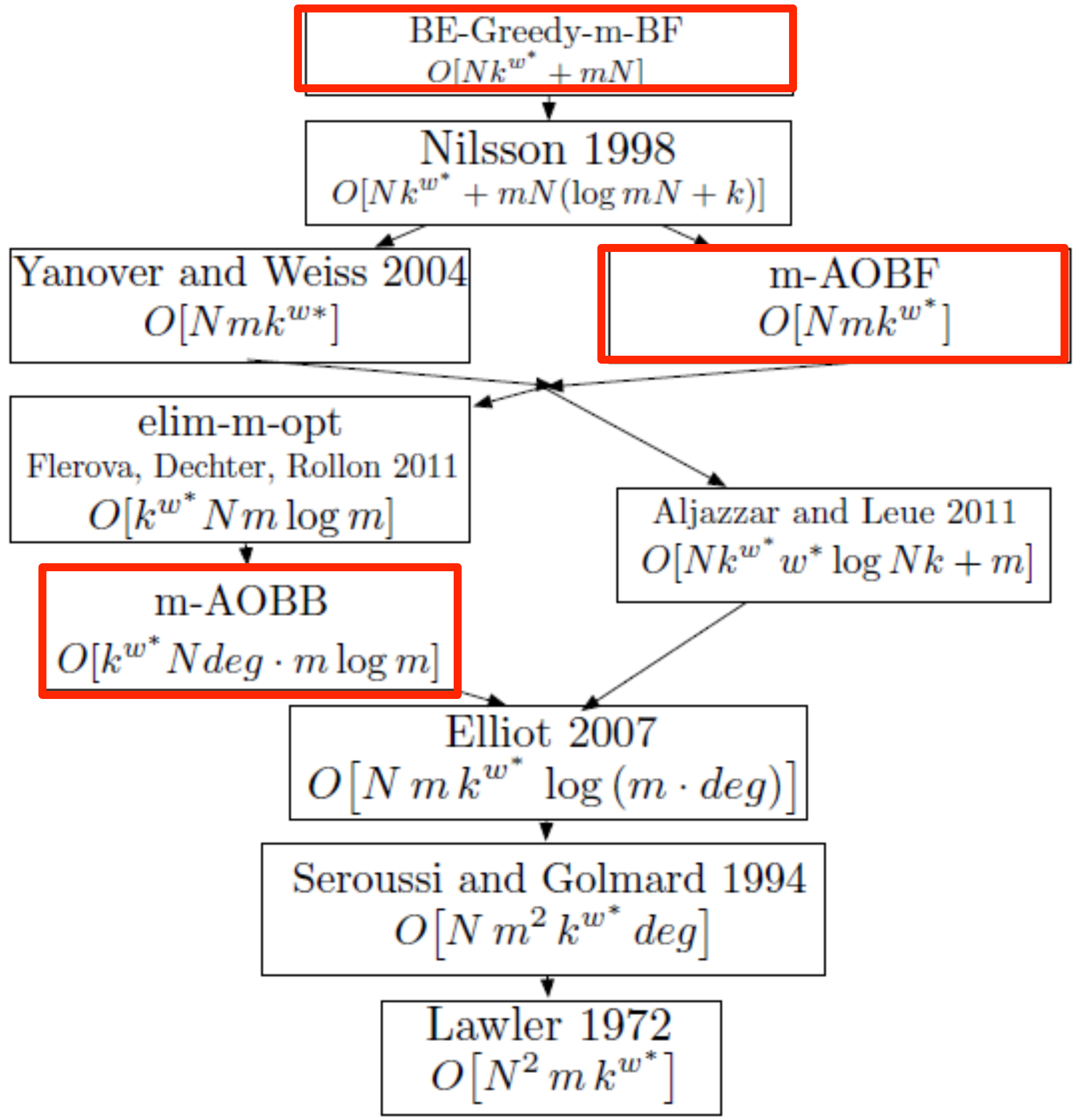
Main idea:

- Generate **exact heuristic** by **Bucket Elimination**
- Run **m-AOBF** with exact heuristics

Properties:

- Only **nodes on the *m* best paths** are expanded
- Time and space complexity: $O(N \cdot k^{w^*} + N \cdot m)$

Worst case time comparison with previous work



Outline:

- Introduction
- Best First search for m-best
  - Background
  - m-A* algorithm
  - Properties of m-A*
- Branch and Bound for m-best
  - Background
  - m-BB algorithm
- Application to graphical models
- **Empirical evaluation**

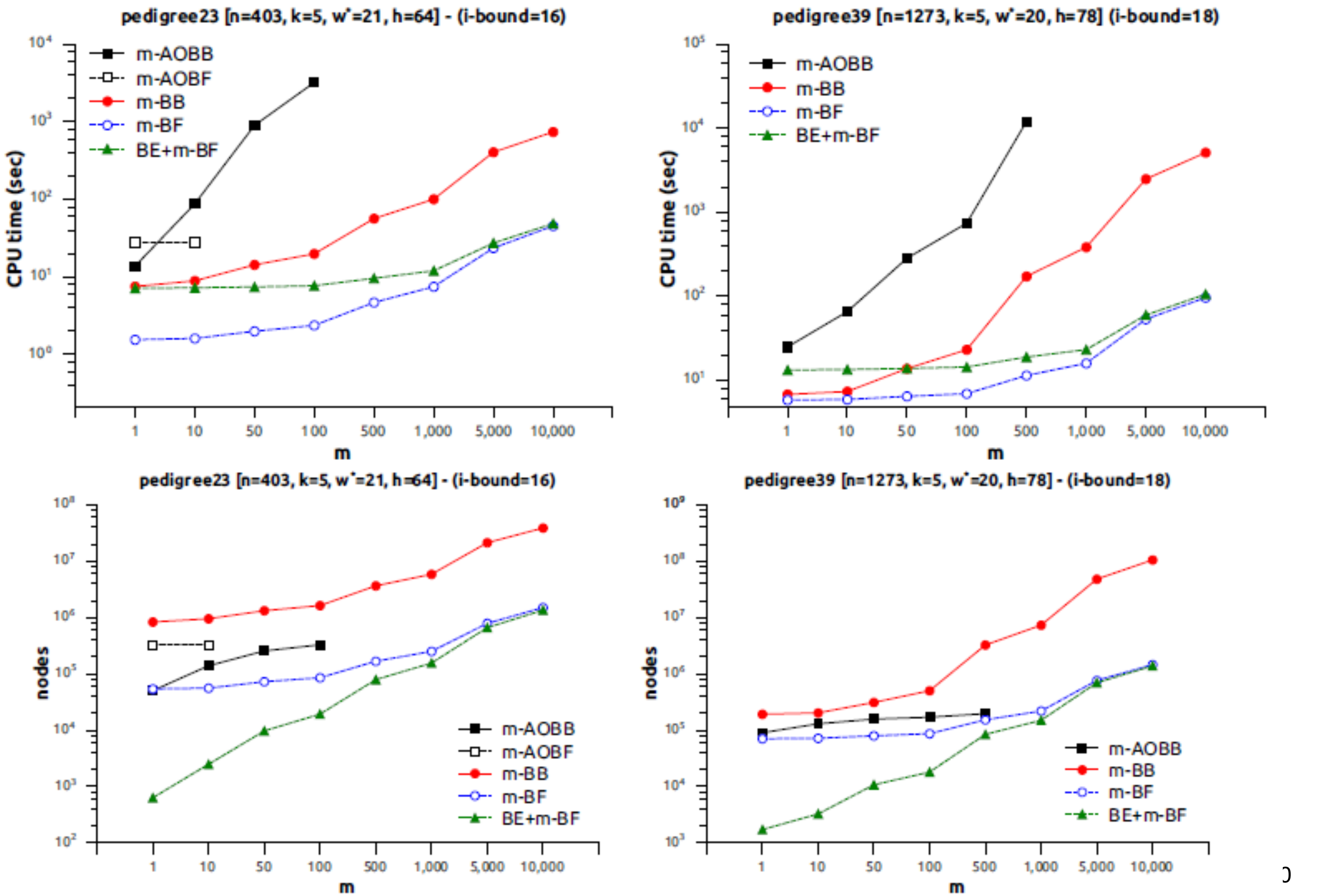
Empirical evaluation

Benchmarks:
- Pedigrees
- n-by-n grids
- ISCAS'89 digital circuits

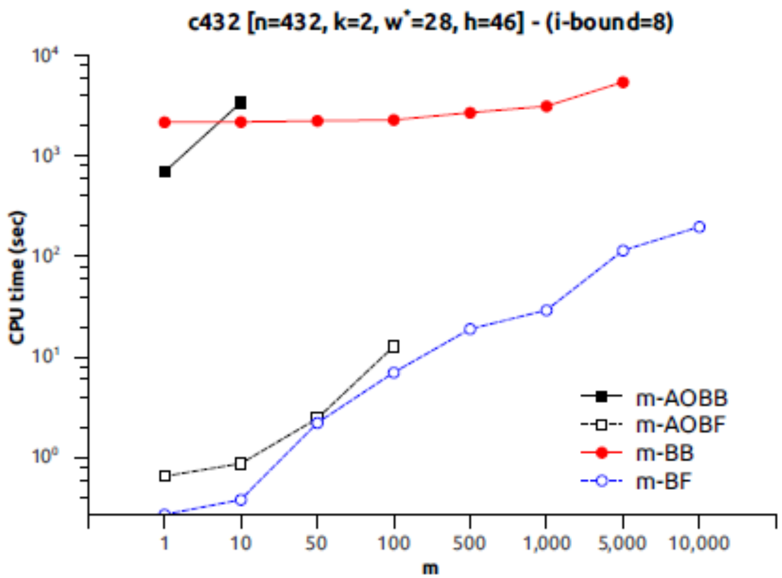
Algorithms:
- m-A*
- m-BB
- m-AOBF
- m-AOBB
- BE+m-BF

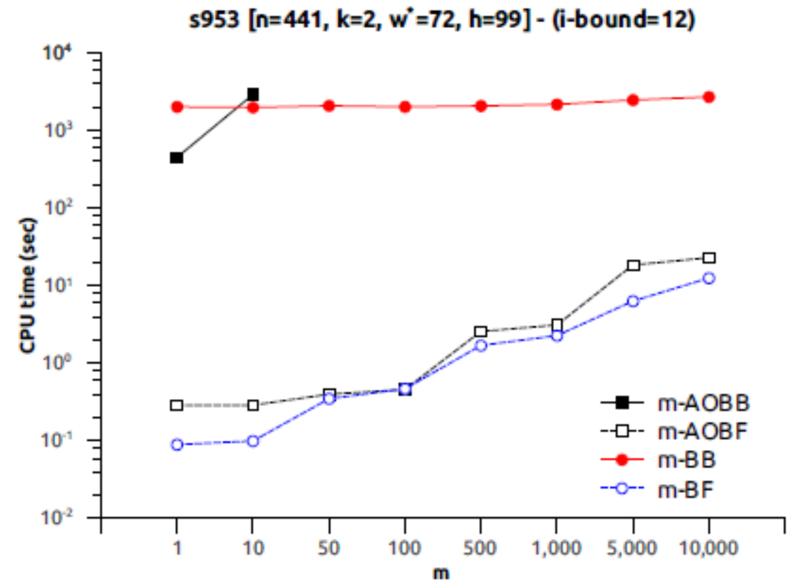
Heuristic: mini-bucket
Memory limit: 4 Gb
Time limit: 12h for pedigrees,
2h for grids and ISCAS

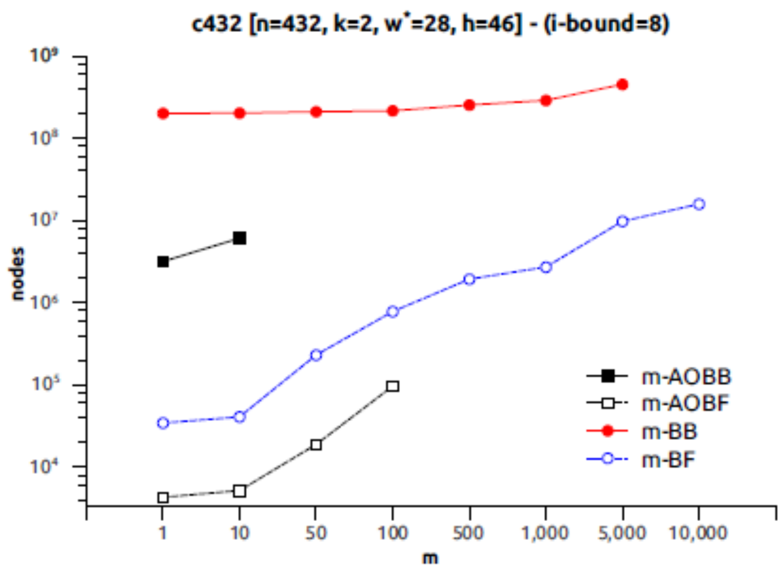
Pedigrees: runtime, number of nodes vs m

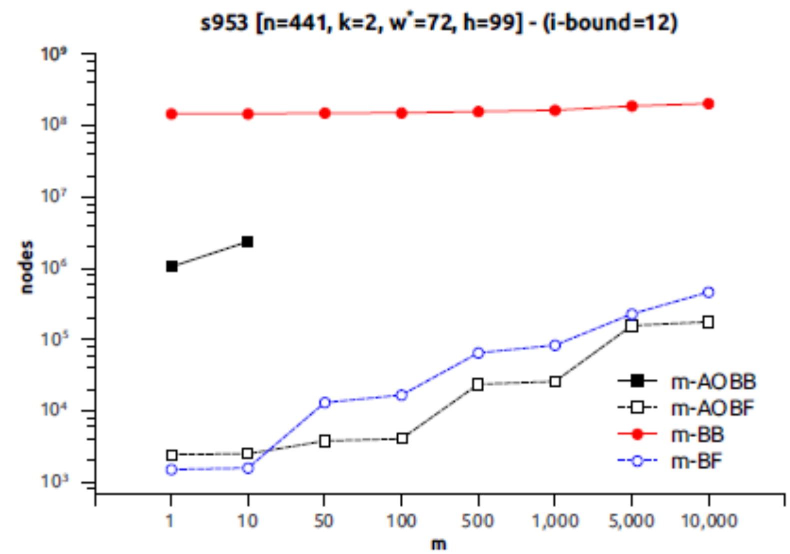


ISCAS instances : runtime, number of nodes vs m



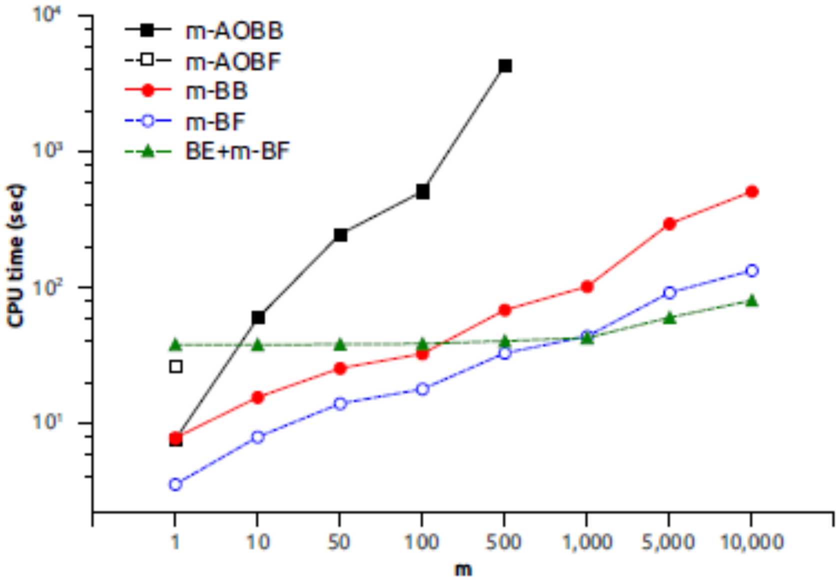




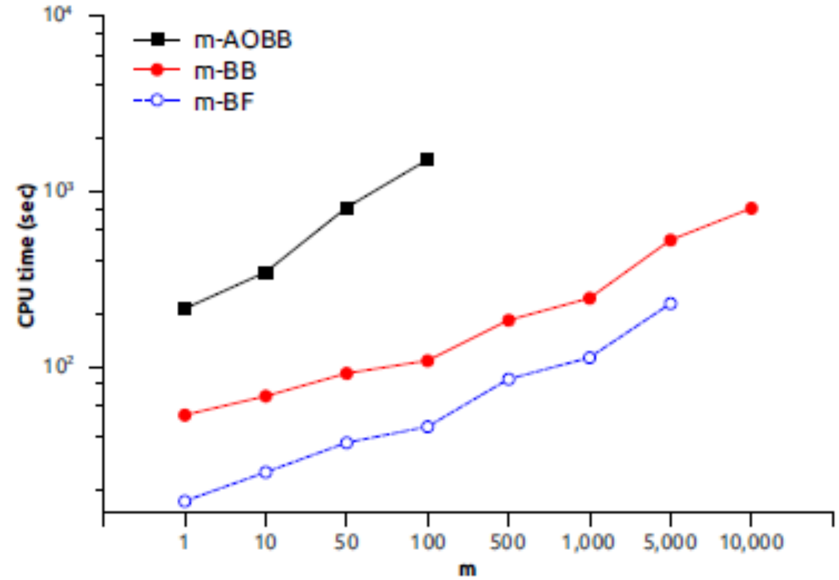


Grids: runtime, number of nodes vs m

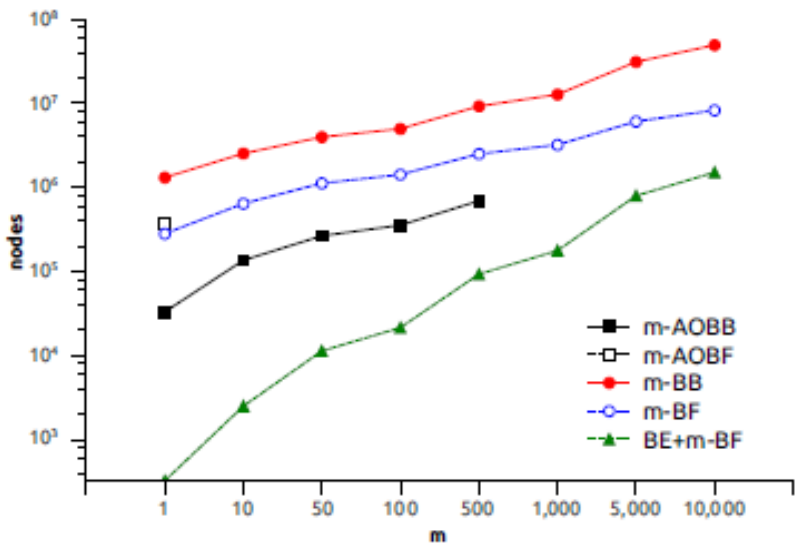
g75-18-5 [n=324, k=2, w*=24, h=85] - (i-bound=16)



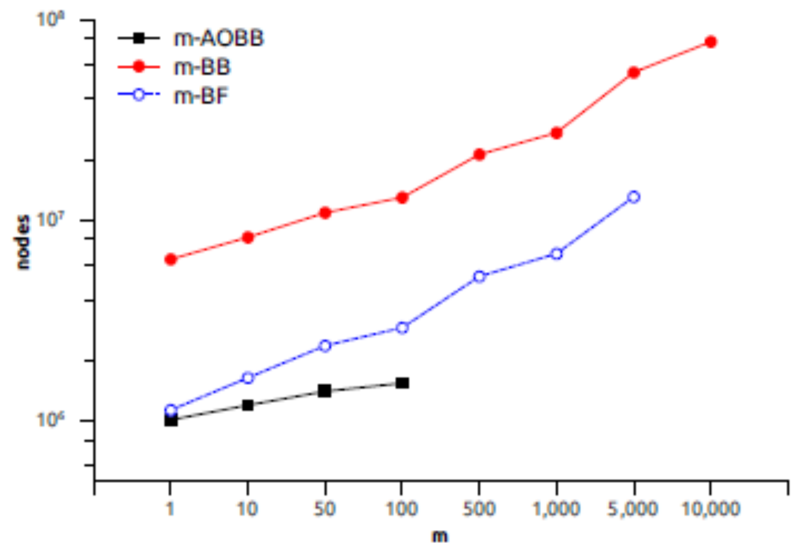
g90-20-5 [n=400, k=2, w*=27, h=99] - (i-bound=16)



g75-18-5 [n=324, k=2, w*=24, h=85] - (i-bound=16)

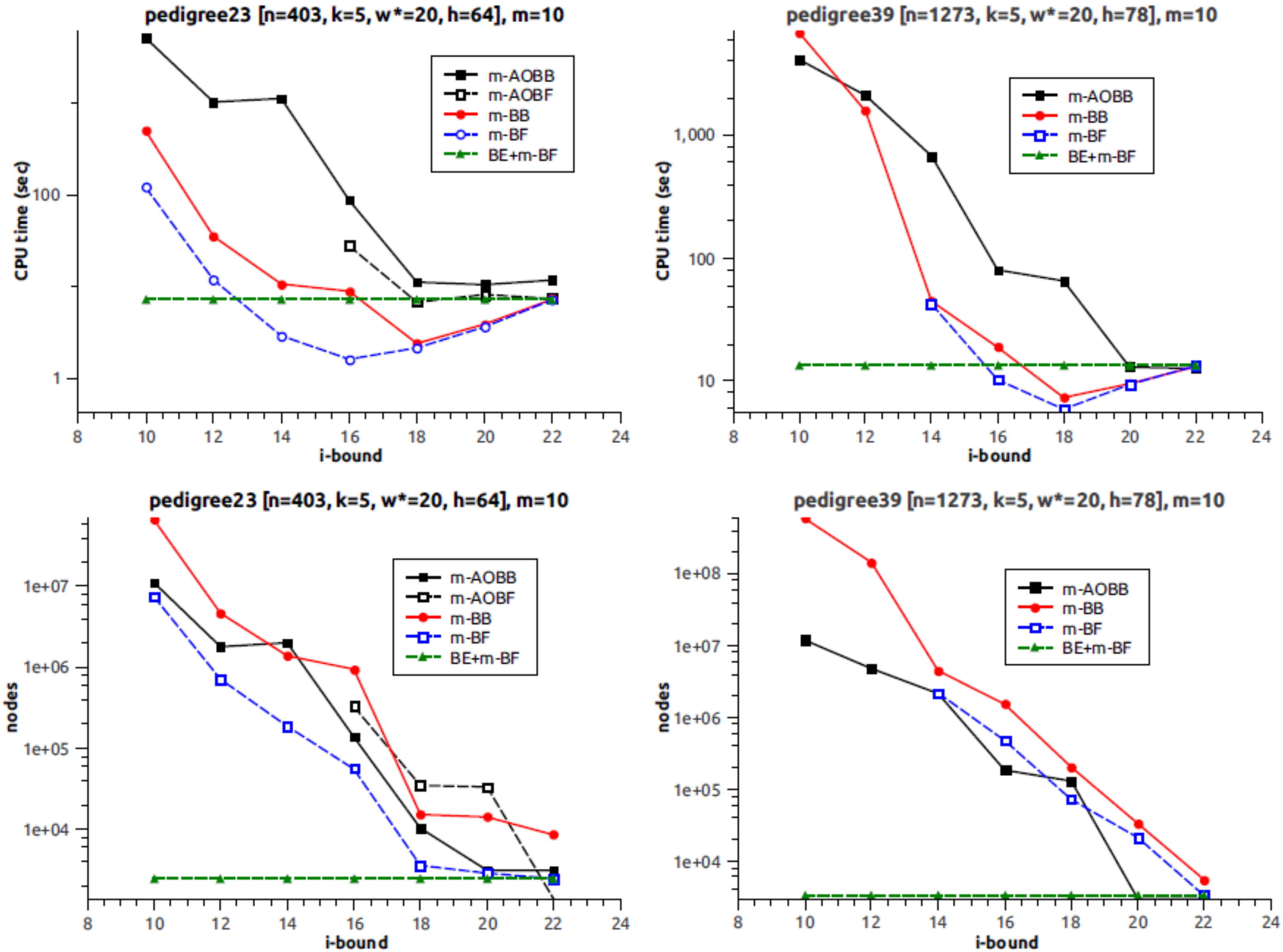


g90-20-5 [n=400, k=2, w*=27, h=99] - (i-bound=16)



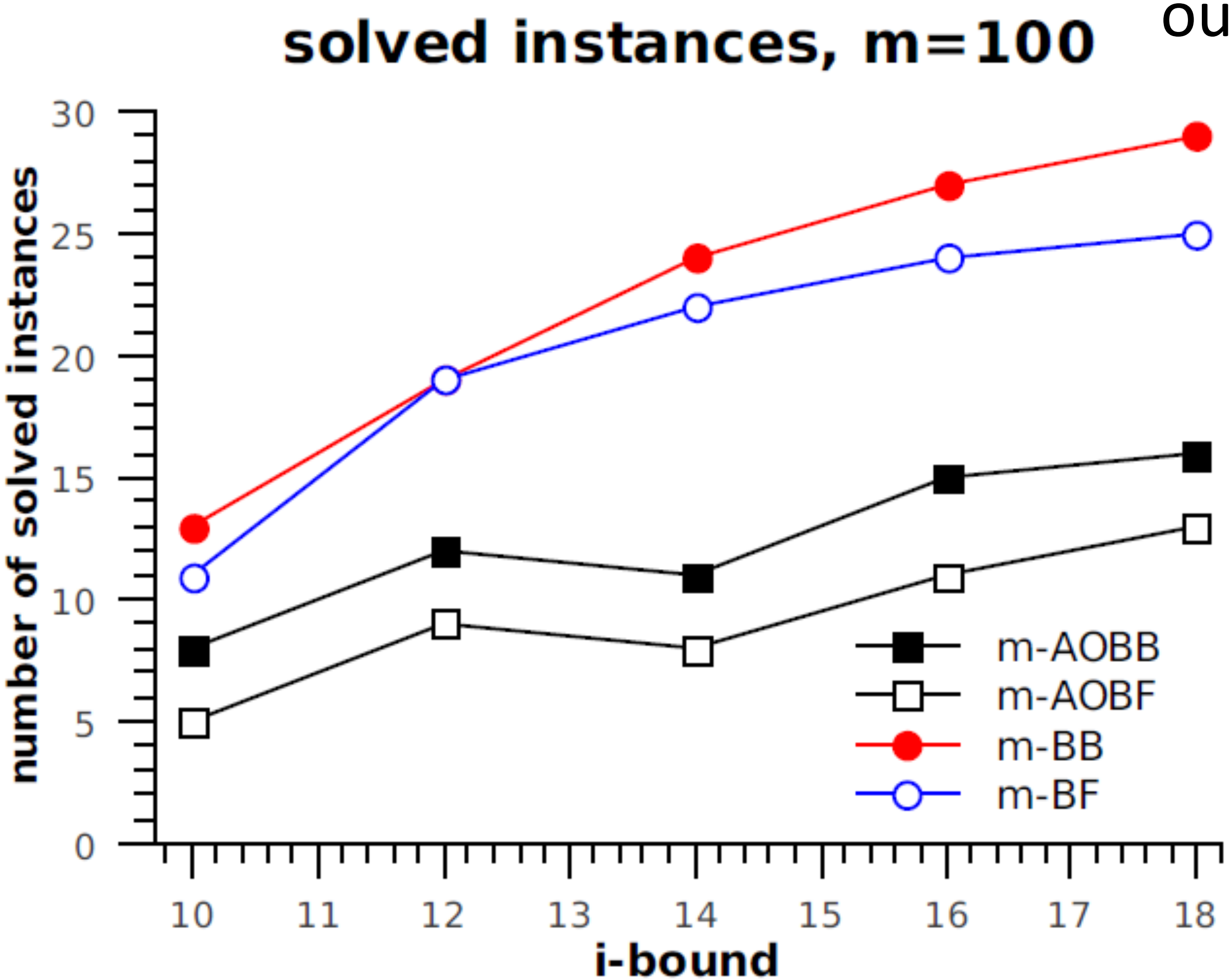
BREN:ICS INFORMATION AND COMPUTER SCIENCES — UNIVERSITY of CALIFORNIA IRVINE

The impact of heuristic strength



Number of instances solved

out of 36



Conclusion:

In this work we:

- **extend** Best First and Branch and Bound **search** to **m-best** solutions
- **explore properties** of new schemes
- apply new schemes to **optimization problems** over **graphical models**

Future work:

- extensions to **graph** search spaces