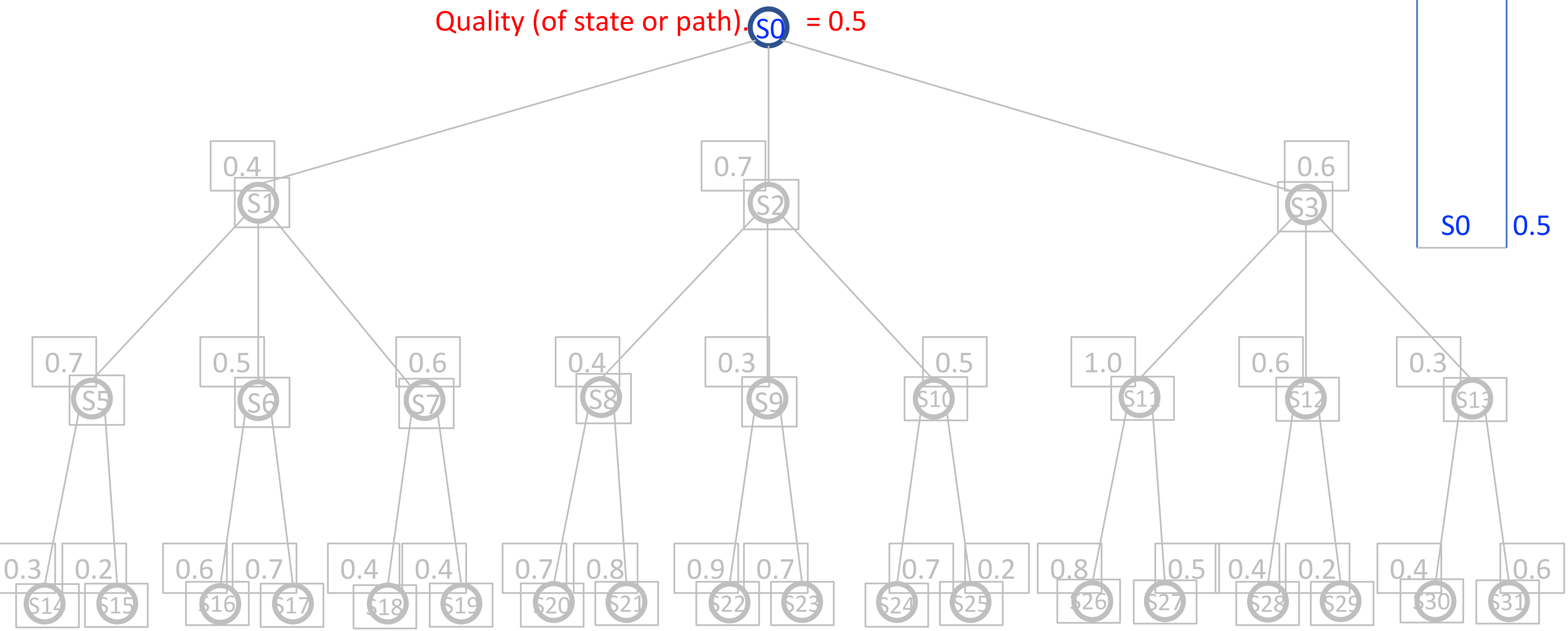
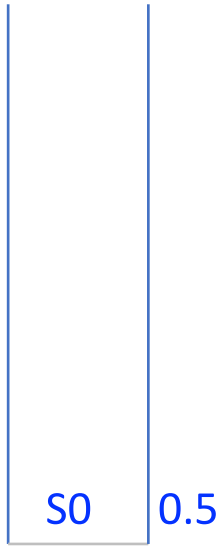
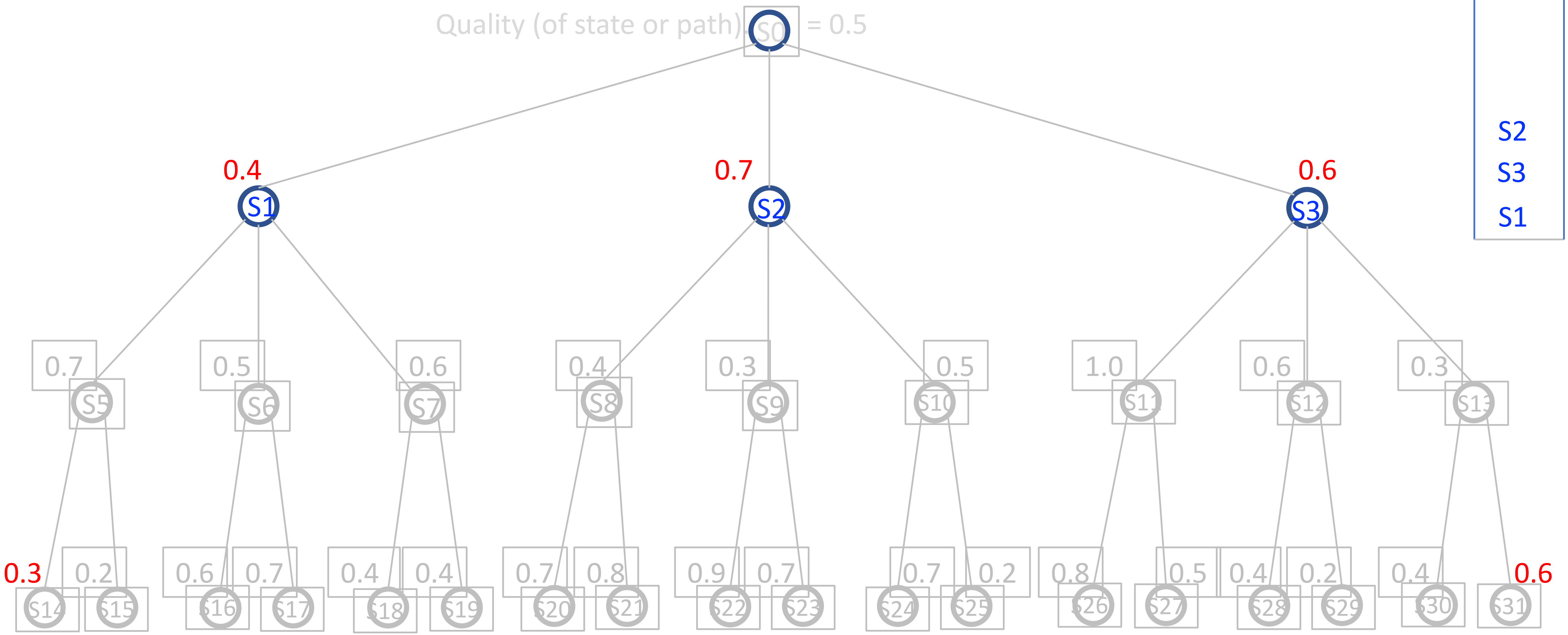
Heuristic depth-bounded depth-first search where B is branching factor and D is depth bound



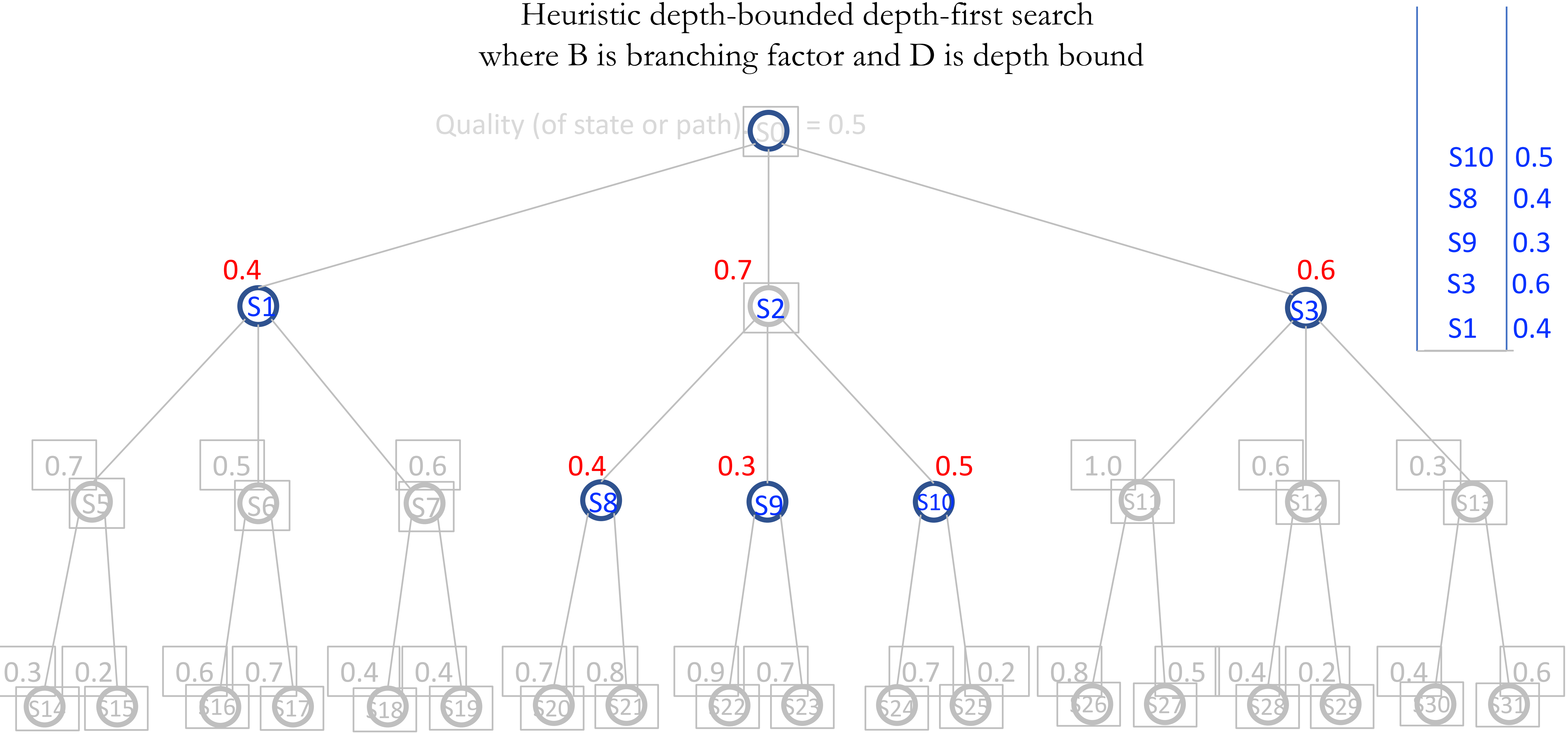


Heuristic depth-bounded depth-first search
where B is branching factor and D is depth bound



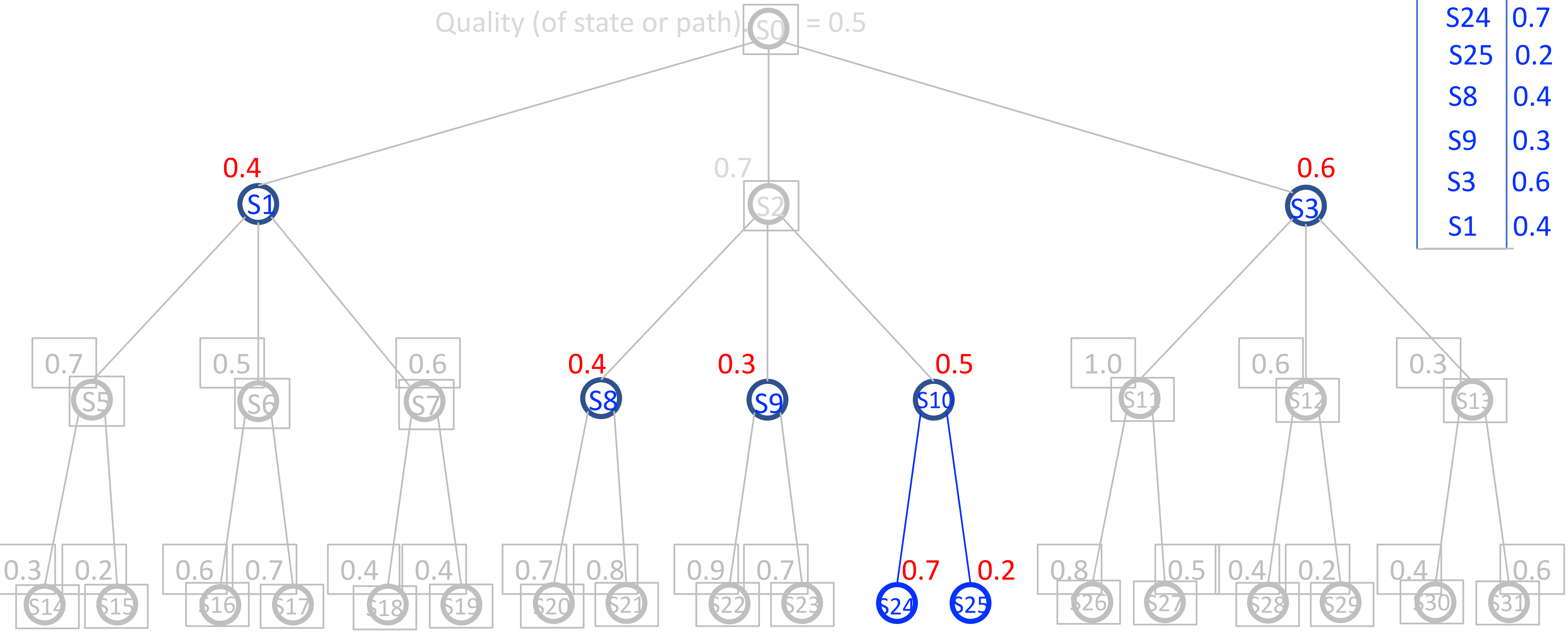
S2	0.7
S3	0.6
S1	0.4

Heuristic depth-bounded depth-first search
where B is branching factor and D is depth bound



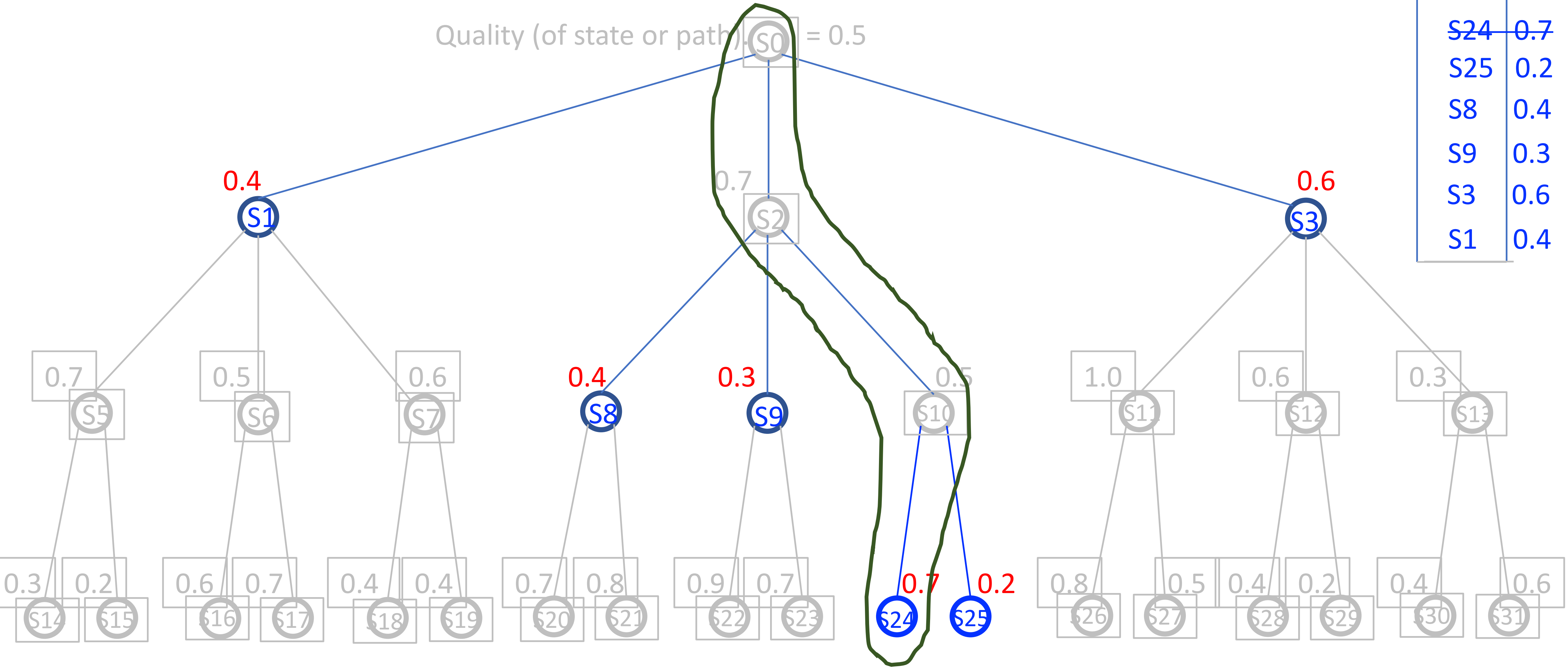
S10	0.5
S8	0.4
S9	0.3
S3	0.6
S1	0.4

Heuristic depth-bounded depth-first search
where B is branching factor and D is depth bound



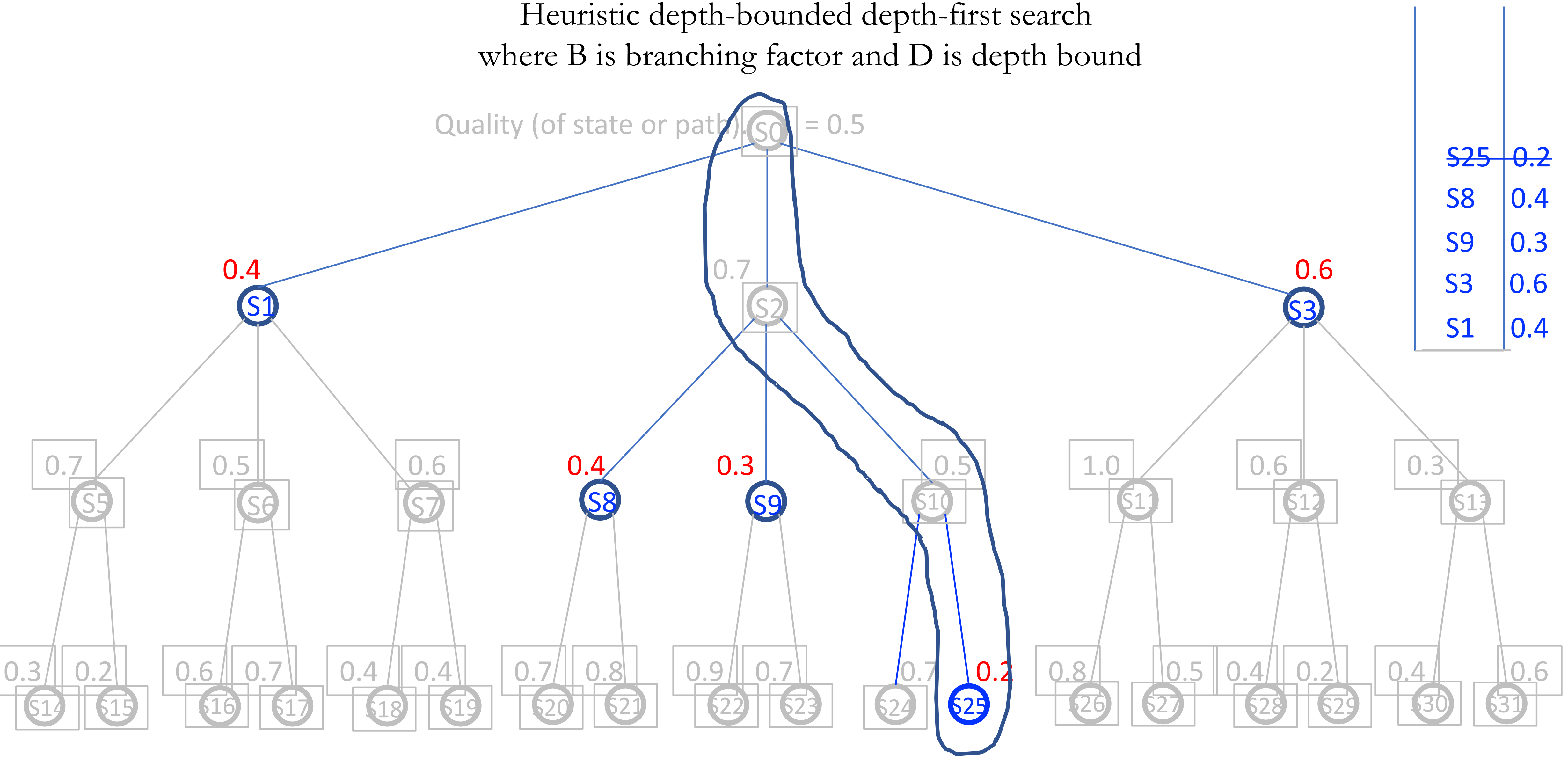
S24	0.7
S25	0.2
S8	0.4
S9	0.3
S3	0.6
S1	0.4

Heuristic depth-bounded depth-first search
where B is branching factor and D is depth bound



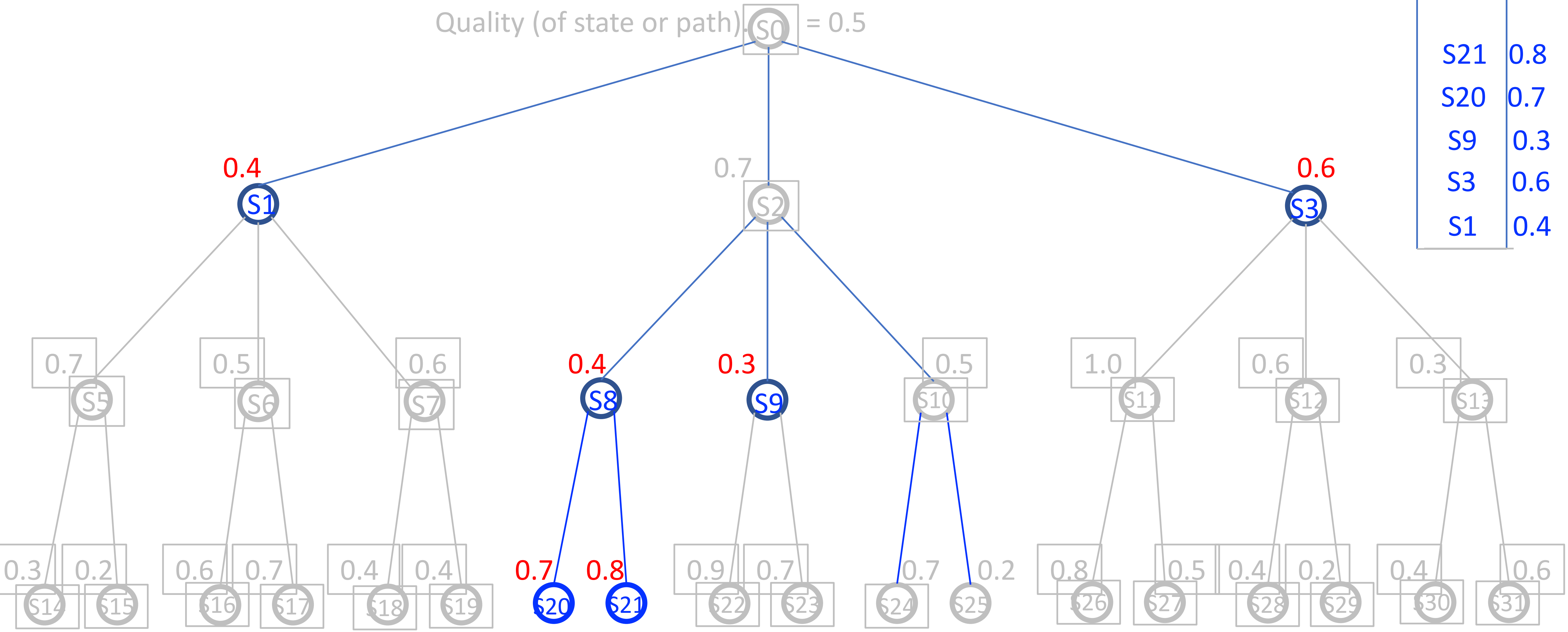
~~S24~~	~~0.7~~
S25	0.2
S8	0.4
S9	0.3
S3	0.6
S1	0.4

Heuristic depth-bounded depth-first search
where B is branching factor and D is depth bound



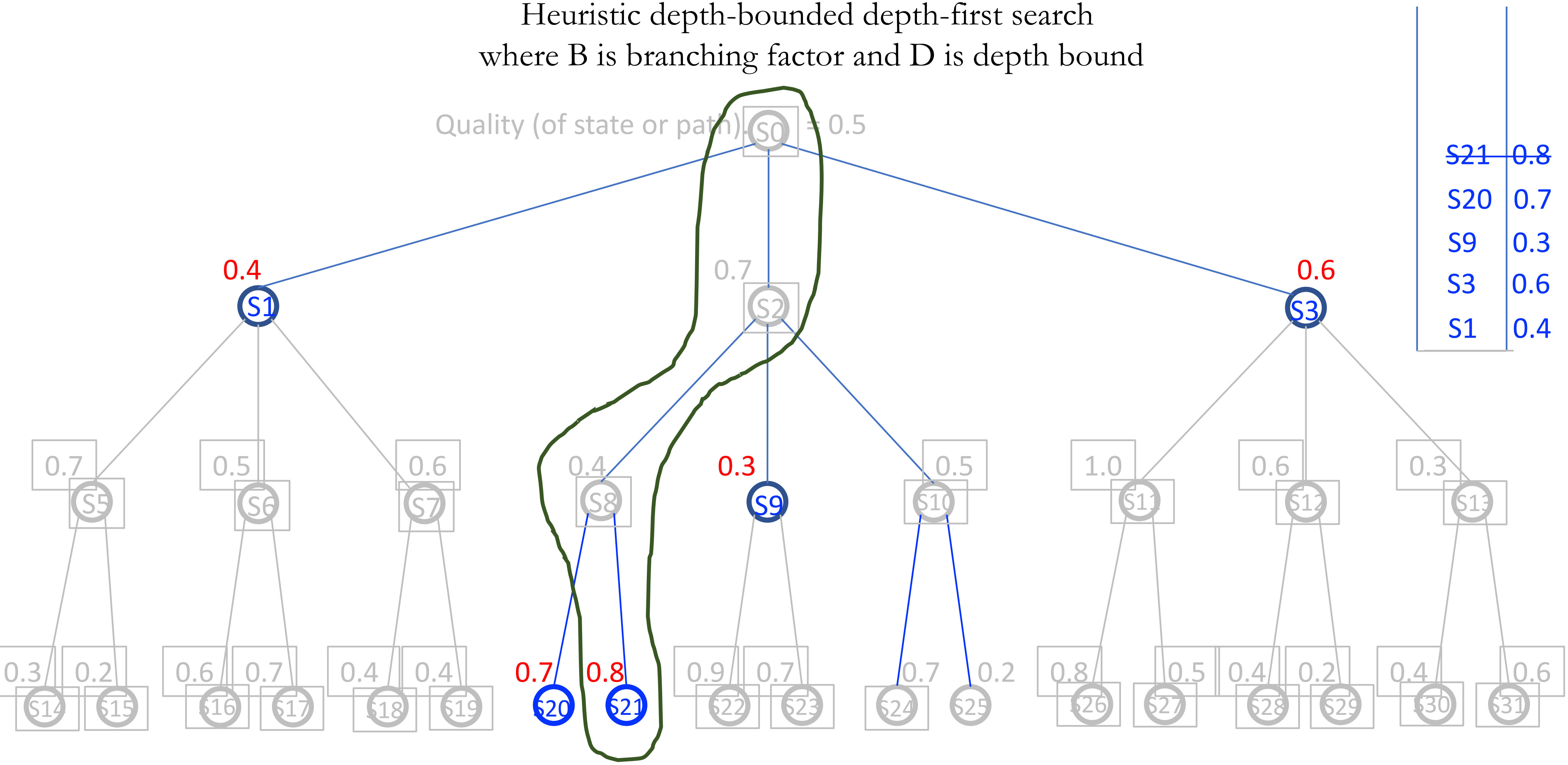
~~S25~~	~~0.2~~
S8	0.4
S9	0.3
S3	0.6
S1	0.4

Heuristic depth-bounded depth-first search
where B is branching factor and D is depth bound



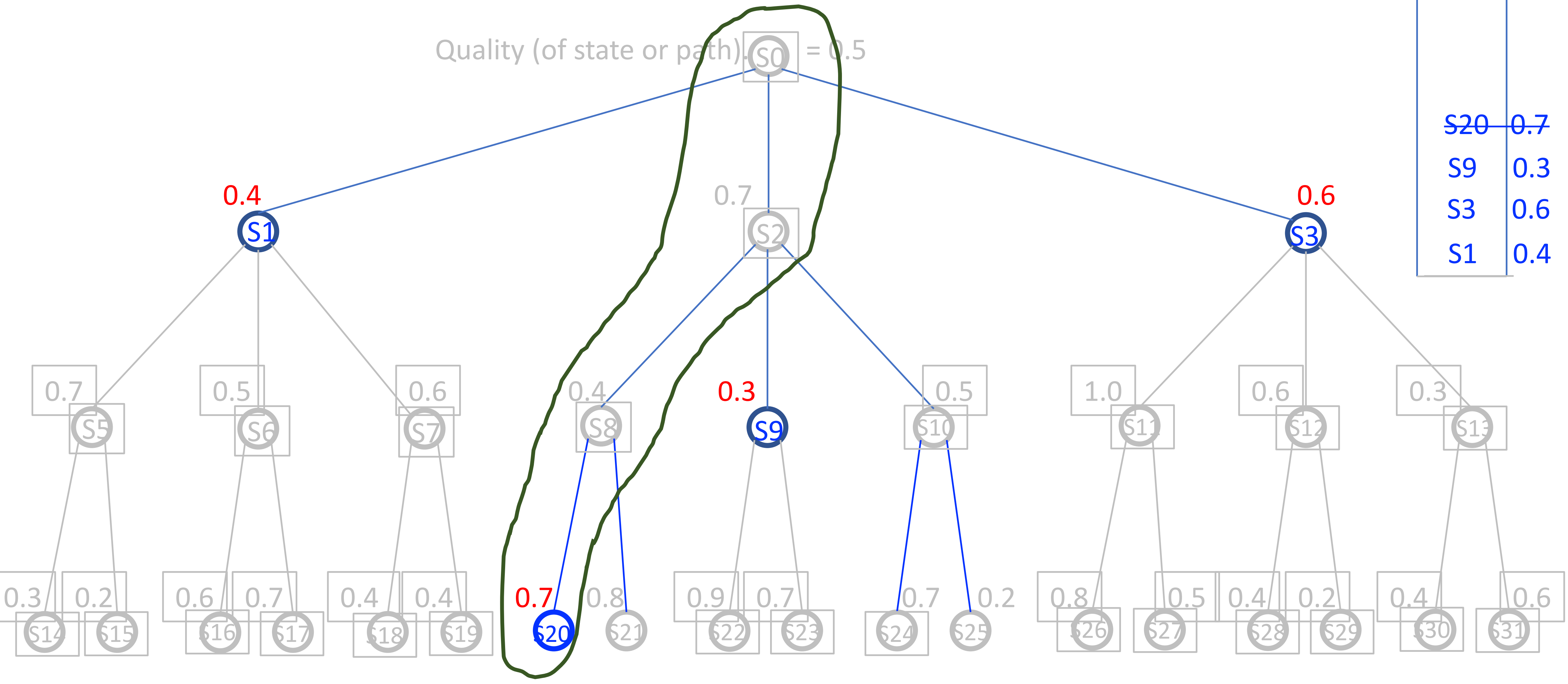
S21	0.8
S20	0.7
S9	0.3
S3	0.6
S1	0.4

Heuristic depth-bounded depth-first search where B is branching factor and D is depth bound



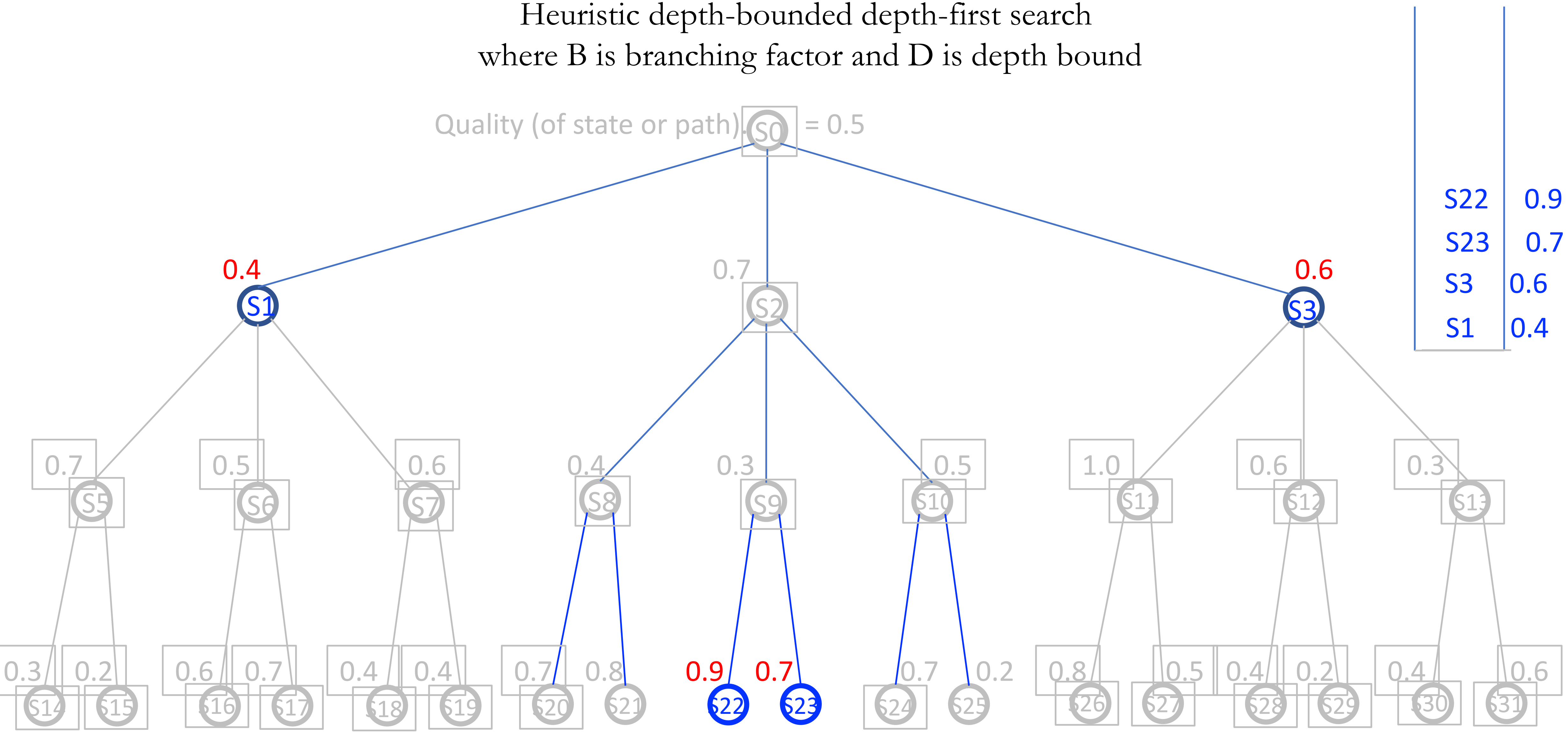
~~S21~~	~~0.8~~
S20	0.7
S9	0.3
S3	0.6
S1	0.4

Heuristic depth-bounded depth-first search
where B is branching factor and D is depth bound



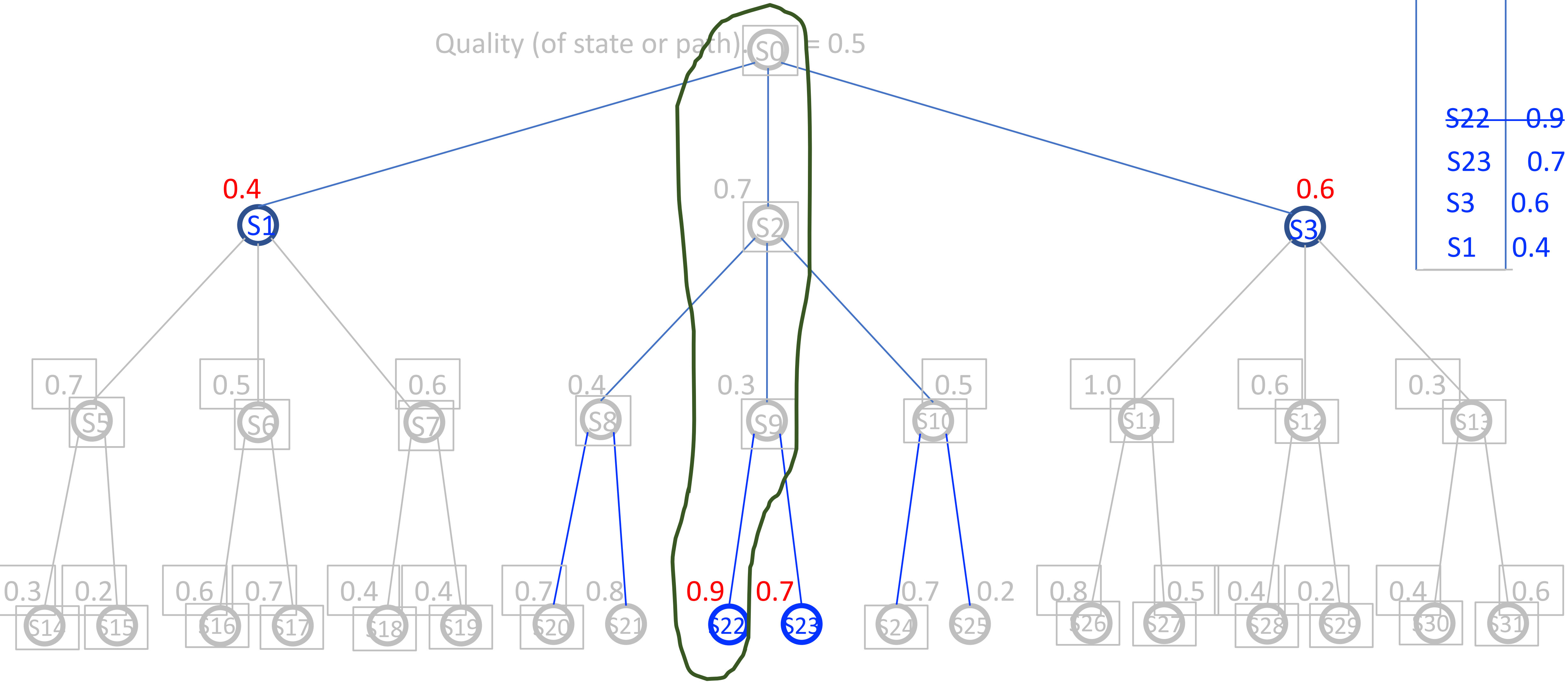
~~S20~~	~~0.7~~
S9	0.3
S3	0.6
S1	0.4

Heuristic depth-bounded depth-first search
where B is branching factor and D is depth bound



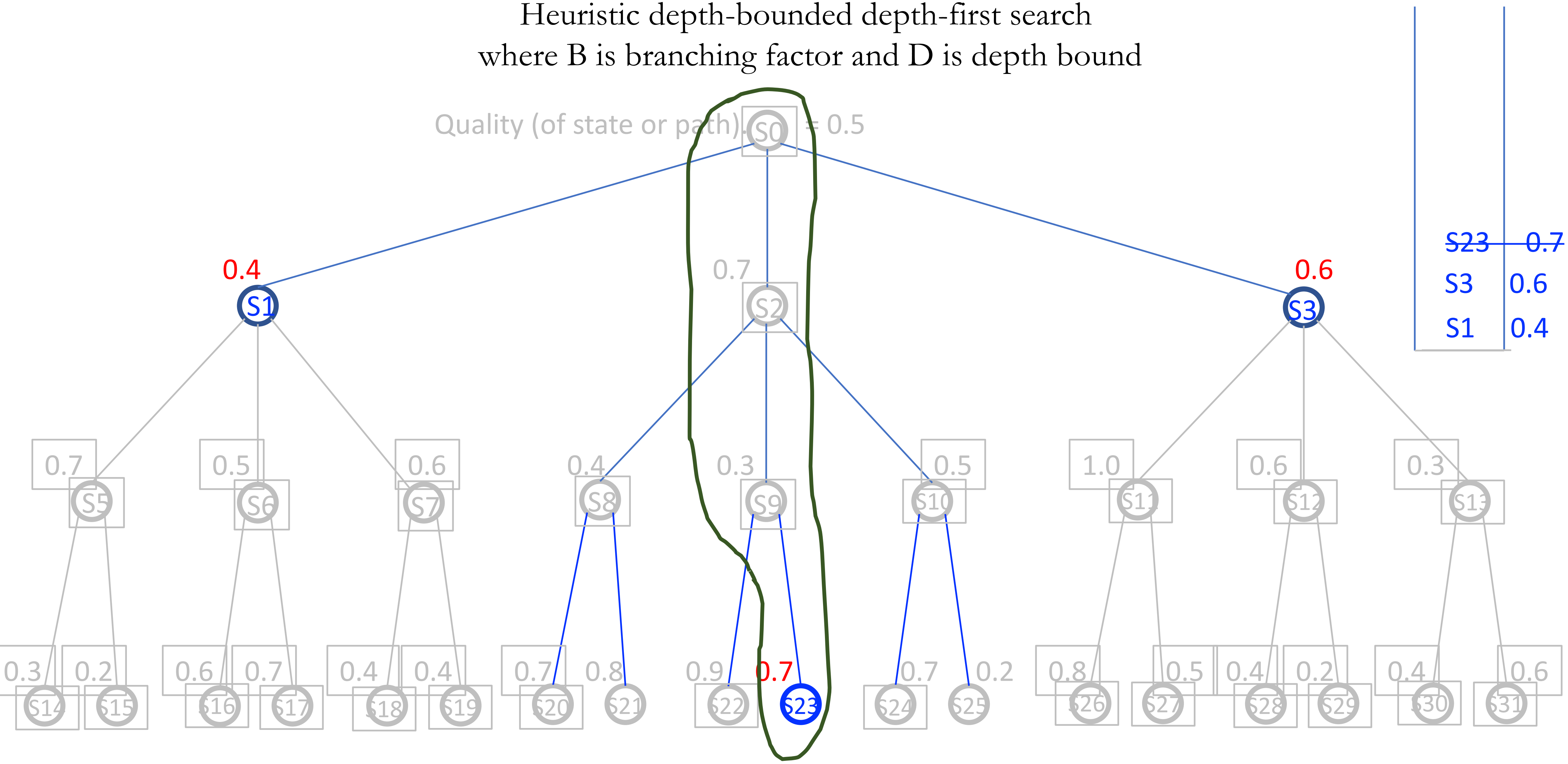
S22	0.9
S23	0.7
S3	0.6
S1	0.4

Heuristic depth-bounded depth-first search
where B is branching factor and D is depth bound



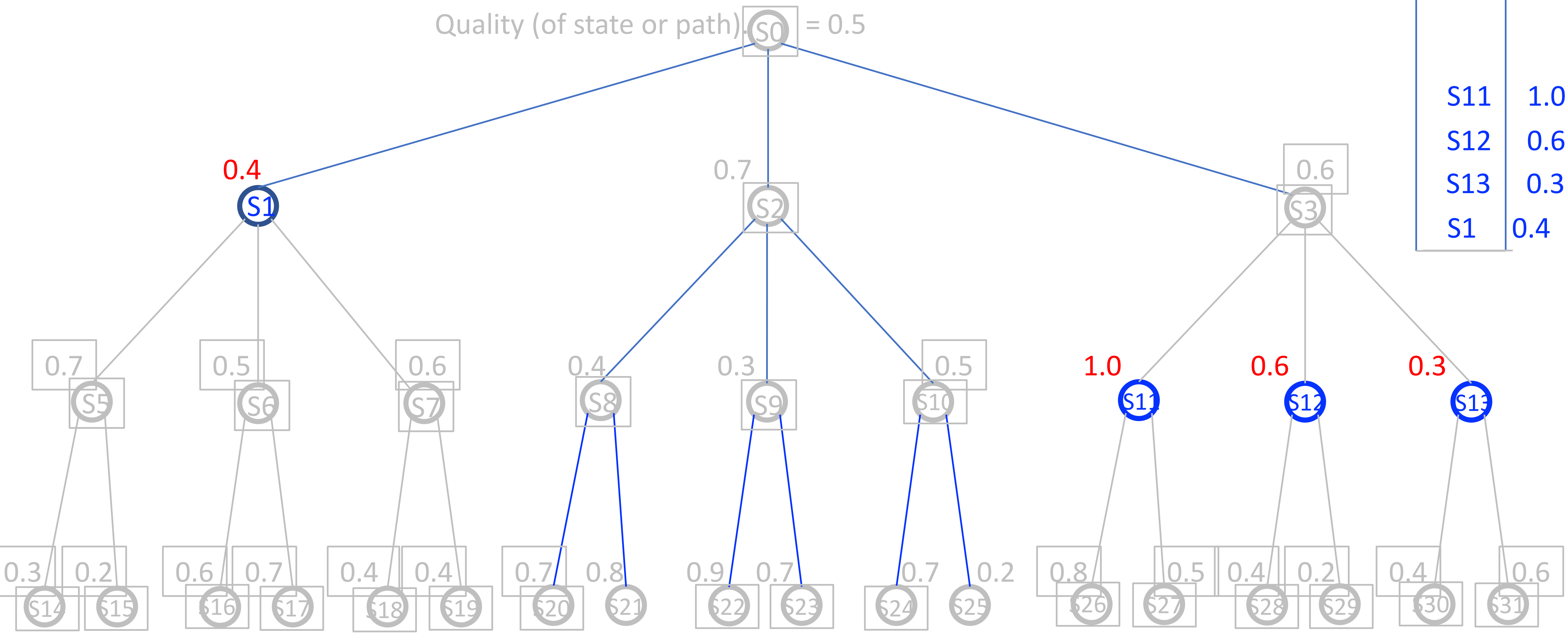
~~S22~~	~~0.9~~
S23	0.7
S3	0.6
S1	0.4

Heuristic depth-bounded depth-first search where B is branching factor and D is depth bound



~~S23~~	~~0.7~~
S3	0.6
S1	0.4

Heuristic depth-bounded depth-first search
where B is branching factor and D is depth bound



S11	1.0
S12	0.6
S13	0.3
S1	0.4

Heuristic depth-bounded depth-first search where B is branching factor and D is depth bound

What recommends HDBDFS as an anytime strategy?

- ”Fast” discovery of initial schedule
- Fast, predictably-paced discovery of subsequent schedules
- Worst-case space requirements are O(B*D) rather than O(B^D) for an unbounded priority queue search
- Less space-motivated reason to ever exclude part of the search space

What are the downsides/limits of HDBDFS?

- Advances local, rather than global best paths
- Very similar solutions (schedules) are enumerated back to back (only varying at the “end points”
  - To address this, perhaps implement a kind of diversity search, where the order of which successors are placed on the stack uses a random draw from a probability distribution that “mirrors” the state (or path) quality