Block 1 – Knowledge Representation and Search

Chapter 6:
Applying heuristic search in RBS

Block 1, Chapter 6- Index

1. Design of heuristic funcions
2. Design of heuristic funcions for the 8-puzzle problem
3. Heuristics for the traveling salesman problem
4. Evaluation of heuristic functions
5. The family problem

Bibliography

S. Russell, P. Norvig. ***Artificial Intelligence . A modern approach.*** Prentice Hall, 3rd edition, 2010 (Chapter 3) http://aima.cs.berkeley.edu/

Alternatively:
S. Russell, P. Norvig. ***Artificial Intelligence . A modern approach.*** Prentice Hall, 2nd edition, 2004 (Chapters 3 and 4) http://aima.cs.berkeley.edu/2nd-ed/

1. Design of heuristic functions

Heuristics are problem-dependent functions.

Given a particular problem, how might one come up with a heuristic function for the problem? In case of using an algorithm A*, how can one invent an admissible heuristic?

Common technique: ***Relaxed problem***

A problem with less restrictions on the operators is called a relaxed problem.

Admissible heuristics can be derived from the cost of an **exact solution to a relaxed version of the problem**. This is usually a good heuristic for the original problem.

2. Design of heuristic functions for the 8-puzzle problem

A tile in square A can move to square B if:

Restriction 1: B is adjacent to A
Restriction 2: B is the empty space (blank)

h1: misplaced tiles

- It removes both restrictions
- Relaxed 8-puzzle for h1 : a tile can move anywhere
- As a result, h1(n) gives a very optimistic estimate (shortest solution)

h2: Manhattan distance

- It removes Restriction 2
- Relaxed 8-puzzle for h2: a tile can move to any adjacent square
- As a result, h2(n) gives an optimistic estimate (shortest solution)

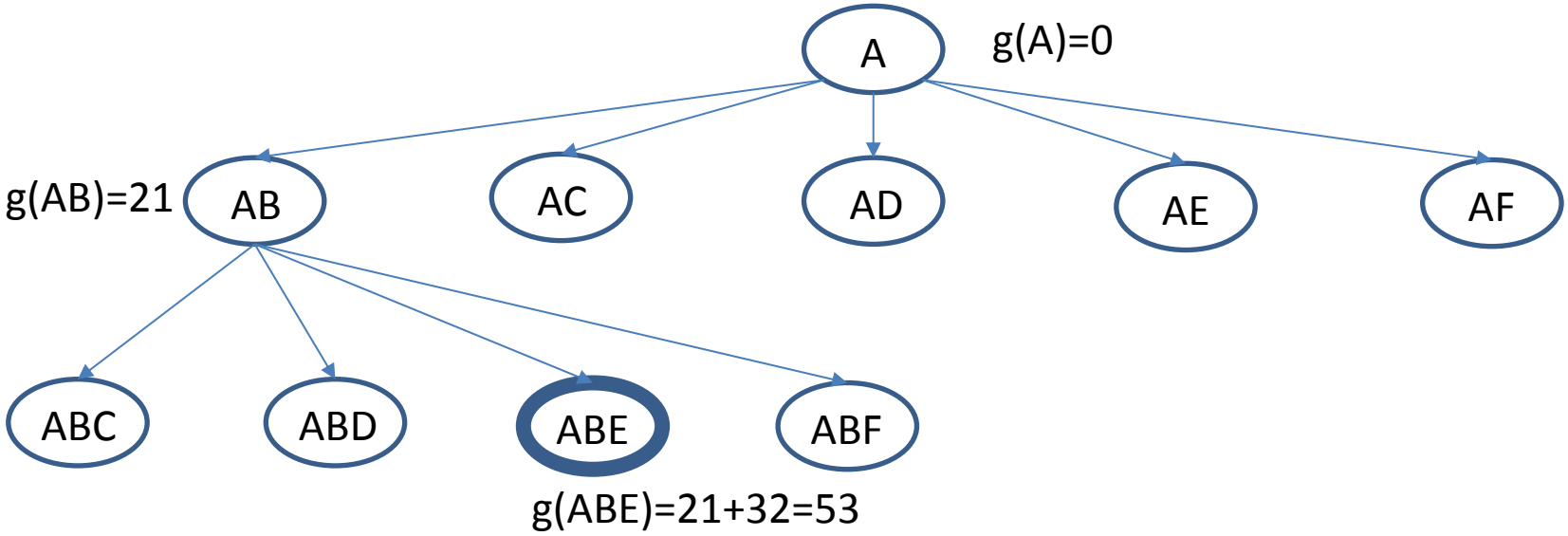
The optimal solution cost of a relaxed problem is no greater than the optimal solution cost of the real problem

3. Heuristics for the traveling salesman problem

6 cities: A,B,C,D,E,F
States: sequences of cities starting in city A which represent partial routes
Start: A
Final: sequences starting and ending in A which are made up of all cities
Rules or operators: add at the end of each state one city which is not in the sequence already
Operators cost: distance between the last city in the sequence and the recently added city (see table)

	A	B	C	D	E	F
A		21	12	15	113	92
B			7	25	32	9
C				5	18	20
D					180	39
E						17

3. Heuristics for the traveling salesman problem



Cities that have not been reached yet: C, D, F, and back to A

h*(ABE) = min(ECDFA, ECFDA, EDCFA, EDFCA, EFCDA, EFDCA)
min(154, 92, 297, 251, 57, 73)

3. Heuristics for the traveling salesman problem

h1(n)=0 in any case (uniform cost)
h2(n)= number of missing arcs multiplied by the arc of minimal cost
[h2(ABE)=4*5=20]
h3(n)= sum of the shortest *p* arcs if there are *p* missing arcs
[h3(ABE)=5+5+7+7=24]
h4(n)= sum of the shortest outgoing arcs of the cities which have not been left yet
[h4(ABE)=17{out E}+5 {out C}+5 {out D}+9 {out F}=36]
h5(n)= sum of the shortest outgoing arcs of the cities which have not been reached yet
[h5(ABE)=12{out A}+5 {out C}+5 {out D}+9 {out F}=31]
h6(n)= shortest distance between last city in 'n' and the start city
h7(n)= number of missing arcs multiplied by their average cost
h8(n)= sum of the highest-cost *p* arcs if *p* arcs are missing

Heuristics 7 and 8 are non-admissible.

4. Evaluation of heuristic functions

Difficult to apply mathematical analysis

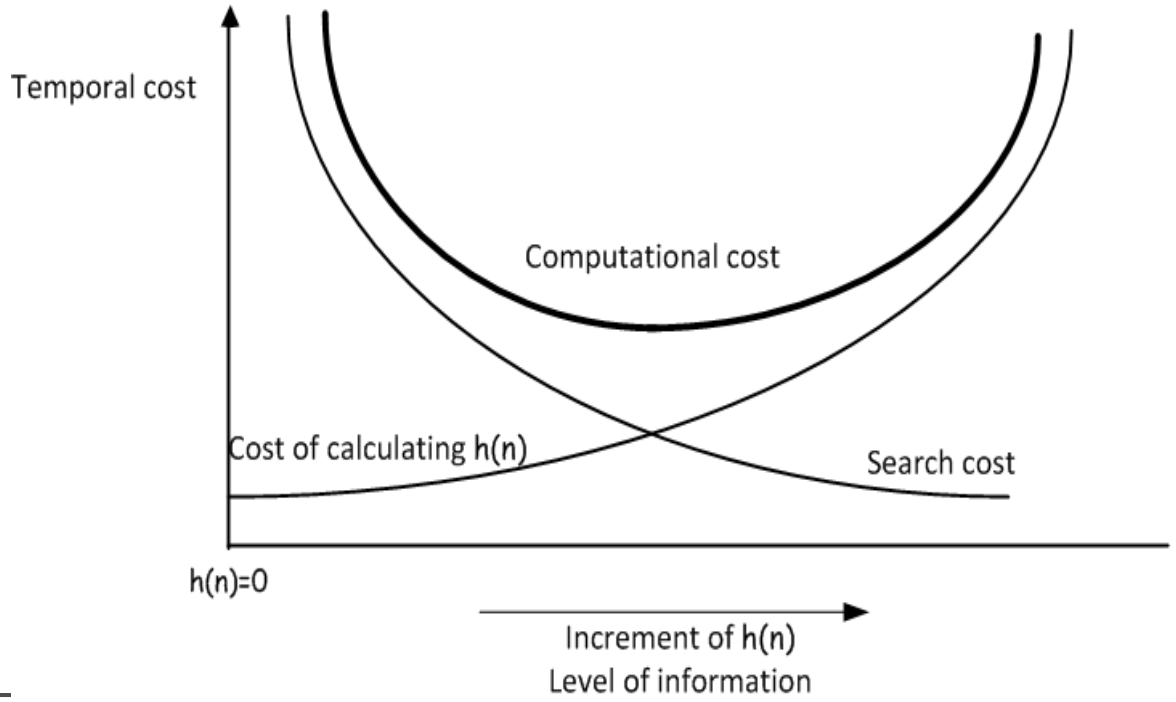
Often used statistical/experimental methods.

Objective: minimize the total cost of the problem, trading off computational expense and path cost

Computational cost (temporal cost):

Search cost: number of nodes generated or applied operators **+**

Cost of calculating h(n): cost for selecting the applicable operator.



4. Evaluation of heuristic functions

Effective branching factor (b*)

If the total number of nodes generated by A* for a particular problem is N, and the solution depth is d, then b* is the branching factor that a uniform tree of depth d would have to have in order to contain N+1 nodes. Thus,

$$N+1=1+b^*+(b^*)^2+ \ldots + (b^*)^d$$

For example, if A* finds a solution at depth 5 using 52 nodes, $b^*=1.92$

- b* defines how sharply a search process is focussed toward the goal.
- b* is reasonably independent of path length (d)
- b* can vary across problem instances, but usually is fairly constant for sufficiently hard problems

- A well-designed heuristic would have a value of b* close to 1, allowing fairly large problems to be solved.
- A value of b* near unity corresponds to a search that is highly focussed toward the goal, with very little branching in other directions.

4. Evaluation of heuristic functions

- 1200 random problems with solution lengths from 2 to 24.

	Search Cost			Effective Branching Factor		
d	IDS	A*(h_1)	A*(h_2)	IDS	A*(h_1)	A*(h_2)
2	10	6	6	2.45	1.79	1.79
4	112	13	12	2.87	1.48	1.45
6	680	20	18	2.73	1.34	1.30
8	6384	39	25	2.80	1.33	1.24
10	47127	93	39	2.79	1.38	1.22
12	3644035	227	73	2.78	1.42	1.24
14	–	539	113	–	1.44	1.23
16	–	1301	211	–	1.45	1.25
18	–	3056	363	–	1.46	1.26
20	–	7276	676	–	1.47	1.27
22	–	18094	1219	–	1.48	1.28
24	–	39135	1641	–	1.48	1.26

Although b* cannot be written explicitly as a function of d and N, it exists a plot of b* versus N for various values of d. This information is used to estimate how many nodes will be generated in a problem of effective branching factor b* and d steps

4. Evaluation of heuristic functions

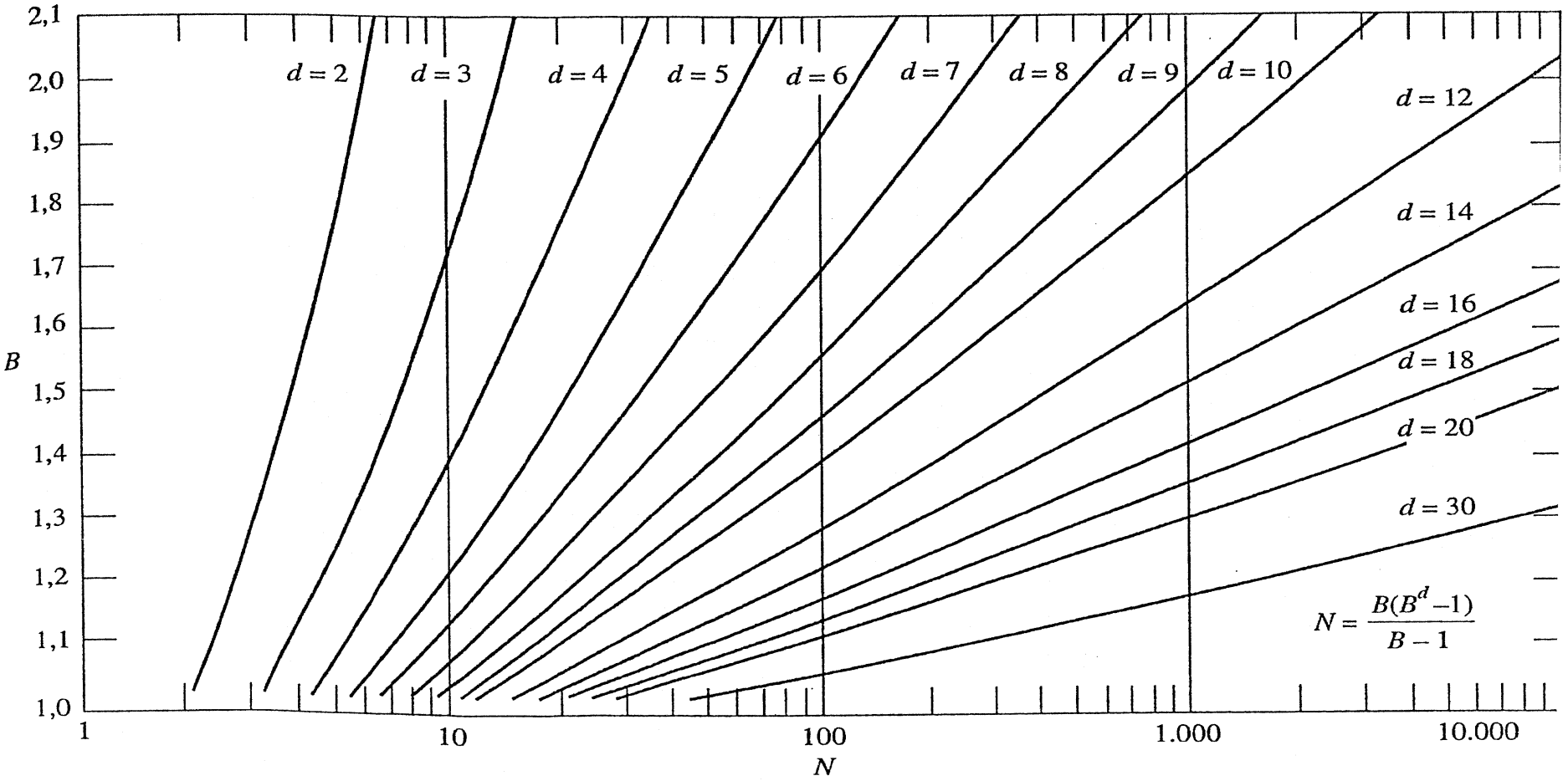


Figura 9.11.

B frente a *N* para distintos valores de *d*.

5. The family problem

Four members of a family are on one side of a bridge and need to cross to the other side. Each person crosses the bridge in one direction or another. Since they have to cross the bridge at night they must use a flashlight. There is only one flashlight available. Each family member crosses the bridge at a different speed and, therefore, it takes a different time (father: 8 sec., mother: 7 sec., son: 3 sec., daughter: 1 sec.). The flashlight is consumed each time a family member crosses the bridge and it cannot be recharged at any time. A **maximum of two people** can cross the bridge at the same time. Since each person moves at a different speed, if two of them cross at the same time they move at the speed of the slowest person. For example:

InitiaFL state: The four members are on the left side of the bridge
Action 1: The father crosses the bridge to the right side with the son (the flashlight consumes 8 sec.)
Action 2: The son crosses to the left side (the flashlight consumes 3 sec.)
Action 3: The daughter and the son cross to the right side together (the flashlight consumes 3 sec.)
etc.

5. The family problem

We want to design and solve this problem with a CLIPS RBS by applying a heuristic search A algorithm. We will assume the following functions:

1) $g(n)$ = time consumption of the flashlight
2) $h(n)$= sum of the crossing times of the first and third slowest members on the left (if they exist) +
   the crossing time of the fastest member on the right (**except when there are two people on the left with the flashlight**. In this case, a goal state is reached once the two people cross to the right side of the river so h(n)=0.

Tasks to do:

1) Select a proper knowledge representation to solve this problem with a RBS. Show the facts that represents the initial state.
2) Write the necessary rules in CLIPS to solve the problem by applying heuristic search.
3) Show the search tree that would result from executing the RBS indicating the value of the evaluation function $f(n)=g(n)+h(n)$ in each node
4) Is the solution found an optimal solution? Is $h(n)$ an admissible function?

5. The family problem

1) State representation:

(family left p^m right q^m flashlight l^s n^s)

where:

p^m , q^m Є {F, M, S, D} / $p^m \cap q^m = \emptyset$
l^s Є {left, right}
n^s Є integer

Initial state:

```
(deffacts data
    (family left F M S D right flashlight left 30)
    (times F 8 M 7 S 3 D 1))
```

5. The family problem

2) Rules in CLIPS:

```
(defrule cross-right-two-people
    (declare (salience (<evaluation_function_value>))
        (family left $?A ?member1 $?B ?member2 $?C
            right $?D flashlight left ?z)
        (times $? ?member1 ?x $?)
        (times $? ?member2 ?y $?)
        (test (>= ?z (max ?x ?y)))
=>
        (assert (family left $?A $?B $?C right $?D ?member1
        ?member2 flashlight right (- ?z (max ?x ?y)))))
```

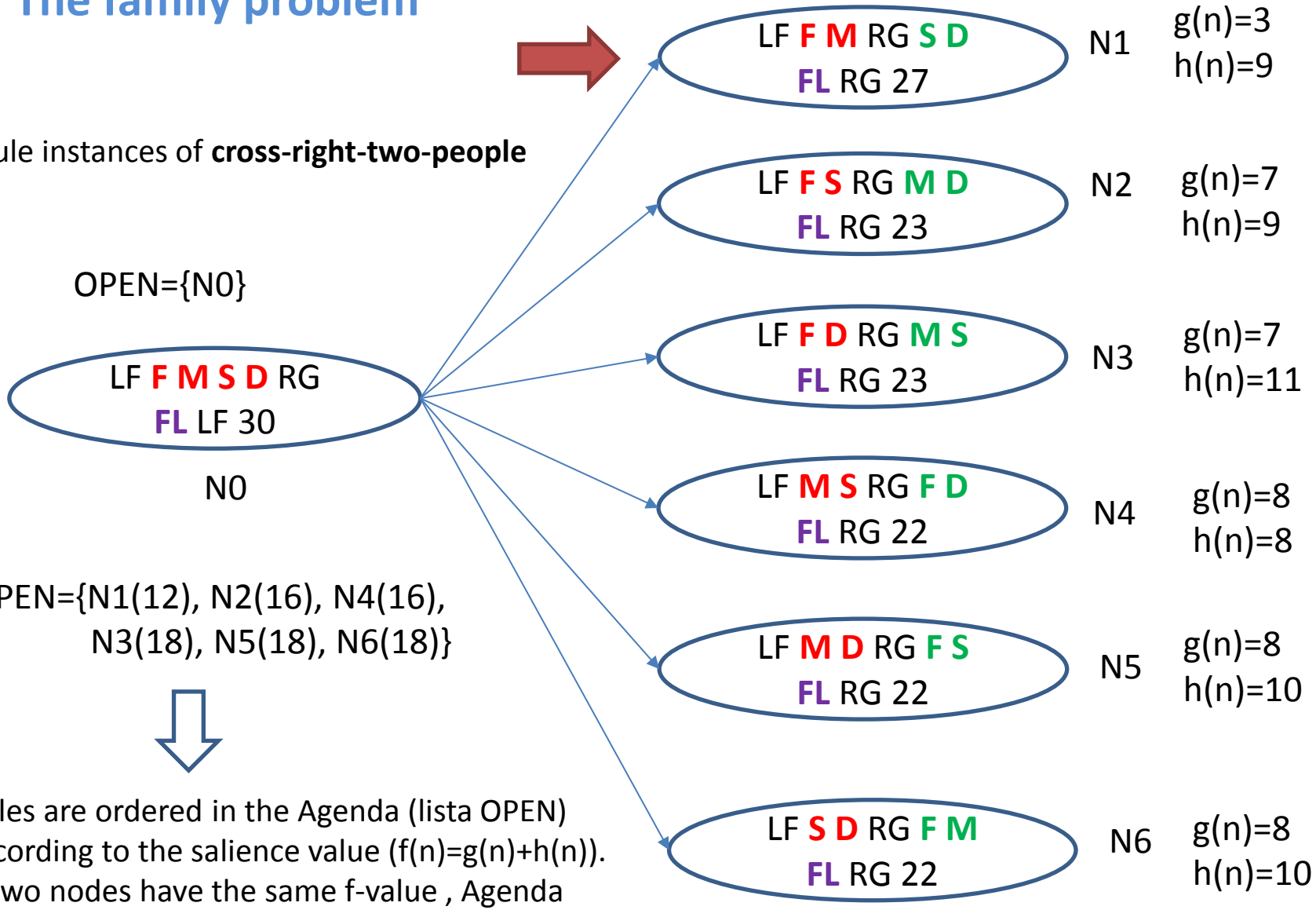
(declare (salience (<evaluation_function_value>)): rules are ordered according to the value of <evaluation_function_value>; the function encodes an A search.

5. The family problem

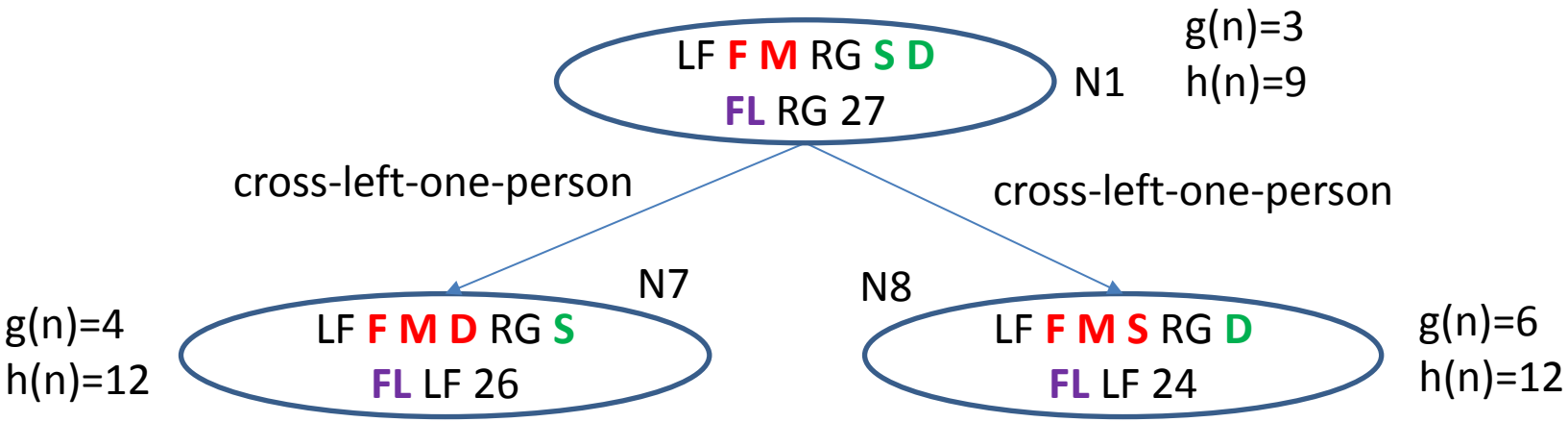
```
(defrule cross-left-one-person
    (declare (salience (<evaluation_function_value>))
      (family left $?A right $?B ?member $?C flashlight right ?z)
      (times $? ?member ?x $?)
      (test (>= ?z  ?x))
     =>
      (assert (family left $?A ?member right $?B  $?C  flashlight left (- ?z  ?x))))

(defrule goal
          (declare (salience 100))
          (family left right $? flashlight ? ?z)
=>
          (printout T "GoaFL reached. Time left in flashlight "  ?z crlf)
          (halt))
```

5. The family problem

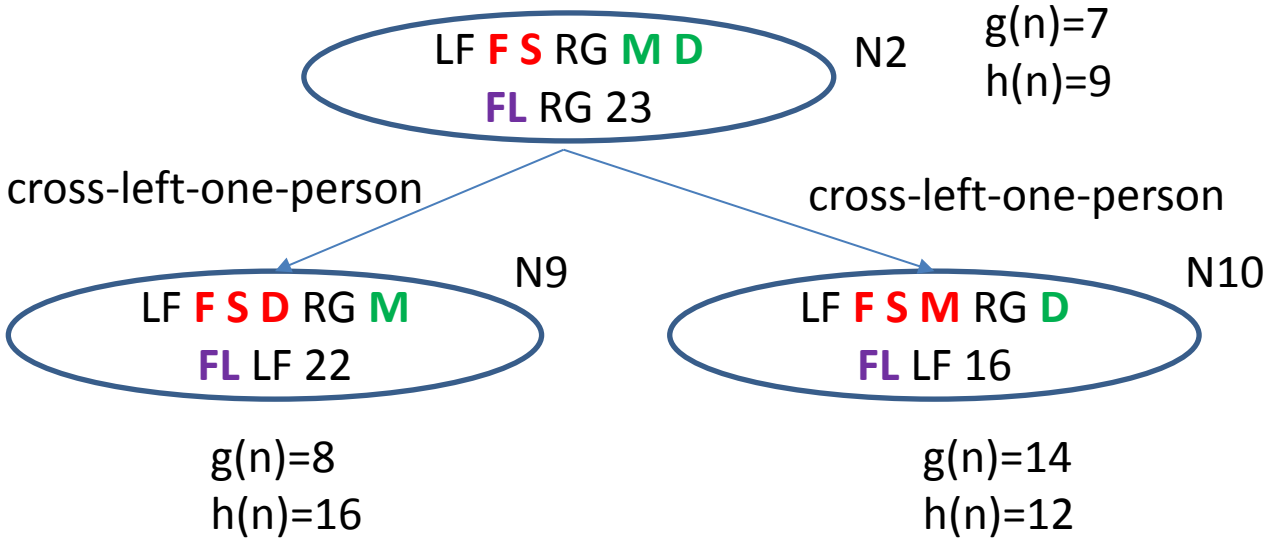


5. The family problem



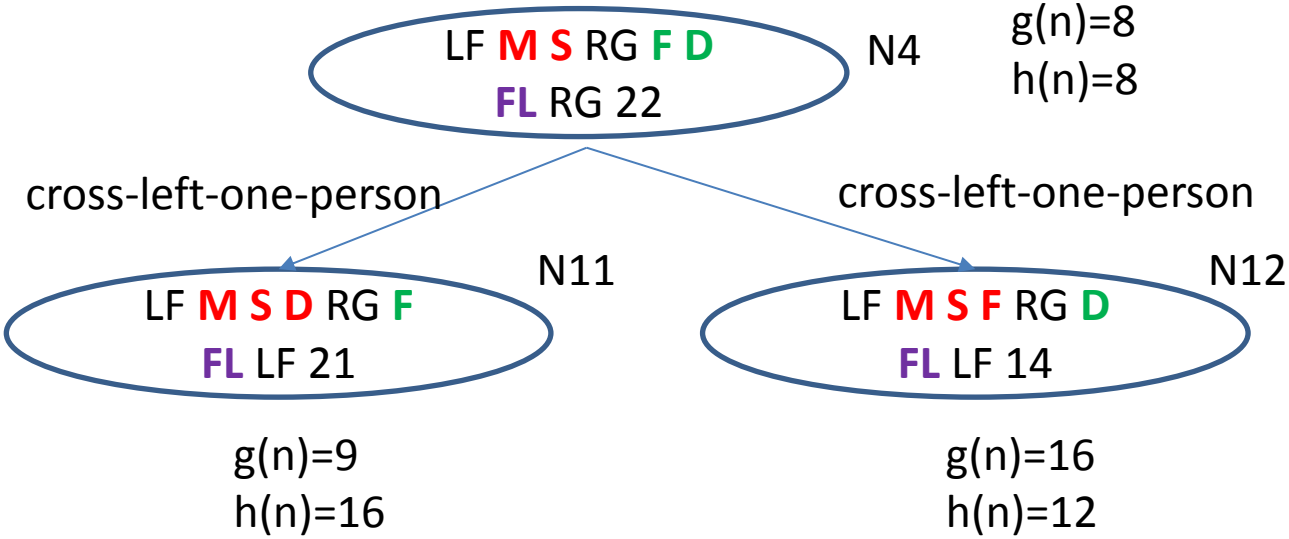
OPEN={N2(16), N4(16), N7(16), N3(18), N5(18), N6(18), N8(18)}

5. The family problem



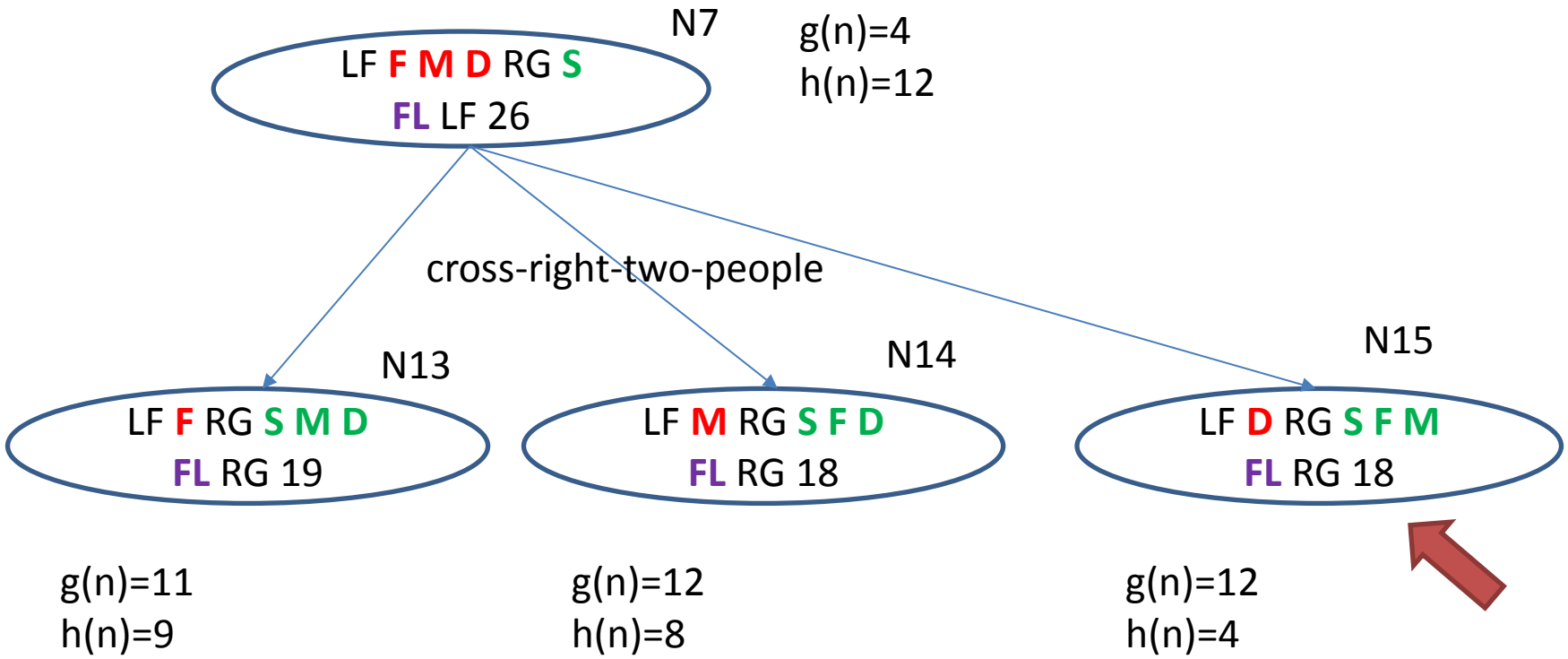
OPEN={N4(16), N7(16), N3(18), N5(18), N6(18), N8(18), N9(24), N10(26)}

5. The family problem



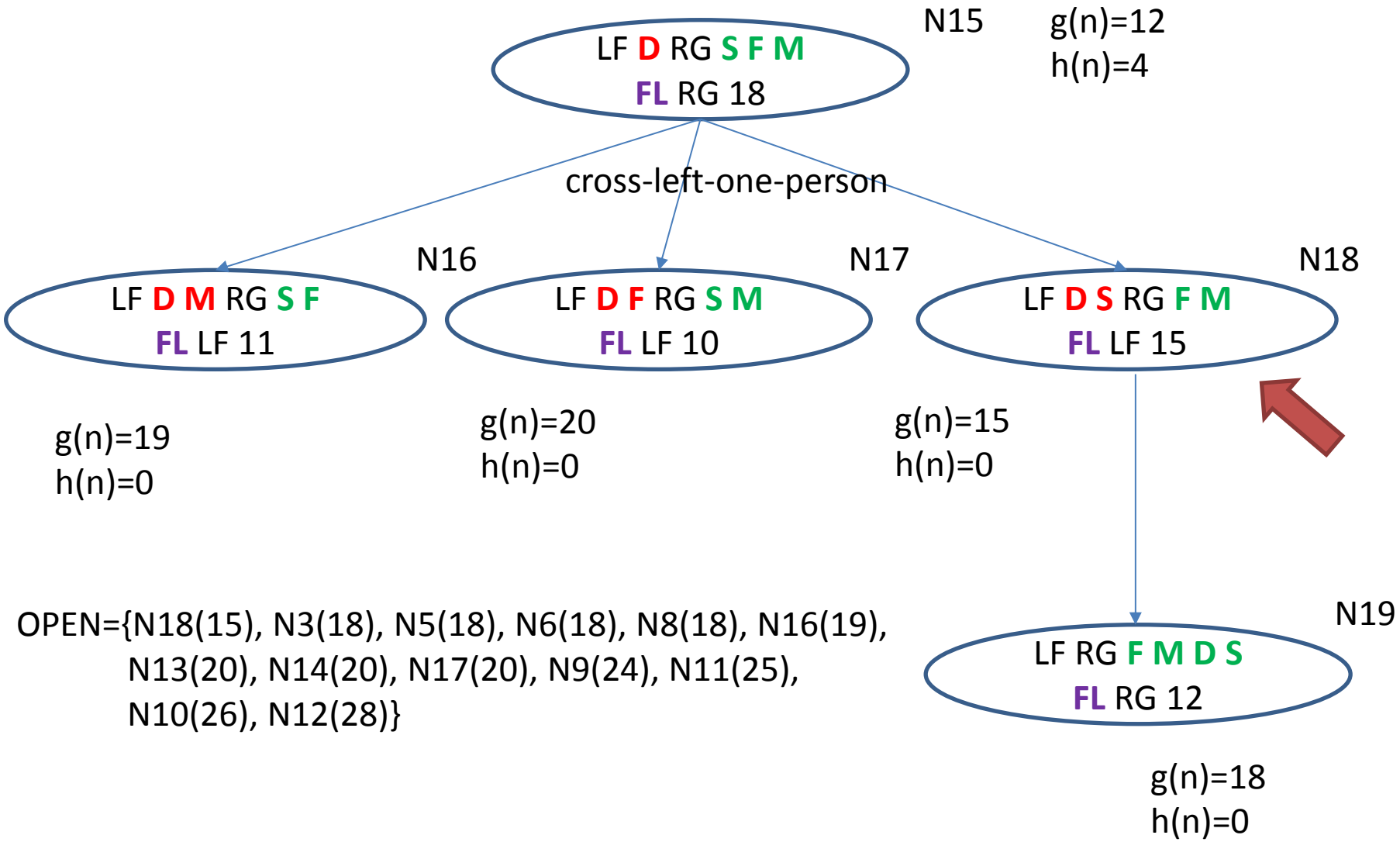
OPEN={N7(16), N3(18), N5(18), N6(18), N8(18), N9(24), N11(25), N10(26), N12(28)}

5. The family problem



OPEN={N15(16), N3(18), N5(18), N6(18), N8(18), N13(20), N14(20),
N9(24), N11(25), N10(26), N12(28)}

5. The family problem



OPEN={N18(15), N3(18), N5(18), N6(18), N8(18), N16(19),
N13(20), N14(20), N17(20), N9(24), N11(25),
N10(26), N12(28)}

5. The family problem

The current OPEN list is:

OPEN={ N3(18), N5(18), N6(18), N8(18), N19(18), N16(19), N13(20), N14(20),
N17(20), N9(24), N11(25), N10(26), N12(28)}

Next, we would expand node, N3, with $f(N3)=18$.

The expansion of nodes N3, N5, N6 and N8 would generate successors with values $f(n) > 18$

Next, we expand N19 and we check this is a solution node ➔ END

5. The family problem

The solution found is optimal

The function h(n) is not admissible. Let's see a counterexample:

Optimal solution from N9:

- Father and Daughter cross right: cost=8
- Daughter crosses left: cost=1
- Daughter and son cross right: cost=3

Therefore, $h^*(n)=8+1+3=12 < h(n)$