HEURISTIC SEARCH FOR M BEST SOLUTIONS WITH APPLICATIONS TO GRAPHICAL MODELS

Rina Dechter and Natalia Flerova

Summary:

- Optimization problem:
  - Finding the m best solutions
- Previous works on the m-best tasks:
  - Compute the m-best solutions by successively computing the best solution, each time using a slightly different reformulation of the original problem.
    - Lawler, 1972; Nilsson, 1998; Yanover and Weiss, 2004
  - Compute the m best solutions in a single pass of algorithm, using message passing/propagation
    - Seroussi and Golmard, 1994 ; Elliot, 2007; Rollon, Flerova, Dechter, 2011

Summary:

- Our contribution:
  - We develop search algorithms to compute the m-best task avoiding the repeated computation inherent in Lawler's scheme and reducing explored state space compared to message-passing schemes
  - In particular, we extend:
    - Best First search (specifically, A*)
    - Depth First Branch and Bound search
  - We focus on Graphical Models and apply our algorithm to the AND/OR search spaces, yielding for the m-best task:
    - AND/OR Best First search
    - AND/OR Branch and Bound

1-best vs. m-best optimization

Variables : $X = \{X_1, X_2, X_3, \ldots, X_n\}$
Finite domain values: $D = \{D_1, D_2, \ldots, D_n\}$

Objective function:

A is a totally ordered set (<)

- **1-best optimization**

$$F(t) \quad \text{such that}$$

$$\forall t' \notin t. \quad F(t) \geq F(t')$$

- **m-best optimization**

$$\{F(t_1), \ldots, F(t_m)\} \quad \text{such that}$$

$$\forall t' \notin t_j. \quad 1 \leq \mathrm{j} \leq \mathrm{m}. \quad F(t_j) \geq F(t')$$

Example: Finding K shortest paths in a graph

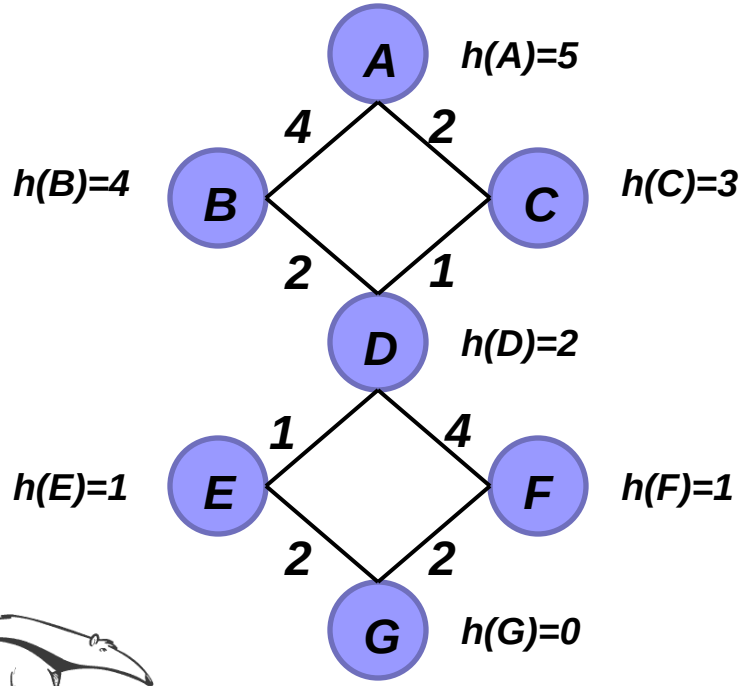
Background: Best First search

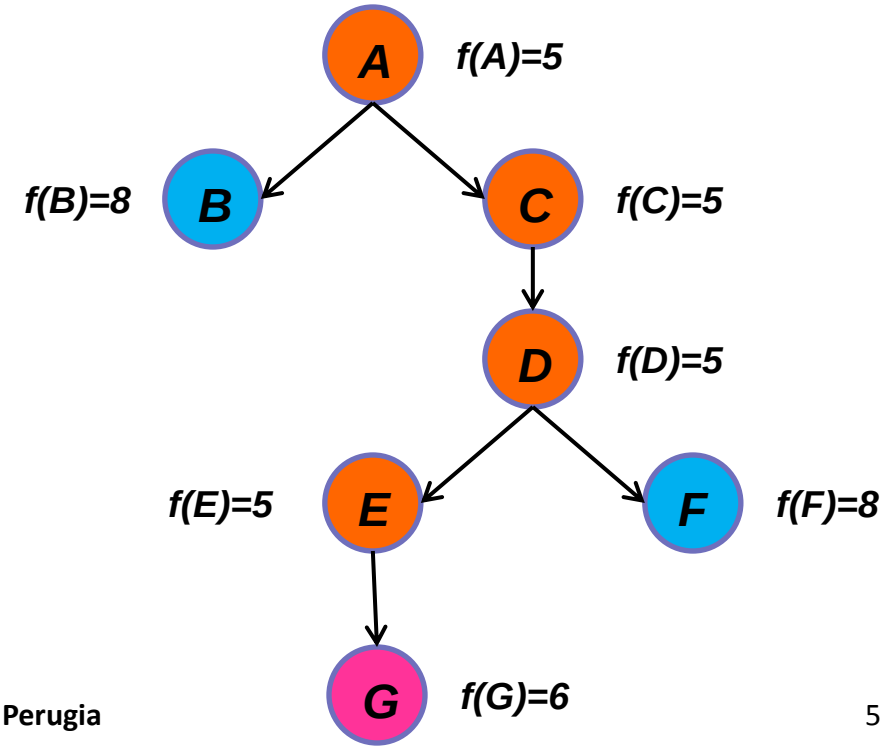
- **Objective function**: *f(n)* *(under-)*estimates the best cost solution path passing through *n*.

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

cost of the best path from start to n

heuristic evaluation function, underestimating *h*(n)*, optimal cost from n to goal

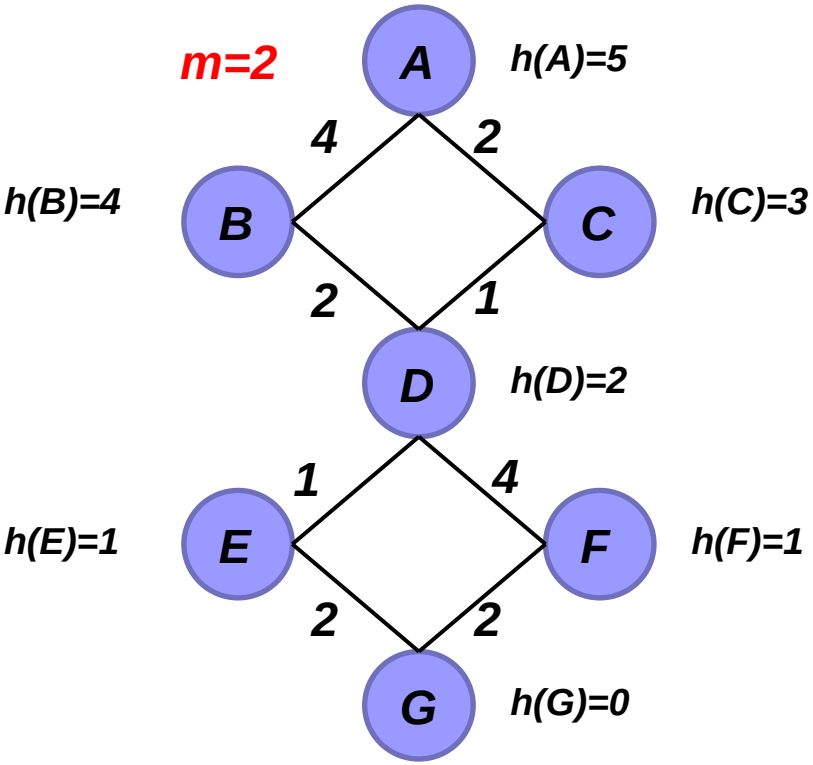


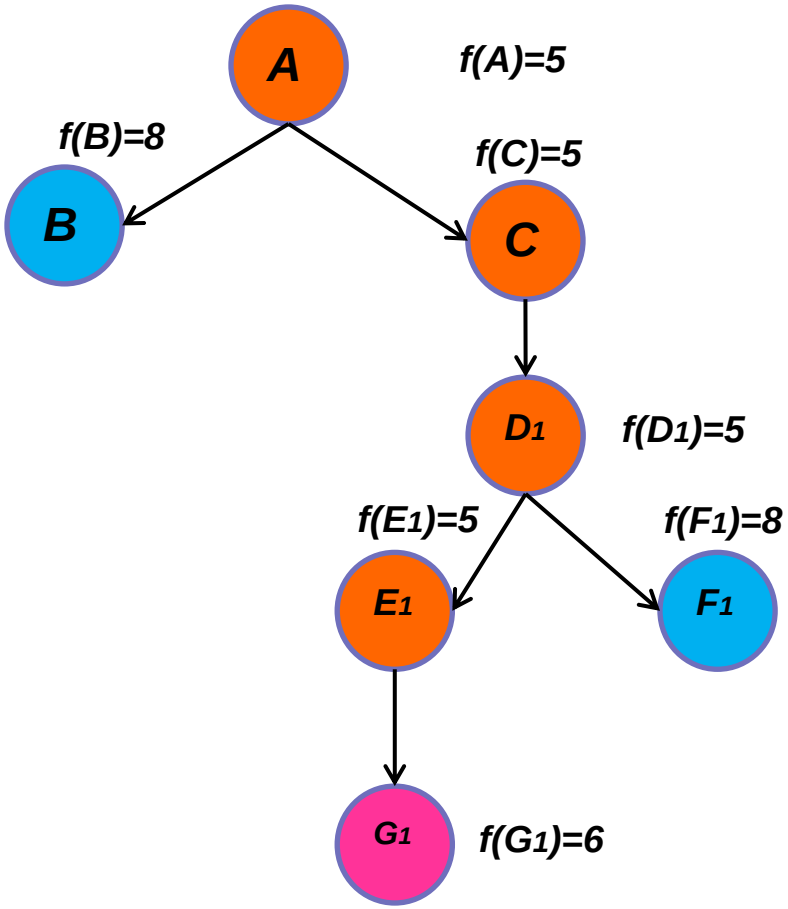


Algorithm m-A*

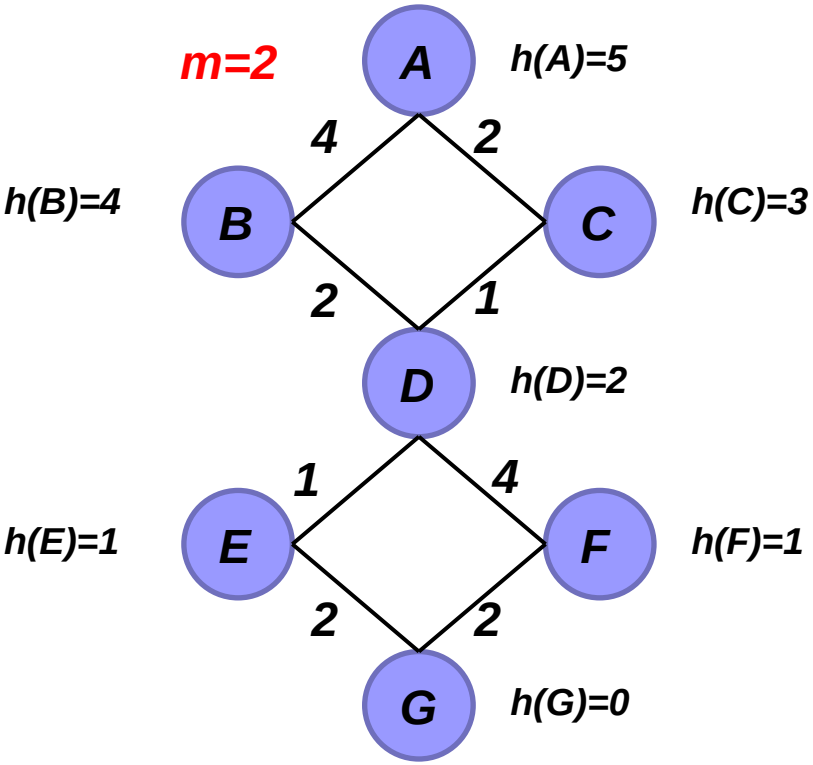
- Explore nodes in Best First manner
- Continue search after finding the 1st solution
- If the node is discovered more than once – keep at most *m* copies both in OPEN and CLOSED
- If heuristic is not consistent – new path to the node might be better than the old one. If we already know *m* paths to node *n*, discard the worse one.

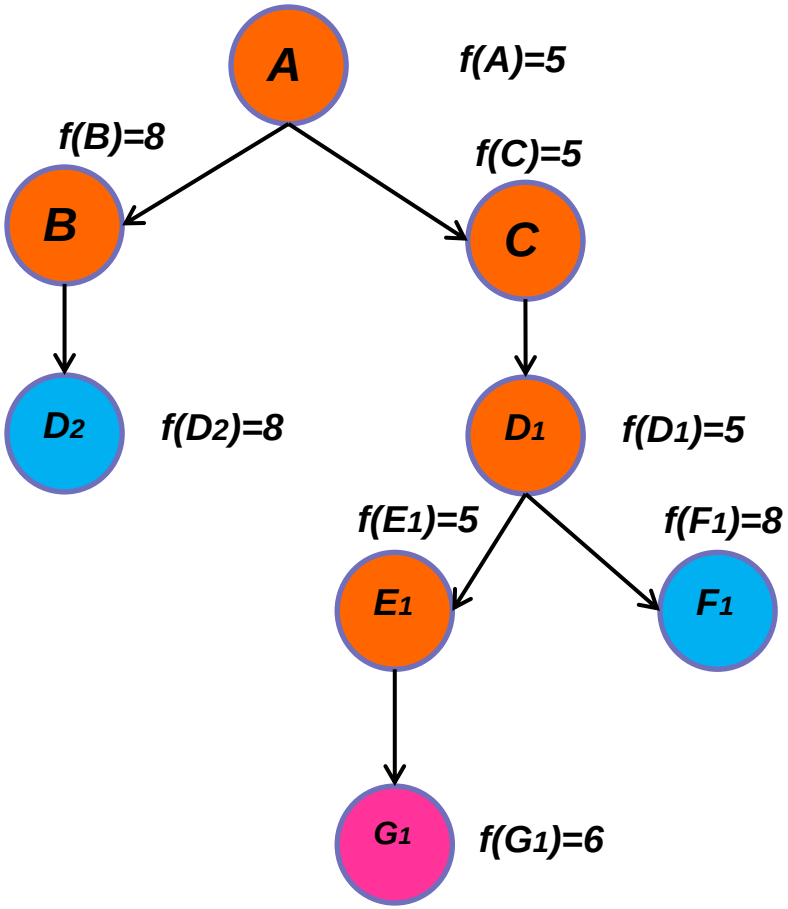
m-A* example


m=2
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)=1
2
2
G
h(G)=0

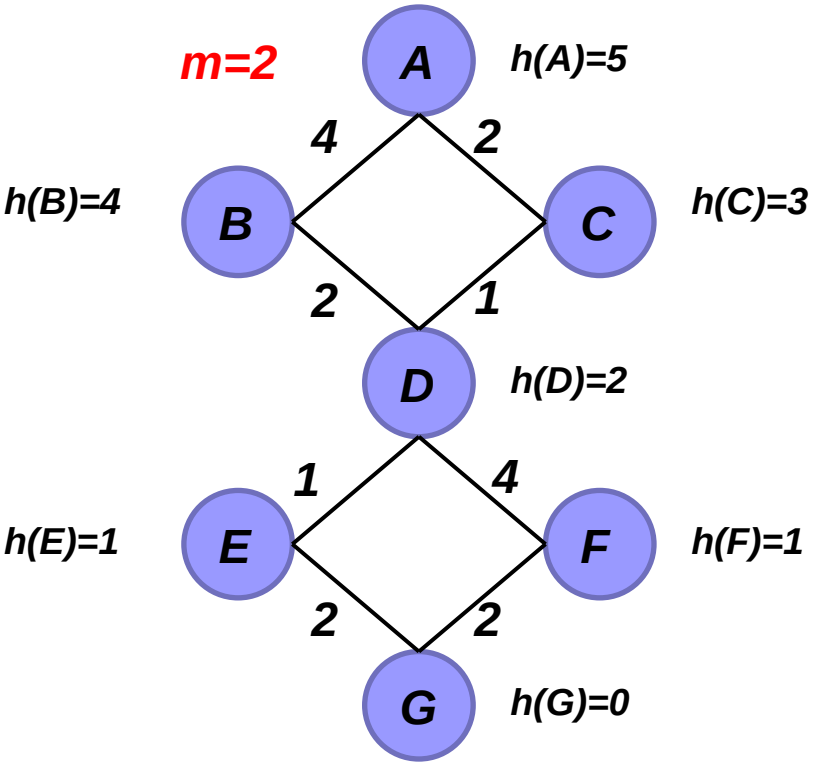

A
f(A)=5
f(B)=8
B
f(C)=5
C
D1
f(D1)=5
f(E1)=5
f(F1)=8
E1
F1
G1
f(G1)=6

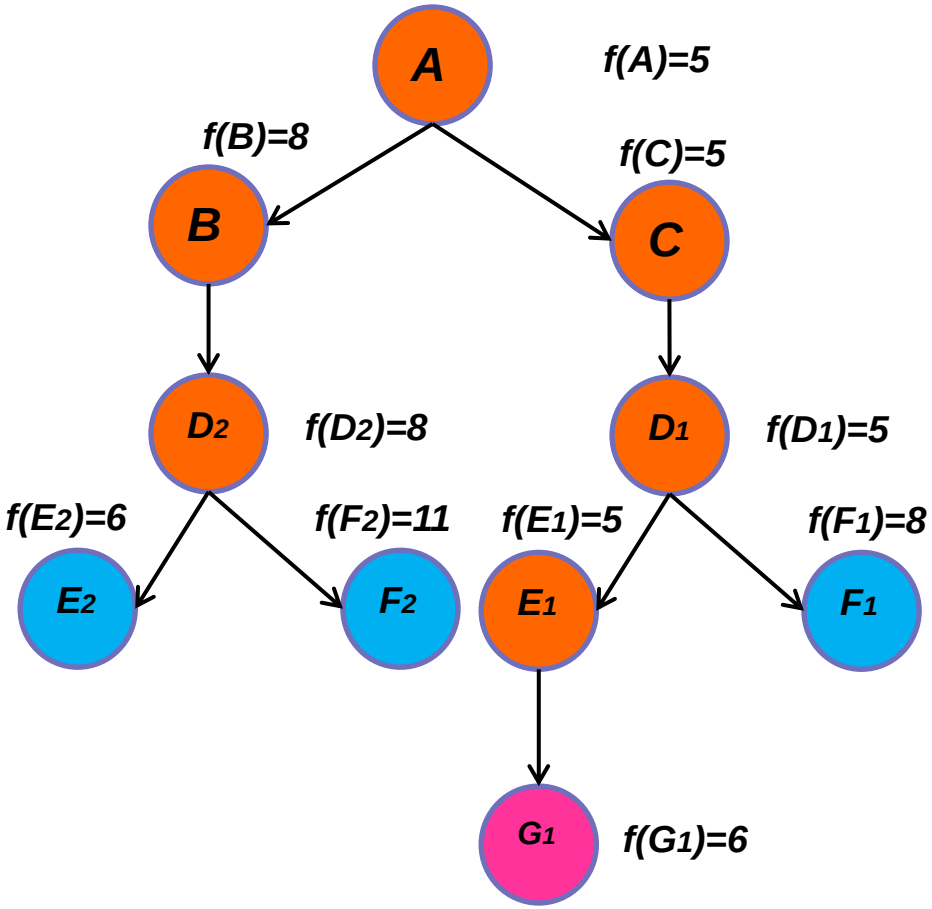
m-A* example


m=2
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)=1
2
2
G
h(G)=0

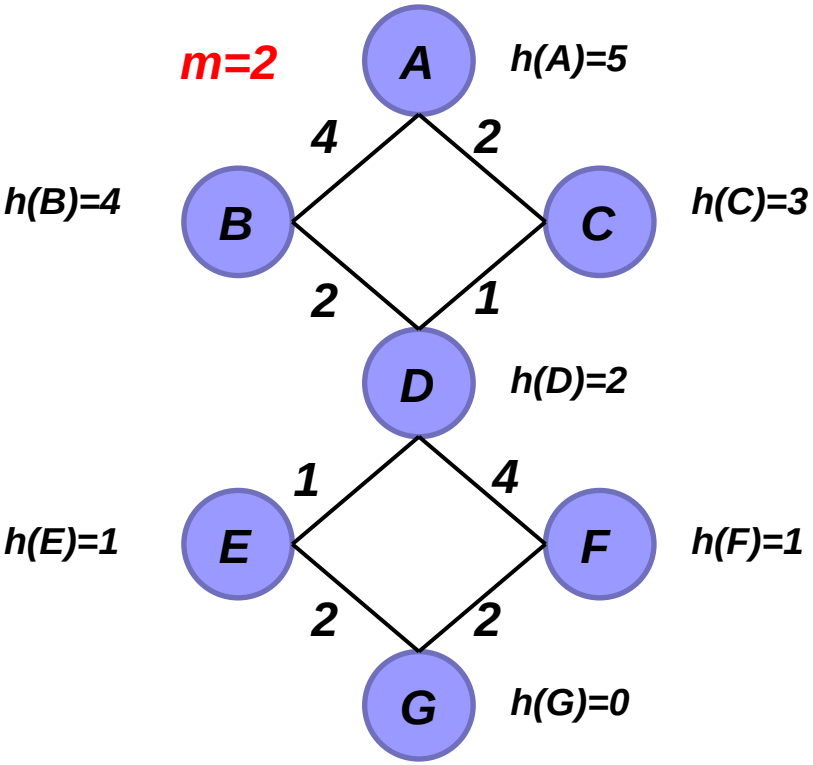

A
f(A)=5
f(B)=8
f(C)=5
B
C
D2
f(D2)=8
D1
f(D1)=5
f(E1)=5
f(F1)=8
E1
F1
G1
f(G1)=6

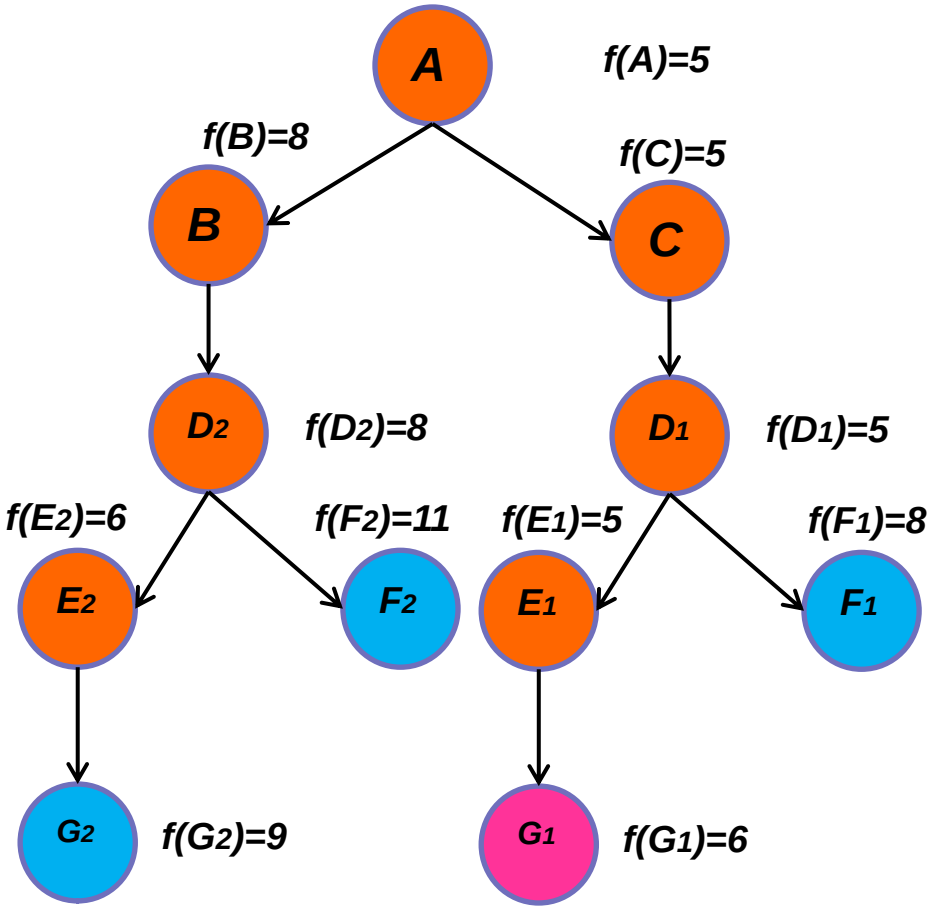
m-A* example


m=2
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)=1
2
2
G
h(G)=0

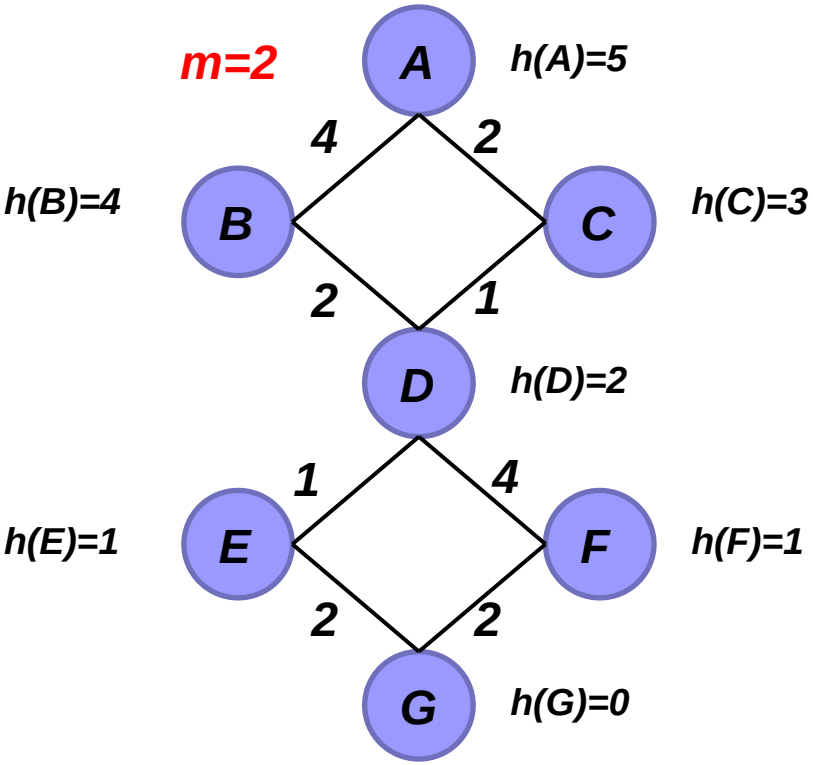

A
f(A)=5
f(B)=8
f(C)=5
B
C
D2
f(D2)=8
D1
f(D1)=5
f(E2)=6
f(F2)=11
f(E1)=5
f(F1)=8
E2
F2
E1
F1
G1
f(G1)=6

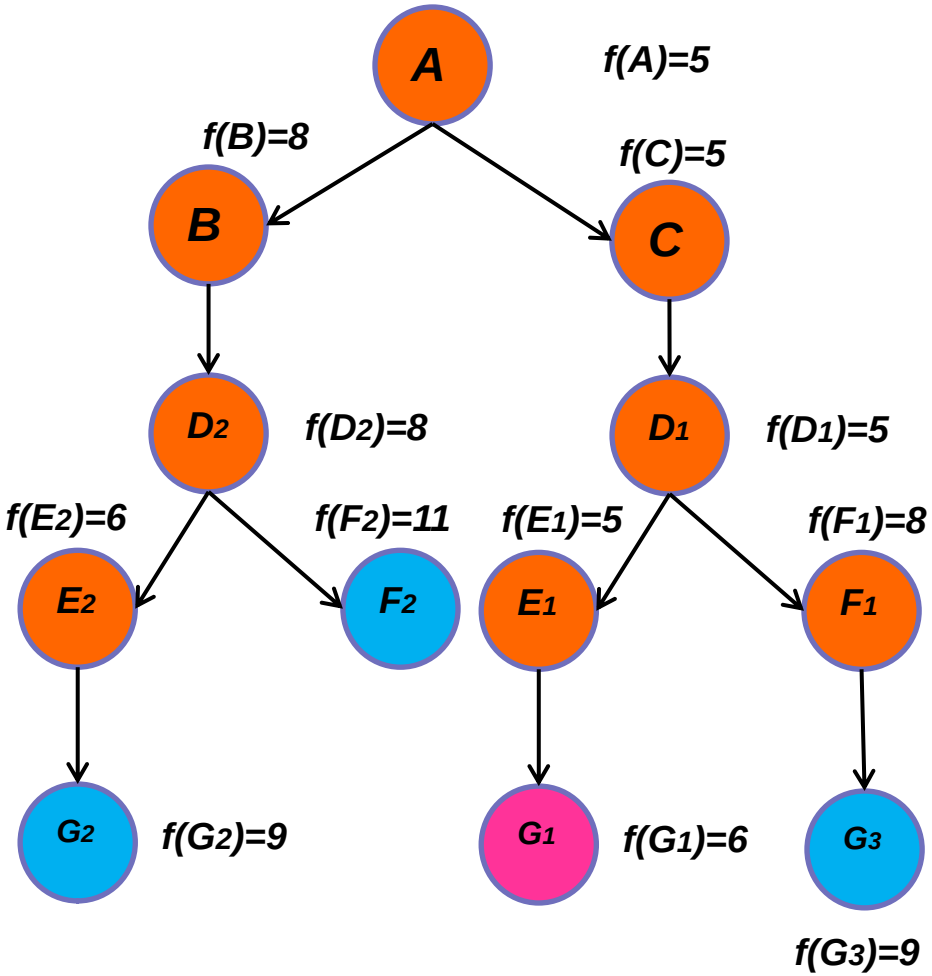
m-A* example


m=2
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)=1
2
2
G
h(G)=0

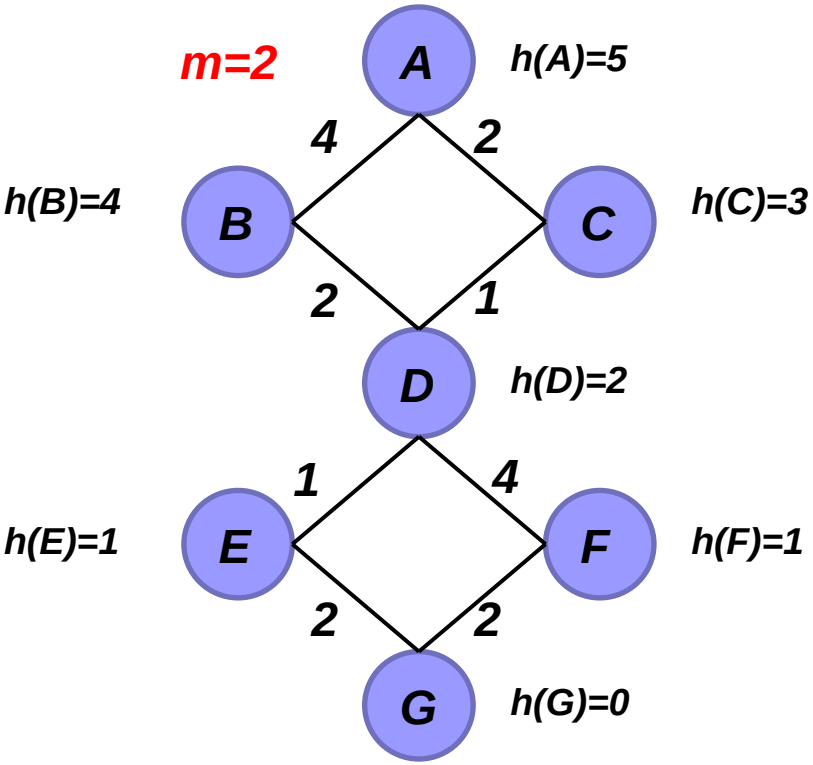

A
f(A)=5
f(B)=8
f(C)=5
B
C
D2
f(D2)=8
D1
f(D1)=5
f(E2)=6
f(F2)=11
f(E1)=5
f(F1)=8
E2
F2
E1
F1
G2
f(G2)=9
G1
f(G1)=6

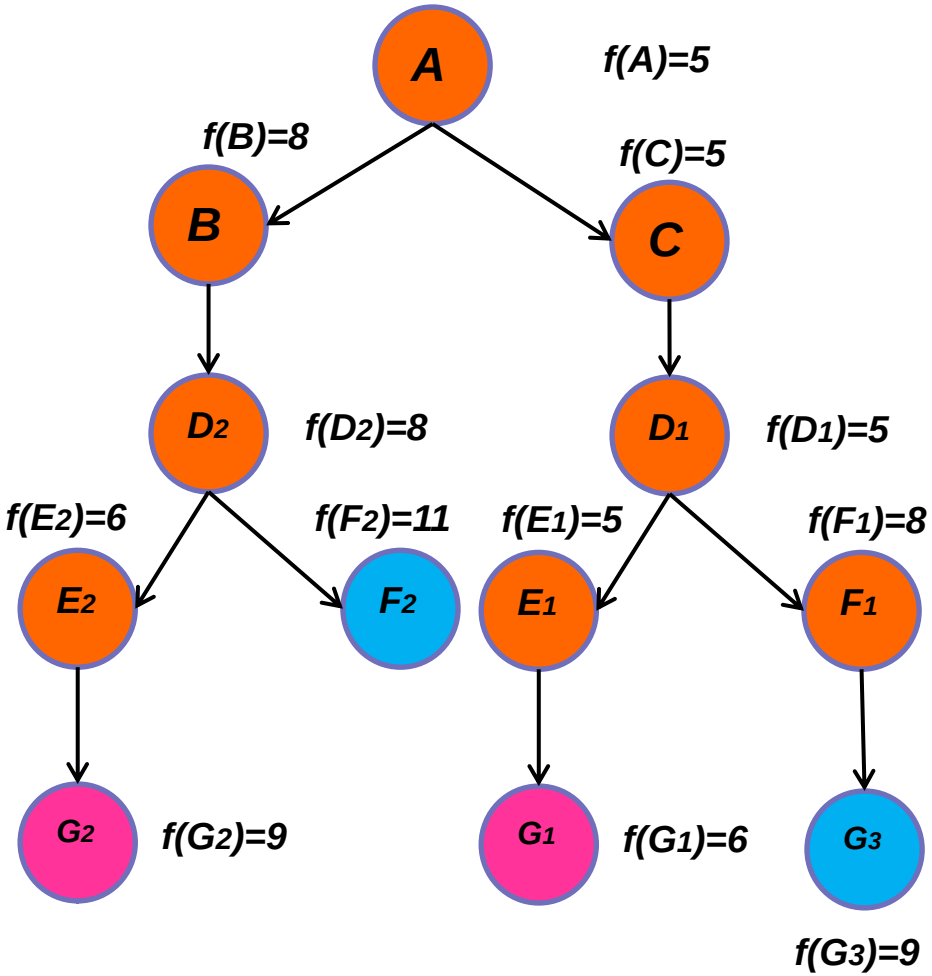
m-A* example


m=2
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)=1
2
2
G
h(G)=0


A
f(A)=5
f(B)=8
f(C)=5
B
C
D2
f(D2)=8
D1
f(D1)=5
f(E2)=6
f(F2)=11
f(E1)=5
f(F1)=8
E2
F2
E1
F1
G2
f(G2)=9
G1
f(G1)=6
G3
f(G3)=9

m-A* example


m=2
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)=1
2
2
G
h(G)=0

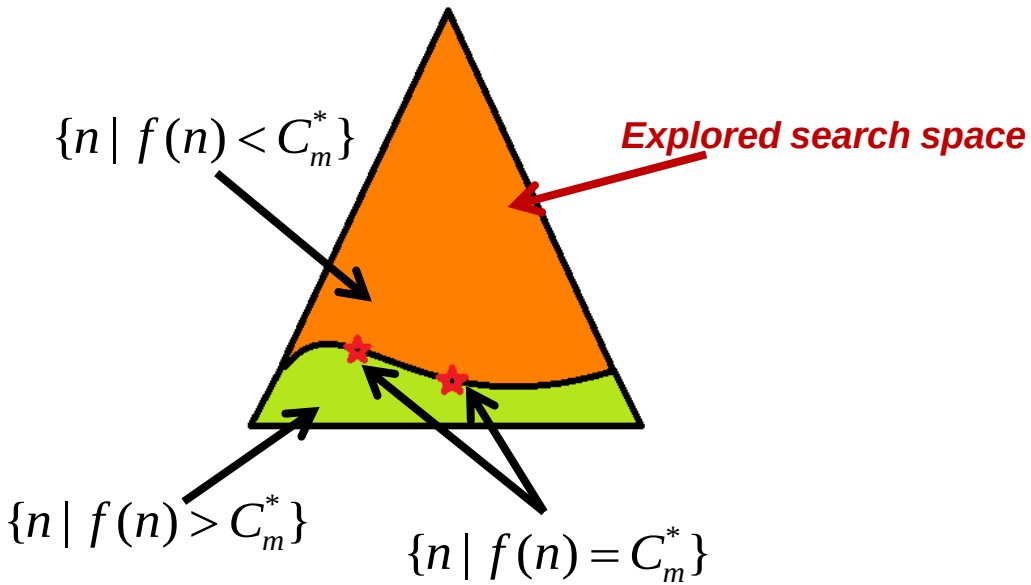

A
f(A)=5
f(B)=8
f(C)=5
B
C
D2
f(D2)=8
D1
f(D1)=5
f(E2)=6
f(F2)=11
f(E1)=5
f(F1)=8
E2
F2
E1
F1
G2
f(G2)=9
G1
f(G1)=6
G3
f(G3)=9

Properties m-A*

1. **Soundness and completeness**: m-A* terminates with the m-best solutions generated in order of their costs.
2. **Optimal efficiency:** any node that is *surely* expanded by m-A* *must be* expanded by any other sound and complete algorithm.
3. **Optimal efficiency for consistent heuristics:** when the heuristic function is consistent, m-A* expands each node at most *m* times.
4. **Dominance:** Given two heuristic functions h_1 and h_2, s.t. for every n $h_1(n) < h_2(n)$, $m\text{-}A^*_1$ will expand every node surely expanded by $m\text{-}A^*_2$.

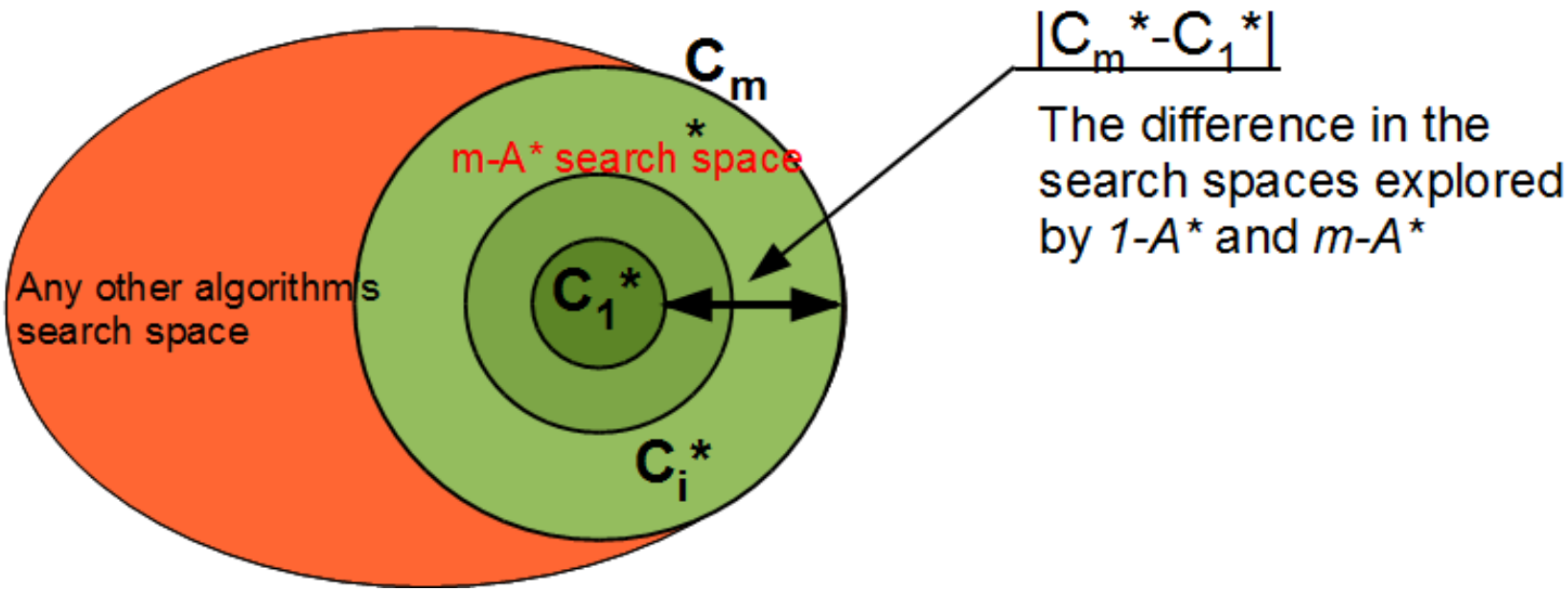
Properties m-A*

- 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



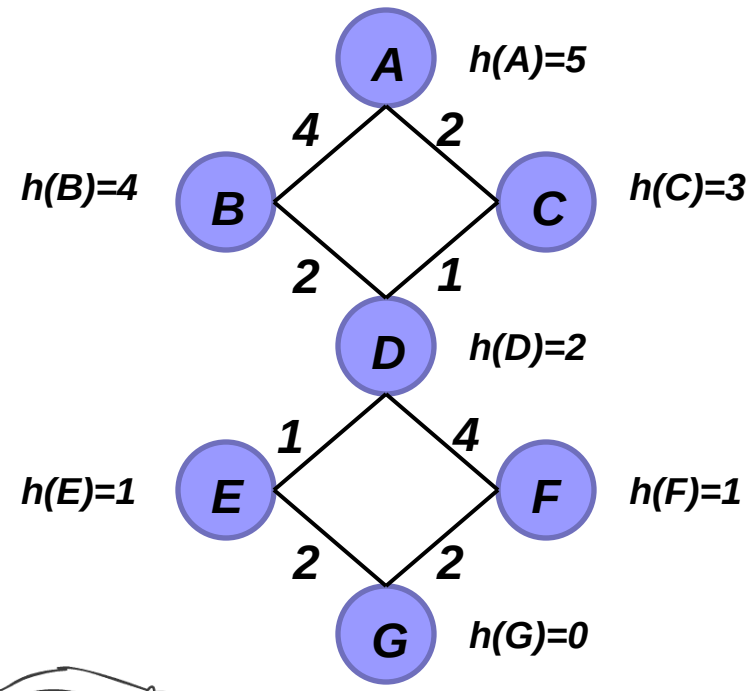
Properties m-A*

- Impact of *m* on the search space size

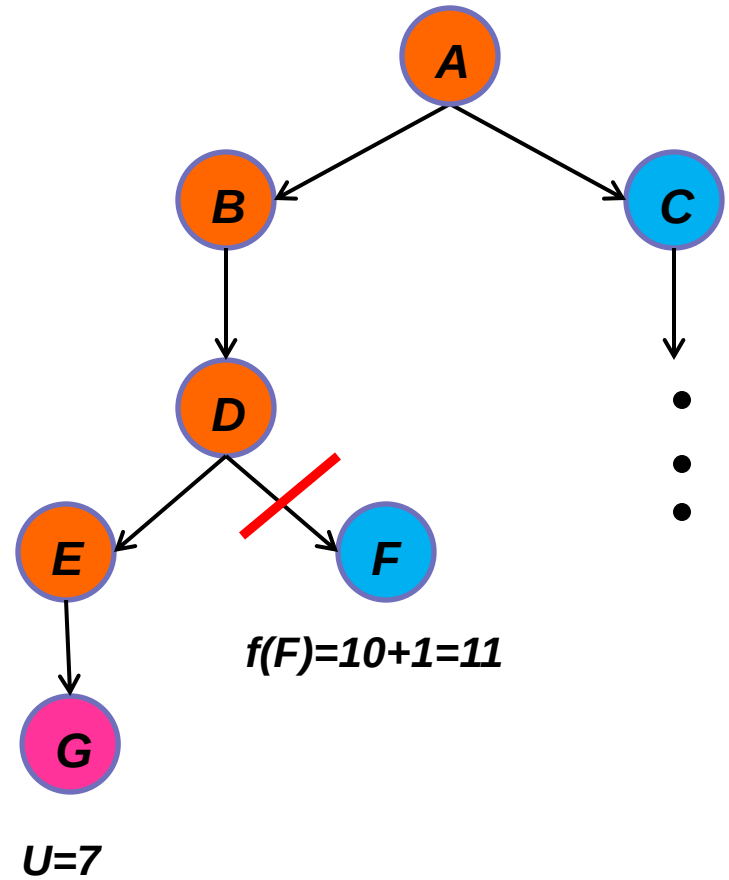


Background: Depth First Branch and Bound

- Maintains the cost of the best solution found so far **U** as an upper bound on the optimal solution



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

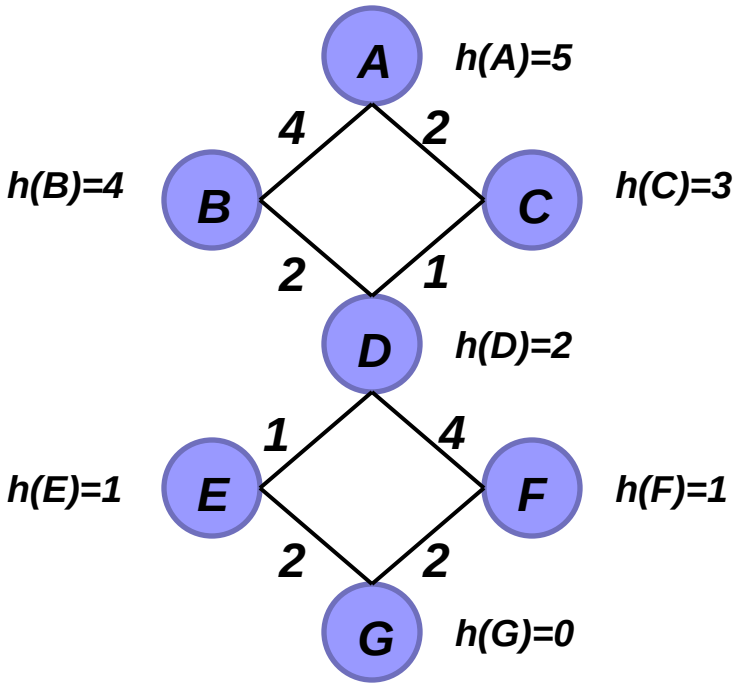


m-BB algorithm

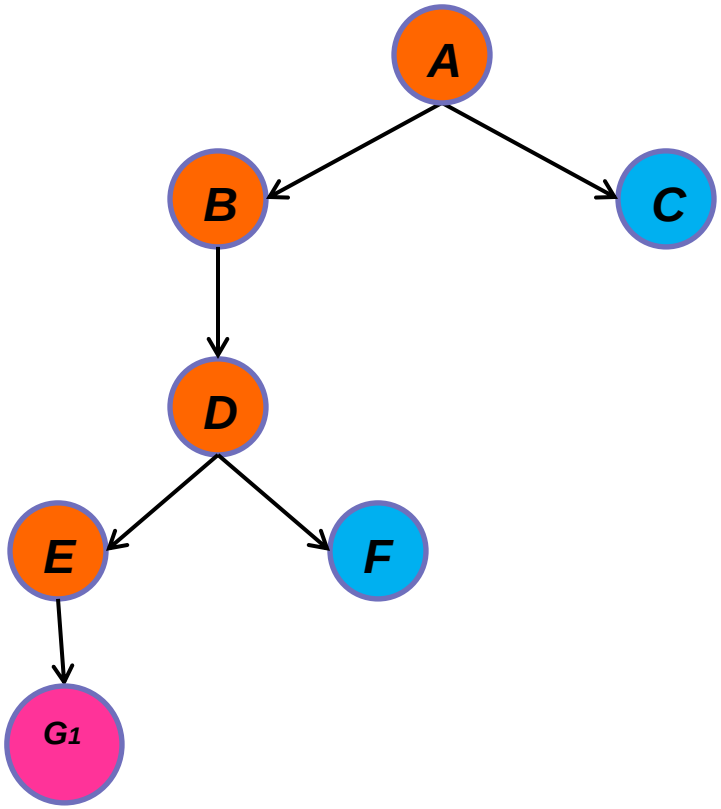
- Algorithm maintains the list of m best out of the solutions found so far
- ***U*$_m$** – the cost of mth best of them
- Prune node *n* if *f(n)*< ***U*$_m$**
- Given a consistent heuristic
  - m-B&B will expand all nodes such that $E_m^* = \{n \mid f(n) < C_m^*\}$
  - Initially, and until m-B&B encounters the true m^{th} -best solution, it will expand also nodes $F_m^* = \{n \mid f(n) > C_m^*\}$ Subsequently, like m-A*, it will only expand nodes for which $f(n) \le C_m^*$

m-BB algorithm

m=2

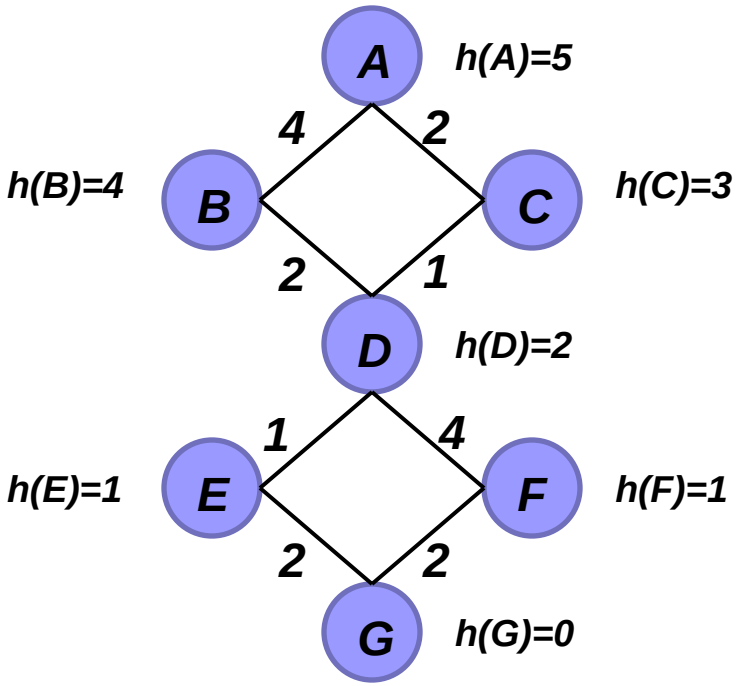


$$U_1 = 7; U_2 = \infty$$

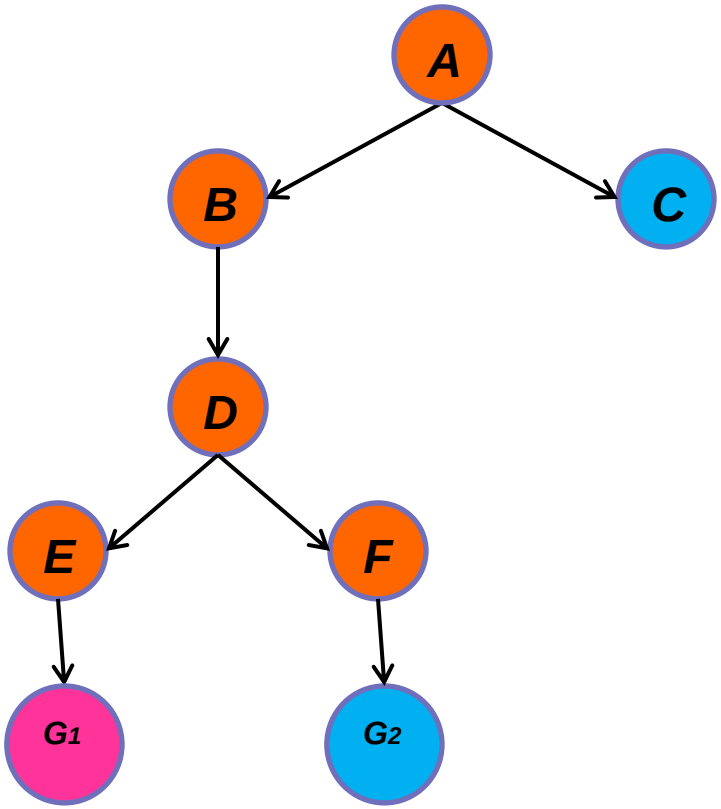


m-BB algorithm

m=2

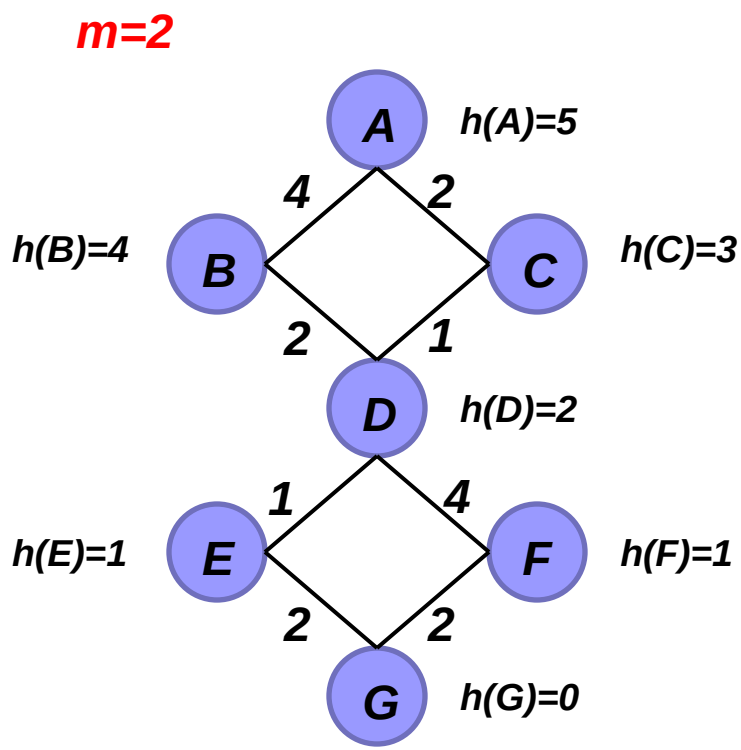


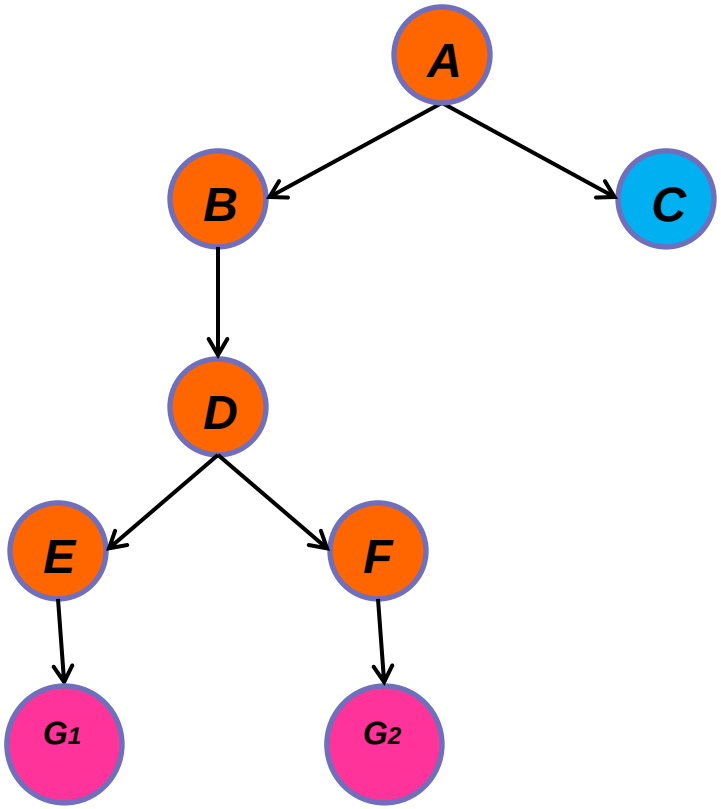
$$U_1 = 7; U_2 = \infty$$



m-BB algorithm

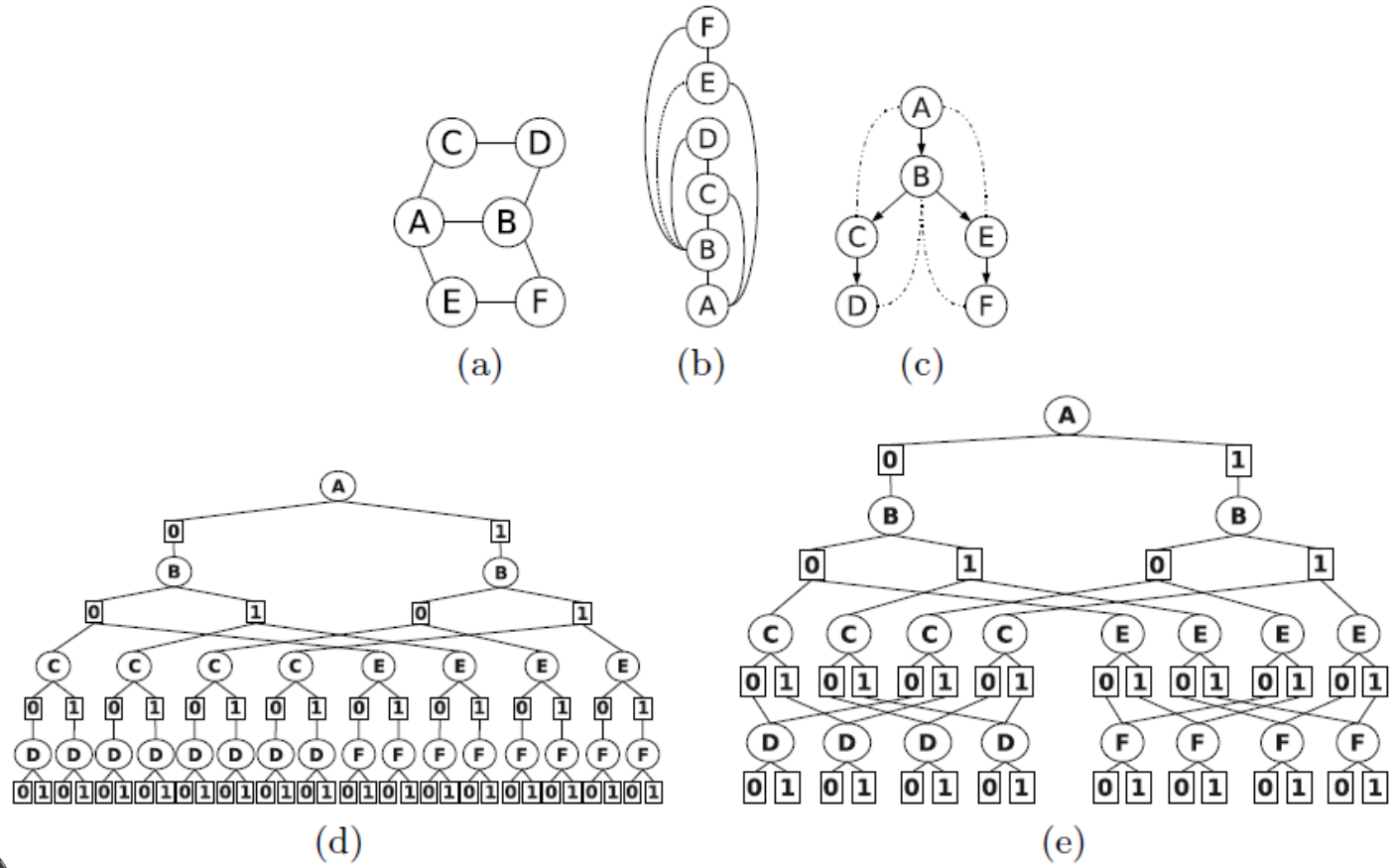
$$U_1 = 7; U_2 = 12$$





AND/OR search spaces

- Optimization problems are known to be effectively solved by searching a **weighted AND/OR search graph** or weighted AND/OR search tree.



m-A* for graphical models (m-AOBF)

- The size of the **AND/OR search tree** is bound by $O(N \cdot k^h)$
- The size of the **context-minimal AND/OR search graph** is bound by $O(N \cdot k^{w^*+1})$
- Applying m-A* to searching *weighted context-minimal AND/OR search graphs* yields *m-AOBF*.
- Each node can be expanded at most *m* times
- The search space explored by m-AOBF is bounded by

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

m-BB for graphical models (m-AOBB)

- At each AND node we combine the m best solutions to the subproblems rooted in its children and pick the best m out of the combination results – time overhead of per AND node $O(\deg \cdot m \cdot \log m)$

AND/OR tree

- Larger search space $O(N \cdot k^h)$
- Overhead due to AND nodes
- **Total run-time complexity**

$$O(m \cdot N \cdot k^h \cdot \deg \cdot \log m)$$

AND/OR graph

- Smaller search space $O(N \cdot k^{w^*+1})$
- Overhead due to AND nodes and caching:

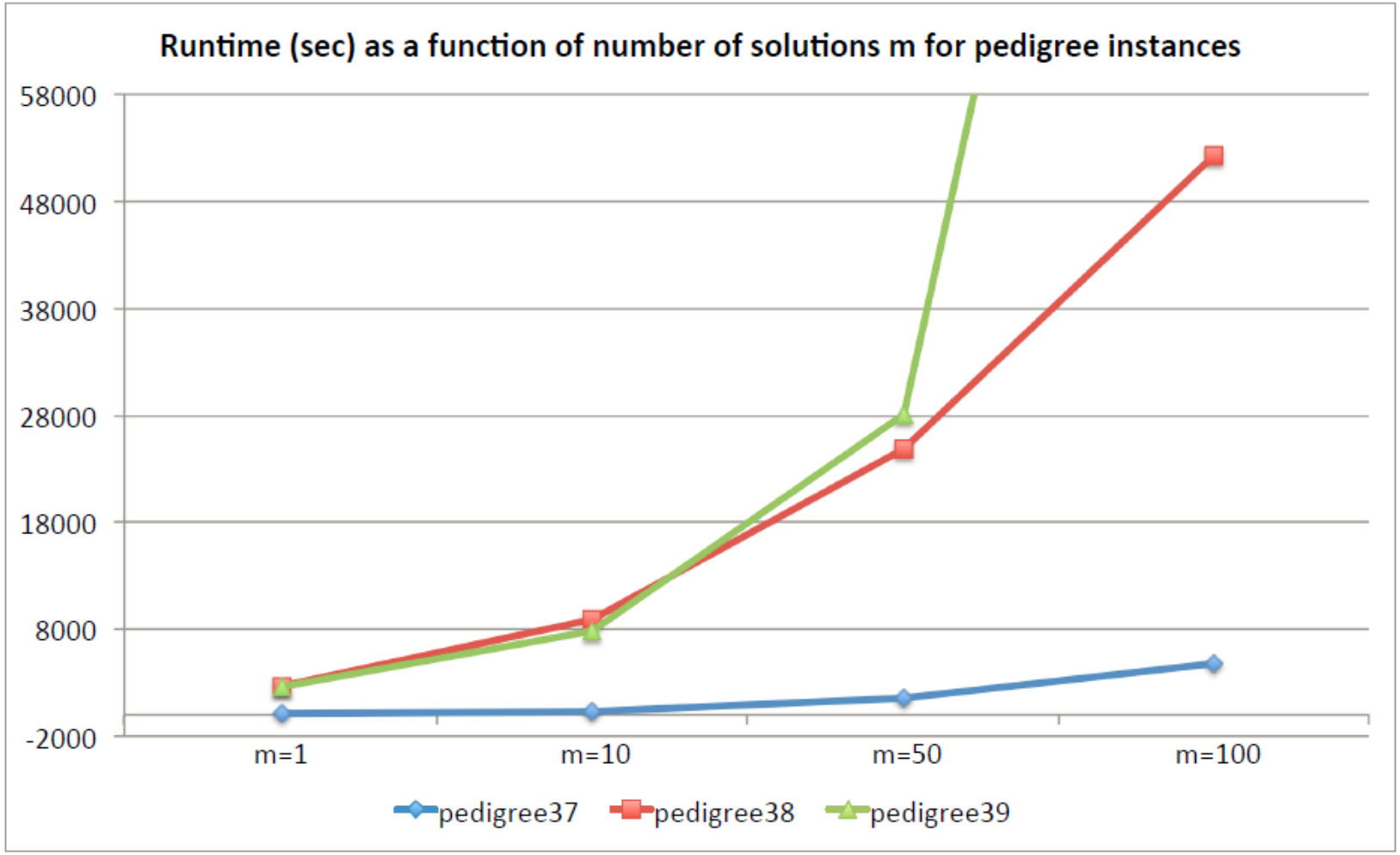
  $O(\deg \cdot m \cdot \log m)$ per AND node
  $O(m \cdot \log m)$ per each subproblem cached (i.e. each OR node)

  Total run-time complexity:
  $O(m \cdot N \cdot k^{w^*+1} \cdot \deg \cdot \log m)$

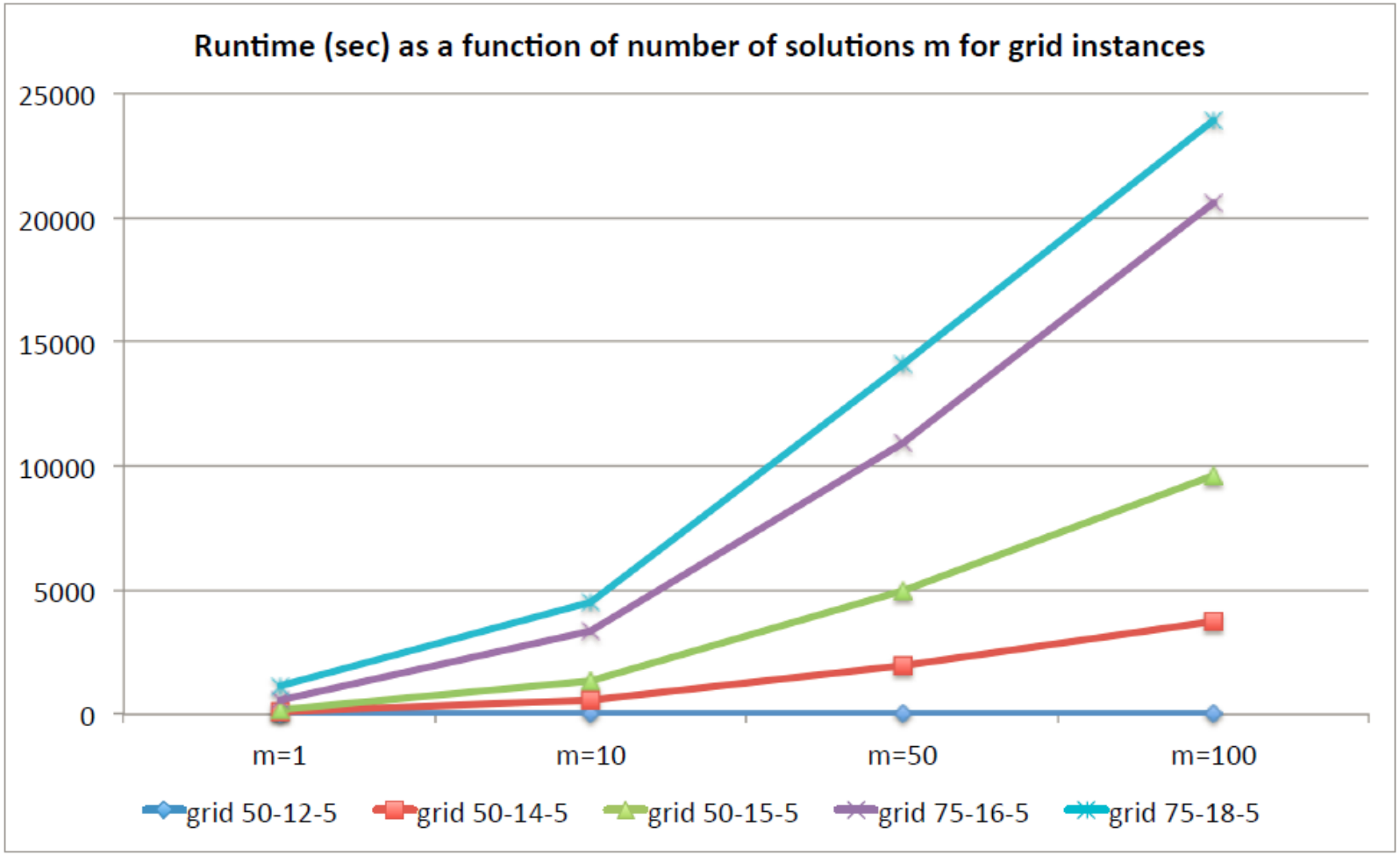
Algorithm BE-Greedy-m-BF

- Generate exact heuristic by running Bucket Elimination along the order
- Run m-A* along the ordering, the algorithm will expand only nodes that lay on the *m* best paths
- The total number of nodes expanded is O(mN), where N bounds the solution length

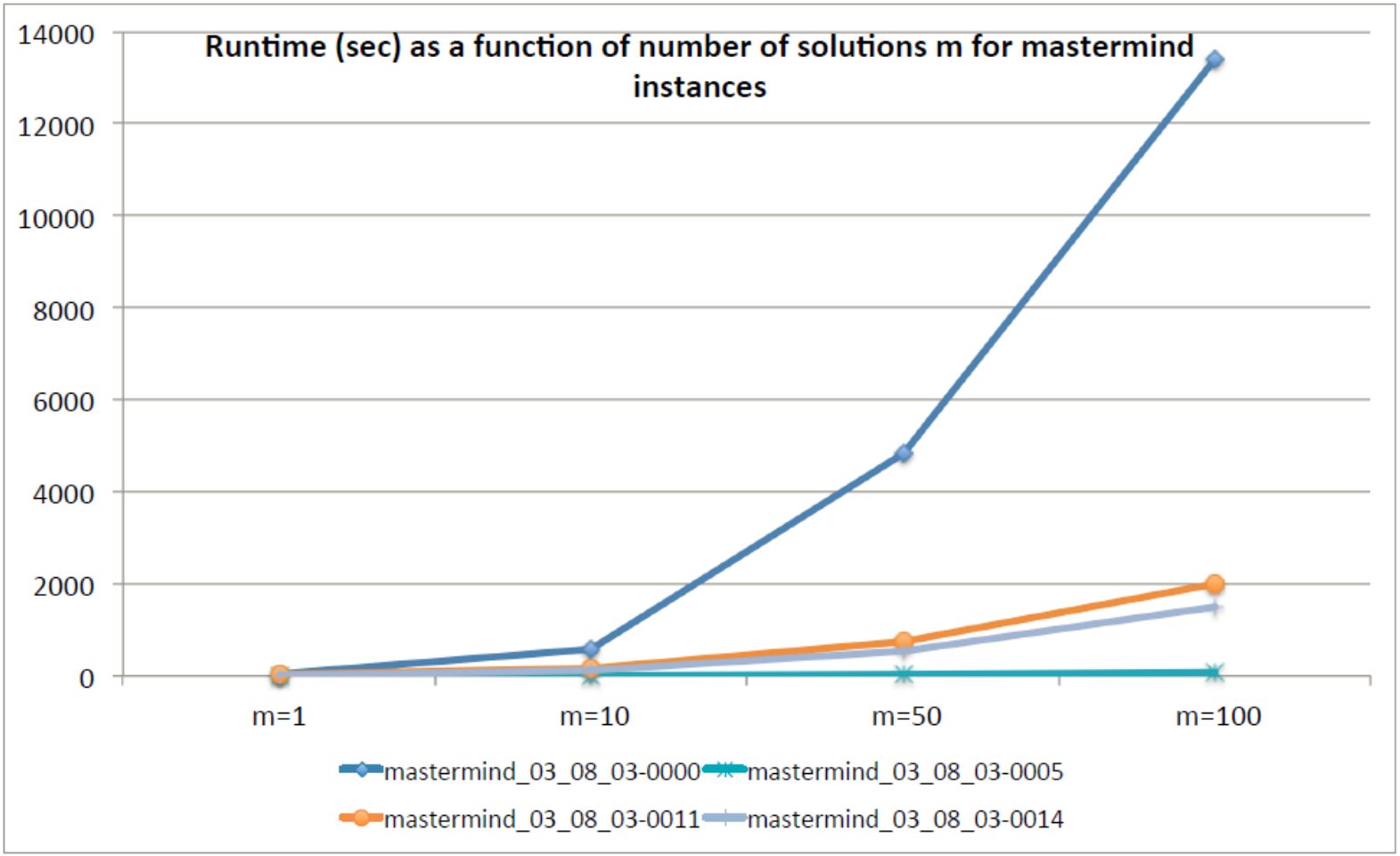
Empirical evaluation



Empirical evaluation



Empirical evaluation



Empirical evaluation

Instance	N	k	w*	h	m=1	m=10	m=50	m=100
pedigree1	298	4	15	59	1	32	464	1751
pedigree37	726	5	20	72	41	240	1537	4825
pedigree38	581	5	16	52	2700	8890	24894	t/o
pedigree39	953	5	20	77	2594	7907	28037	t/o
pedigree50	478	6	16	54	685	2835	21903	t/o
grid50-12-5	143	2	15	48	1	2	13	35
grid50-14-5	195	2	18	64	87	599	1995	3739
grid50-15-5	224	2	19	76	180	1382	5011	9591
grid75-16-5	256	2	21	73	539	3327	10917	20572
grid75-18-5	324	2	24	85	1147	4533	14078	23948
mm-03-0000	1220	2	18	43	43	580	4845	13381
mm-03-0001	1193	2	15	38	1	2	8	23
mm-03-0002	1193	2	15	39	1	3	17	48
mm-03-0004	1193	2	15	38	0	3	17	46
mm-03-0005	1193	2	15	38	0	5	25	70
mm-03-0011	1172	2	18	42	18	148	757	2016
mm-03-0014	1172	2	18	43	15	107	544	1488
mm-04-0012	2224	2	29	56	842	6859	31952	t/o
mm-04-0013	2224	2	29	58	375	4880	23854	t/o
mm-04-0014	2224	2	29	60	655	8080	36740	t/o
mm-04-0015	2224	2	29	57	818	13299	t/o	t/o

Conclusion:

- We present two new search-based algorithms for solving the m-best task extending known algorithm that explore AND/OR space in Best First and in Depth First manner, using heuristic information.
- Future work:
  - Improve the emplementation
  - Perform more extensive the empirical evaluation