Chapter S:III

III. Informed Search

- Best-First Search Basics
- Best-First Search Algorithms
- Cost Functions for State-Space Graphs
- Evaluation of State-Space Graphs
- Algorithm A*

- BF* Variants
- Hybrid Strategies

- Best-First Search for AND-OR Graphs
- Relation between GBF and BF
- Cost Functions for AND-OR Graphs
- Evaluation of AND-OR Graphs

Cost Functions for AND-OR Graphs

Overview

Similar to BF, also GBF defines a schema for the design of search strategies. Up to this point, the evaluation functions f_1 remained unspecified.

Questions:

- How to compute f_1?
- How to evaluate a solution graph?
- How to evaluate a search space graph?
- How to identify a most promising solution base?

These question are answered analogously to the BF case.

Notice the differences: solution graphs are considered instead of solution paths.

Remarks:

- In fact, implicitly, we have already given answers during the Illustration of GBF before. What will be added in the following are the foundations for these answers, along with a definite semantics.

Cost Functions for AND-OR Graphs

Overview (continued) [Overview for BF]

The answers are developed in several steps by the following concepts:

1. Recursive cost functions (for graphs)
2. Solution cost (for a given solution graph)
3. Optimum solution cost (for a complete search space graph)
4. Estimated solution cost (for a given solution base)
5. Estimated optimum solution cost (for a partial search space graph)

Names of the respective cost functions:

		Solution	
		given	optimum searched
Exploration	complete	C_H	C^*
	partial	$\widehat{C}_H$	$\widehat{C}$

Cost Functions for AND-OR Graphs

Overview (continued) [Overview for BF]

The answers are developed in several steps by the following concepts:

1. Recursive cost functions (for graphs)
2. Solution cost (for a given solution graph)
3. Optimum solution cost (for a complete search space graph)
4. Estimated solution cost (for a given solution base)
5. Estimated optimum solution cost (for a partial search space graph)

Names of the respective cost functions:

		Solution	
		given	optimum searched
Exploration	complete	$C_H(s)$	$C^*(s)$
	partial	$\widehat{C}_H(s)$	$\widehat{C}(s) \rightsquigarrow H$

H = most promising solution base.

Cost Functions for AND-OR Graphs

If solution graphs are known, the solution cost for a solution graph can be determined.

Definition 43 (Cost Function C_H)

Let G be an acyclic AND-OR graph and let M be an ordered set. A function C_H, which assigns to each solution graph H and each node n in G a cost value $C_H(n)$ in M, is called a cost function (for G).

Usage and notation of C_H :

- No provisions are made how to compute $C_H(n)$ for a solution graph H. $C_H(s)$ specifies the cost of a solution graph H for s :

$$f_1(H) = C_H(s).$$

Remarks:

- Again, as ordered set M usually $\mathbf{R} \cup \{\infty\}$ is chosen.
- $C_H(n)$ should be seen as binary function with arguments H and n.
- The cost value $C_H(n)$ is meaningful only if n is a node in H.
- Solution cost does not measure efforts for finding a solution. Solution cost aggregate properties of operations and decompositions in a solution graph to form a cost value.
- Instead of cost functions we may employ merit functions or, even more general, weight functions. The respective notations are Q_H for merits, and W_H for weights.

Cost Functions for AND-OR Graphs

If the entire search space graph rooted at a node s is known, the optimum solution cost for the root node s can be determined.

Definition 44 (Optimum Solution Cost C^*, Optimum Solution)

Let G be an acyclic AND-OR graph with root node s and let $C_H(n)$ denote a cost function for G.

The optimum solution cost for a node n in G, $C^*(n)$, is defined as

$$C^*(n) = \inf\{C_H(n) \mid H \text{ is solution graph for } n \text{ in } G\}$$

A solution graph with solution cost $C^*(n)$ is called optimum solution graph for n. The optimum solution cost for s, $C^*(s)$, is abbreviated as C^*.

Cost Functions for AND-OR Graphs

If the entire search space graph rooted at a node s is known, the optimum solution cost for the root node s can be determined.

Definition 44 (Optimum Solution Cost C^*, Optimum Solution)

Let G be an acyclic AND-OR graph with root node s and let $C_H(n)$ denote a cost function for G.

The optimum solution cost for a node n in G, $C^*(n)$, is defined as

$$C^*(n) = \inf\{C_H(n) \mid H \text{ is solution graph for } n \text{ in } G\}$$

A solution graph with solution cost $C^*(n)$ is called optimum solution graph for n. The optimum solution cost for s, $C^*(s)$, is abbreviated as C^*.

Remarks:

- If G contains no solution graph for n, let $C^*(n) = \infty$.

Cost Functions for AND-OR Graphs

If the entire search space graph rooted at a node s is known, the optimum solution cost extending a solution base for s can be determined.

Definition 45 (Optimum Solution Cost C^*_H for a Solution Base)

Let G be an acyclic AND-OR graph with root node s and let $C_H(n)$ denote a cost function for G.

The optimum solution cost, $C^*_H(n)$, for a node n in a solution base H in G is defined as

$$C^*_H(n) = \inf\{C_{H'}(n) \mid H' \text{ is solution graph in } G \text{ extending } H\}$$

Remarks:

- In the setting of Definition 45 we assume:

$$C^*(s) = \min\{C^*_H(s) \mid H \text{ is solution base maintasined by an algorithm}\}$$

  The solution bases maintained by algorithm GBF are the maximal solution bases for s in the explored part G_e of the search space graph G.

  Therefore, it is essential for search algorithms to keep available solution bases that are important for this result. Optimistically estimating $C^*_H(n)$ in GBF will direct the search into promising directions.

Cost Functions for AND-OR Graphs

If the search space graph rooted at a node s is known partially, the optimum solution cost extending a solution base for s can be estimated.

Definition 46 (Estimated Solution Cost $\widehat{C}_H$ for a Solution Base)

Let G be an acyclic AND-OR graph with root node s and let $C_H(n)$ denote a cost function for G.

The estimated solution cost, $\widehat{C}_H(n)$, for a node n in a solution base H in G returns an estimate of $C^*_H(n)$.

$\widehat{C}_H(n)$ is optimistic, if and only if $\widehat{C}_H(n) \leq C^*_H(n)$.

Usage of $\widehat{C}_H$:

- In GBF we use $f_1(H) = \widehat{C}_H(s)$ with solution bases H for s in the explored part of the search space graph G.
- $f_1(H)$ is optimistic, if $f_1(H) \leq C^*_H(s)$.

Cost Functions for AND-OR Graphs

If the search space graph rooted at a node s is known partially, the optimum solution cost extending a solution base for s can be estimated.

Definition 46 (Estimated Solution Cost $\widehat{C}_H$ for a Solution Base)

Let G be an acyclic AND-OR graph with root node s and let $C_H(n)$ denote a cost function for G.

The estimated solution cost, $\widehat{C}_H(n)$, for a node n in a solution base H in G returns an estimate of $C^*_H(n)$.

$\widehat{C}_H(n)$ is optimistic, if and only if $\widehat{C}_H(n) \leq C^*_H(n)$.

Usage of $\widehat{C}_H$:

- In GBF we use $f_1(H) = \widehat{C}_H(s)$ with solution bases H for s in the explored part of the search space graph G.
- $f_1(H)$ is optimistic, if $f_1(H) \leq C^*_H(s)$.

Cost Functions for AND-OR Graphs

If the search space graph rooted at a node s is known partially, the optimum solution cost for s can be estimated.

Definition 47 (Estimated Optimum Solution Cost $\widehat{C}$ [Overview])

Let G be an acyclic AND-OR graph with root node s and let $C_H(n)$ denote a cost function for G. Further, let G_e be a finite (explored) subgraph of G with root node s.

The estimated optimum solution cost for a node n in G, $\widehat{C}(n)$, is defined as follows:

$$\widehat{C}(n) = \min\{\widehat{C}_H(n) \mid H \text{ is maximal solution base in } G_e\}$$

A solution base H for s with $\widehat{C}_H(s) = \widehat{C}(s)$ is called most promising solution base (for s).

Cost Functions for AND-OR Graphs

If the search space graph rooted at a node s is known partially, the optimum solution cost for s can be estimated.

Definition 47 (Estimated Optimum Solution Cost $\widehat{C}$ [Overview])

Let G be an acyclic AND-OR graph with root node s and let $C_H(n)$ denote a cost function for G. Further, let G_e be a finite (explored) subgraph of G with root node s.

The estimated optimum solution cost for a node n in G, $\widehat{C}(n)$, is defined as follows:

$$\widehat{C}(n) = \min\{\widehat{C}_H(n) \mid H \text{ is maximal solution base in } G_e\}$$

A solution base H for s with $\widehat{C}_H(s) = \widehat{C}(s)$ is called most promising solution base (for s).

For an algorithm searching an acyclic AND-OR-graph G, the finite explored part G_e of G defines the solution bases considered: the tip nodes of a solution base have to be tip nodes in G_e.

Cost Functions for AND-OR Graphs

Recursive Cost Functions

The computation of the evaluation functionsr f_1 would be infeasible

1. if each possible solution base had to be analyzed in isolation, or
2. if the cost of a solution base had to be computed from scratch for each additionally explored node.

➜ Utilization of recursive cost functions to implement the evaluation function f_1.

Efficiency benefits of recursive functions:

1. Shared computation.
   Already computed evaluation results for solution bases H are exploited for other solution bases that contain H.
2. Selective updating.
   Only the predecessors of a newly explored node need to be updated.

Cost Functions for AND-OR Graphs

Recursive Cost Functions (continued)

Definition 48 (Recursive Cost Function, Cost Measure)

A cost function C_H for a solution graph H is called recursive, if for each node n in H it holds:

$$C_H(n) = \begin{cases} F[E(n)] & n \text{ is goal node and leaf in } H \\ F[E(n), C_H(n')] & n \text{ is inner OR node in } H, \\ & n' \text{ direct successor of } n \text{ in } H \\ F[E(n), C_H(n_1), \ldots, C_H(n_k)] & n \text{ is inner AND node in } H, \\ & n_1, \ldots, n_k \text{ direct successors of } n \text{ in } H \end{cases}$$

- $n_1, n_2, \ldots, n_k$ denote the direct successors of n in H,
- $E(n) \in \mathbf{E}$ denotes a set of *local* properties of n with respect to H,
- F is a function that prescribes how local properties of n are accounted (better: combined) with properties of the direct successors of n:

$$F : \mathbf{E} \times M^k \to M, \quad \text{where } M \text{ is an ordered set.}$$

F is called cost measure.

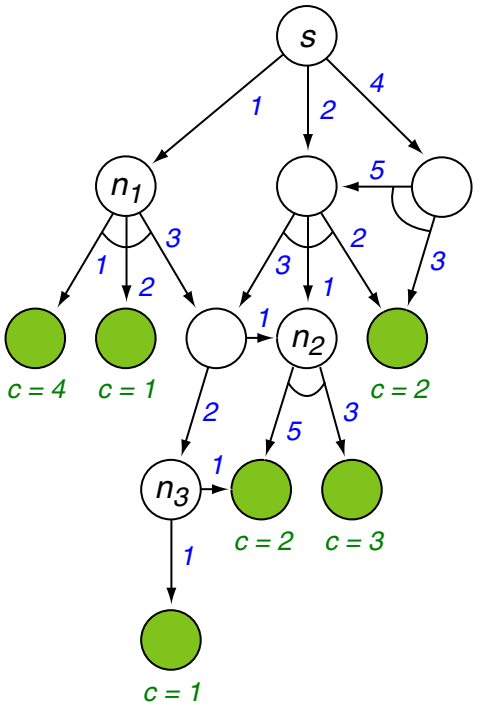
Remarks:

- Observe that for each node n in a solution graph H the complete subgraph of H that is rooted at n forms a solution graph for n (neglecting additional solution constraints).
- As a shorthand we use $c(n) := F[E(n)]$ for the remaining cost of a nontrivial goal node. Often we have $(c(n) = 0$.
- The computation of $C_H(n)$ is called cost propagation. If $C_H(n)$ fulfills the conditions of a recursive cost function, it can be computed bottom-up, similar to the solved-labeling procedure.
- The function $C_H(n)$ employs the the same $E(n)$ and the same cost measure F for all possible solution graphs of a search space graph.
  As a consequence, cost value computation is context insensitive: If a solution graph for node n is contained in two solution graphs H and H', then $C_H(n) = C_{H'}(n)$ holds.

Cost Functions for AND-OR Graphs

Illustration of C_H [Overview, Illustration of C^*]

Underlying problem-reduction graph:



$E(n) = c(n, n') =$ cost for edge (n, n')

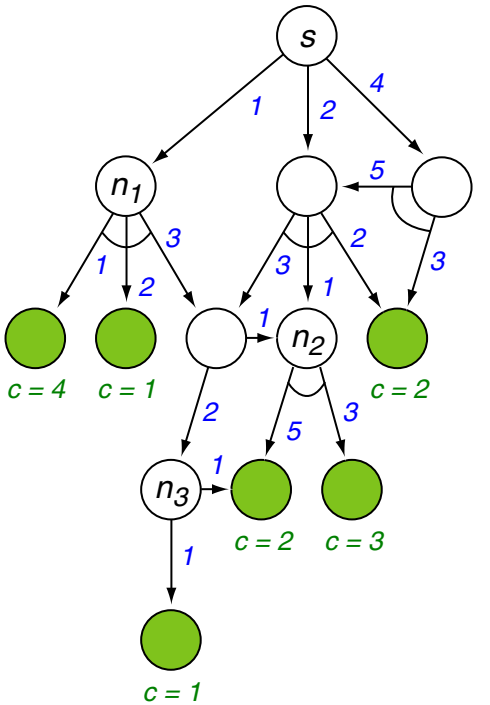
$$C_H(n) = \begin{cases} c(n) & n \text{ is goal node and leaf in } H \\ c(n, n') + C_H(n') & n \text{ is inner OR node in } H, \\ & n' \text{ direct successor of } n \text{ in } H \\ \max_i\{c(n, n_i) + C_H(n_i)\} & n \text{ is inner AND node in } H, \\ & n_i \text{ direct successors of } n \text{ in } H \end{cases}$$

Cost Functions for AND-OR Graphs

Illustration of C_H [Overview, Illustration of C^*]

$H \equiv$ solution graph for s :

$C_H(s) = 12$



$E(n) = c(n, n') =$ cost for edge (n, n')

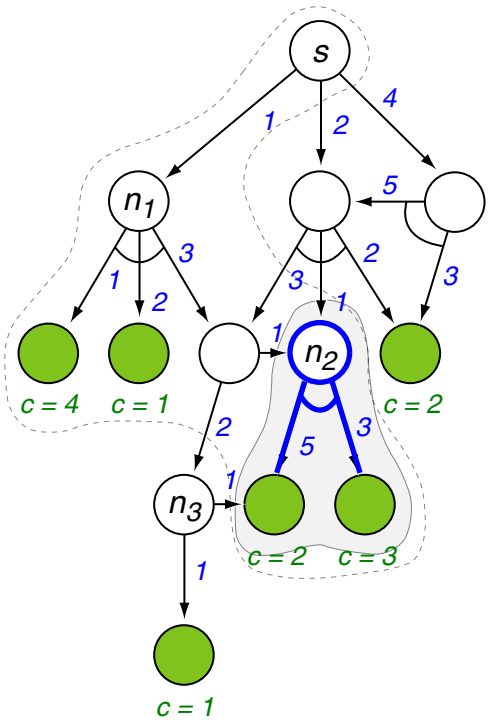
$$C_H(n) = \begin{cases} c(n) & n \text{ is goal node and leaf in } H \\ c(n, n') + C_H(n') & n \text{ is inner OR node in } H, \\ & n' \text{ direct successor of } n \text{ in } H \\ \max_i\{c(n, n_i) + C_H(n_i)\} & n \text{ is inner AND node in } H, \\ & n_i \text{ direct successors of } n \text{ in } H \end{cases}$$

Cost Functions for AND-OR Graphs

Illustration of C_H [Overview, Illustration of C^*]

Solution graph for n_2 :

$C_H(n_2) = 7$



$E(n) = c(n, n') =$ cost for edge (n, n')

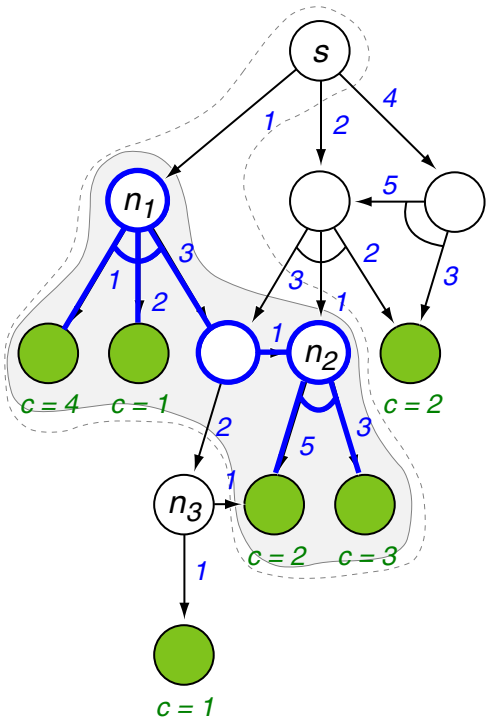
$$C_H(n) = \begin{cases} c(n) & n \text{ is goal node and leaf in } H \\ c(n, n') + C_H(n') & n \text{ is inner OR node in } H, \\ & n' \text{ direct successor of } n \text{ in } H \\ \max_i\{c(n, n_i) + C_H(n_i)\} & n \text{ is inner AND node in } H, \\ & n_i \text{ direct successors of } n \text{ in } H \end{cases}$$

Cost Functions for AND-OR Graphs

Illustration of C_H [Overview, Illustration of C^*]

Solution graph for n_1 :

$C_H(n_1) = 11$



$E(n) = c(n, n') =$ cost for edge (n, n')

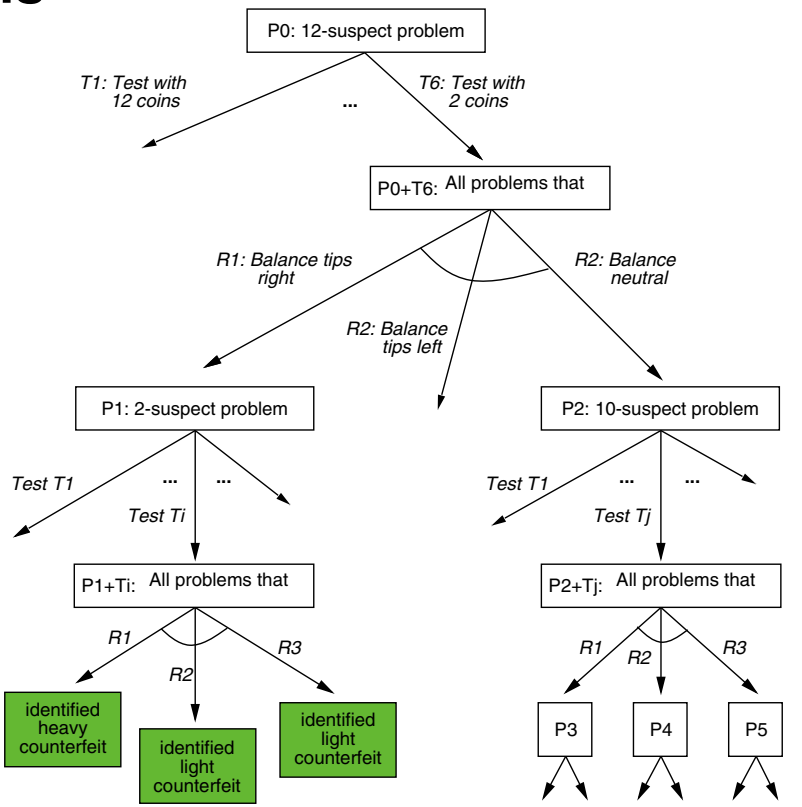
$$C_H(n) = \begin{cases} c(n) & n \text{ is goal node and leaf in } H \\ c(n, n') + C_H(n') & n \text{ is inner OR node in } H, \\ & n' \text{ direct successor of } n \text{ in } H \\ \max_i\{c(n, n_i) + C_H(n_i)\} & n \text{ is inner AND node in } H, \\ & n_i \text{ direct successors of } n \text{ in } H \end{cases}$$

Cost Functions for AND-OR Graphs

Illustration of C_H [Overview, Illustration of C^*, $\widehat{C}$]

Solution graph for the counterfeit problem:

- Leaf nodes correspond to identified coin.
- OR nodes (action, strategy) specify the chosen test.
- AND nodes (reaction) model the weighing outcomes.



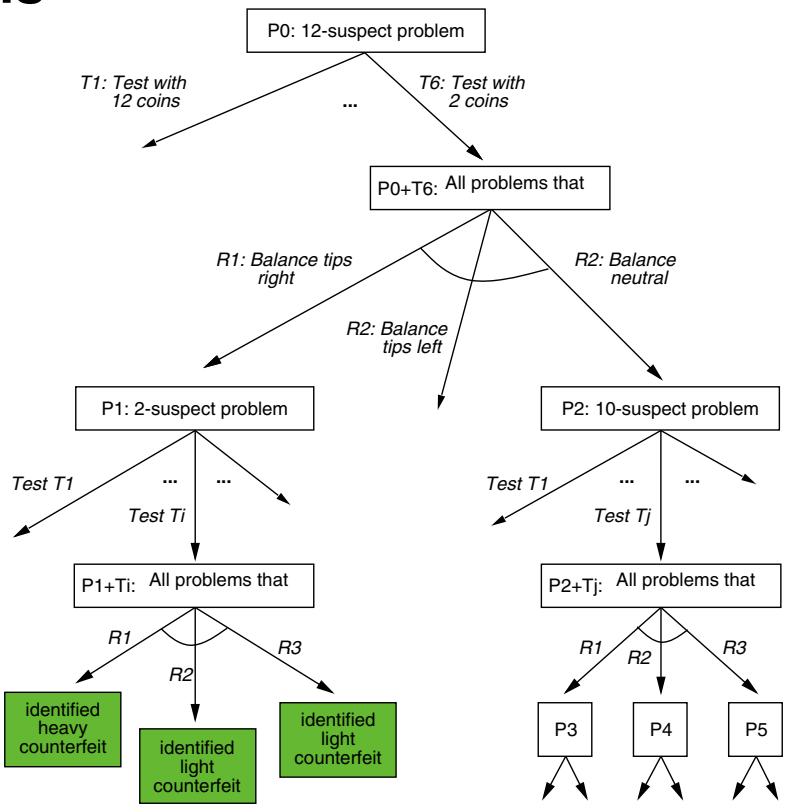
$$C_H(n) = \begin{cases} 0 & n \text{ is goal node and leaf in } H \\ 1 + C_H(n') & n \text{ is inner OR node in } H, \\ & n' \text{ direct successor of } n \text{ in } H \\ \max\{C_H(n_i) \mid i = 1,2,3\} & n \text{ is inner AND node in } H, \\ & n_i \text{ direct successors of } n \text{ in } H \end{cases}$$

Cost Functions for AND-OR Graphs

Illustration of C_H [Overview, Illustration of C^*, $\widehat{C}$]

Solution graph for the counterfeit problem:

- Leaf nodes correspond to identified coin.
- OR nodes (action, strategy) specify the chosen test.
- AND nodes (reaction) model the weighing outcomes.



$$C_H(n) = \begin{cases} 0 & n \text{ is goal node and leaf in } H \\ 1 + C_H(n') & n \text{ is inner OR node in } H, \\ & n' \text{ direct successor of } n \text{ in } H \\ \sum_{i=1,2,3} p_i(n) \cdot C_H(n_i) & n \text{ is inner AND node in } H, \\ & n_i \text{ direct successors of } n \text{ in } H \end{cases}$$

Remarks:

- The recursive cost function $C_H(n)$ with $\max\{C_H(n_i) \mid i = 1, 2, 3\}$ in the AND-node branch models the maximum required number of tests (worst case costs) using the weighing operations in the solution graph H.

- The recursive cost function $C_H(n)$ with $\sum_{i=1,2,3} p_i(n) \cdot C_H(n_i)$ in the AND-node branch models the expected test effort. The $p_i, i = 1, 2, 3$ are the probabilities for the test outcomes and could be quantified as follows:

$$p_i(n) = \begin{cases} \frac{1}{2} \cdot \frac{k(n)}{12-u(n)} & i \in \{1, 3\} \\ 1 - \frac{k(n)}{12-u(n)} & i = 2 \end{cases}$$

  where $k(n)$, $k(n) \leq 12$, is number of suspicious coins that are weighed in subproblem n, and $u(n)$, $u(n) < 12$, is the number of coins classified as "not-suspect" in subproblem n.

Cost Functions for AND-OR Graphs

Recursive Cost Functions (continued)

If the search space graph rooted at a node s is known partially and a recursive cost function is used, cost estimates for a solution base

1. can be built upon estimates for optimum solution cost of non-goal leaf nodes in this solution base and,
2. can be computed by taking the estimations of h for granted and propagating the cost values bottom-up. Keyword: *Face-Value Principle*

Definition 49 (Heuristic Function h)

Let G be an acyclic AND-OR graph. A function h, which assigns each node n in G an estimate $h(n)$ of the optimum solution cost value $C^*(n)$, the optimum cost of a solution graph for n, is called heuristic function (for G).

Cost Functions for AND-OR Graphs

Recursive Cost Functions (continued)

Corollary 50 (Estimated Solution Cost $\widehat{C}_H$ for a Solution Base)

Let G be an acyclic AND-OR graph with root node s and let $C_H(n)$ denote a cost function for G.

Further, let the cost function be recursive based on F and E, and let h be a heuristic function.

Using the face-value principle, the estimated solution cost for solution base H in G is computed as follows:

$$\widehat{C}_H(n) = \begin{cases} c(n) & n \text{ is goal node and leaf in } H \\ h(n) & n \text{ is leaf in } H \text{ but no goal node} \\ F[E(n), \widehat{C}_H(n')] & n \text{ is inner OR node in } H, \\ & n' \text{ direct successor of } n \text{ in } H \\ F[E(n), \widehat{C}_H(n_1), \ldots, \widehat{C}_H(n_k)] & n \text{ is inner AND node in } H, \\ & n_i \text{ direct successors of } n \text{ in } H \end{cases}$$

Chapter S:III

III. Informed Search

- Best-First Search Basics
- Best-First Search Algorithms
- Cost Functions for State-Space Graphs
- Evaluation of State-Space Graphs
- Algorithm A*

- BF* Variants
- Hybrid Strategies

- Best-First Search for AND-OR Graphs
- Relation between GBF and BF
- Cost Functions for AND-OR Graphs
- Evaluation of AND-OR Graphs

Evaluation of AND-OR Graphs

Recursive Cost Functions and Efficiency

If the search space graph is an acyclic AND-OR graph rooted at a node s and is known partially and a recursive cost function is used that is defined via a

1. monotone cost measure F, i.e., for e, c, c' with $c_1 \leq c'_1, \ldots, c_k \leq c'_k$ we have

$$F[e, c_1, \ldots, c_k] \leq F[e, c'_1, \ldots, c'_k]$$

the (estimated) optimum solution cost can be computed bottom-up.

A solution base can be determined which has the estimated optimum solution cost as its estimated solution cost.

If additionally the recursive cost function is based on an

2. underestimating heuristic function h, i.e., $h(n) \leq C^*(n)$

then the estimated solution cost $\widehat{C}_H(s)$ is underestimating optimum solution cost $C^*_H(s)$ for a solution base H.

Evaluation of AND-OR Graphs

Recursive Cost Functions and Efficiency (continued)

Corollary 51 (Optimum Solution Cost C^* [BF, Overview])

Let G be an acyclic AND-OR graph rooted at s. Let $C_H(n)$ be a recursive cost function for G based on E and a monotone cost measure F.

The optimum solution cost $C^*(n)$ for a node n in G can be computed as follows:

$$C^*(n) = \begin{cases} c(n) & n \text{ is goal node and leaf in } G \\ \infty & n \text{ is unsolvable leaf node in } G \\ \min_i\{F[E(n), C^*(n_i)]\} & n \text{ is inner OR node in } G, \\ & n_i \text{ direct successors of } n \text{ in } G \\ F[E(n), C^*(n_1), \ldots, C^*(n_k)] & n \text{ is inner AND node in } G, \\ & n_i \text{ direct successors of } n \text{ in } G \end{cases}$$

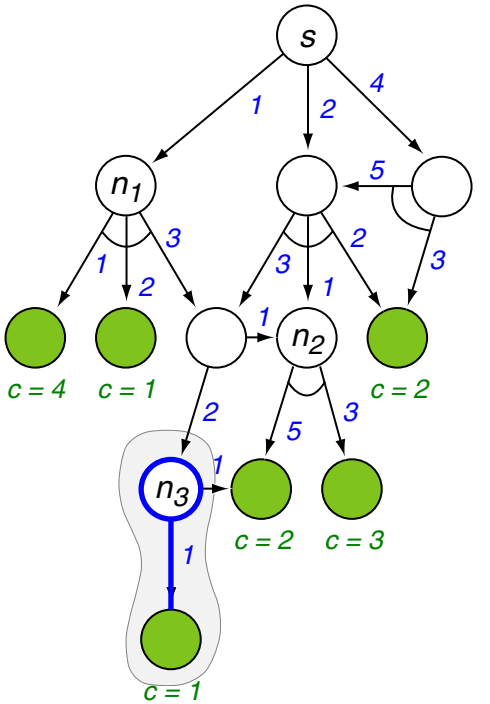
Compare to Bellman's equations.

Evaluation of AND-OR Graphs

Illustration of C^* [Overview, Illustration of C_H]

Optimum solution graph for s:

$$C^*(s) \equiv C^* = 8$$



$E(n) = c(n, n') =$ cost for edge (n, n')

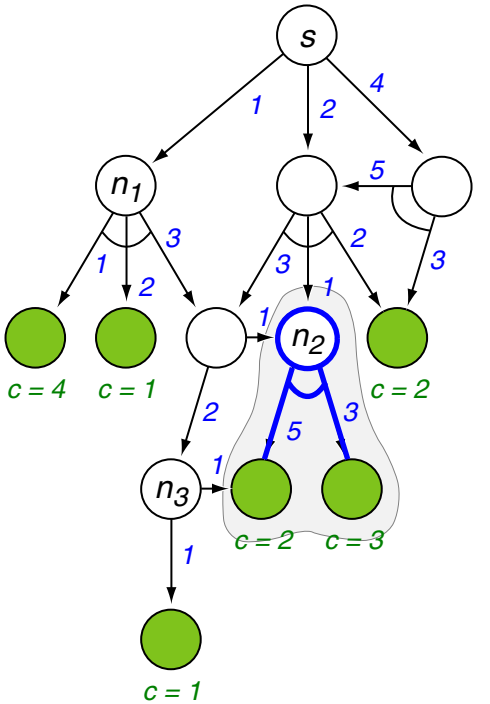
$$C^*(n) = \begin{cases} c(n) & n \text{ is goal node and leaf in } G \\ \min_i\{c(n, n_i) + C^*(n_i)\} & n \text{ is inner OR node in } G, \\ & n_i \text{ direct successors of } n \text{ in } G \\ \max_i\{c(n, n_i) + C^*(n_i)\} & n \text{ is inner AND node in } G, \\ & n_i \text{ direct successors of } n \text{ in } G \end{cases}$$

Evaluation of AND-OR Graphs

Illustration of C^* [Overview, Illustration of C_H]

Optimum solution graph for n_3:

$C^*(n_3) = 2$



$E(n) = c(n, n') =$ cost for edge (n, n')

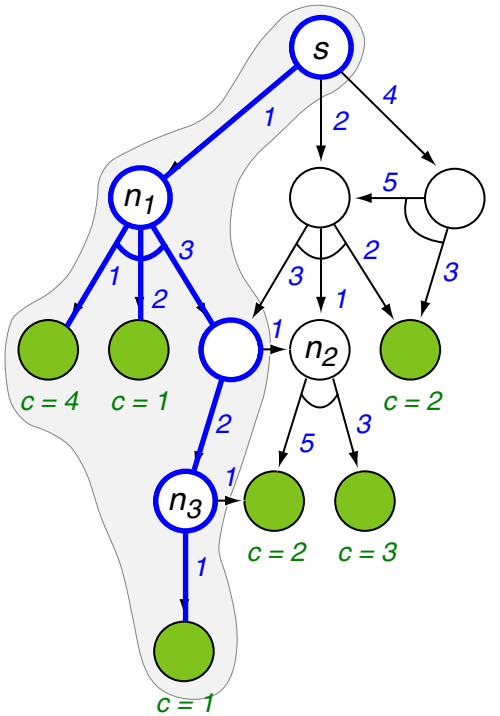
$$C^*(n) = \begin{cases} c(n) & n \text{ is goal node and leaf in } G \\ \min_i\{c(n, n_i) + C^*(n_i)\} & n \text{ is inner OR node in } G, \; n_i \text{ direct successors of } n \text{ in } G \\ \max_i\{c(n, n_i) + C^*(n_i)\} & n \text{ is inner AND node in } G, \; n_i \text{ direct successors of } n \text{ in } G \end{cases}$$

Evaluation of AND-OR Graphs

Illustration of C^* [Overview, Illustration of C_H]

Optimum solution graph for n_2:

$C^*(n_2) = 7$



$E(n) = c(n, n') =$ cost for edge (n, n')

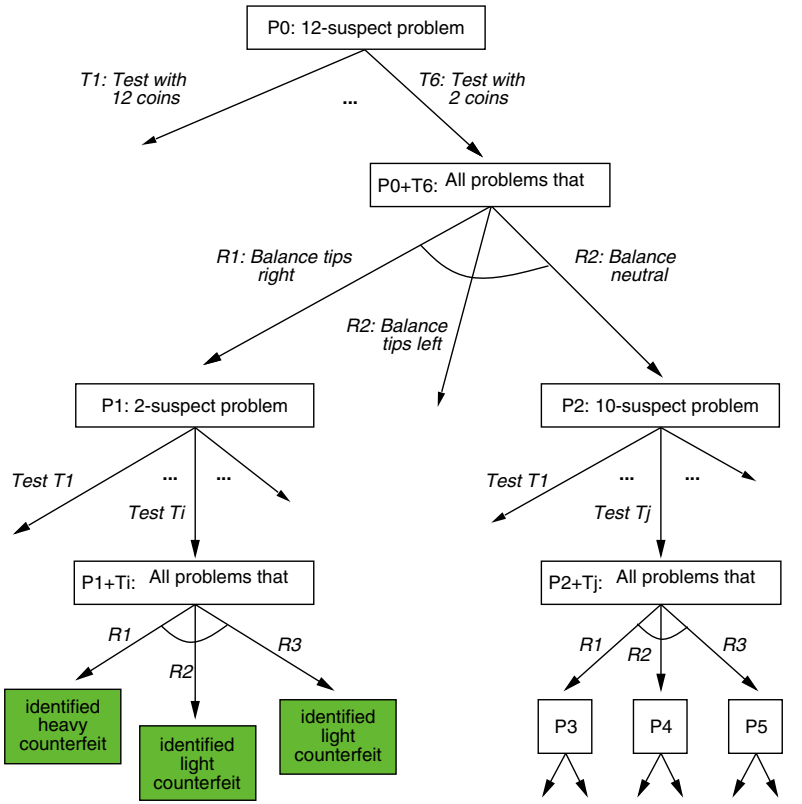
$$C^*(n) = \begin{cases} c(n) & n \text{ is goal node and leaf in } G \\ \min_i\{c(n, n_i) + C^*(n_i)\} & n \text{ is inner OR node in } G, \; n_i \text{ direct successors of } n \text{ in } G \\ \max_i\{c(n, n_i) + C^*(n_i)\} & n \text{ is inner AND node in } G, \; n_i \text{ direct successors of } n \text{ in } G \end{cases}$$

Evaluation of AND-OR Graphs

] Illustration of C^* [Overview, Illustration of C_H $\widehat{C}$]

Search space graph for the counterfeit problem:

- Leaf nodes correspond to identified coin.
- OR nodes (action, strategy) specify the possible tests.
- AND nodes (reaction) model the weighing outcomes.



$$C^*(n) = \begin{cases} 0 & n \text{ is goal node and leaf in } G \\ \min_i\{1 + C^*(n_i)\} & n \text{ is inner OR node in } G, \; n_i \text{ direct successors of } n \text{ in } G \\ \sum_{i=1,2,3} p_i \cdot C^*(n_i) & n \text{ is inner AND node in } G, \; n_i \text{ direct successors of } n \text{ in } G \end{cases}$$

Remarks:

- If the underlying search space graph is infinite, it may be impossible to compute $C^*(n)$. Exploiting properties of E and F for a search space graph can make the computation feasible.

Evaluation of AND-OR Graphs

Recursive Cost Functions and Efficiency (continued)

Corollary 52 (Estimated Optimum Solution Cost $\widehat{C}$ [BF, Overview])

Let G be an acyclic AND-OR graph with root node s. Let h be a heuristic function and let $C_H(n)$ be a recursive cost function for G based on E and a monotone cost measure F. Further, let G_e be a finite (explored) subgraph of G with root node s.

Using the face-value principle, the estimated optimum solution cost $\widehat{C}(n)$ for a node n in G can be computed as follows:

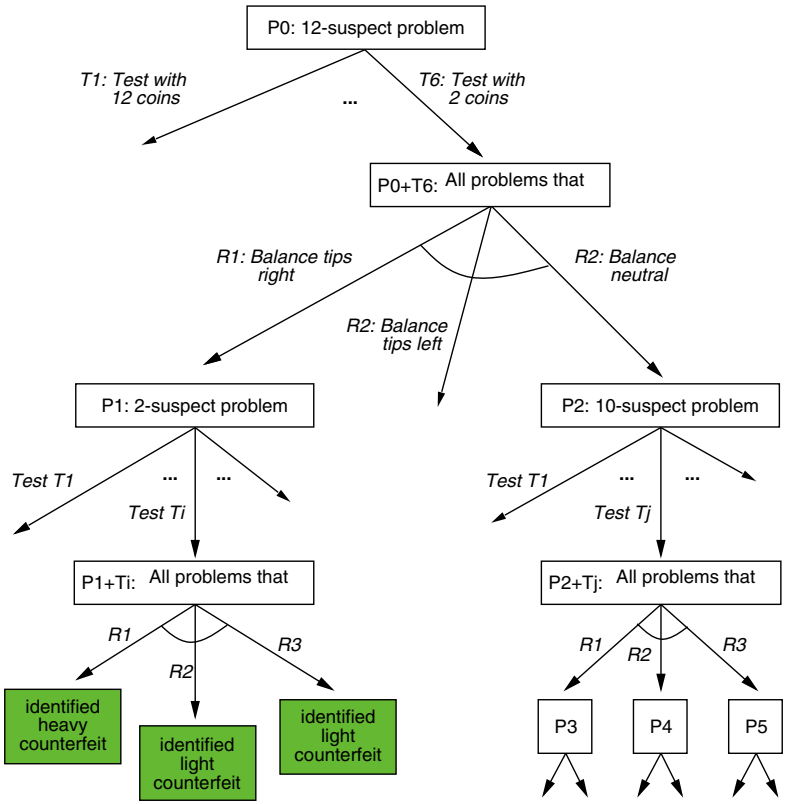
$$\widehat{C}(n) = \begin{cases} c(n) & n \text{ is goal node and leaf in } G_e \\ h(n) & n \text{ is leaf in } G_e \text{ but no goal node} \\ \min_i\{F[E(n), \widehat{C}(n_i)]\} & n \text{ is inner OR node in } G_e, \\ & n_i \text{ direct successors of } n \text{ in } G \\ F[E(n), \widehat{C}(n_1), \ldots, \widehat{C}(n_k)] & n \text{ is inner AND node in } G_e, \\ & n_i \text{ direct successors of } n \text{ in } G \end{cases}$$

Remarks:

- The above calculation rules for $\widehat{C}$ requires a useful handling regarding the value ∞ (assuming $h(n) = \infty$ for unsolvable nodes n). In particular, the cost measure F should return ∞ if one of its arguments is ∞.
- The above calculation rules for $\widehat{C}$ presume the—already computed—estimated optimum solution cost values for the direct successors of n. Their existence is guaranteed here (also for infinitely large search space graphs) since the search (= the computation of $\widehat{C}$) uses a finite portions of the search space graph, the explored subgraphs of the search space graph.
- $\widehat{C}(n)$ computes for a node n the minimum of the estimated costs among all maximal solution bases rooted at n in G_e. In particular, $\widehat{C}(s)$ computes the estimated optimum solution cost for the entire problem, and it hence defines a most promising solution base H for s in G_e.
- Finally, cost propagation is reasonable (also in the sense of "is efficient") only for problems whose solution obey Bellman's principle of optimality.

Evaluation of AND-OR Graphs

Illustration of $\widehat{C}(n)$ [Overview, Illustration of C_H, C^*]



Search space graph G for the counterfeit problem:

- Leaf nodes correspond to identified coin.
- OR nodes (action, strategy) specify the possible tests.
- AND nodes (reaction) model the weighing outcomes.

G_e is explored subgraph of G generated so far.

$$\widehat{C}(n) = \begin{cases} 0 & n \text{ is goal node and leaf in } G_e \\ h(n) & n \text{ is leaf in } G_e \text{ but no goal node} \\ \min_i\{1 + \widehat{C}(n_i)\} & n \text{ is inner OR node in } G_e, \\ & n_i \text{ direct successors of } n \text{ in } G \\ \max\{\widehat{C}(n_i) \mid i = 1,2,3\} & n \text{ is inner AND node in } G_e, \\ & n_i \text{ direct successors of } n \text{ in } G \end{cases}$$

Evaluation of AND-OR Graphs

Relation to the Algorithm GBF

GBF$^*(s,$ *successors*$, \bot, \star, f_1, f_2)$

```
      ...
  2.  LOOP
  3.    IF (OPEN = ∅) THEN RETURN(Fail);
4.a     H = min_solution_base(s, G_e, f_1);   // Find most prom. sol. base in G_e.
        ...
```

Evaluation of AND-OR Graphs

Relation to the Algorithm GBF

GBF*$(s,$ *successors*$, \bot, \star, f_1, f_2)$

```
      ...
 2.   LOOP
 3.     IF (OPEN = ∅) THEN RETURN(Fail);
4.a     H = min_solution_base(s, G_e, f_1);   // Find most prom. sol. base in G_e.
        ...
```

Corollary 53 (Most Promising Solution Base H [Overview])

Let G be an acyclic AND-OR graph with root node s. Let h be a heuristic function and let $C_H(n)$ be a recursive cost function for G based on E and a monotone cost measure F. Further, let G_e be a finite (explored) subgraph of G with root node s.

Using the face-value principle, a most promising solution base H for s can be characterized by the following conditions:

1. If H contains an inner OR node n, then H contains exactly one link to a direct successor n' in H, where
$$F[E(n), \widehat{C}(n')] = \min_i\{F[E(n), \widehat{C}(n_i)]\} \quad n_i \text{ direct successor of } n$$
2. If H contains an inner AND node, then H contains all links to its direct successors in H.

Evaluation of AND-OR Graphs

Relation to the Algorithm GBF [BF]

GBF*$(s, \textit{successors}, \bot, \star, f_1, f_2)$

```
      ...
 2.   LOOP
 3.     IF (OPEN = ∅) THEN RETURN(Fail);
4.a     H = min_solution_base(s, f_1);
        solved_labeling(H);    // Check if H is a solution graph.
        IF ⋆(s) THEN RETURN(H);    // Delayed termination.
        ...
```

Delayed termination:

➜ Algorithm GBF becomes Algorithm GBF*.

Define $f_1(H)$ as $\widehat{C}_H(s)$ for a solution base H in G_e:

➜ f_1 is a recursive evaluation function.

➜ Using an additive cost measure, algorithm GBF* becomes Algorithm AO*.

Remarks:

- If $\widehat{C}$ is based on a recursive cost function involving a monotone cost measure, the determination of a most promising solution base H can happen bottom-up along with the computation of $\widehat{C}(n)$ for n in G, simply by propagating the cheapest cost of the nodes and storing the source of this cheapest cost.
- The bottom-up propagation yields a minimal solution graph (or solution base) for each node in G_e.
- Leaf nodes in G_e are nodes on OPEN that are either solved rest problems or nodes that are currently not labeled “unsolvable”.
- Recall that $\widehat{C}_H(s)$ computes the estimated solution cost for a given solution base H. If h is optimistic and if, in addition, F is monotone, then $f_1(H)$ (defined as $\widehat{C}_H(s)$) is optimistic for all solution bases H.
  A proof of this claim is given in the lab class.

Evaluation of AND-OR Graphs

Taxonomy of Best-First Algorithms