EARIN

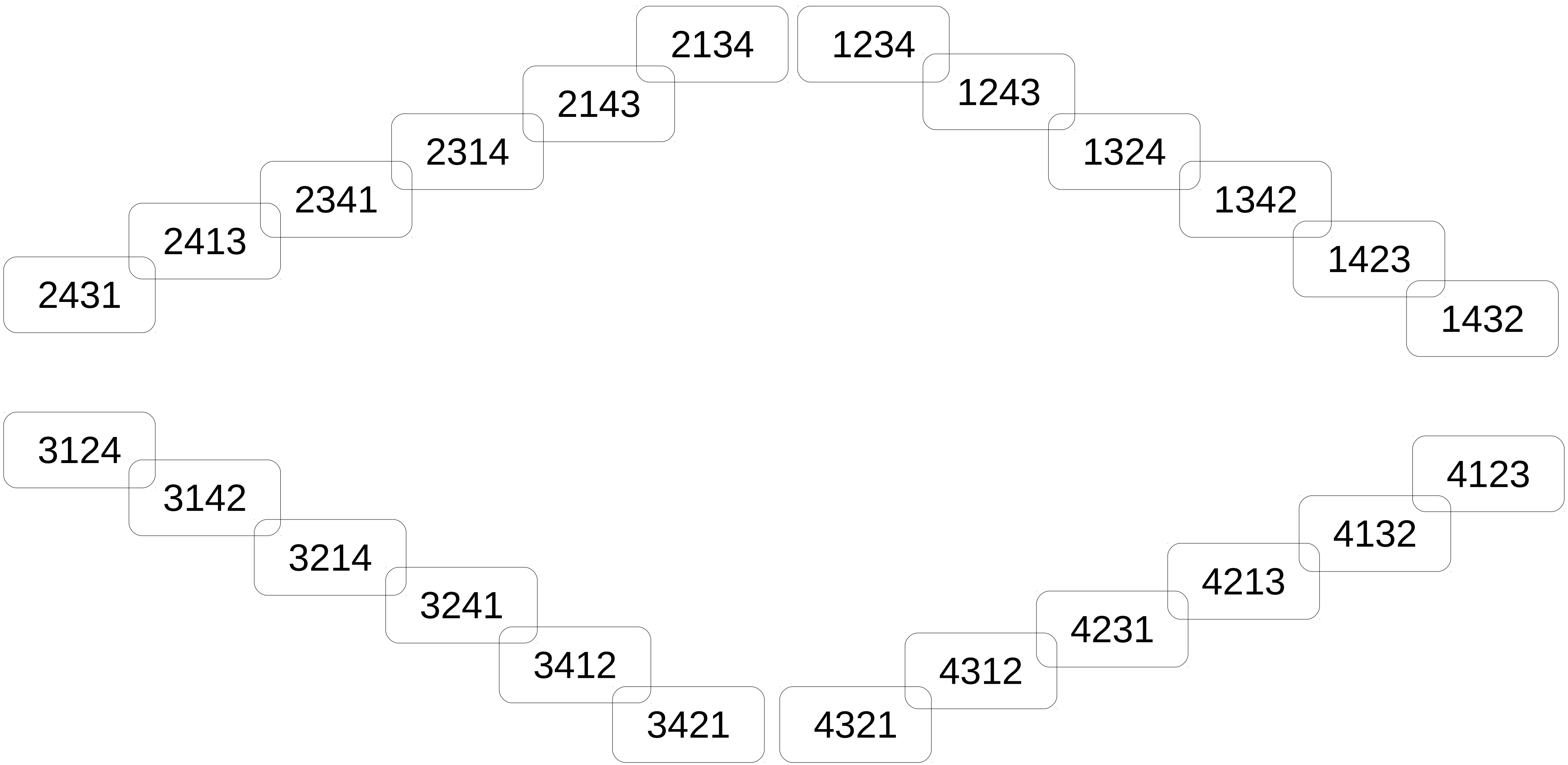
Jarosław Arabas

Problem solving by searching

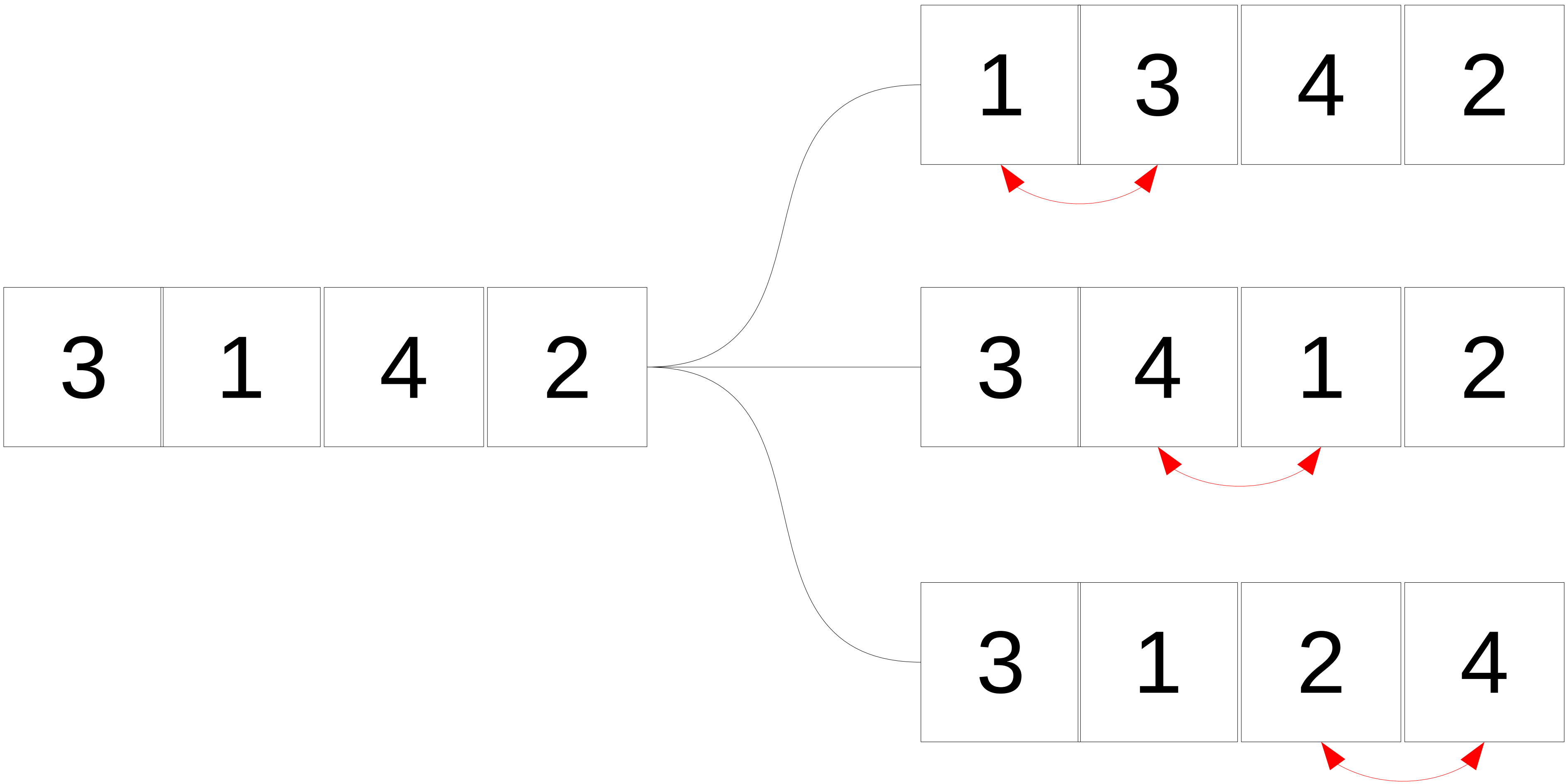
Sort a table of integers

```
void sort (int *tab, unsigned n){
  int i=1,j=0,sorted=0;
  do{
     sorted=1;
     for(j=0;j<n-i;++j){
        if (tab[j+1]<tab[j]){
           int temp=tab[j];
           tab[j]=tab[j+1];tab[j+1]=temp;
           sorted=0;}
     ++i;
  }while(i<n && !sorted);
}
```

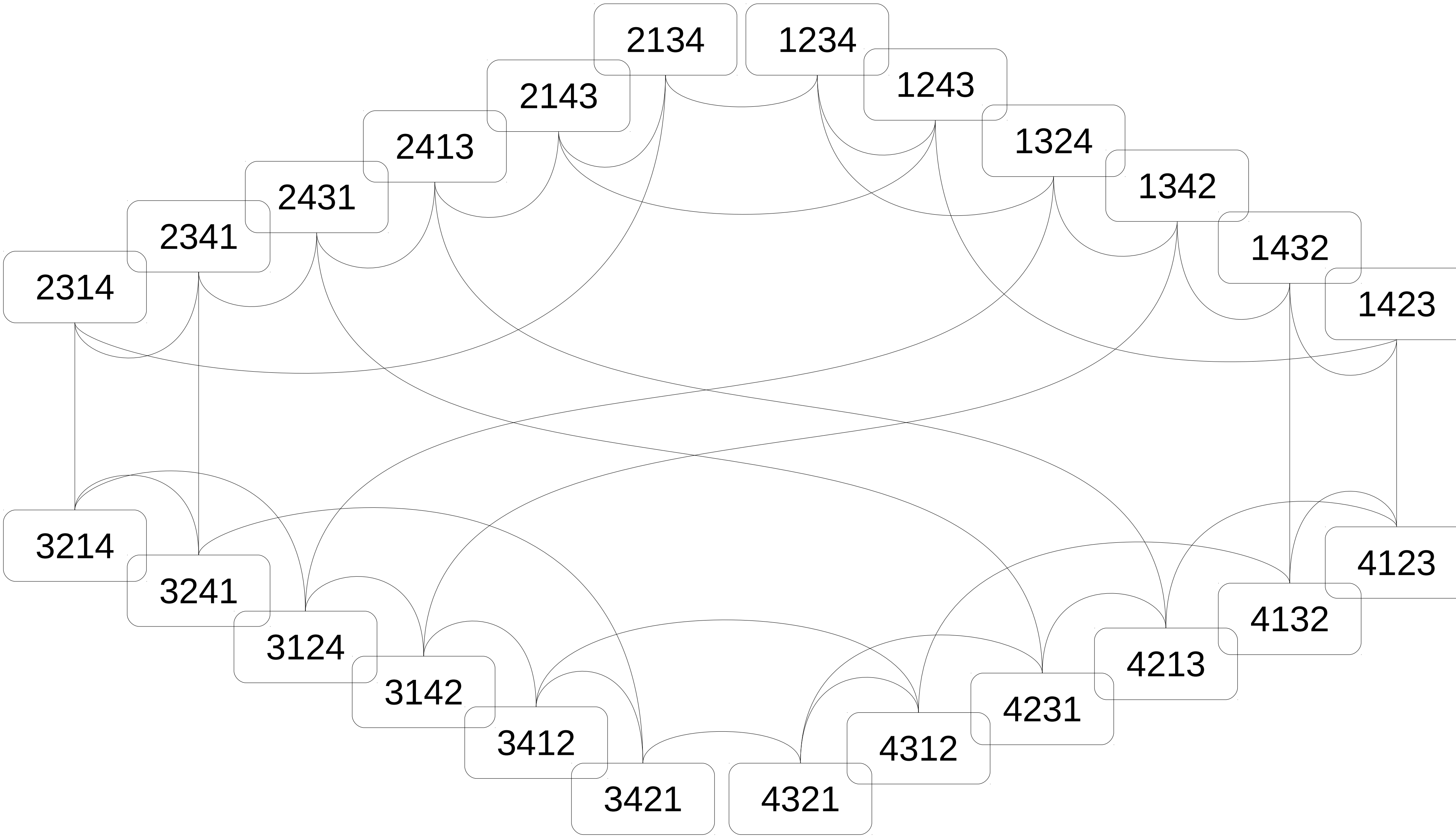
Set of possible table orderings



Relations between soluions

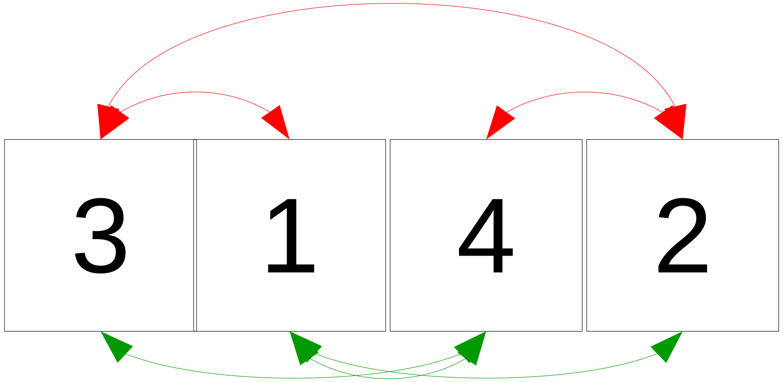


Search space



Objective function

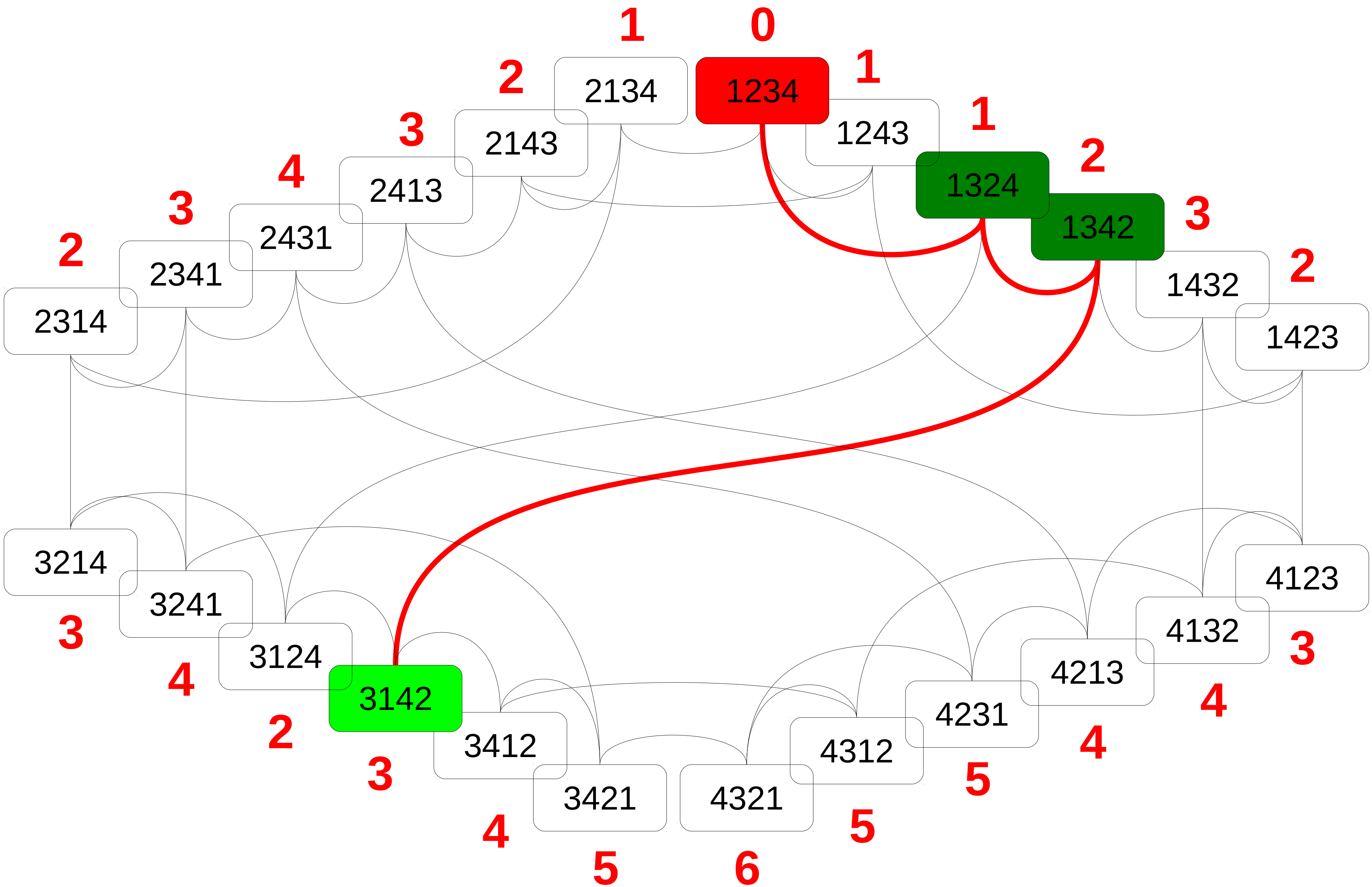
- Measure of the solutions quality



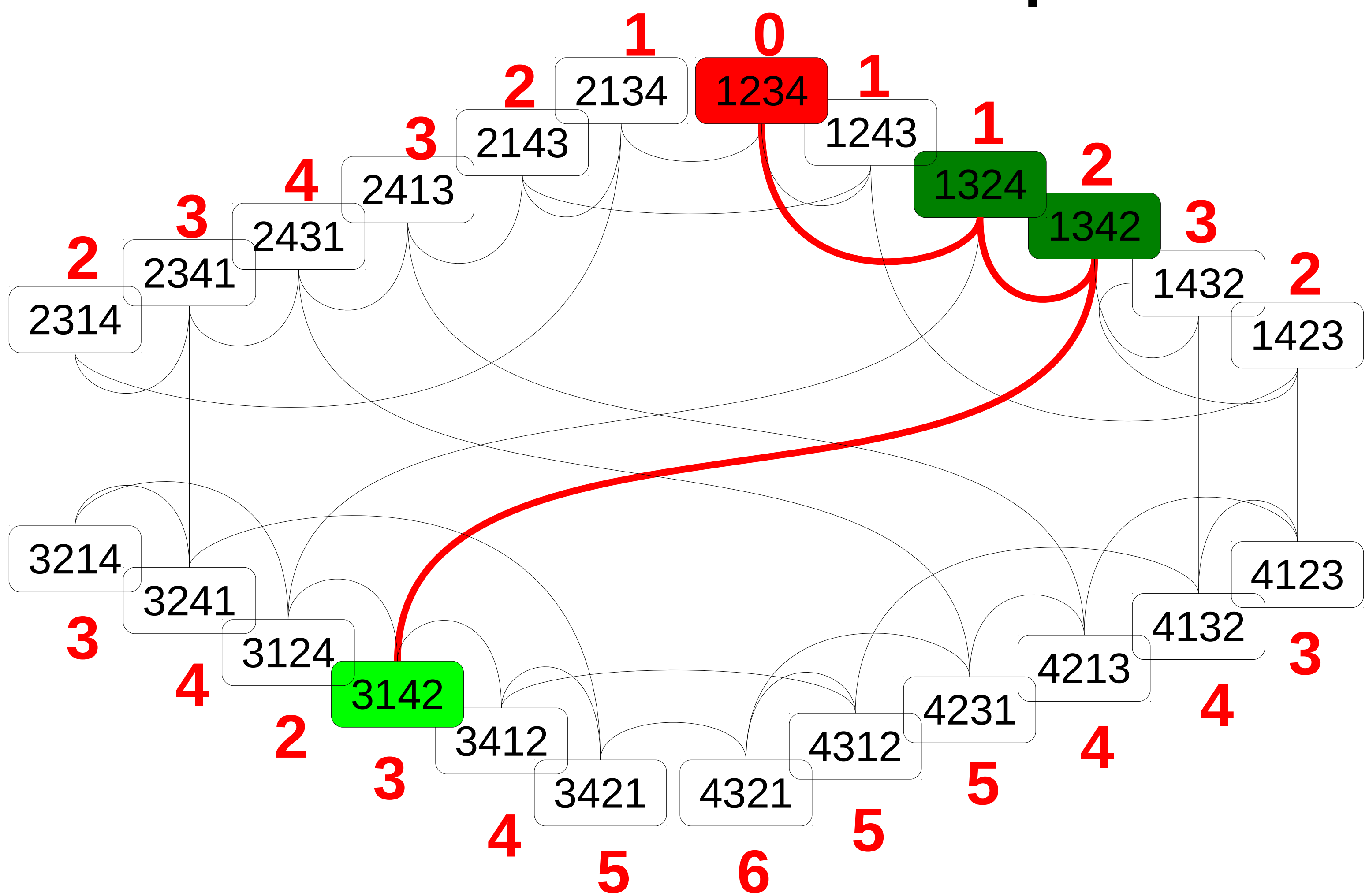
3 improperly ordered pairs

3 properly ordered pairs

Example path in the search space



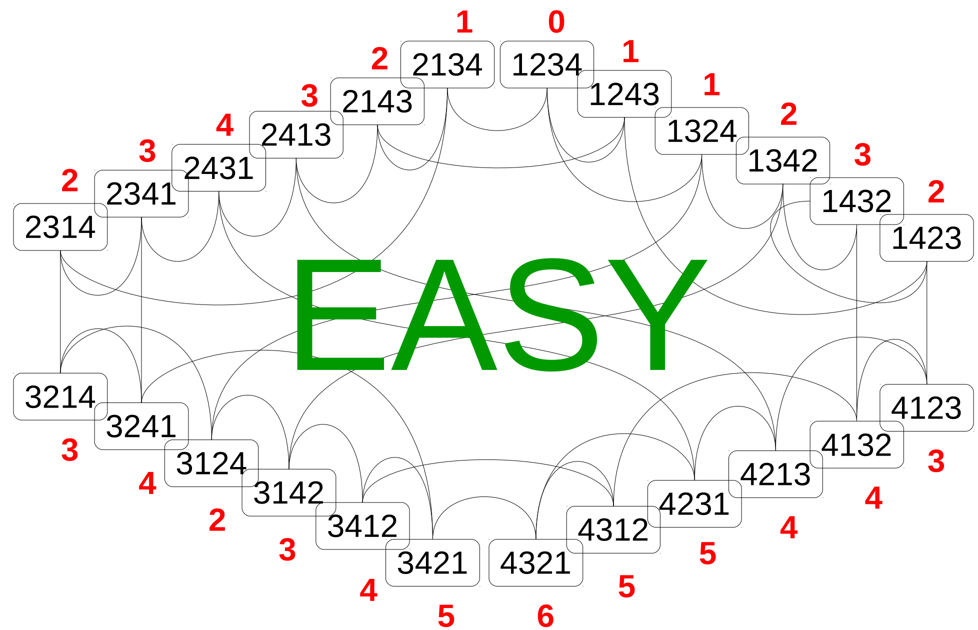
Path in the search space



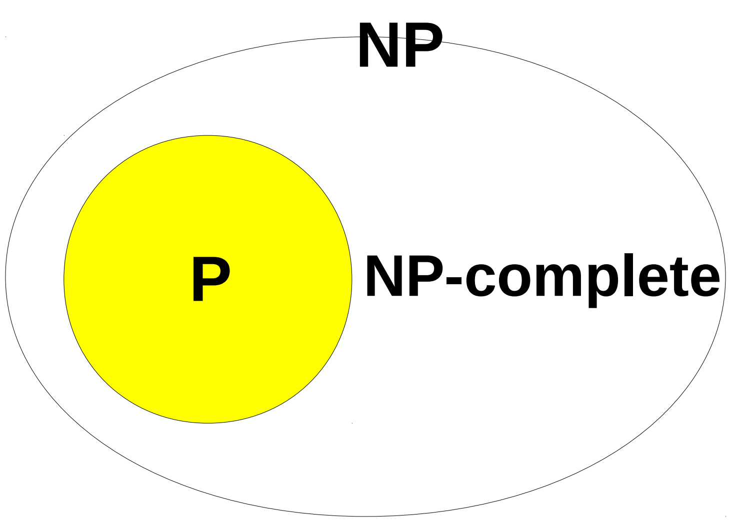
3142 → 1342 → 1324 → 1234

Search space serialization

“Easy” and “hard” problems

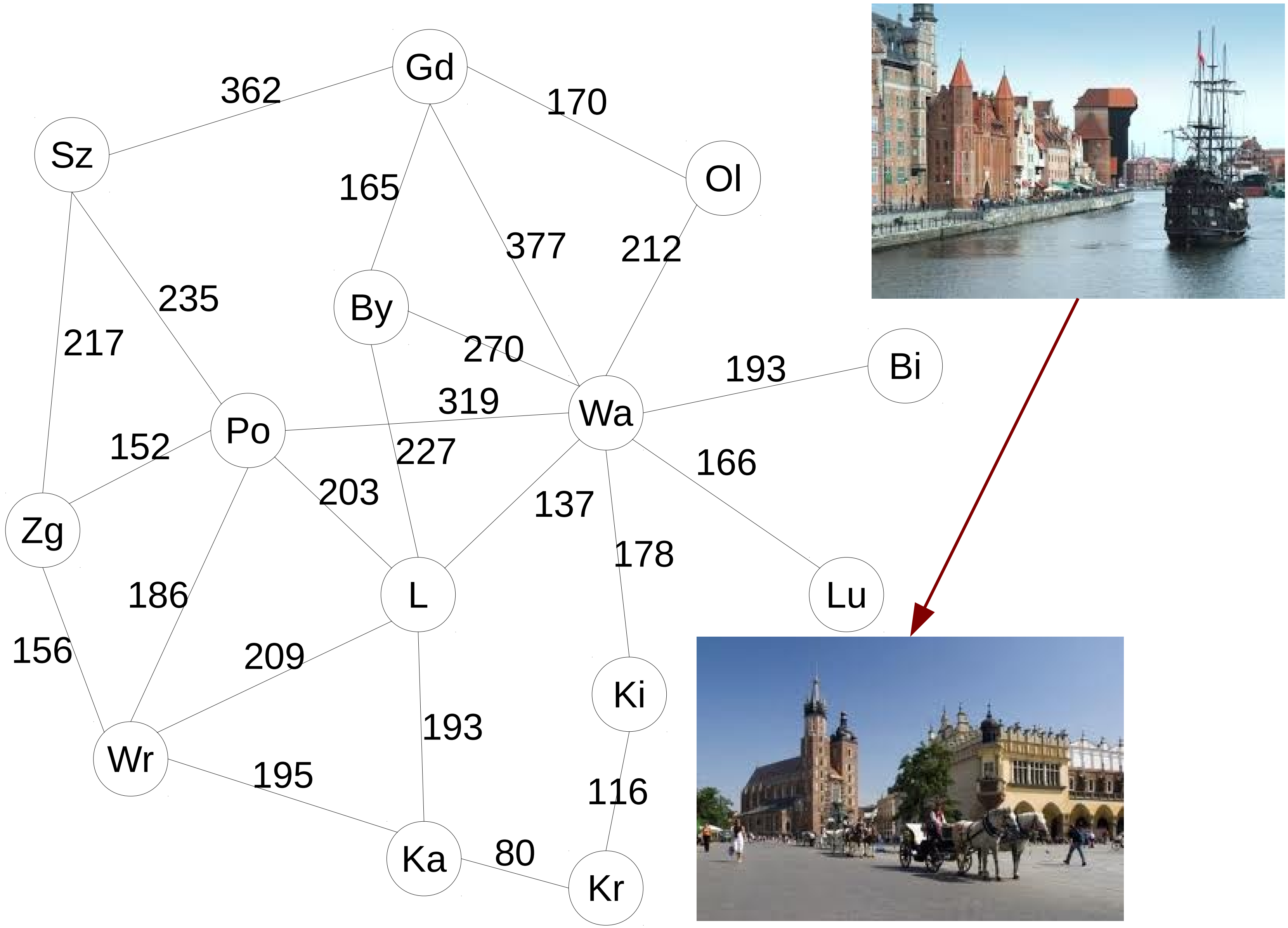


Hierarchy of decision problems

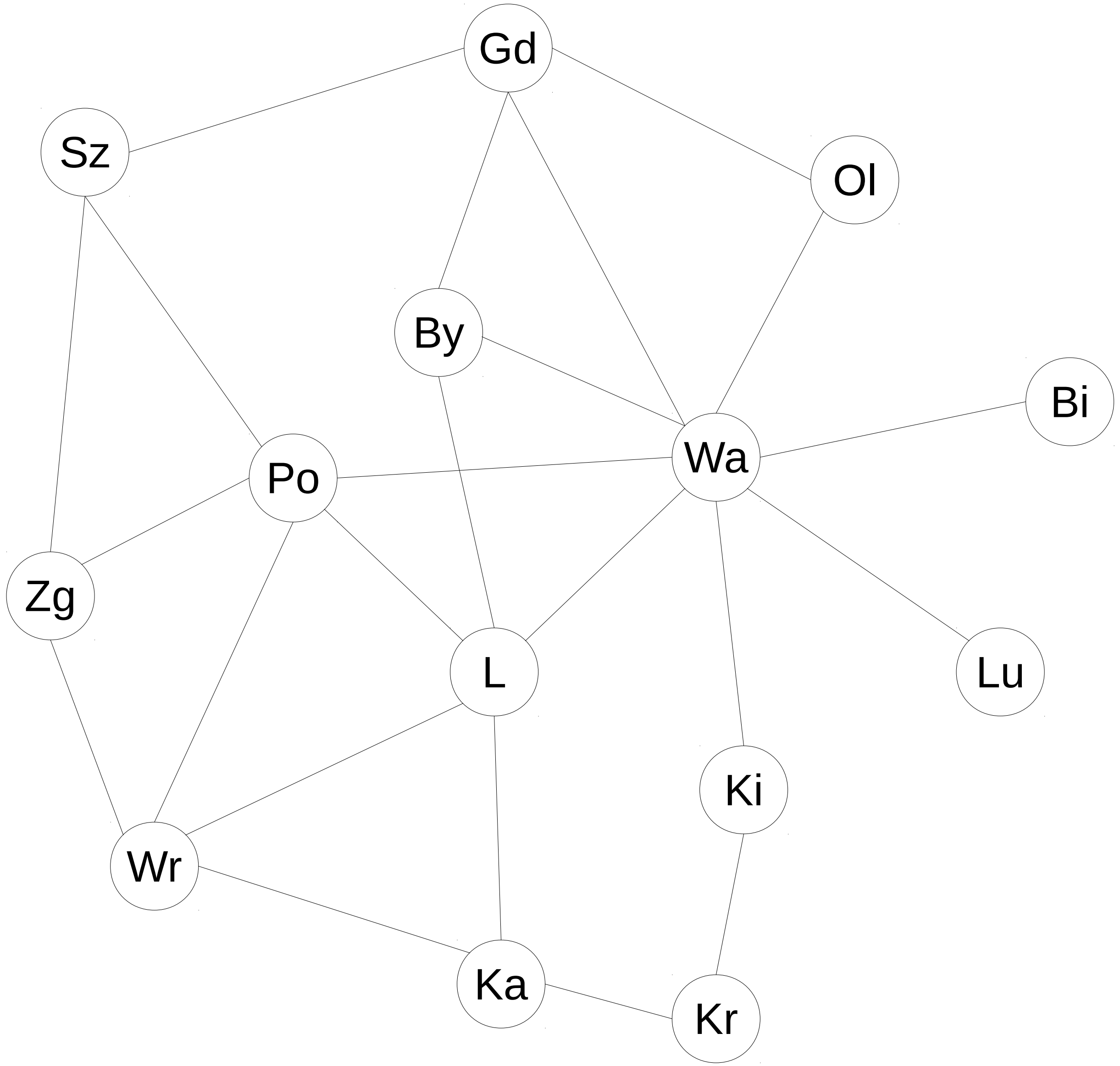


- Small neighborhoods (polynomial of n)
- Short paths (polynomial of n)
- Single optimum of the objective function

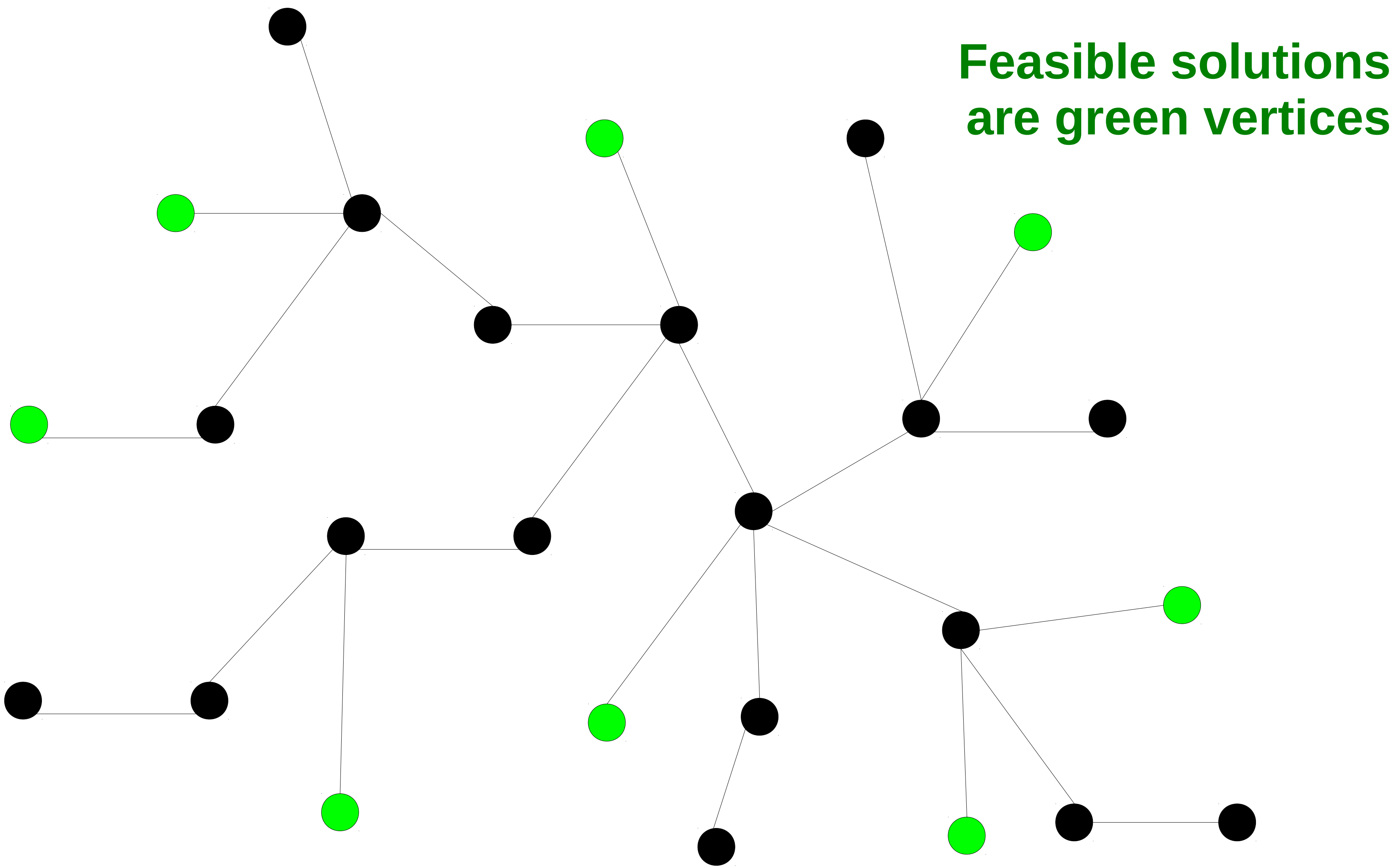
Example problem
shortest trip from Gd to Kr



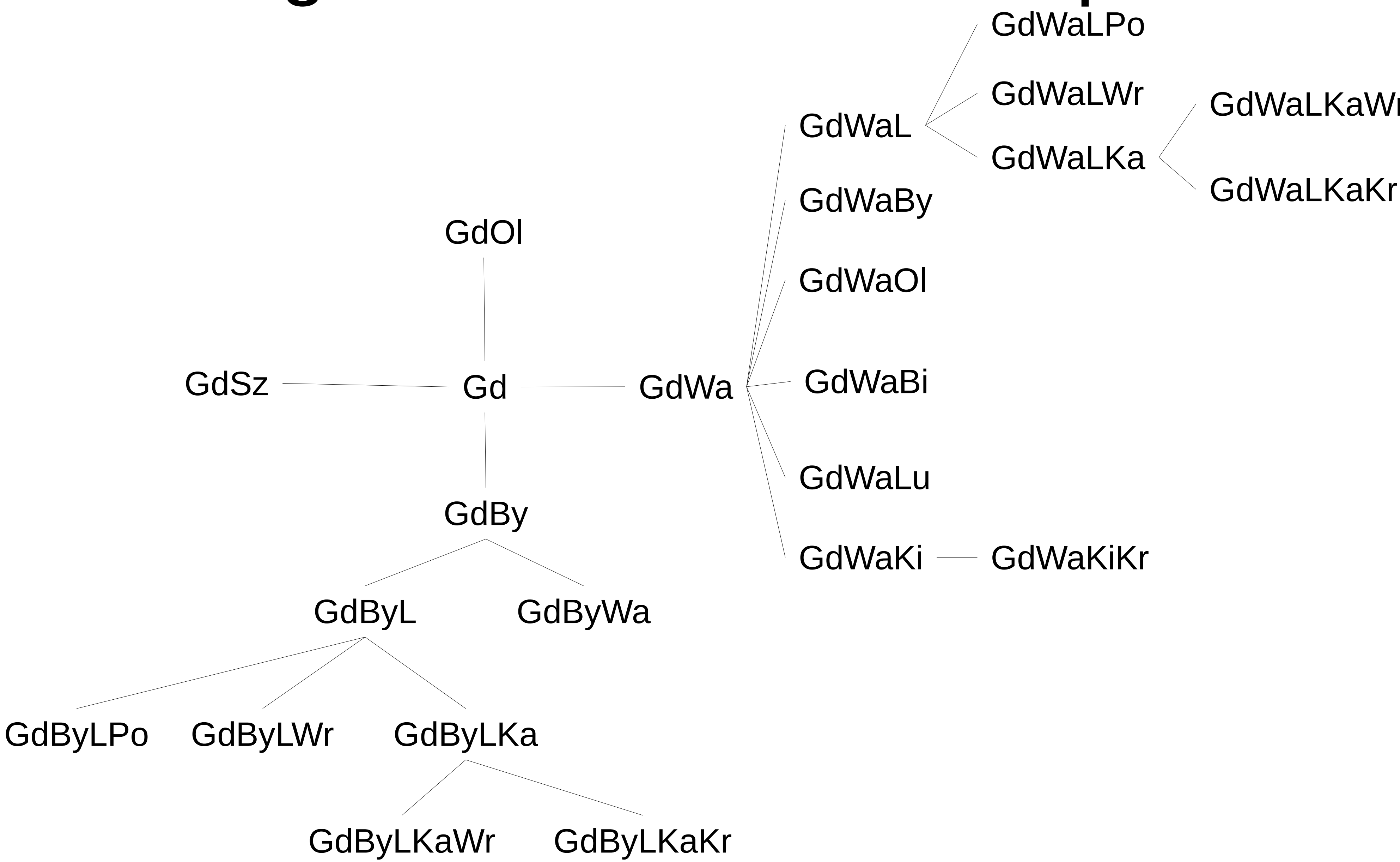
Uninformed search



Structure of the search space



Fragment of the search space



Breadth-first and depth-first search

procedure ***breadth first***

$A \leftarrow \{s_0\}$

while $A \neq \emptyset$

$\quad \boldsymbol{x} \leftarrow popFIFO(A)$

$\quad Y \leftarrow neighbors(\boldsymbol{x})$

$\quad A \leftarrow A \cup Y$

procedure ***depth first***

$A \leftarrow \{s_0\}$

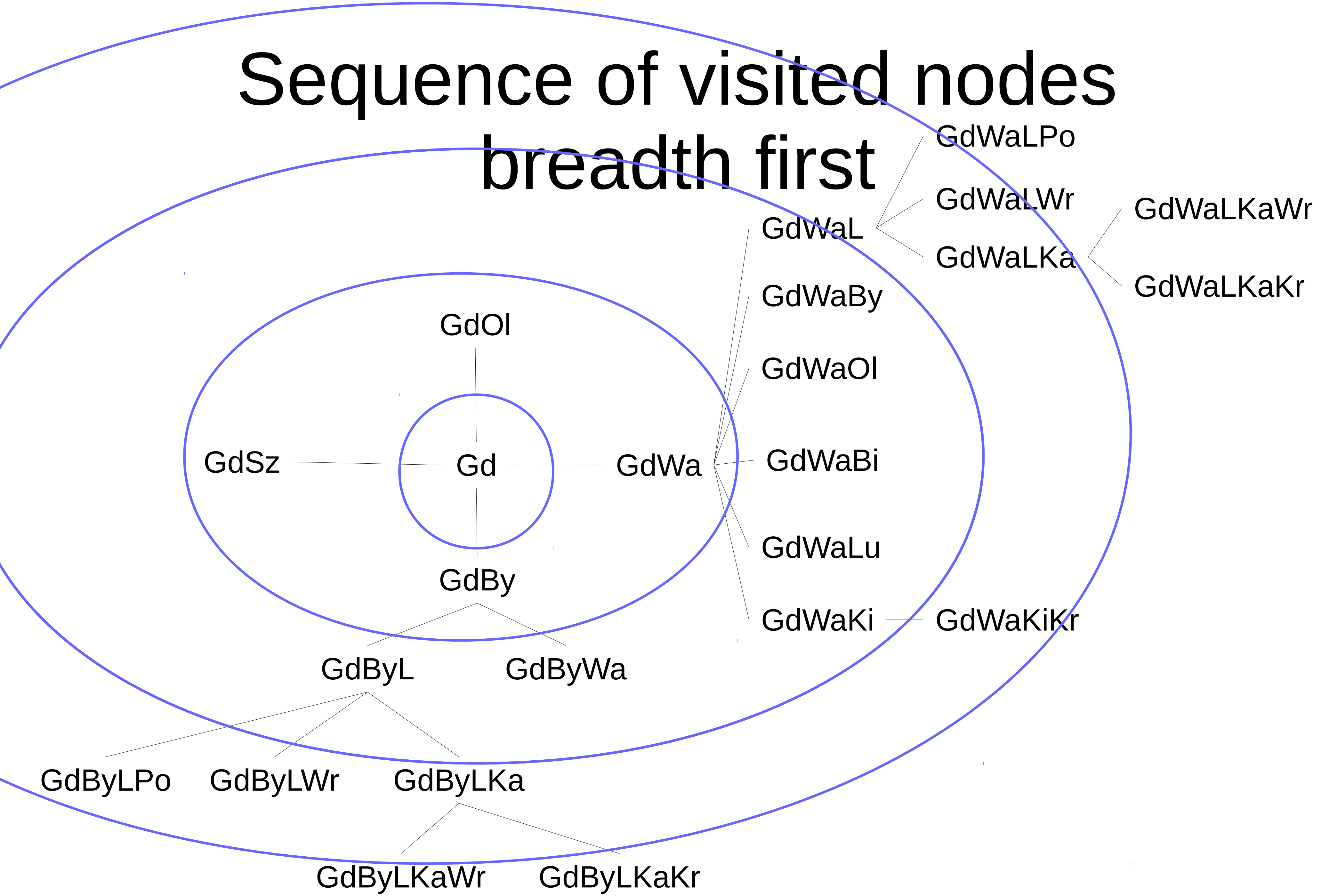
while $A \neq \emptyset$

$\quad \boldsymbol{x} \leftarrow popLIFO(A)$

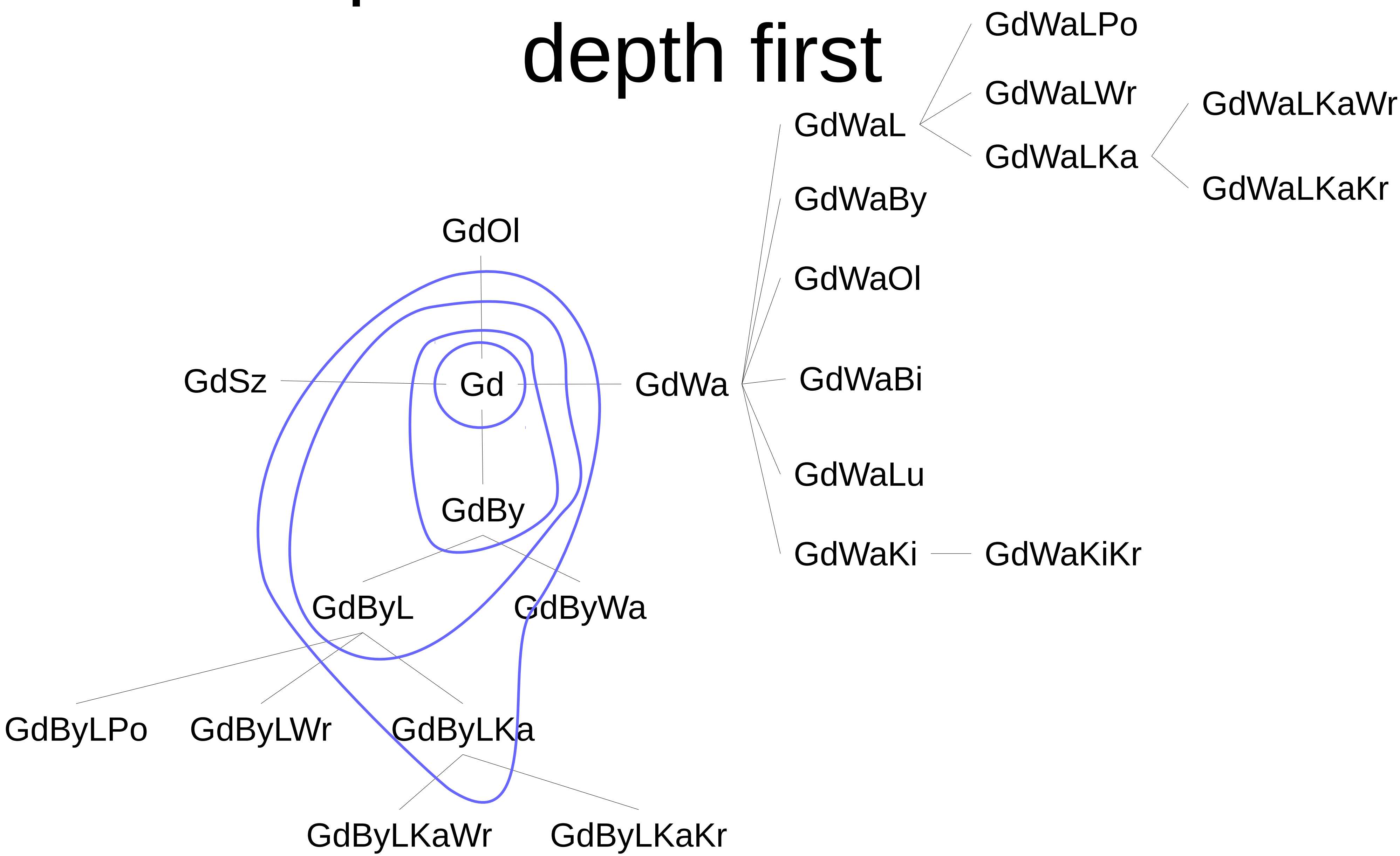
$\quad Y \leftarrow neighbors(\boldsymbol{x})$

$\quad A \leftarrow A \cup Y$

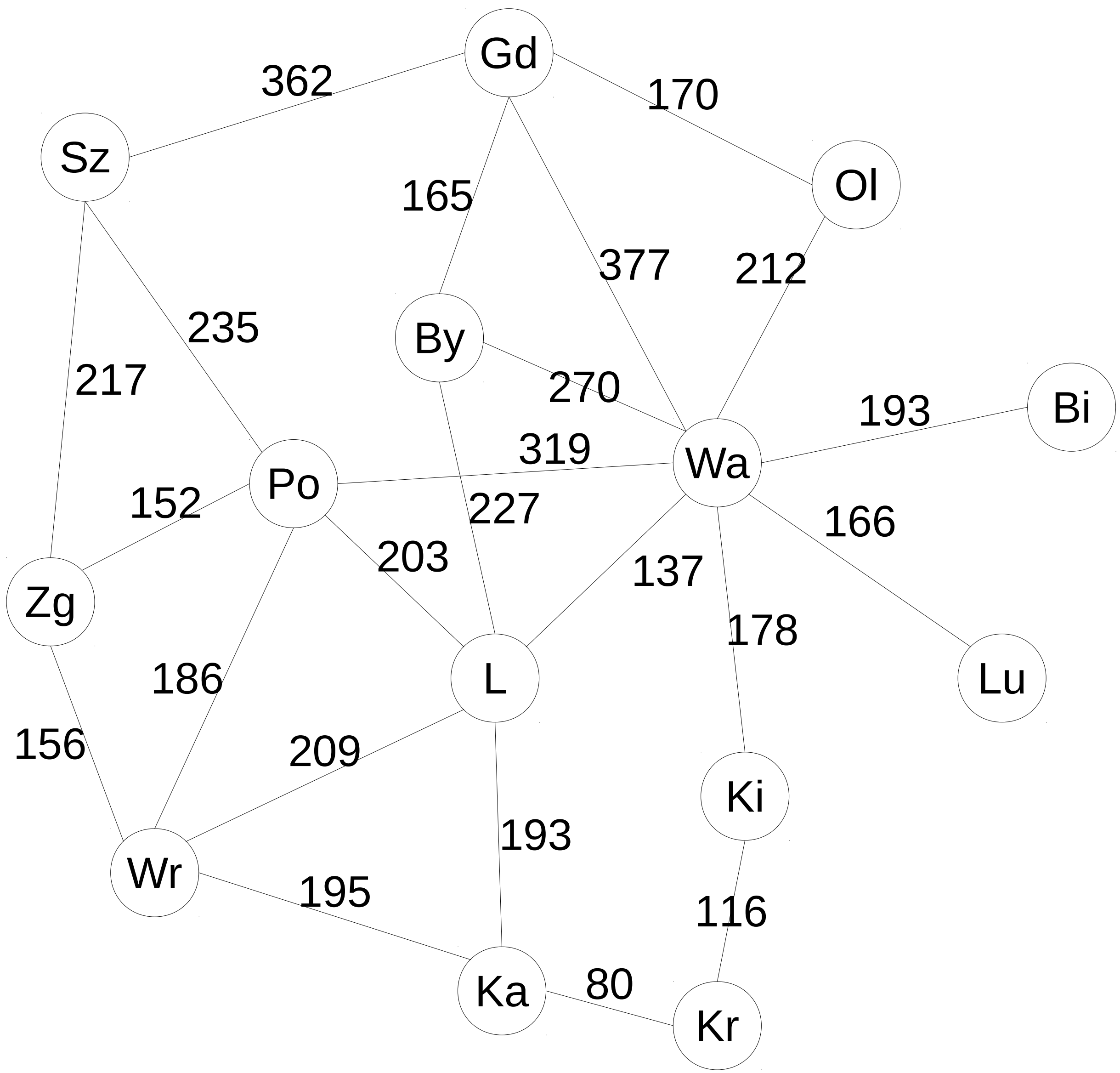
Sequence of visited nodes breadth first



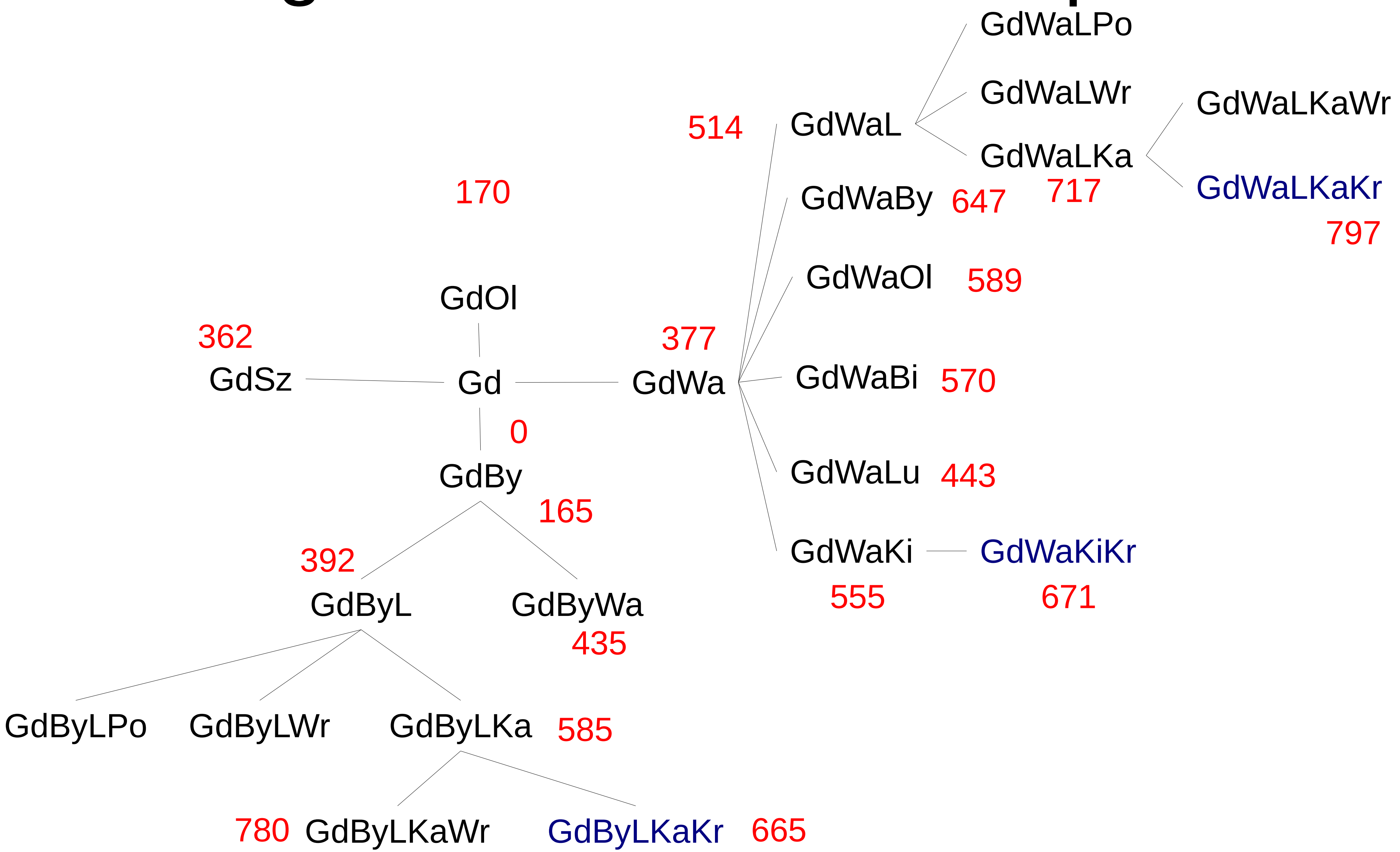
Sequence of visited nodes depth first



Informed search



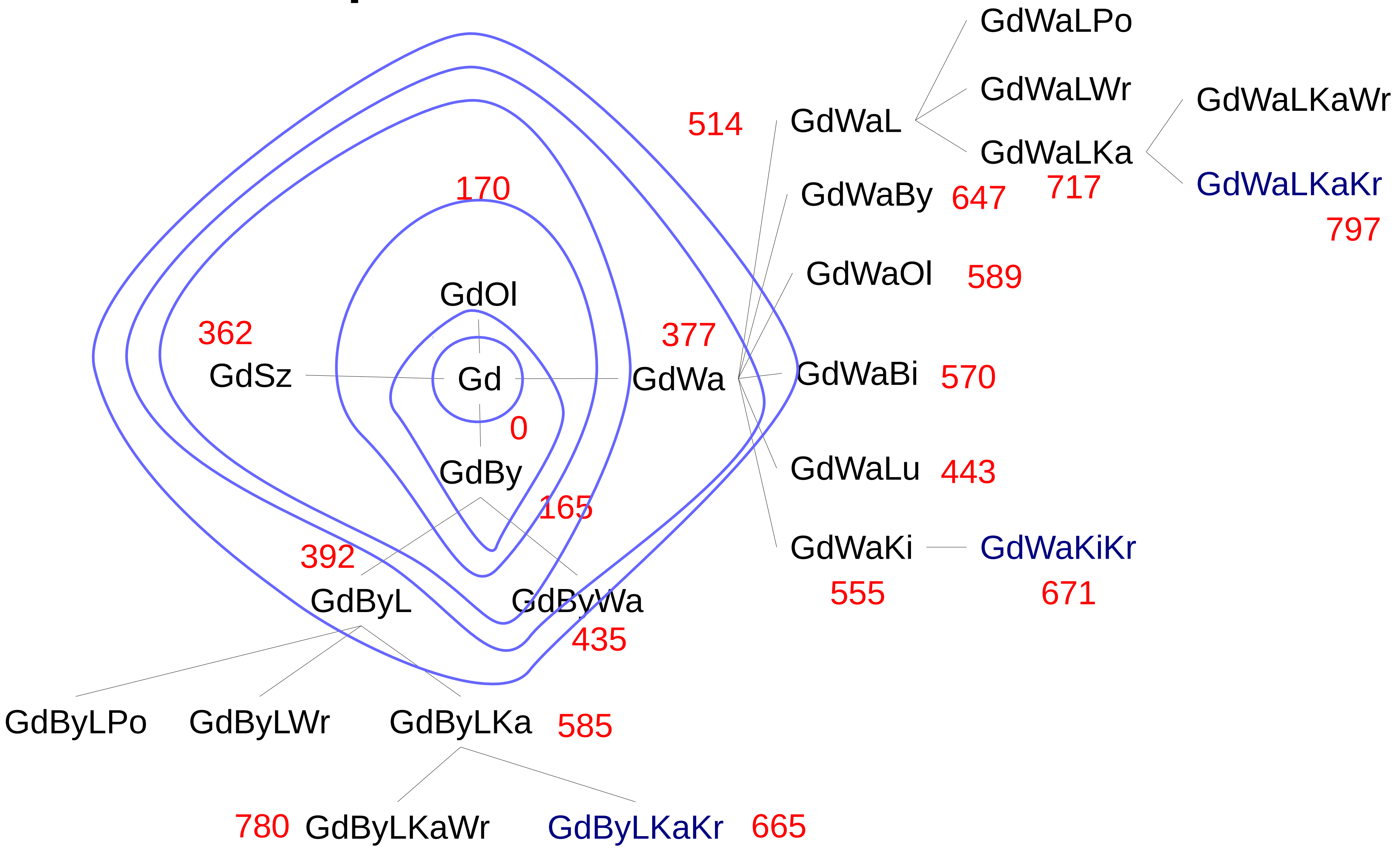
Fragment of the search space

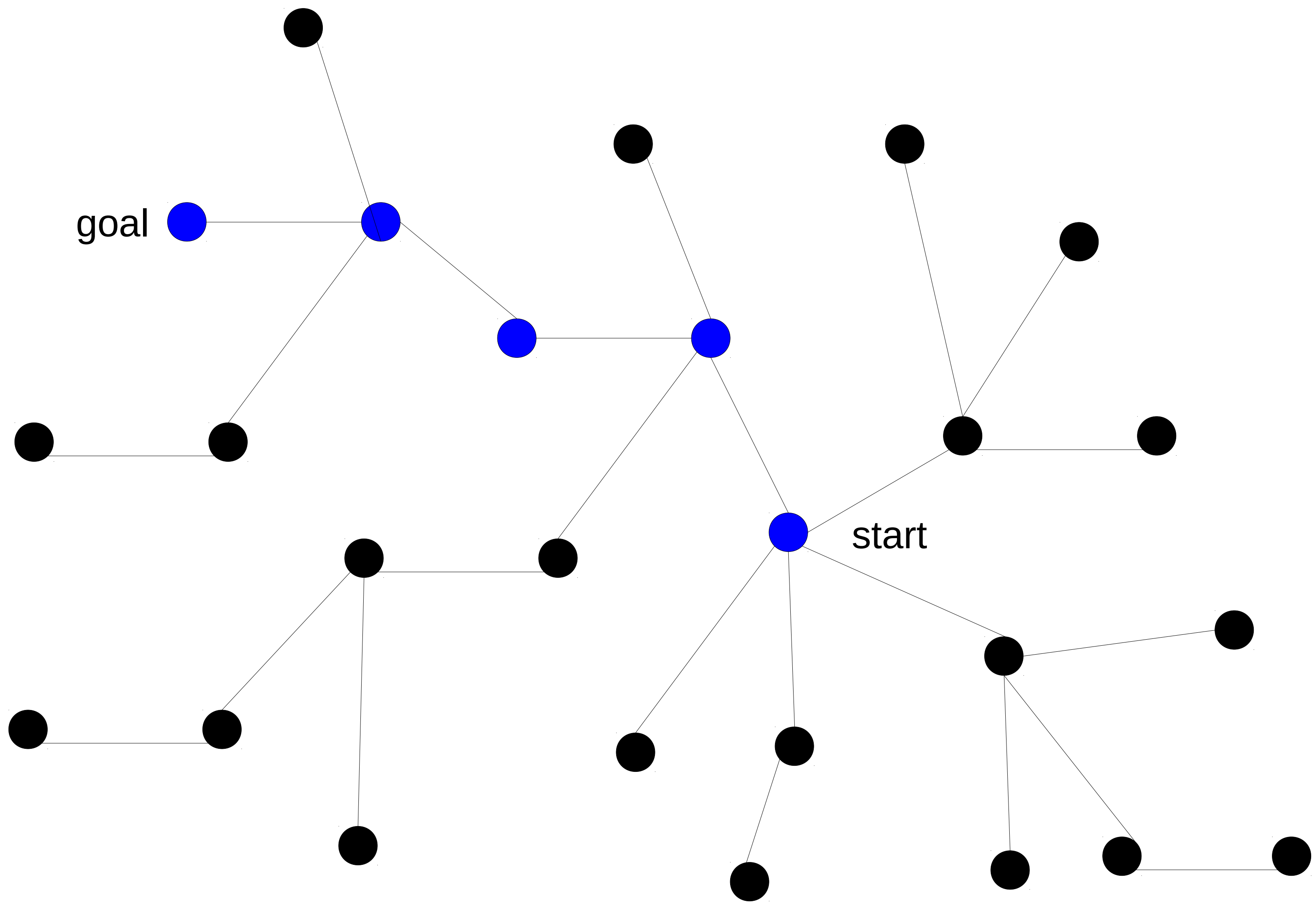


Best-first search

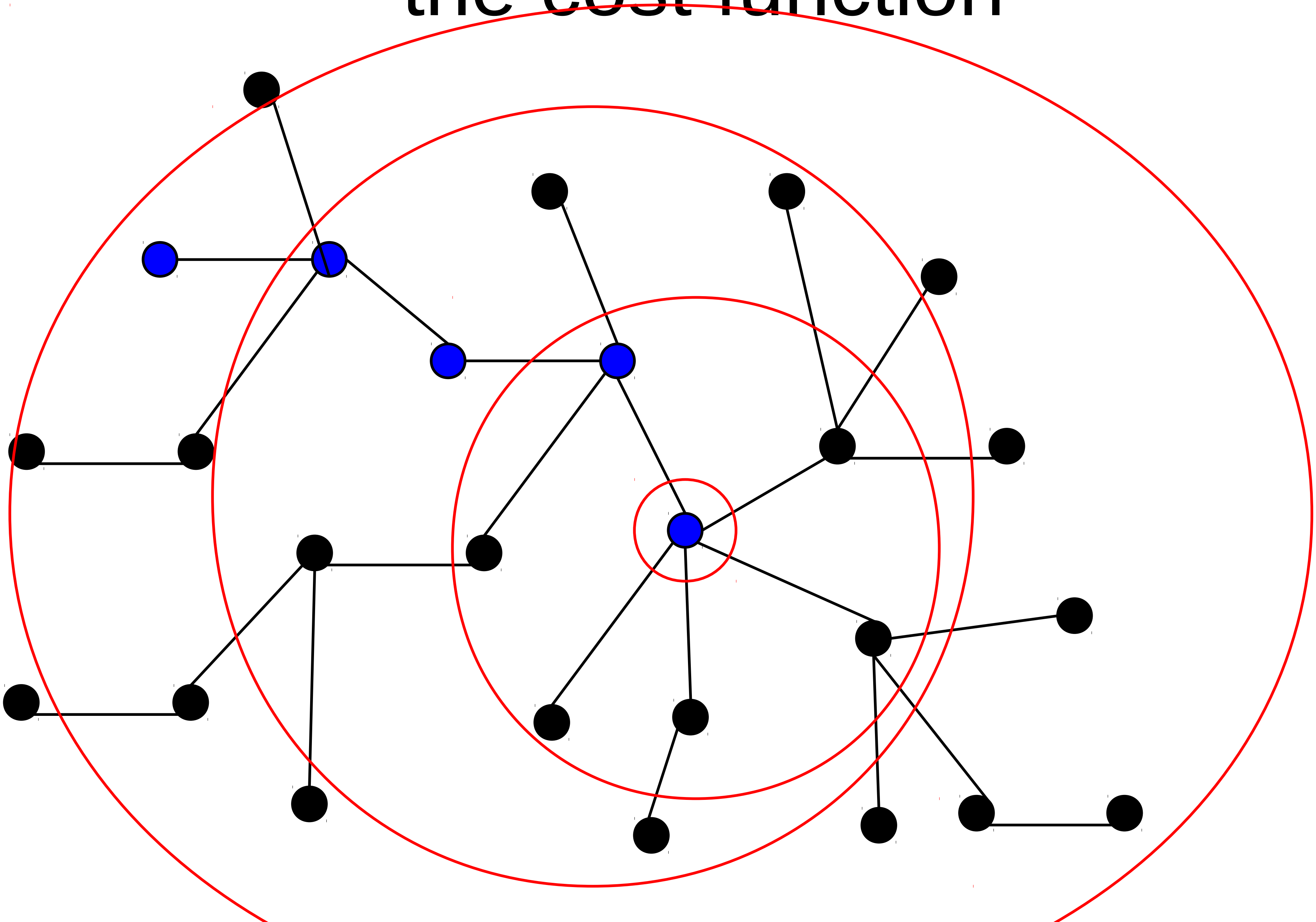
$$\textit{procedure}\ \boldsymbol{best\ first}$$
$$A \leftarrow \{s_0\}$$
$$\boldsymbol{while}\ A \neq \emptyset$$
$$\quad \boldsymbol{x} \leftarrow popProrityList(A)$$
$$\quad Y \leftarrow neighbors(\boldsymbol{x})$$
$$\quad A \leftarrow A \cup Y$$

Sequence of visited nodes

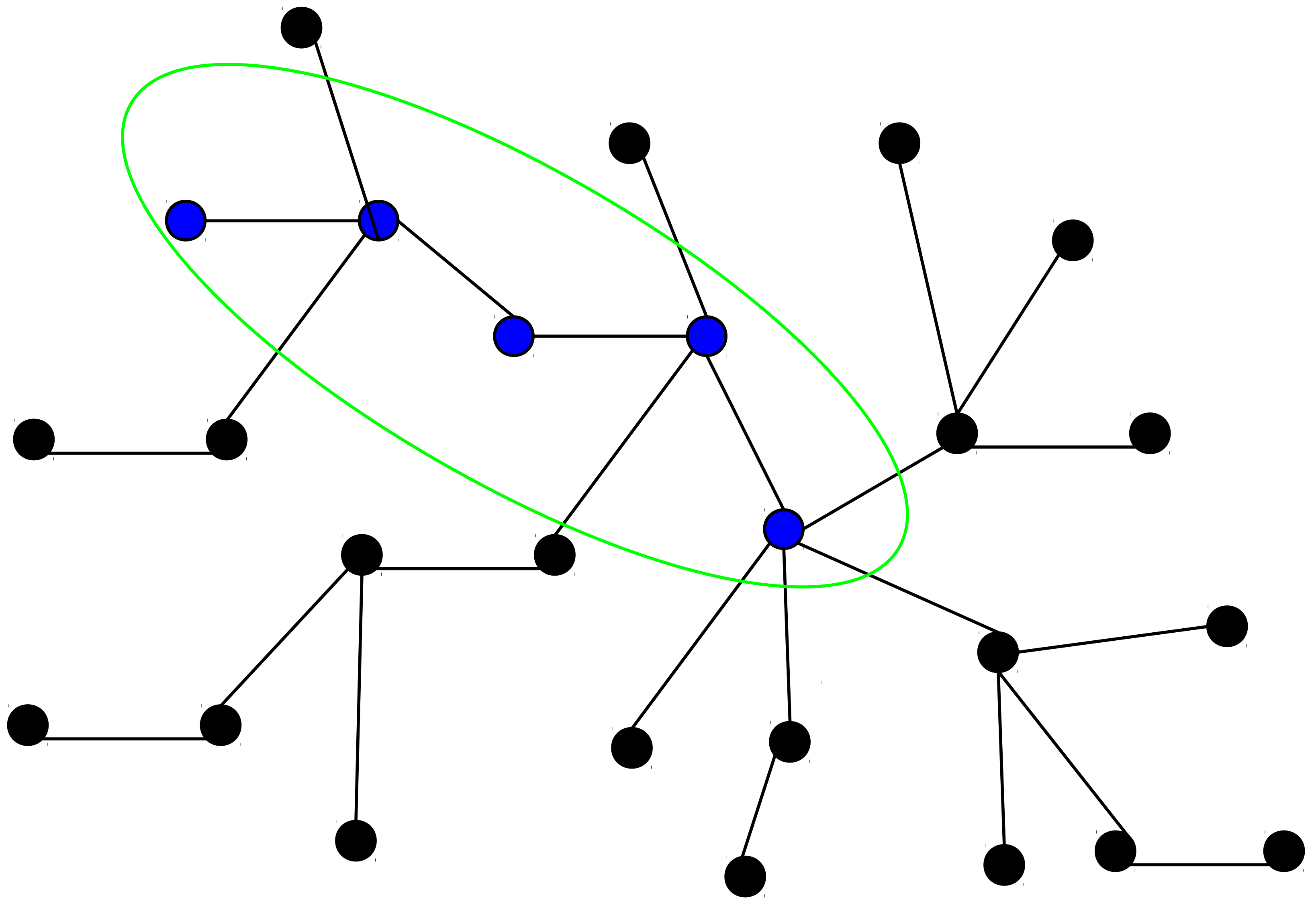



goal
start

Best first based on the cost function



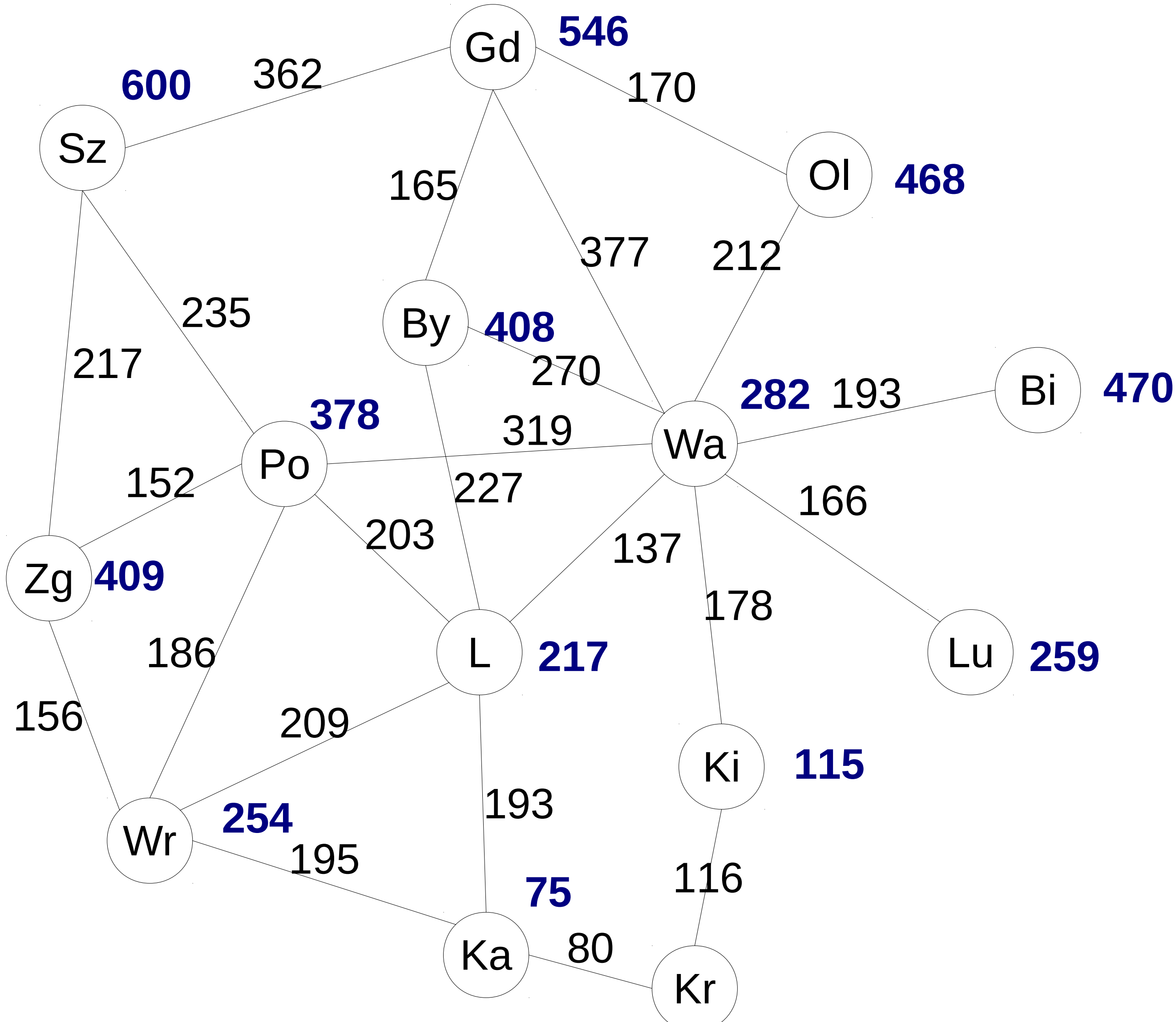
Best first based on an oracle



WANTED

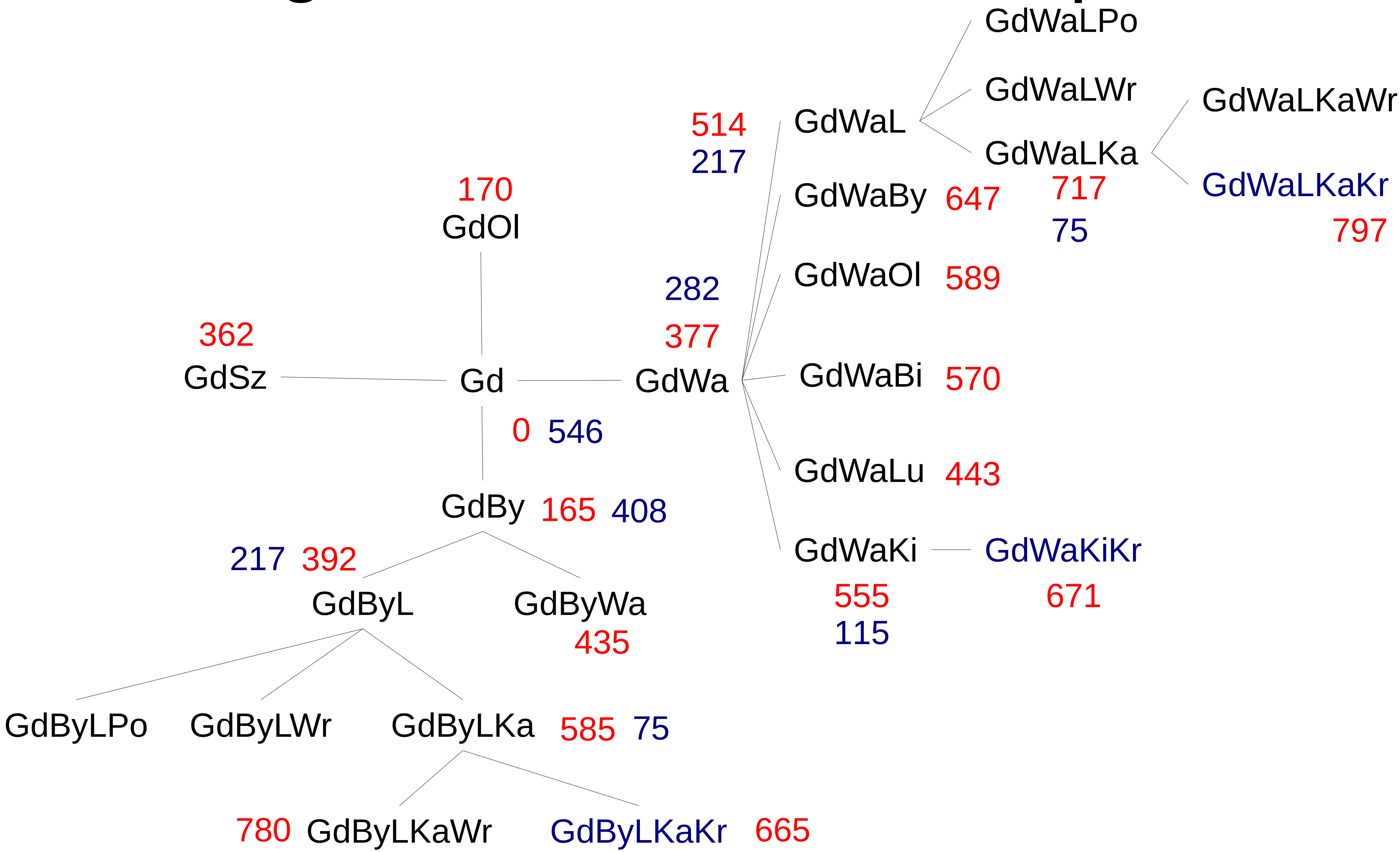
an oracle

Distances and their estimates

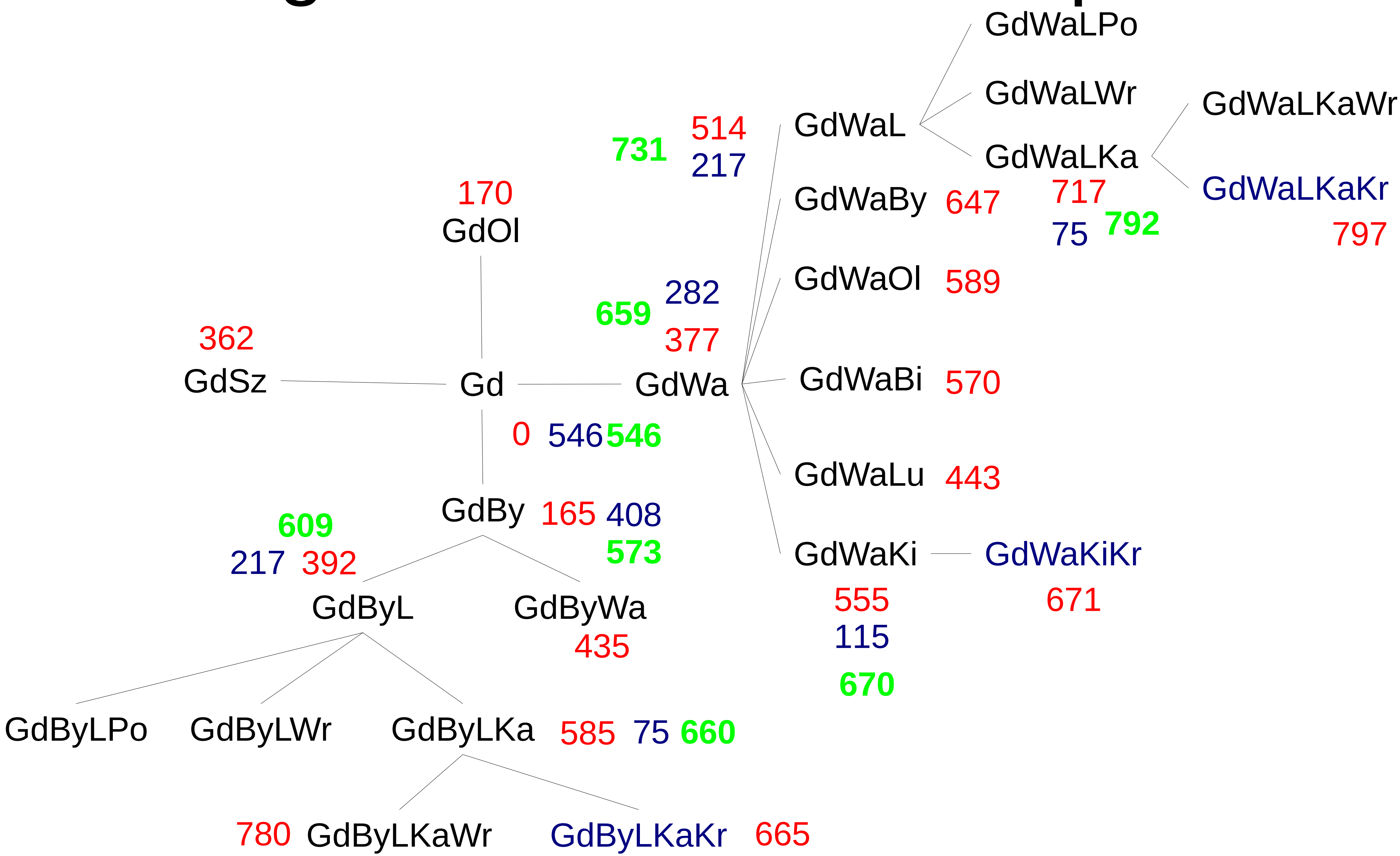


Straight line distances to Kraków

Fragment of the search space



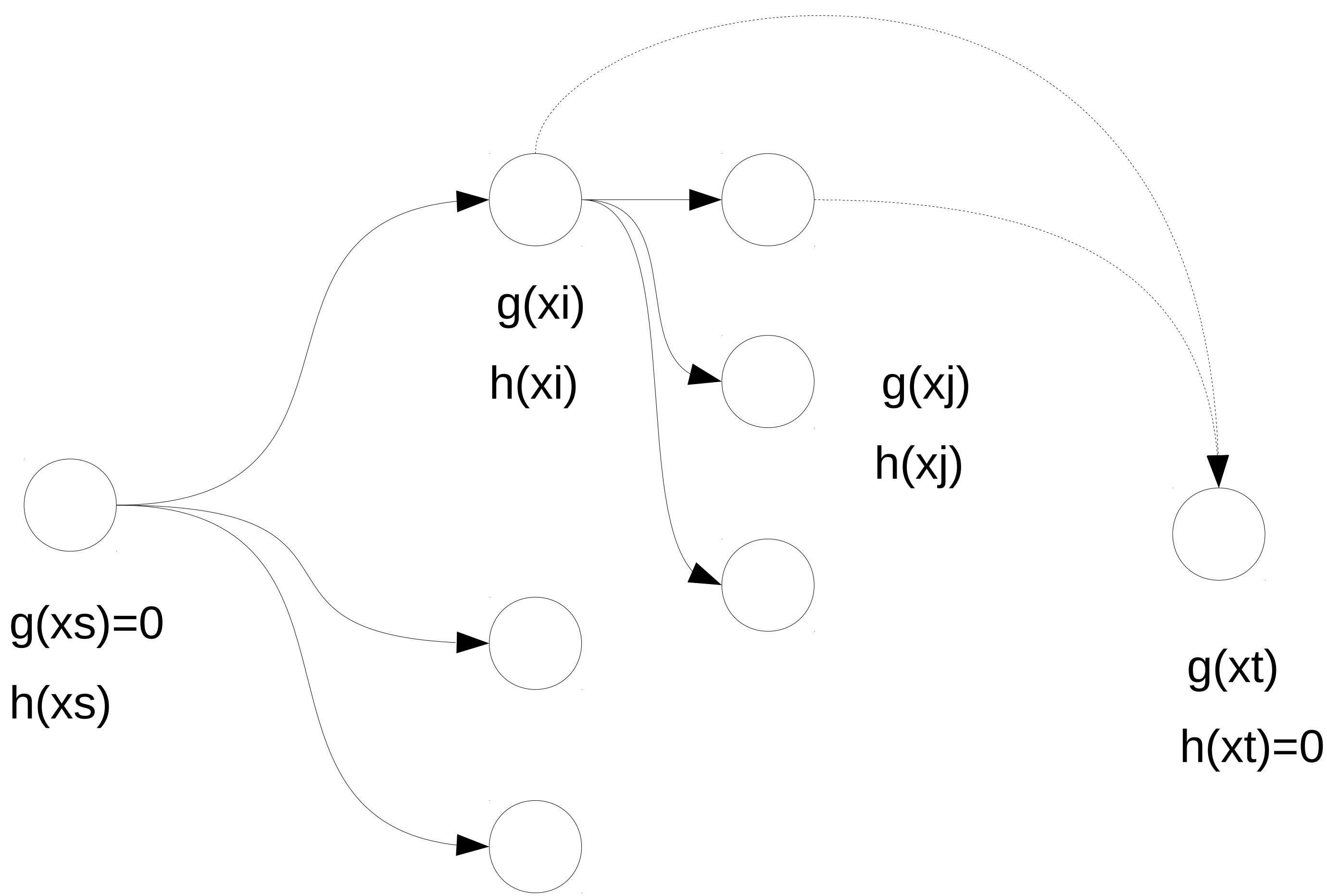
Fragment of the search space



Heuristic function

- Tree structured search space
- Objective function for intermediate nodes needed
- Cost for the intermediate node is known
- Heuristic function estimates how much the cost will increase after the best accessible terminal node is attained

Heuristic function (for minimization)



Optimistic (lower bound of the cost increase)

admissible: $g(x)+h(x)<=g(xt)$

Error of the estimate never increases while approaching to the terminal node

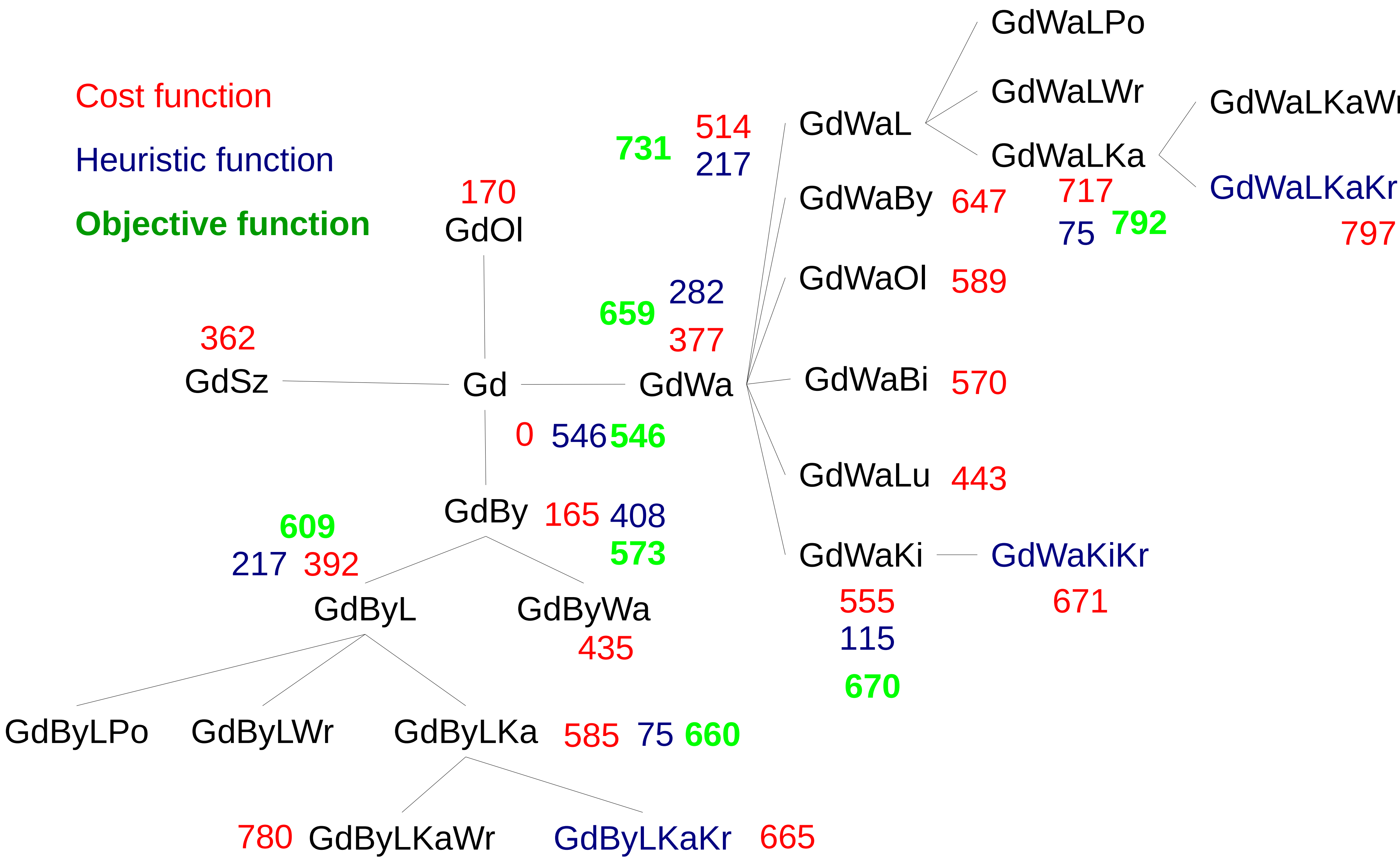
monotonous: $g(xj)+h(xj)>=g(xi)+h(xi)$

Heuristic function (for minimization)

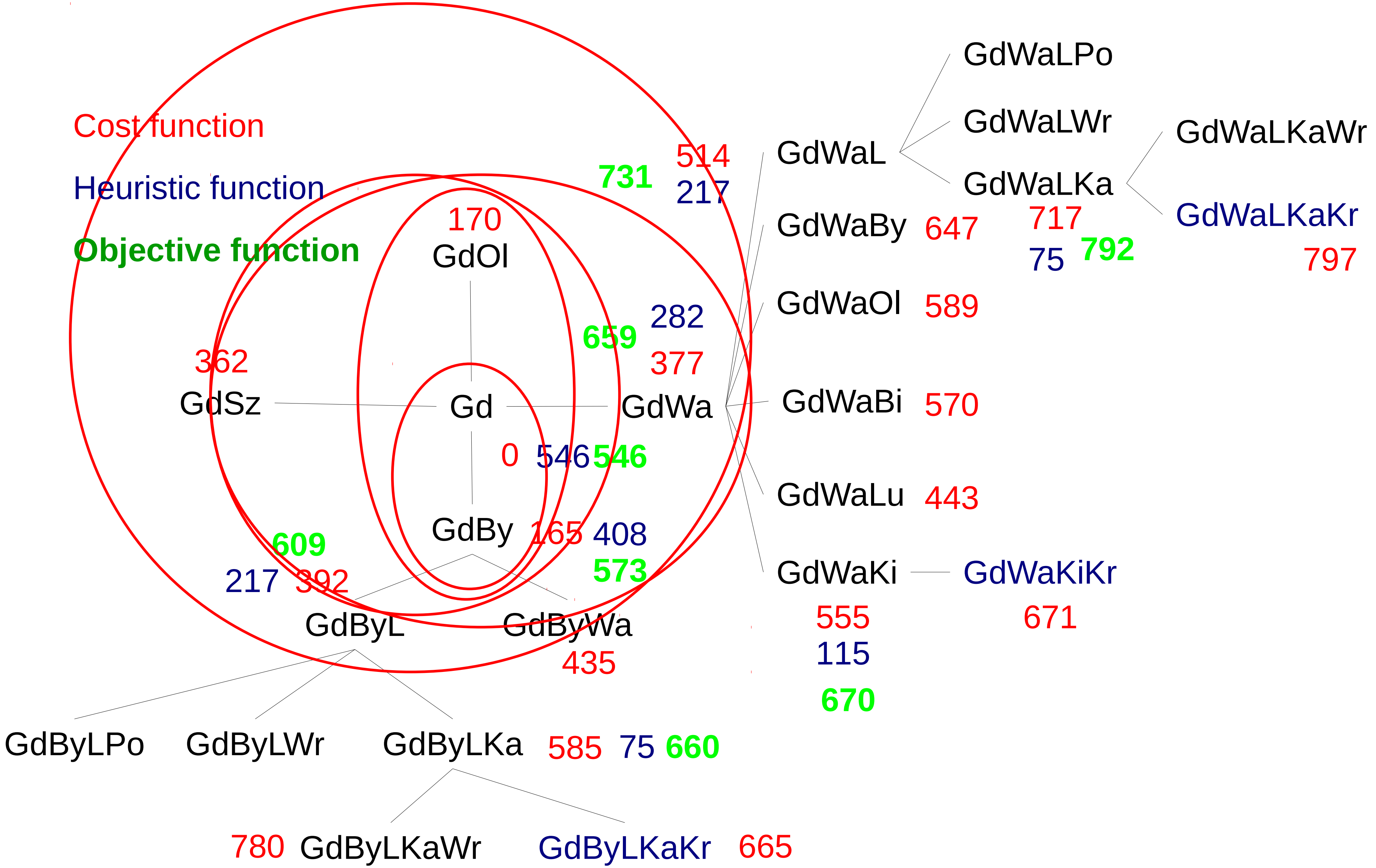
If the heuristic function is admissible and monotonic
and the solution is in a finite distance from the starting point

then A* will yield that solution after a finite number of visited nodes

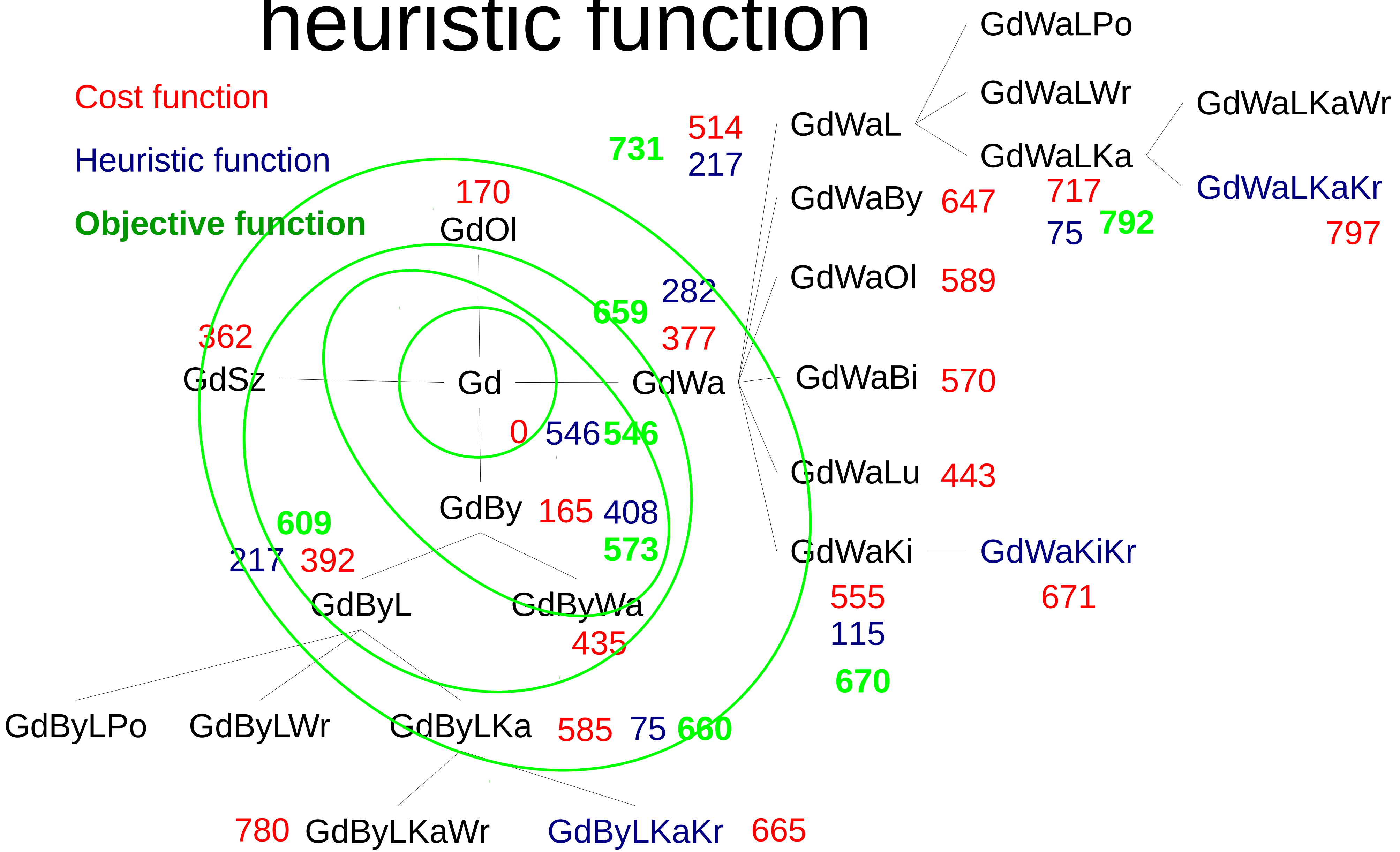
which is no greater than for the best-first that uses the cost function


Cost function
Heuristic function
Objective function
GdOl
170
Gd
0 546 546
GdSz
362
GdWa
659 282 377
GdWaL
731 514 217
GdWaLPo
GdWaLWr
GdWaLKa
717 75 792
GdWaLKaWr
GdWaLKaKr
797
GdWaBy 647
GdWaOl 589
GdWaBi 570
GdWaLu 443
GdWaKi
555 115 670
GdWaKiKr
671
GdBy 165 408 573
GdByL
609 217 392
GdByWa
435
GdByLPo
GdByLWr
GdByLKa 585 75 660
780 GdByLKaWr
GdByLKaKr 665

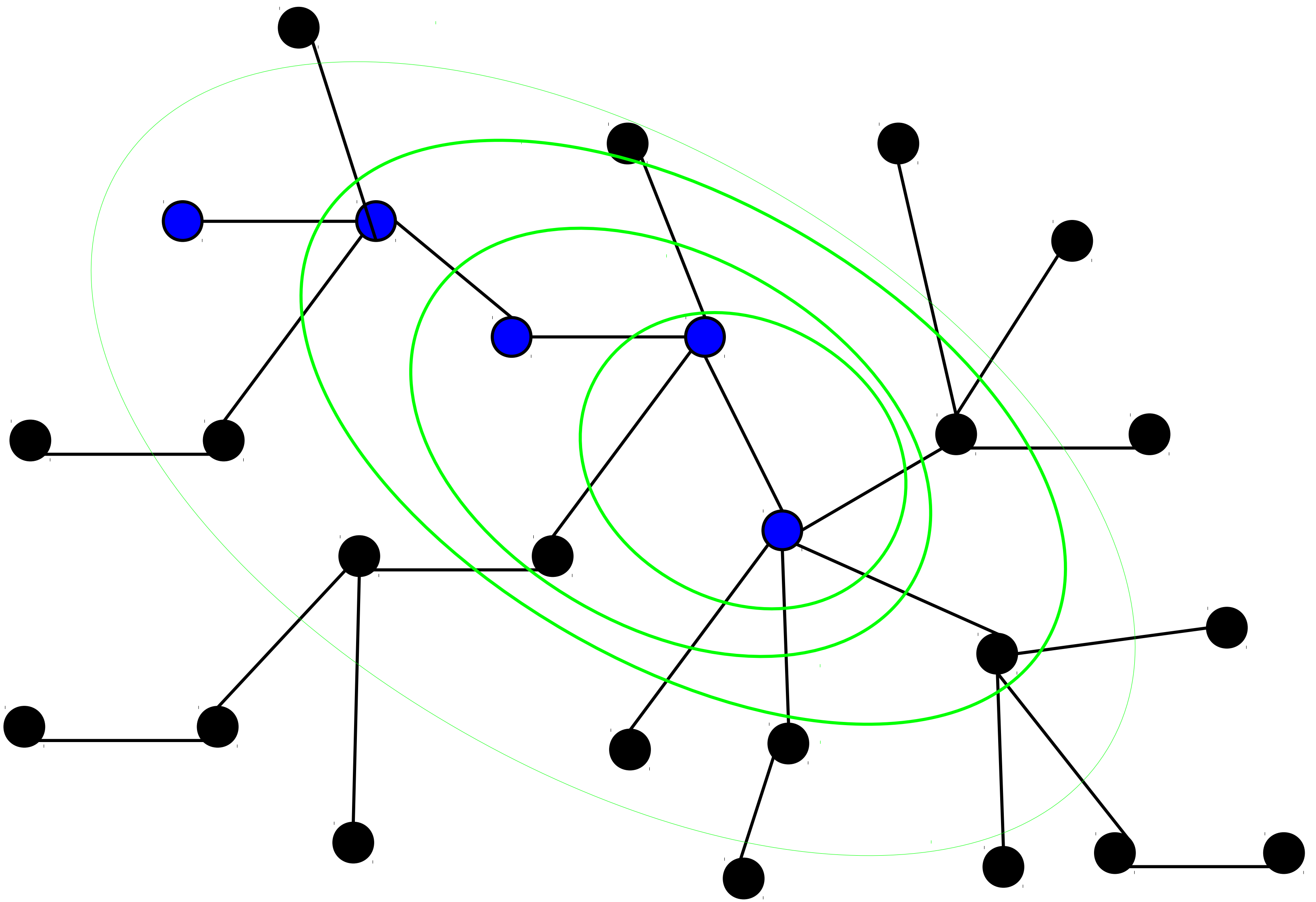
Best first based on the cost



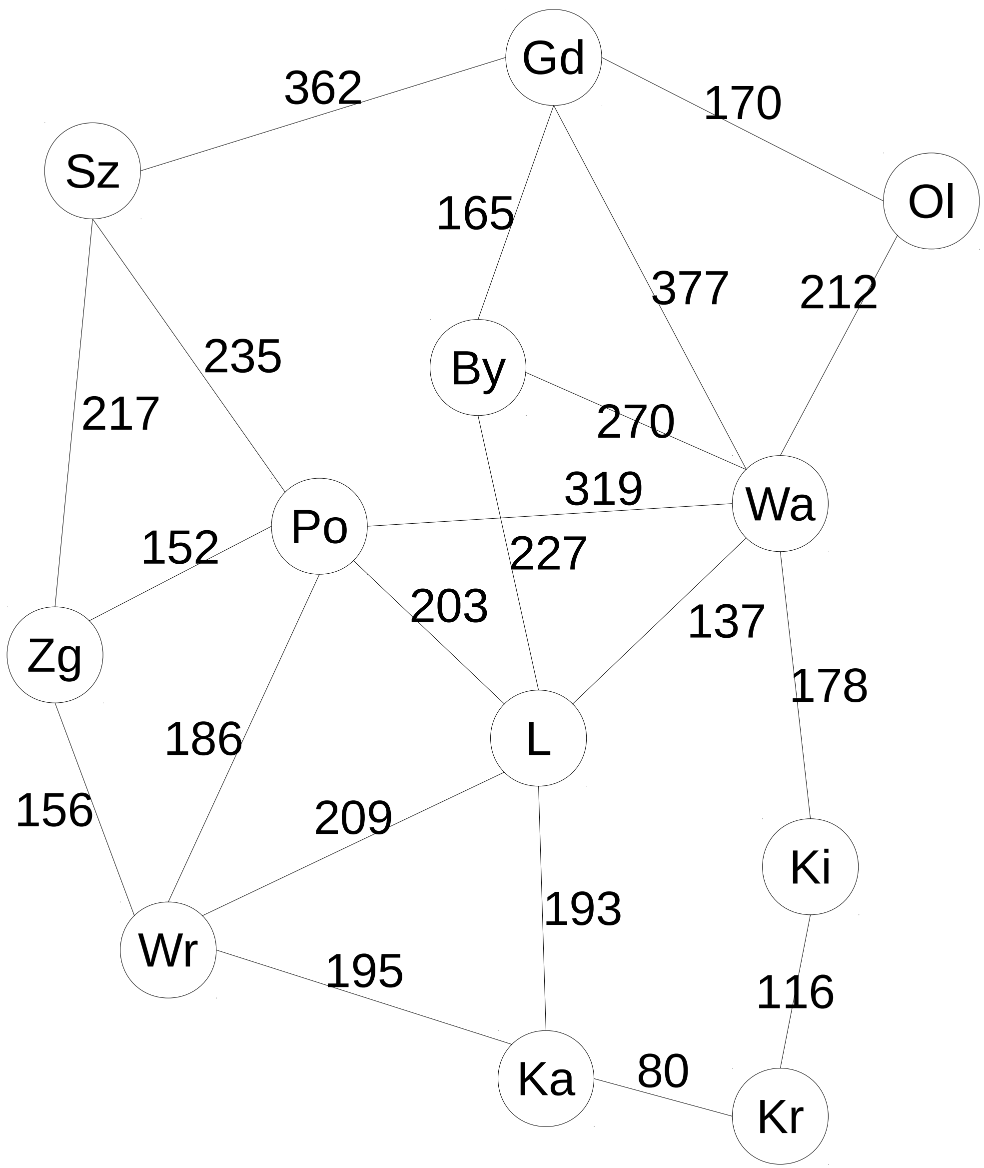
Best first based on the cost + heuristic function



Realistic heuristic function

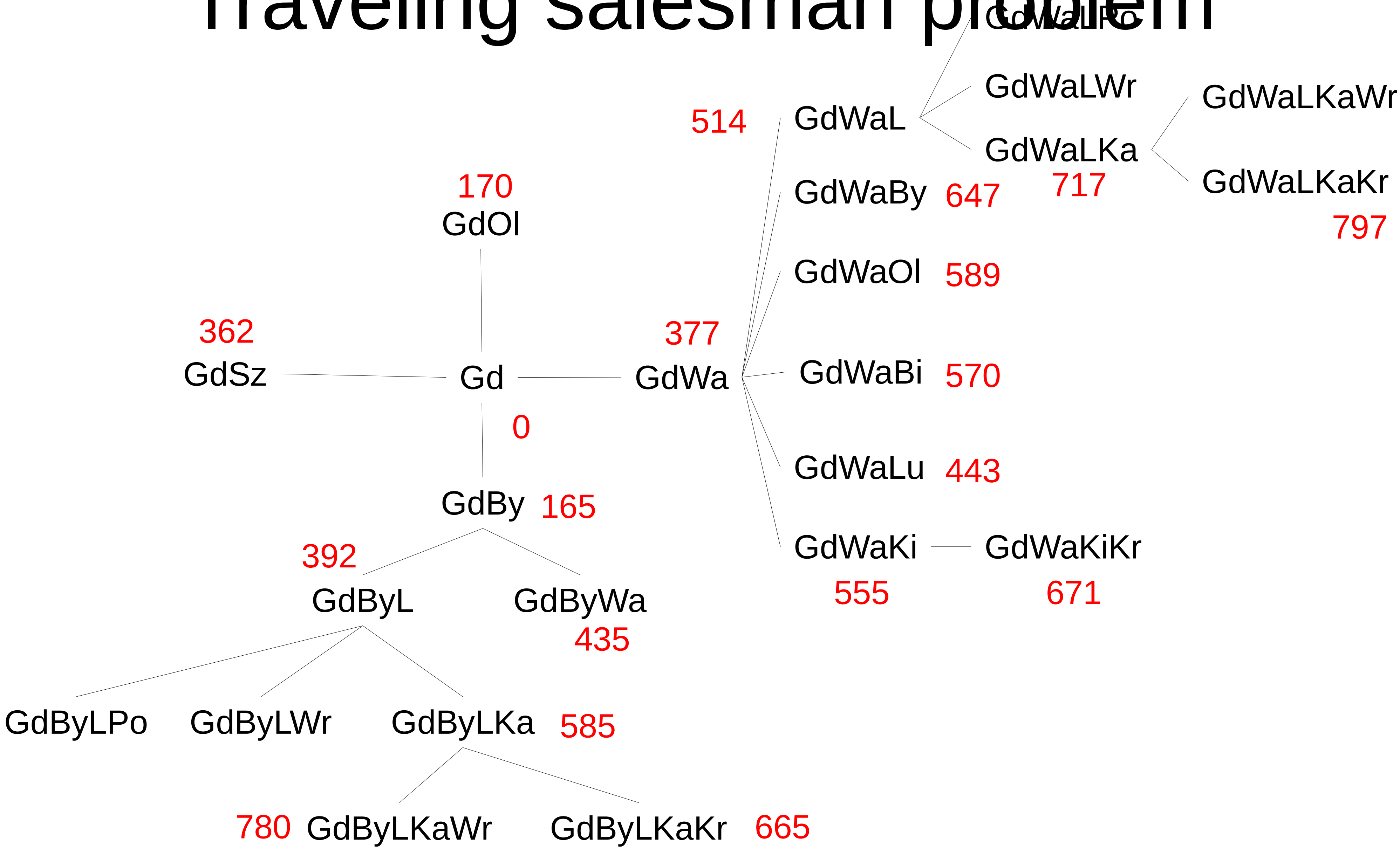


Traveling salesman problem

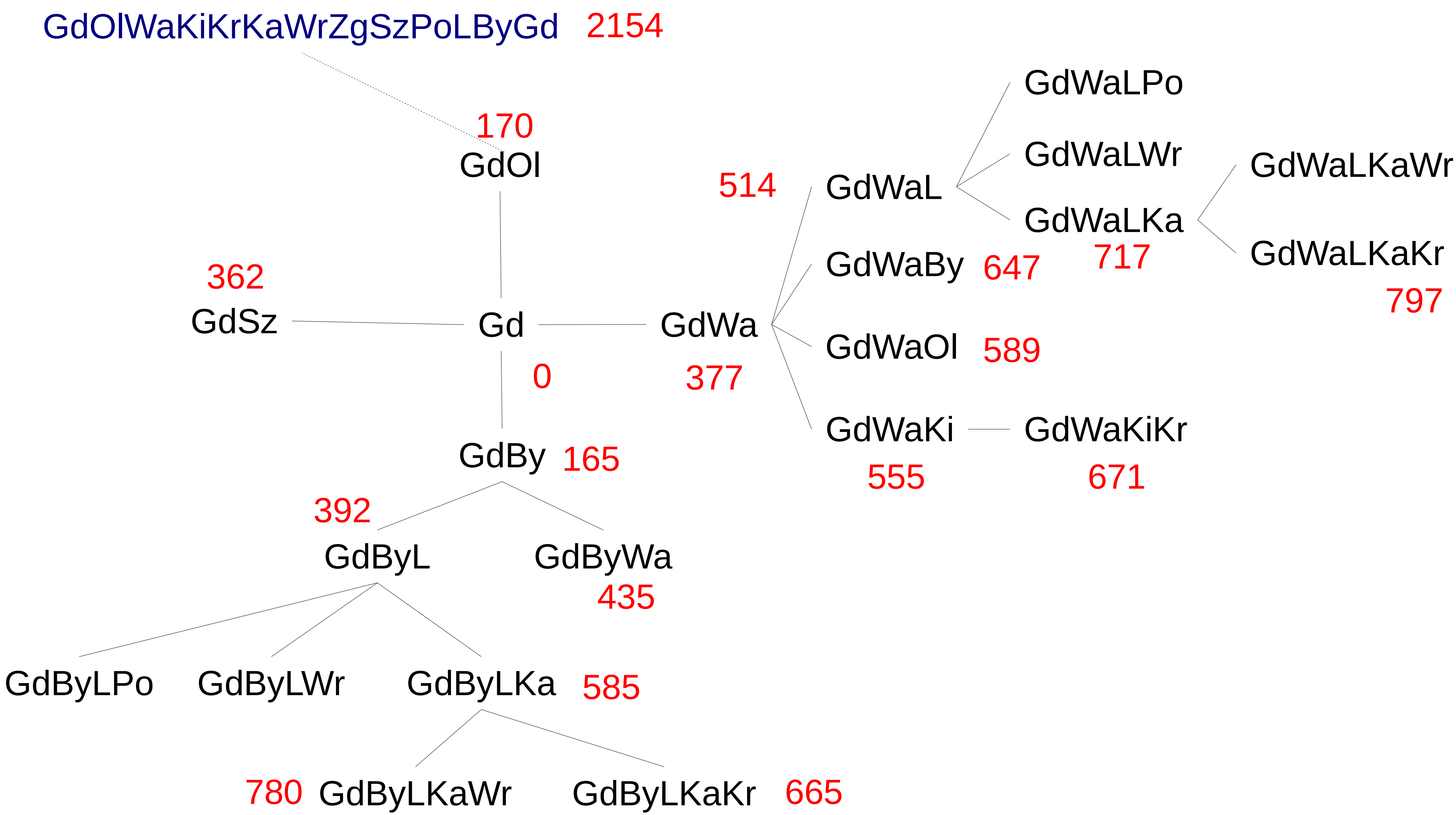


Find the cheapest tour to visit all cities

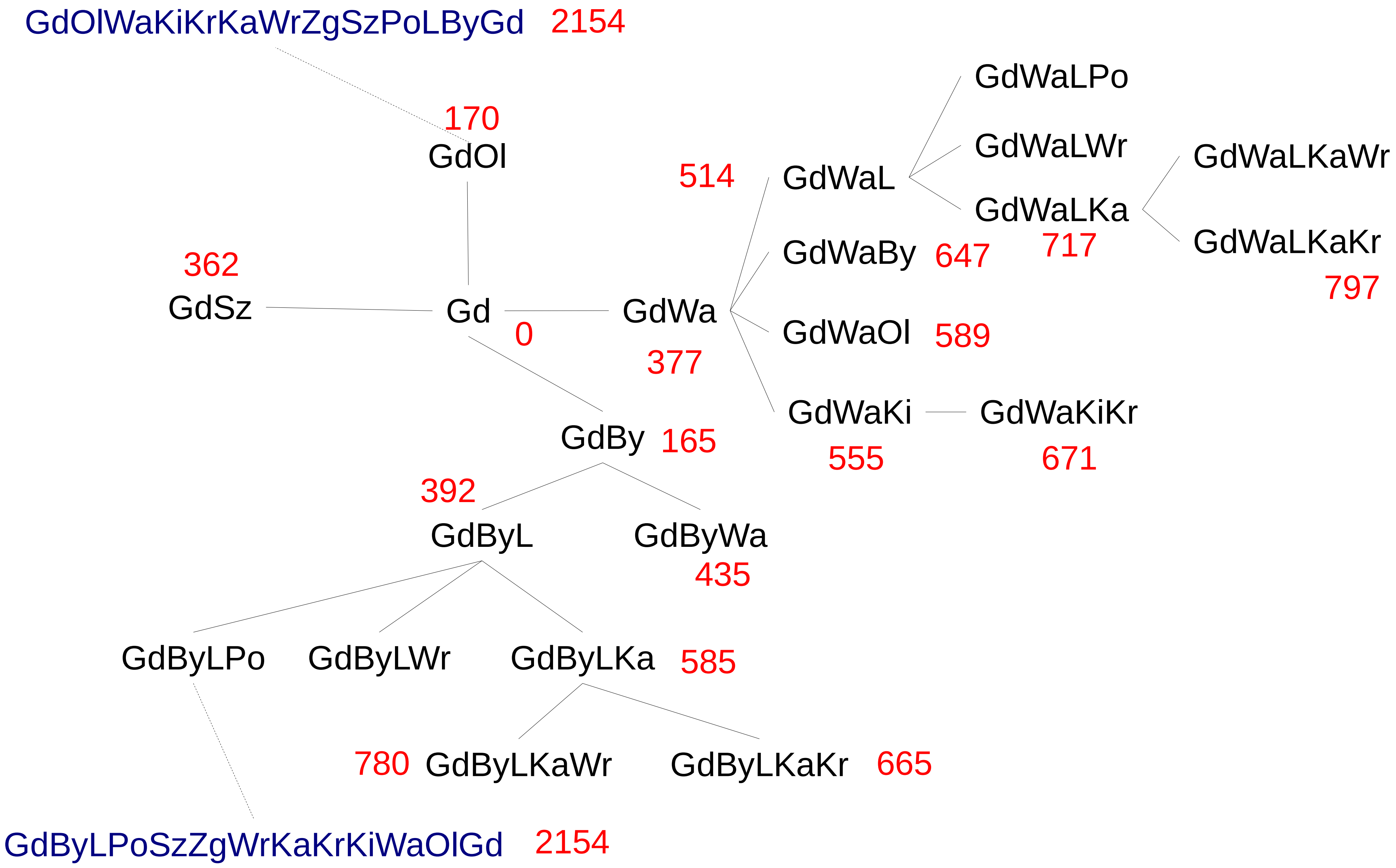
Traveling salesman problem



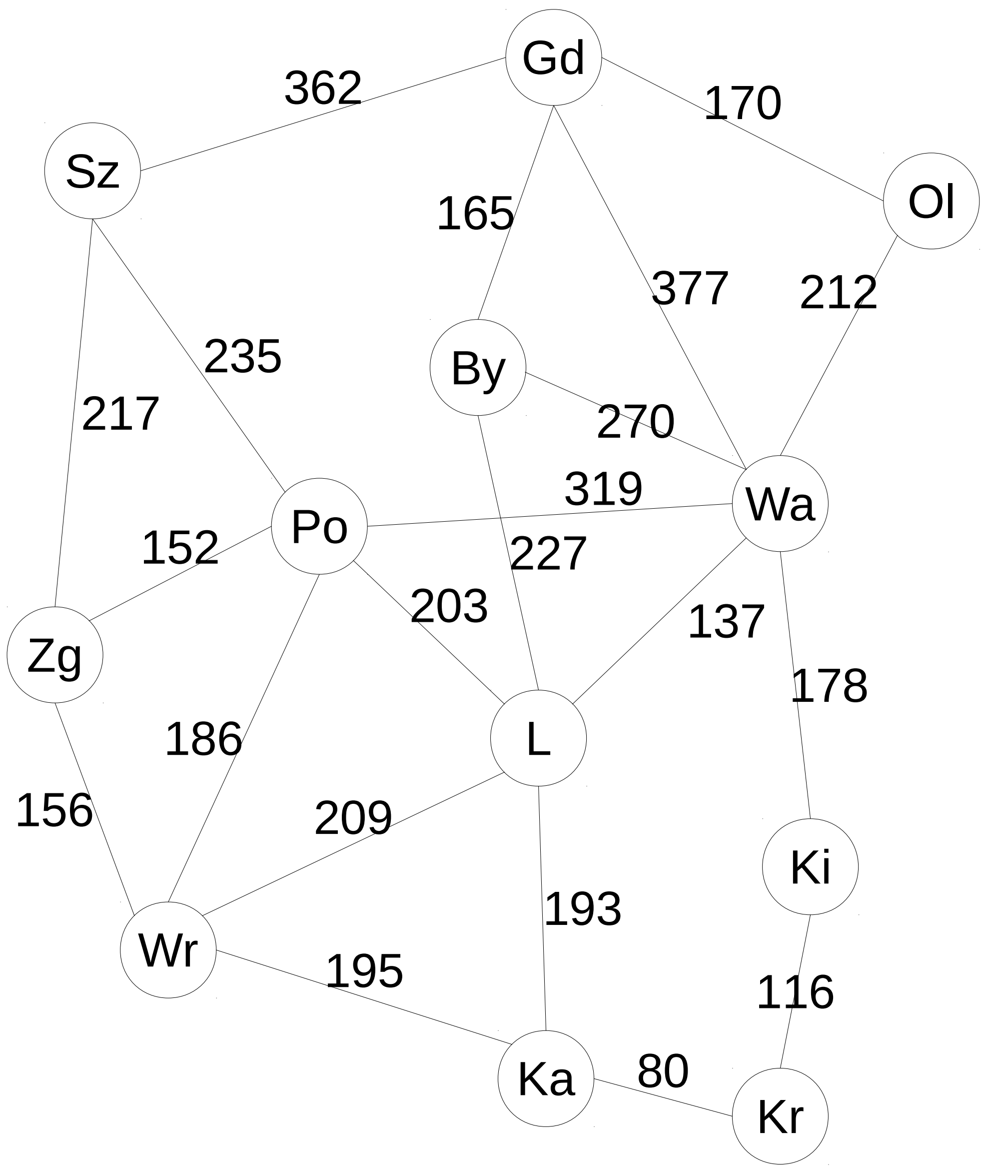
Traveling salesman problem



Traveling salesman problem

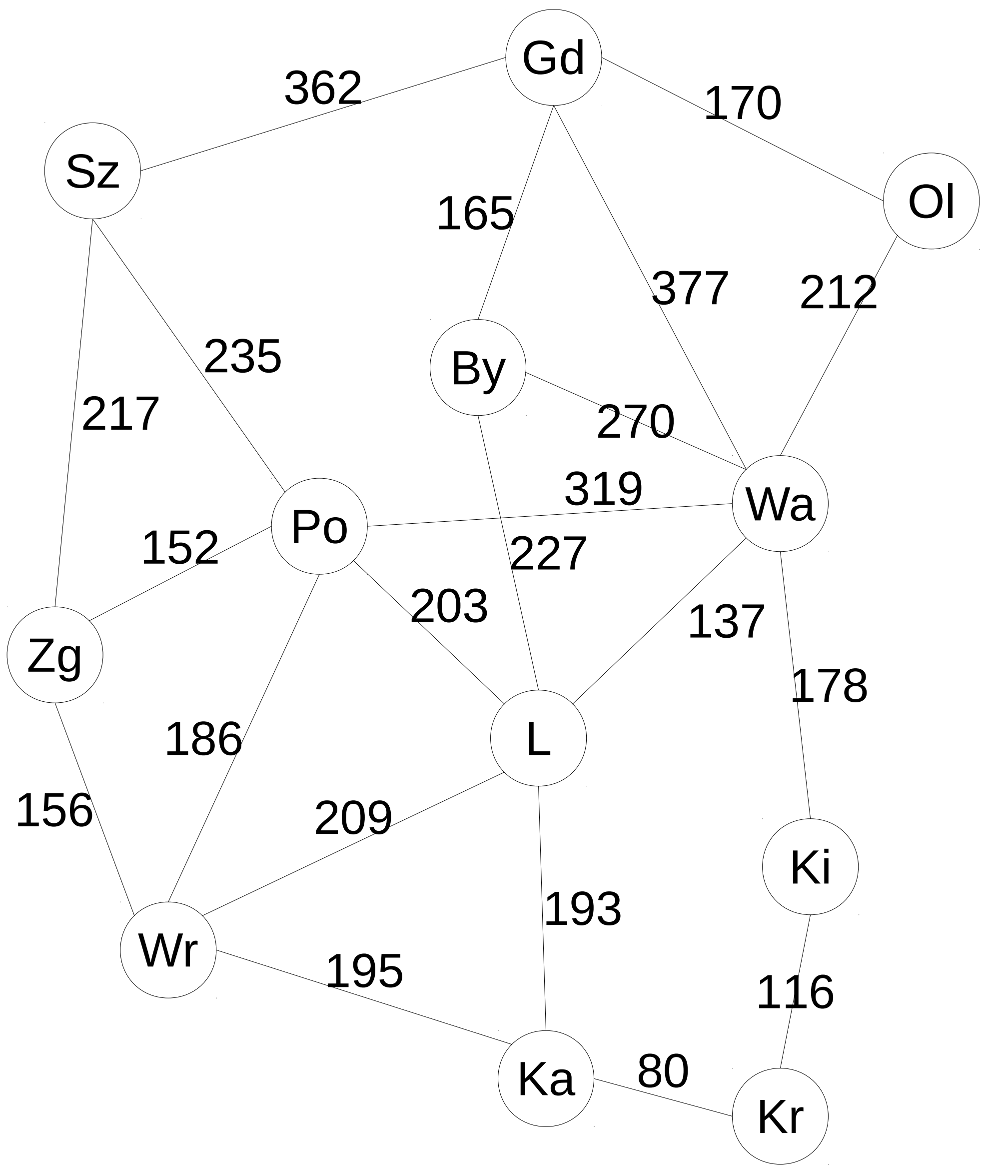


Traveling salesman problem



Heuristic function?

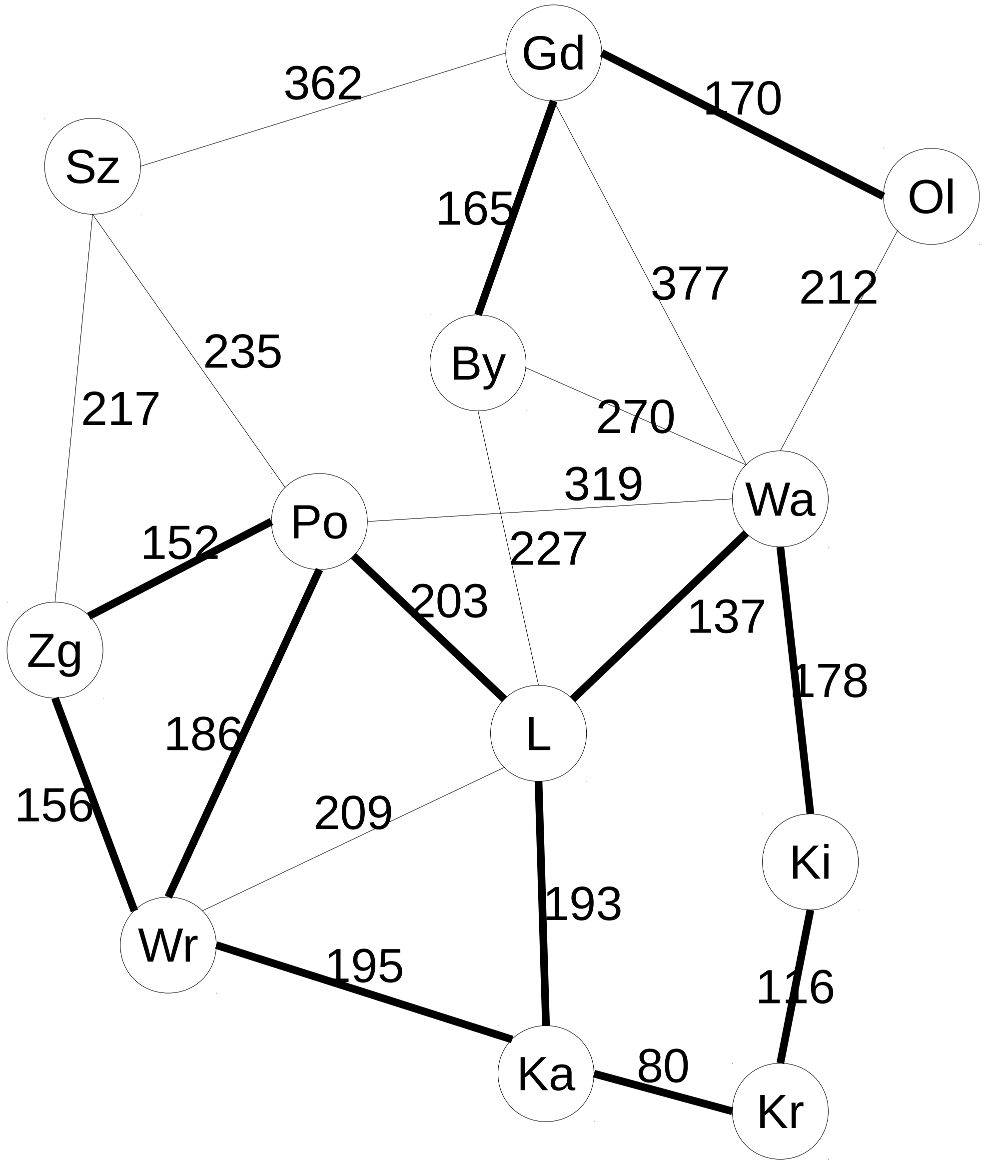
Heuristic function



Sorted list of edge weights

80
116
137
152
156
165
170
178
186
193
195
203
209
212
217

Heuristic function

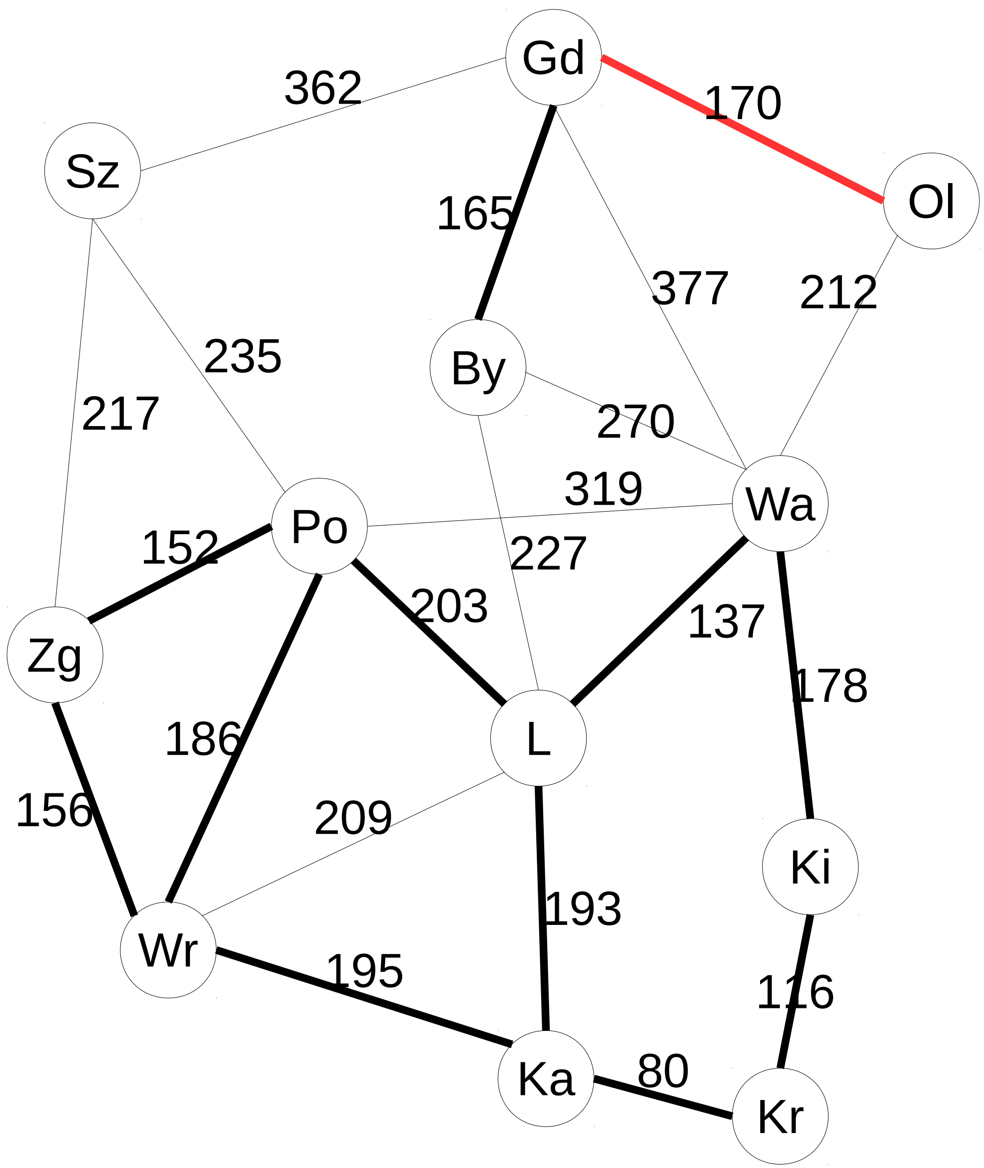


Sorted list of edge weights

80
116
137
152
156
165
170
178
186
193
195
203

1931

Path cost and heuristic function

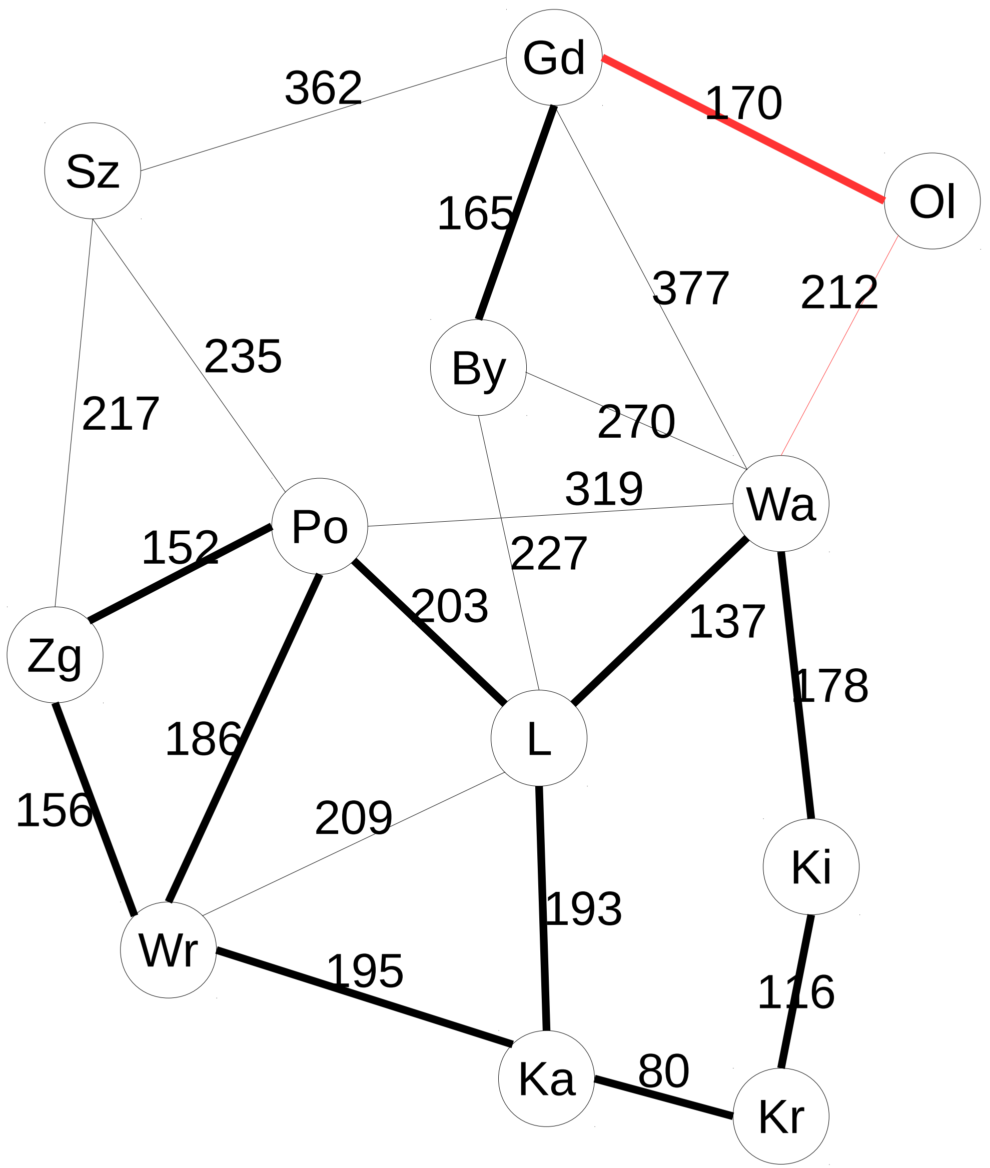


Sorted list of edge weights

80
116
137
152
156
165
~~170~~
178
186
193
195
203

1761
170
1931

Path cost and heuristic function

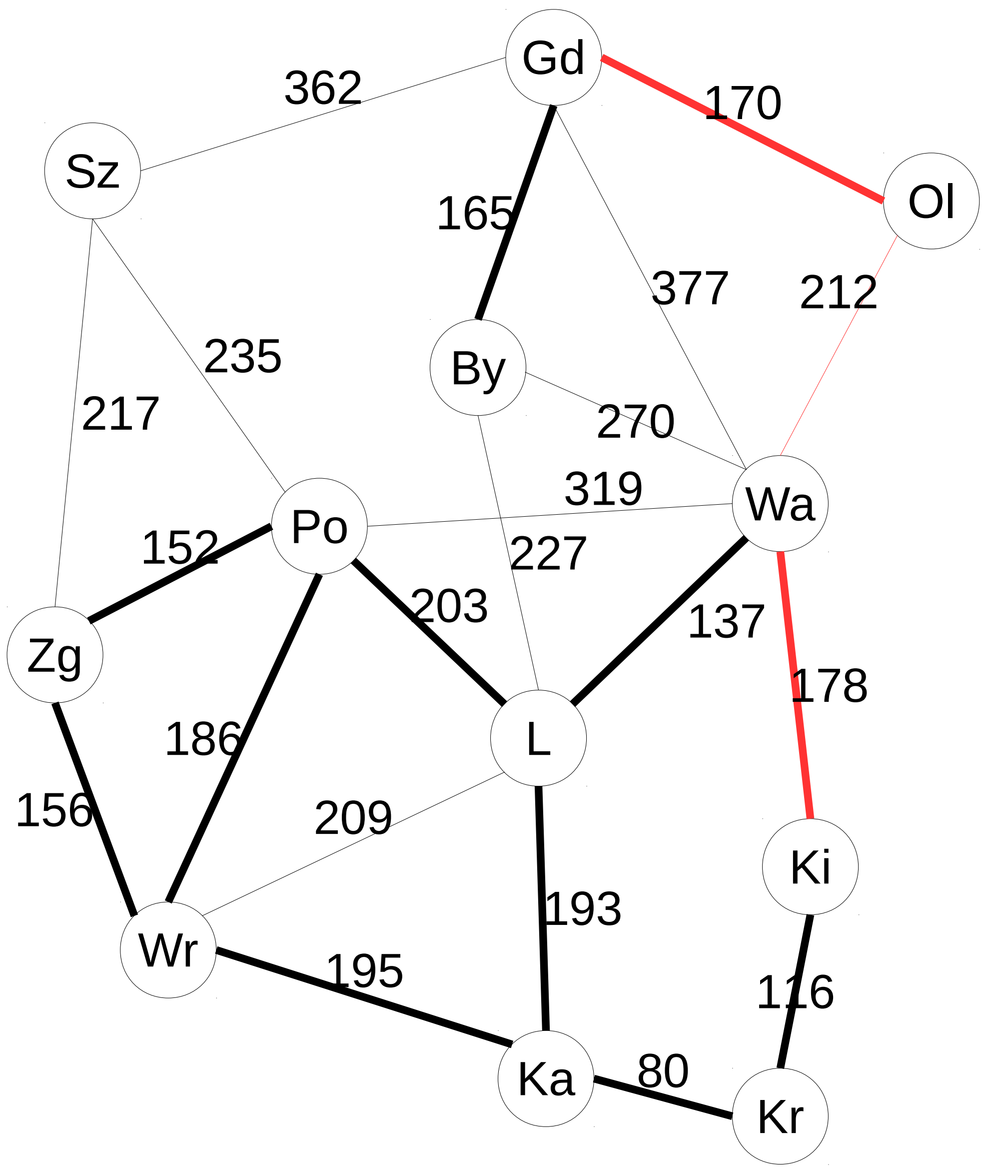


Sorted list of edge weights

80
116
137
152
156
165
~~170~~
178
186
193
195
~~203~~

1558
382
1940

Path cost and heuristic function

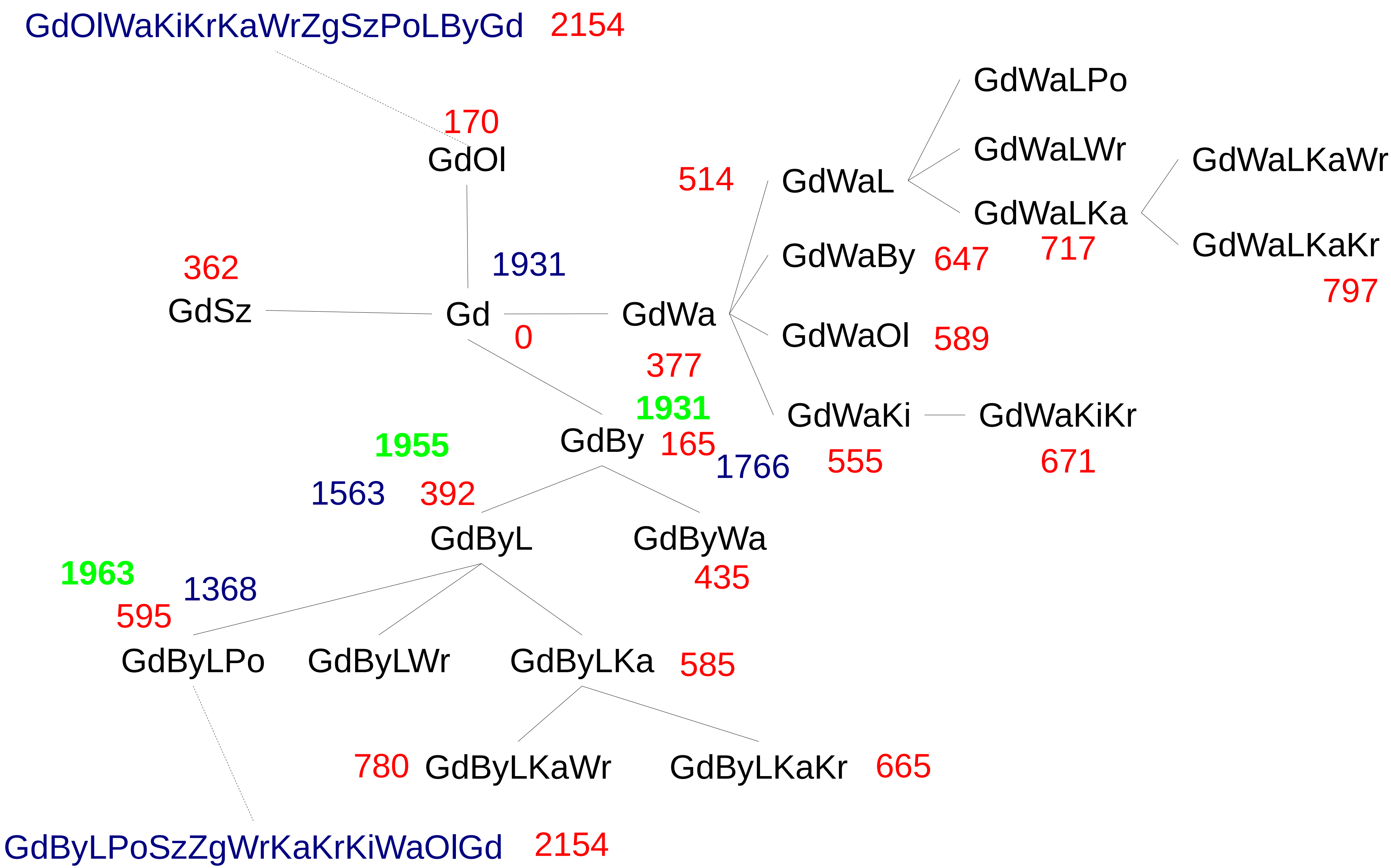


Sorted list of edge weights

80
116
137
152
156
165
~~170~~
~~178~~
186
193
195
~~203~~

1380
560
1940

Traveling salesman problem



Solving 15 puzzle

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

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

Solution representation?

Admissible transformations?

Cost function?

Heuristic function?

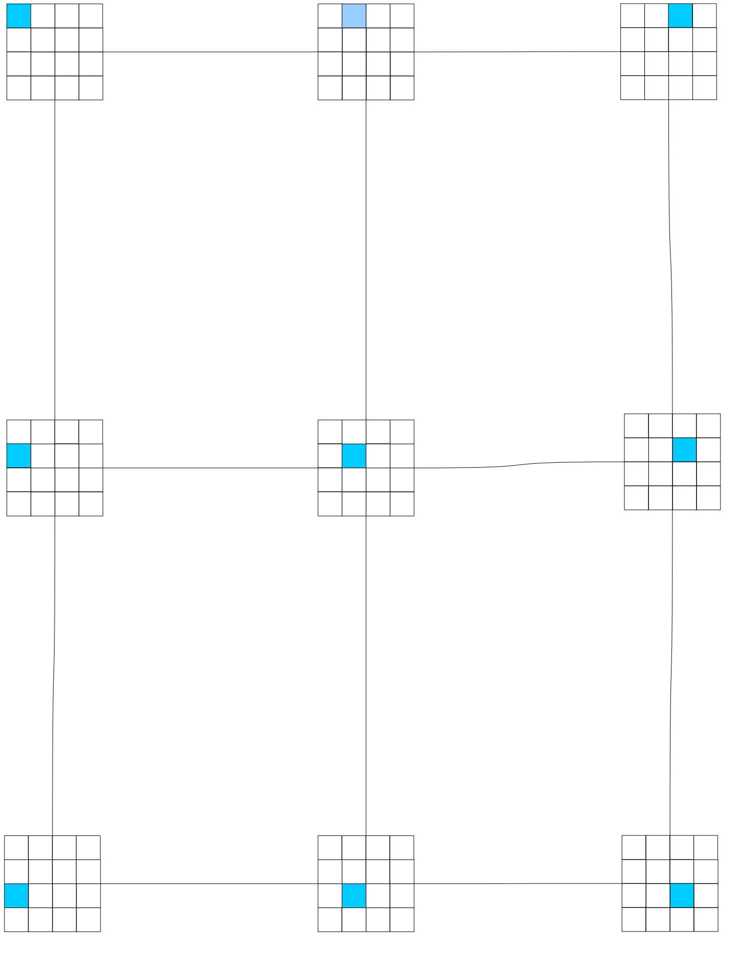
Solving 15 puzzle attempt #1

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

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

Solution representation?	Positions of tiles
Admissible transformations?	Move a tile into empty place
Cost function?	Number of wrongly placed tiles
Heuristic function?	?

Solving 15 puzzle attempt #1



Solving 15 puzzle attempt #2

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

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

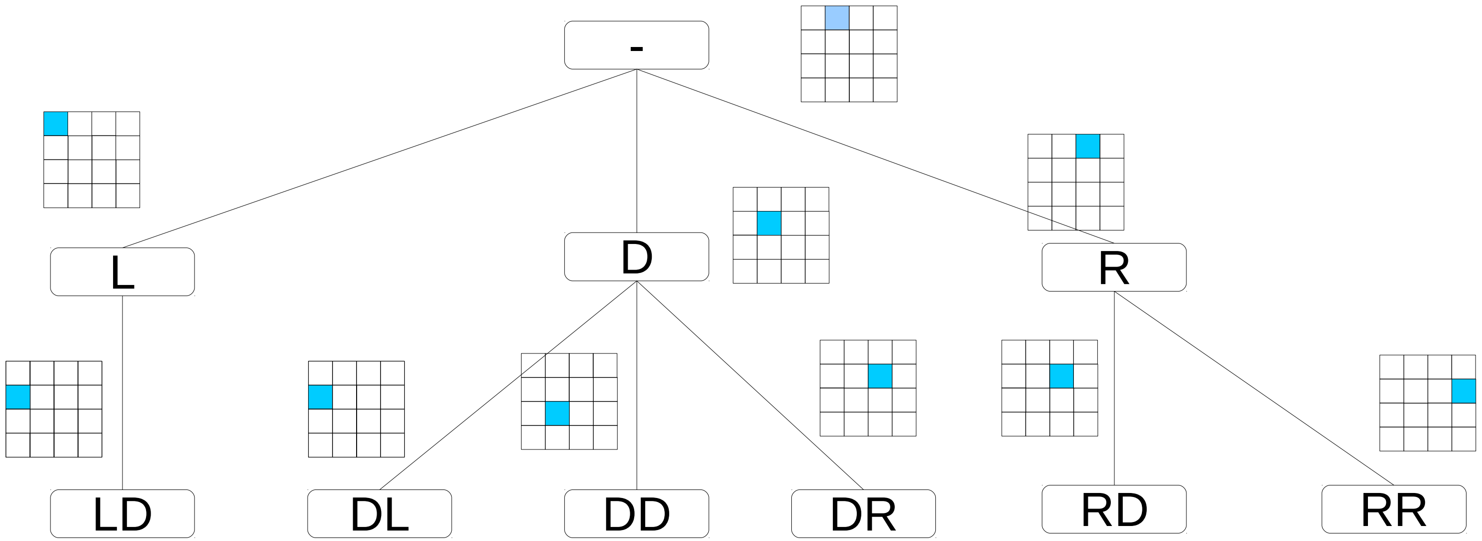
Solution representation? Sequence of moves

Admissible transformations? Move a tile into empty place

Cost function? Number of moves

Heuristic function? ?

Solving 15 puzzle attempt #2



Solving 15 puzzle
attempt #2 - heuristic function

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

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

30

2:4
4:2
5:4
6:2
7:2
8:2

10:2
11:2
12:4
13:2
14:2
15:2

Knapsack problem

- N items
- Each item has its weight $w_i>0$ and profit $p_i>0$
- Choose items such that total profit is maximized and total weight does not exceed W

$$max \sum_{i=1}^{n} x_i p_i$$

$$\sum_{i=1}^{n} x_i w_i \leqslant W$$

$$x_i \in \{0,1\}$$

Knapsack problem

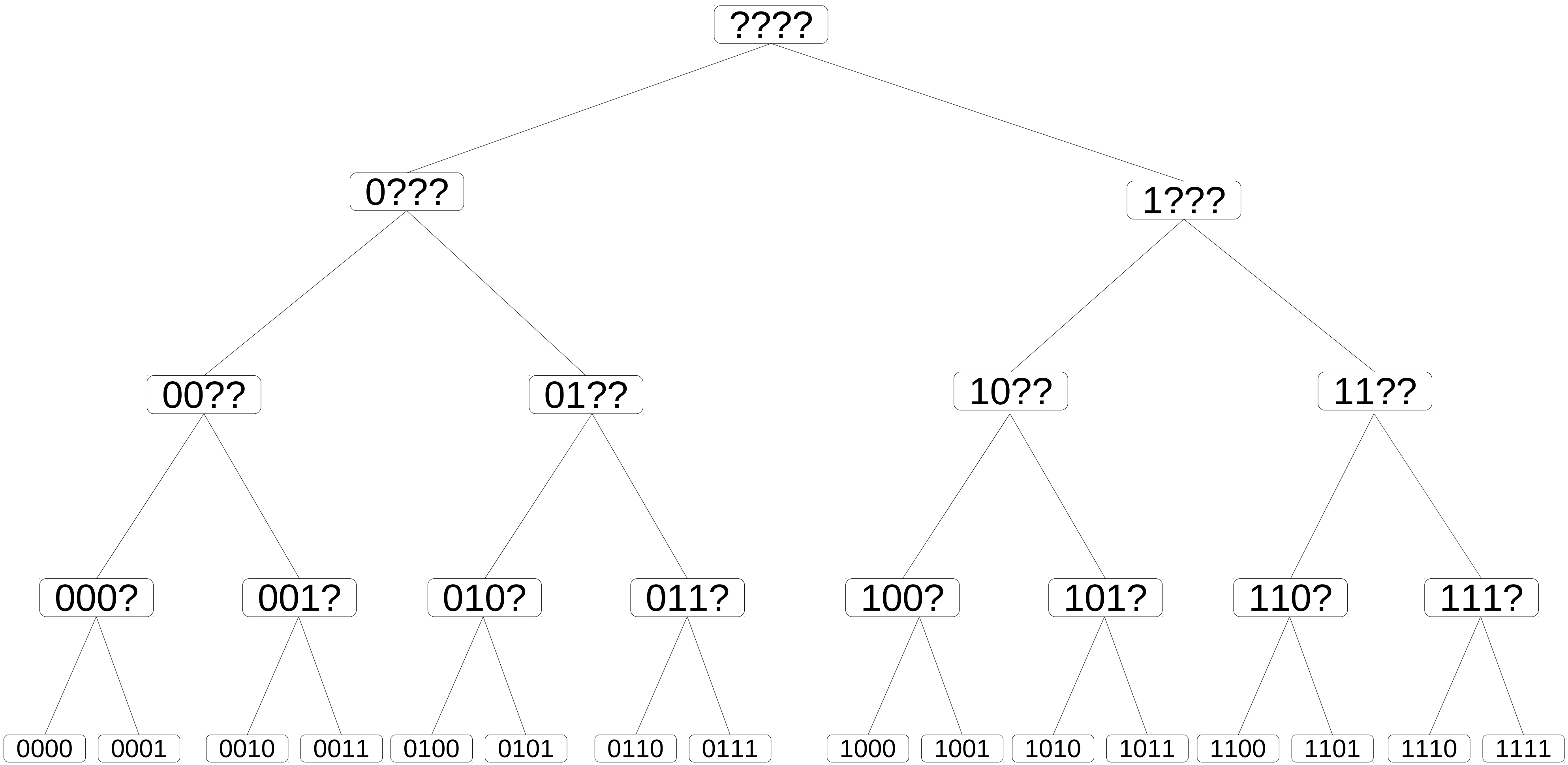
- N items
- Each item has its weight $w_i>0$ and profit $p_i>0$
- Choose items such that total profit is maximized and total weight does not exceed W

$$max \sum_{i=1}^{n} x_i p_i$$

$$\sum_{i=1}^{n} x_i w_i \leqslant W$$

$$x_i \in \{0,1\}$$

Knapsack problem search space



Knapsack problem profit and heuristic function

- Profit function

$$g(x)=\sum_{i:x_i=1} p_i$$

- Heuristic function

Items are sorted w.r.t. p_i/w_i (descending)

$$h(x)=\sum_{i:x_i=?} y_i p_i$$

$$\sum_{i:x_i=?} y_i w_i = W - \sum_{i:x_i=1} x_i w_i$$

$$y_i \in [0,1]$$

Knapsack problem example profit and heuristic function

- Items

i	1	2	3	4	5	6
p_i	20	5	10	5	6	3
w_i	6	2	5	3	4	3
p_i/w_i	3.33	2.5	2	1.66	1.5	1

- W=13
- Consider the solution $x = ?0?1?1$
- Total profit: $g(x) = p_4 + p_6 = 5 + 3 = 8$
- Total weight: $w(x) = w_4 + w_6 = 3 + 3 = 6$
- Vector y: $y = [1,0,1/5,0,0,0]$
- Heuristic function: $h(x) = 20 \cdot 1 + 10 \cdot 1/5 = 22$