# Chapter S:III

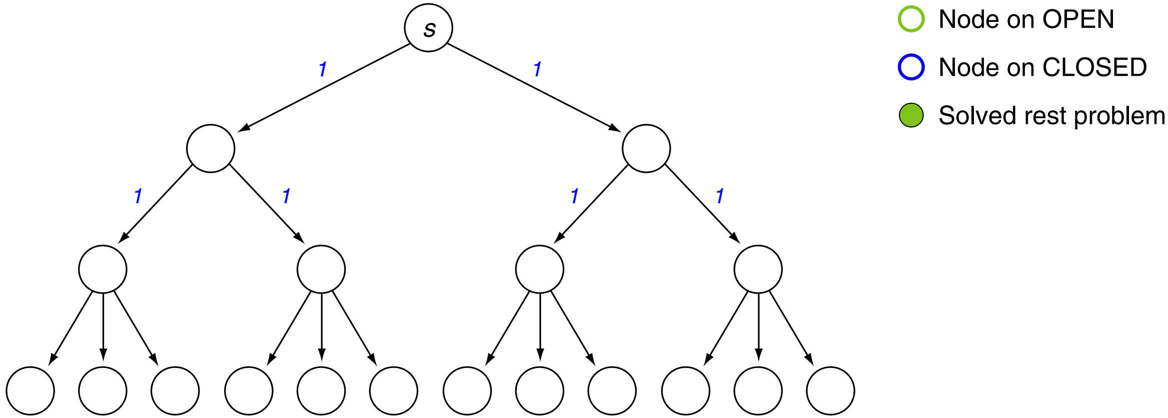
## III. Informed Search

# BF* Variants

For trees $G$: Breadth-first search is a special case of A*, where $h = 0$ and $c(n, n') = 1$ for all successors $n'$ of $n$.

# BF* Variants

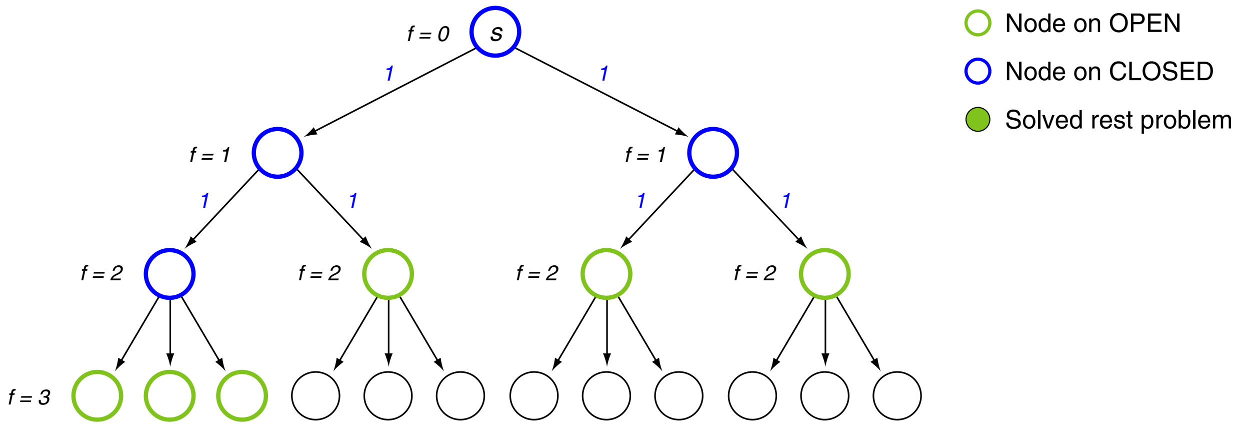
For trees $G$: Breadth-first search is a special case of A*, where $h = 0$ and $c(n, n') = 1$ for all successors $n'$ of $n$.



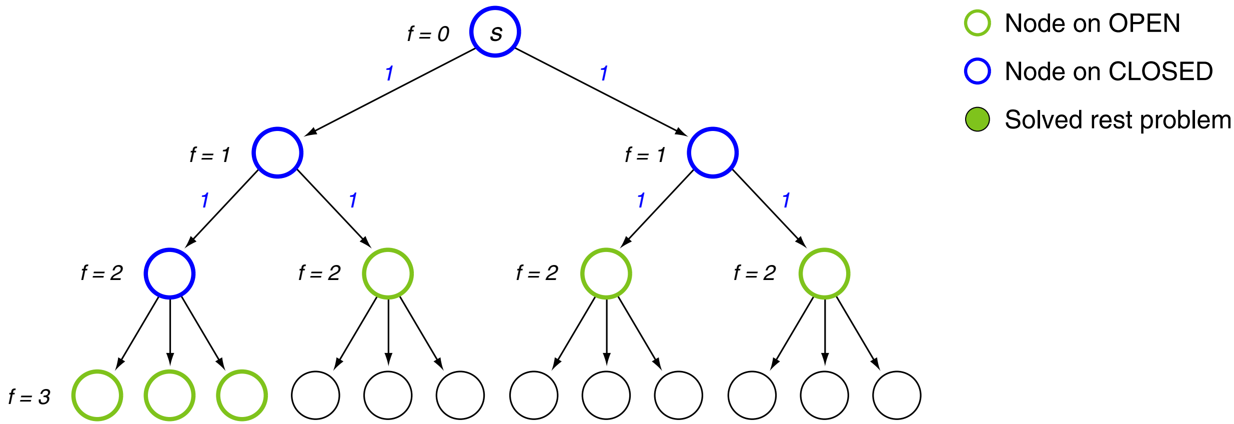
# BF* Variants

For trees $G$: Breadth-first search is a special case of A*, where $h = 0$ and $c(n, n') = 1$ for all successors $n'$ of $n$.



# BF* Variants

For trees $G$: Breadth-first search is a special case of A*, where $h = 0$ and $c(n, n') = 1$ for all successors $n'$ of $n$.




## Proof (sketch)

1. $g(n)$ defines the depth of $n$ (consider path from $n$ to $s$).
2. $f(n) = g(n)$.
3. Breadth-first search $\equiv$ the depth difference of nodes on OPEN is $\leq 1$.
4. Assumption: Let $n_1$, $n_2$ be on OPEN, having a larger depth difference: $f(n_2) - f(n_1) > 1$.
5. $\Rightarrow$ For the direct predecessor $n_0$ of $n_2$ holds: $f(n_0) = f(n_2) - 1 > f(n_1)$.
6. $\Rightarrow$ $n_1$ must have been expanded before $n_0$ (consider minimization of $f$ under A*).
7. $\Rightarrow$ $n_1$ must have been deleted from OPEN. Contradiction to 4.

# BF* Variants

For trees $G$: Uniform-cost search is a special case of A*, where $h = 0$.

## Proof (sketch)

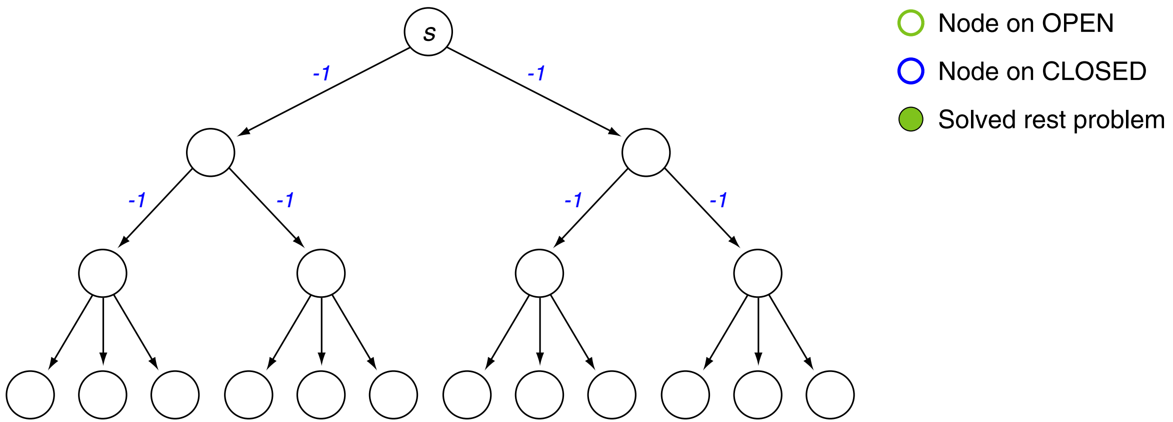
See lab class.

# BF* Variants

For trees $G$: Depth-first search is a special case of Z*, where $f(n') = f(n) - 1$, $f(s) = 0$, for all successors $n'$ of $n$.

# BF* Variants

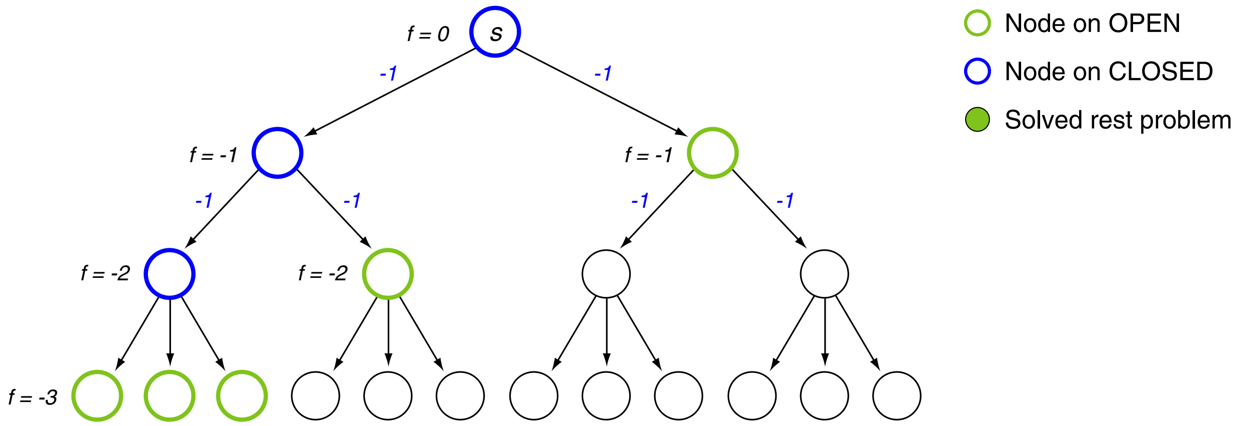
For trees $G$: Depth-first search is a special case of Z*, where $f(n') = f(n) - 1$, $f(s) = 0$, for all successors $n'$ of $n$.



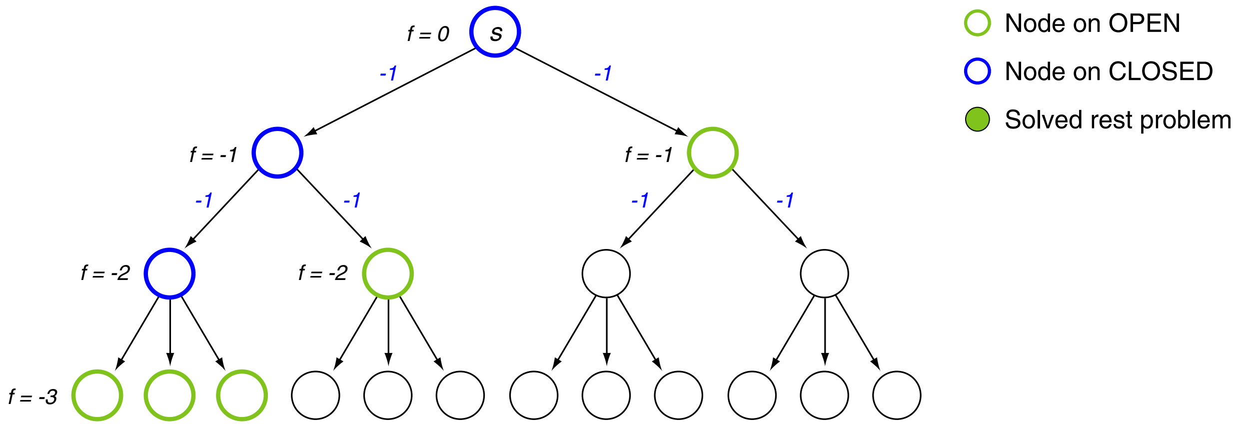
# BF* Variants

For trees $G$: Depth-first search is a special case of Z*, where $f(n') = f(n) - 1$, $f(s) = 0$, for all successors $n'$ of $n$.



# BF* Variants

For trees $G$: Depth-first search is a special case of Z*, where $f(n') = f(n) - 1$, $f(s) = 0$, for all successors $n'$ of $n$.




## Proof (sketch)

1. $f(n') < f(n) \;\Rightarrow n'$ was inserted on OPEN after $n$.
   $f(n') \leq f(n) \;\Leftrightarrow n'$ was inserted on OPEN after $n$.
2. Depth-first search $\equiv$ the most recently inserted node on OPEN is expanded.
3. Let $n_2$ be the most recently inserted node on OPEN.
4. Assumption: Let $n_1$ have been expanded before $n_2 \;\wedge\; f(n_1) \neq f(n_2)$.
5. $\Rightarrow f(n_1) < f(n_2)$ (consider minimization of $f$ under Z*).
6. $\Rightarrow n_1$ was inserted on OPEN after $n_2$.
7. $\Rightarrow n_2$ is not the most recently inserted node on OPEN. Contradiction to 3.

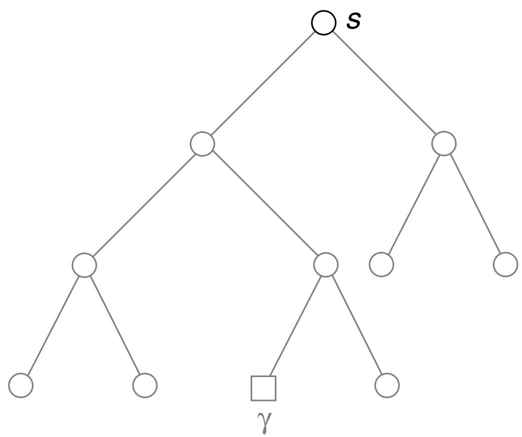
# BF* Variants

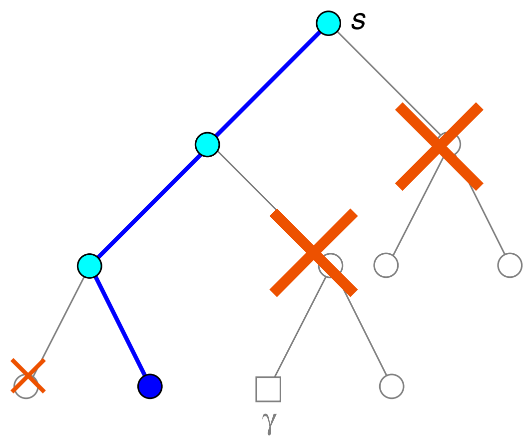
## OPEN List Restriction: Hill-Climbing (HC)

Hill-climbing is an informed, irrevocable search strategy.

HC characteristics:

- local or greedy optimization:
  take the direction of steepest ascend (sometimes: descend)
- "never look back" :
  alternatives are not remembered ➜ no OPEN/CLOSED lists
- usually low computational effort
- a strategy that is often applied by humans



# BF* Variants

| | |
|---|---|
| Algorithm: | HC (Hill-Climbing) |
| Input: | $s$. Start node representing the initial problem.<br>*successors*$(n)$. Returns the successors of node $n$.<br>$\star(n)$. Predicate that is *True* if $n$ is a goal node.<br>$f(n)$. Evaluation function for a node $n$. |
| Output: | A goal node or the symbol *Fail*. |

# BF* Variants [DFS] [BT]

Algorithm: HC (Hill-Climbing)

Input: $s$. Start node representing the initial problem.
*successors*$(n)$. Returns the successors of node $n$.
$\star(n)$. Predicate that is *True* if $n$ is a goal node.
$f(n)$. Evaluation function for a node $n$.

Output: A goal node or the symbol *Fail*.

HC$(s,$ *successors*$, \star, f)$

1. $n = s$;
2. $n_{\text{opt}} = s$;
3. **LOOP**
4. IF $\star(n)$ THEN RETURN$(n)$;
5. **FOREACH** $n'$ IN *successors*$(n)$ **DO** // Expand $n$.
   *add_backpointer*$(n', n)$;
   IF $(f(n') > f(n_{\text{opt}}))$ THEN $n_{\text{opt}} = n'$; // Remember optimum successor.
   **ENDDO**
6. IF $(n_{\text{opt}} = n)$
   THEN RETURN(*Fail*); // We could not improve.
   ELSE $n = n_{\text{opt}}$; // Continue with the best successor.
7. **ENDLOOP**

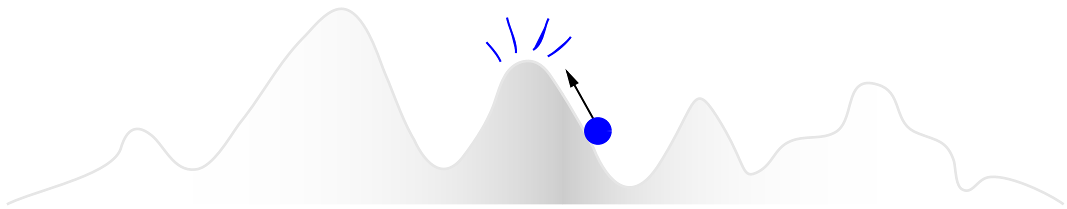
# BF* Variants

## HC Discussion

HC issue:

The first property of a systematic control strategy, *"Consider all objects in $S$."*, is violated by hill-climbing if no provisions are made.

- The forecast of the evaluation function (cost function, merit function) may be—at least sometimes—wrong and misguiding the search.
- Search will probably terminate at a local optimum.
- Alternative paths are not considered since each step is irrevocable.



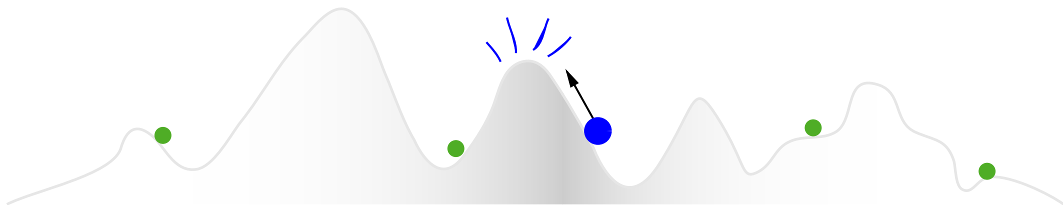
# BF* Variants

## HC Discussion

HC issue:

The first property of a systematic control strategy, *"Consider all objects in $S$."*, is violated by hill-climbing if no provisions are made.

- The forecast of the evaluation function (cost function, merit function) may be—at least sometimes—wrong and misguiding the search.
- Search will probably terminate at a local optimum.
- Alternative paths are not considered since each step is irrevocable.



Workaround: Perform multiple restarts (e.g. random-restart hill climbing).

Workaround issue: The second property of a systematic control strategy, *"Consider each object in $S$ only once."*, is violated if no provisions are made.

# BF* Variants

## HC Discussion (continued)

Hill-climbing can be the favorite strategy in certain situations:

(a) We are given a highly informative evaluation function to control search.

(b) The operators are commutative. Commutativity is given, if all operators are independent of each other.

➜ The application of an operator will

1. neither prohibit the applicability of any other operator,
2. nor modify the outcome of its application.

Example: Expansion of the nodes in a complete graph.

Remarks:

- Given commutativity, an irrevocable search strategy can be applied without hesitation: finding the optimum may be postponed but is never prohibited. Keywords: *greedy algorithm, greedy strategy, matroid*
- Given commutativity, hill-climbing can be considered a systematic strategy.
- Typically, hill-climbing is operationalized as an *informed strategy*, i.e., information about the goal (or about a concept to reach the goal) is exploited. If such external or look-ahead information is not exploited, hill-climbing must be considered an uninformed strategy.
- Q. What could be a provision to avoid a violation of the second property of a systematic control strategy?

# BF* Variants

## OPEN List Restriction: Best-First Beam Search [Rich & Knight 1991]

Characteristics:

- Best-first search is used with an OPEN list of limited size $k$.
- If OPEN exceeds its size limit, nodes with worst $f$-values are discarded until size limit is adhered to.

Operationalization:

1. A *cleanup_closed* function is needed to prevent CLOSED from growing uncontrollably.

Remarks:

- For $k = 1$ this is identical to an hill-climbing search.
- In breadth-first beam search [Lowerre 1976] all (at most) $k$ nodes of the current level are expanded and only the best $k$ of all these successors are kept and used for the next level.

# Hybrid Strategies

## Spectrum of Search Strategies

The search strategies

- Hill-climbing
- Informed backtracking
- Best-first search

form the extremal points within the spectrum of search strategies, based on the following dimensions:

R **Recovery.**
How many previously suspended alternatives (nodes) are reconsidered after finding a dead end?

S **Scope.**
How many alternatives (nodes) are considered for each expansion?

# Hybrid Strategies

## Spectrum of Search Strategies

The search strategies

- Hill-climbing — irrevocable decisions, consideration of newest alternatives
- Informed backtracking — tentative decisions, consideration of newest alternatives
- Best-first search — tentative decisions, consideration of all alternatives

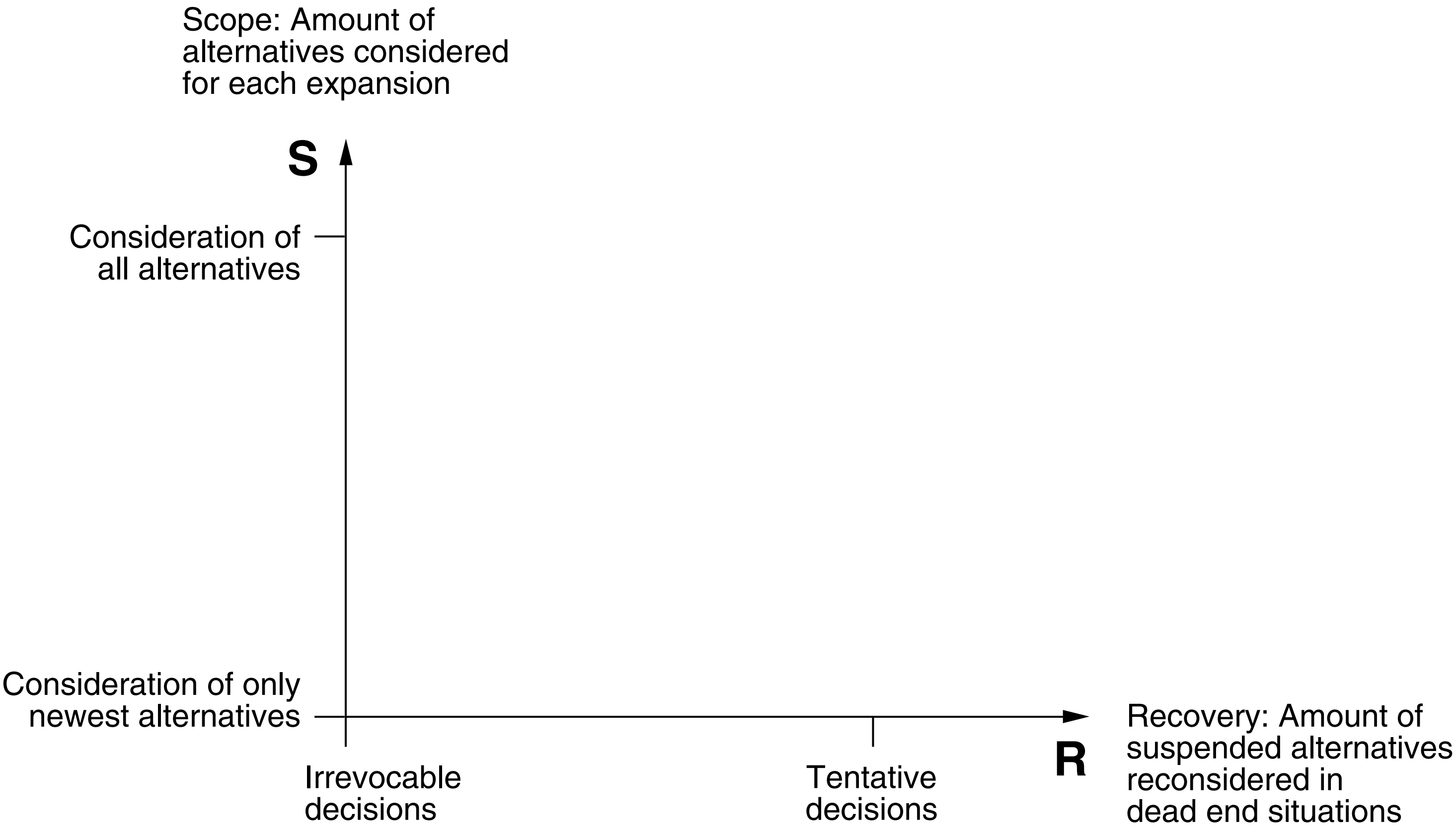
form the extremal points within the spectrum of search strategies, based on the following dimensions:

R **Recovery.**
How many previously suspended alternatives (nodes) are reconsidered after finding a dead end?

S **Scope.**
How many alternatives (nodes) are considered for each expansion?

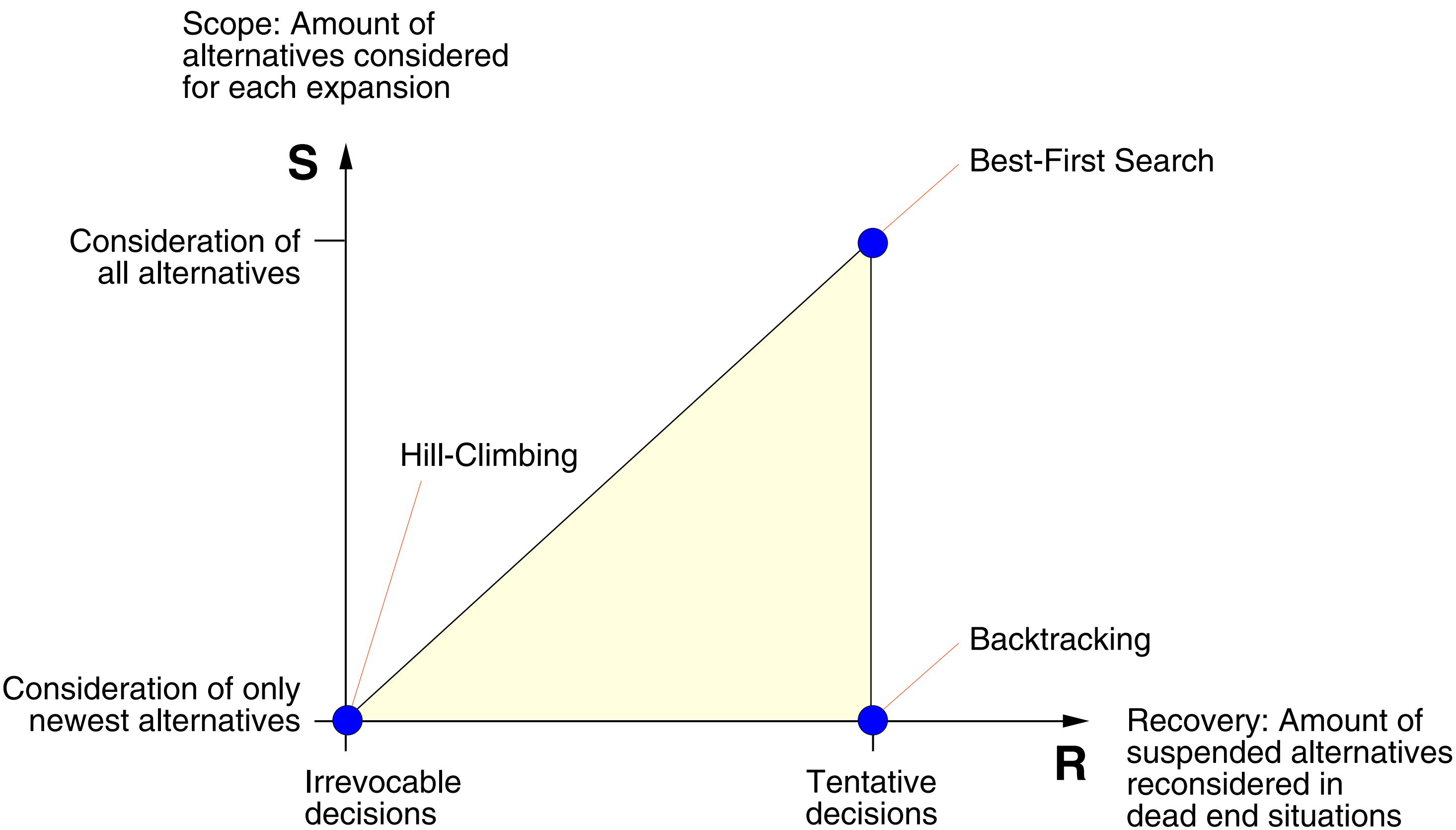
# Hybrid Strategies

## Spectrum of Search Strategies



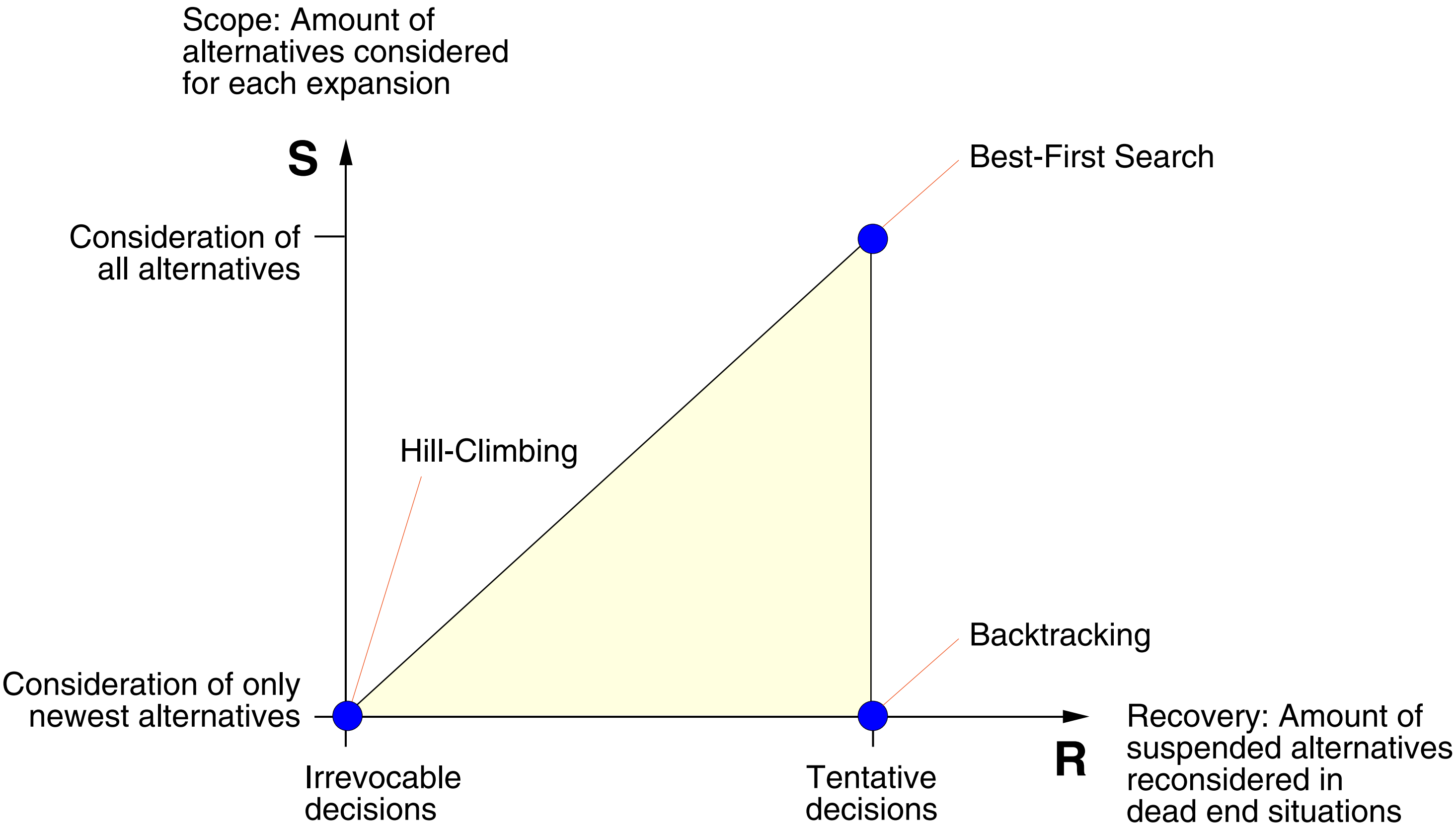
# Hybrid Strategies

## Spectrum of Search Strategies



# Hybrid Strategies

## Spectrum of Search Strategies



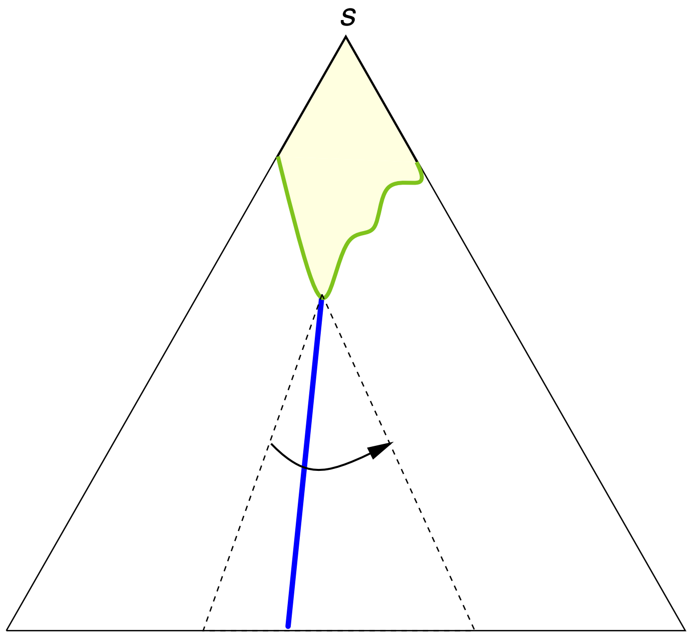

- The large scope of best-first search requires a high memory load.
- This load can be reduced by mixing it with backtracking.

Remarks:

- Recall that the memory consumption of best-first search is an (asymptotically) exponential function of the search depth.
- Hill-climbing is the most efficient strategy, but its effectiveness (solution quality) can only be guaranteed for problems that can be solved with a greedy approach.
- Informed backtracking requires not as much memory as best-first search, but usually needs more time as its scope is limited.
- Without a highly informed heuristic $h$, the degeneration of best-first strategies down to a uniform-cost search is typical and should be expected as the normal case.

# Hybrid Strategies

## Strategy 1: BF at Top




Characteristics:

- Best-first search is applied at the top of the search space graph.
- Backtracking is applied at the bottom of the search space graph.

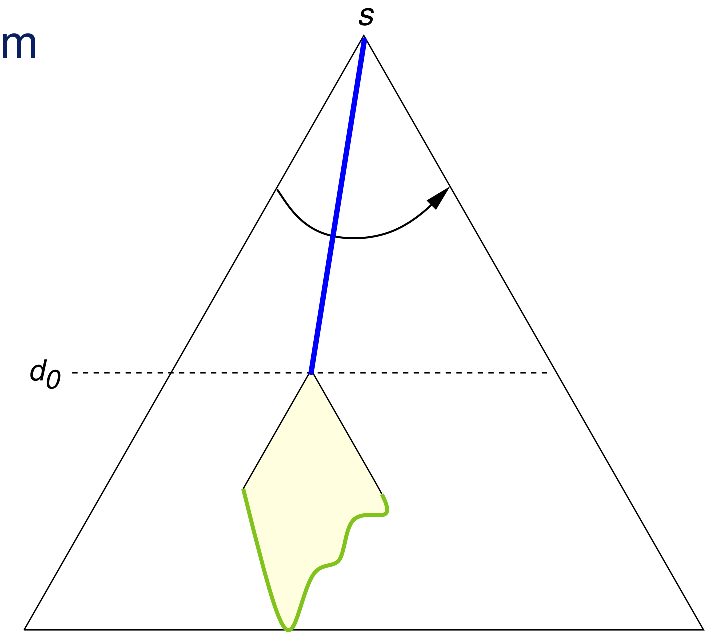
Operationalization:

1. Best-first search is applied until a memory allotment of size $M_0$ is exhausted.
2. Then backtracking starts with a most promising node $n'$ on OPEN.
3. If backtracking fails, it restarts with the next most promising OPEN node.

# Hybrid Strategies

## Strategy 2: BF at Bottom




Characteristics:

- Backtracking is applied at the top of the search space graph.
- Best-first search is applied at the bottom of the search space graph.

Operationalization:

1. Backtracking is applied until the search depth bound $d_0$ is reached.
2. Then best-first search starts with the node at depth $d_0$.
3. If best-first search fails, it restarts with the next node at depth $d_0$ found by backtracking.

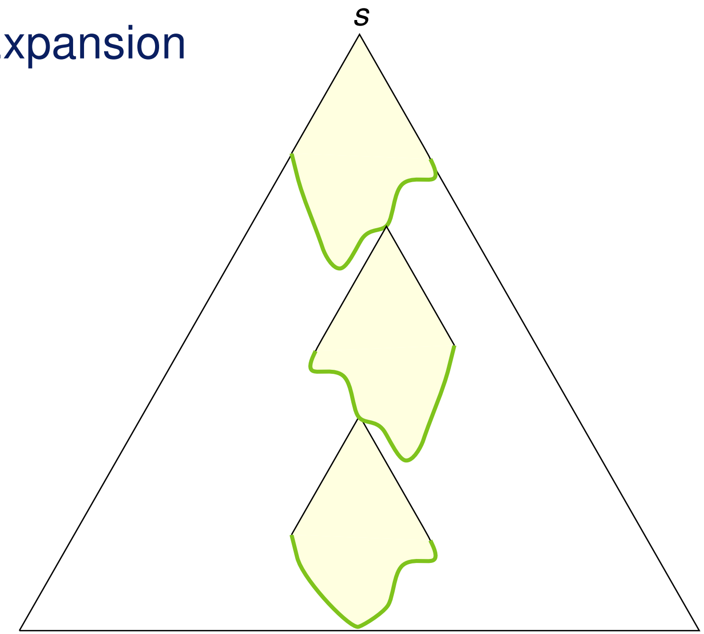
Remarks:

- The depth bound $d_0$ in Strategy 2 must be chosen carefully in order to avoid that the best-first search does not run out of memory. Hence, this strategy is more involved than Strategy 1 where the switch between best-first search and backtracking is triggered by the exhausted memory.

- If a sound depth bound $d_0$ is available, Strategy 2 (best-first search at bottom) is usually superior to Strategy 1 (best-first search at top). Q. Why?

# Hybrid Strategies

## Strategy 3: Extended Expansion




Characteristics:

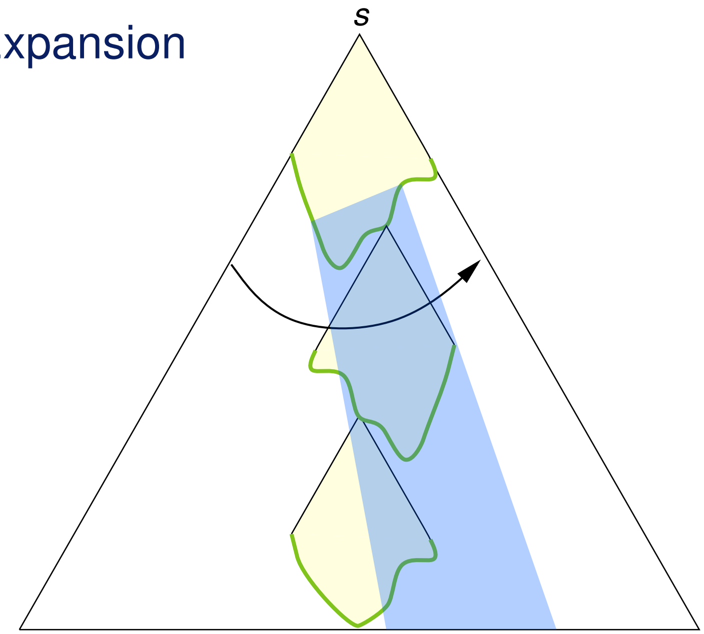
- Best-first search acts locally to generate a restricted number of promising nodes.
- Informed depth-first search acts globally, using best-first as an "extended node expansion".

Operationalization:

1. An informed depth-first search selects the nodes $n$ for expansion.
2. But a best-first search with a memory allotment of size $M_0$ is used to "expand" $n$.
3. The nodes on OPEN are returned to the depth-first search as "direct successors" of $n$.

# Hybrid Strategies

## Strategy 3: Extended Expansion




Characteristics:

- Best-first search acts locally to generate a restricted number of promising nodes.
- Informed depth-first search acts globally, using best-first as an “extended node expansion”.

Operationalization:

1. An informed depth-first search selects the nodes $n$ for expansion.
2. But a best-first search with a memory allotment of size $M_0$ is used to “expand” $n$.
3. The nodes on OPEN are returned to the depth-first search as “direct successors” of $n$.

Remarks:

- Strategy 3 is an informed depth-first search whose node expansion is operationalized via a memory-restricted best-first search.
- Q. What is the asymptotic memory consumption of Strategy 3 in relation to the search depth?

# Hybrid Strategies

## Strategy 4: IDA* [Korf 1985]

Characteristics:

- Depth-first search is used in combination with an iterative deepening approach for $f$-values.
- Nodes are considered only if their $f$-values do not exceed a given threshold.

Operationalization:

1. *limit* is initialized with $f(s)$.
2. In depth-first search, only nodes are considered with $f(n) \leq$ *limit*.
3. If depth-first search fails, *limit* is increased to the minimum cost of all $f$-values that exceeded the current threshold and depth-first search is rerun.

Remarks:

- IDA* always finds a cheapest solution path if the heuristic is admissible, or in other words never overestimates the actual cost to a goal node.
- IDA* uses space linear in the length of a cheapest solution.
- IDA* expands the same number of nodes, asymptotically, as A* in an exponential tree search.

# Hybrid Strategies

## Strategy 5: Focal Search [Ibaraki 1978]

Characteristics:

- An informed depth-first search is used as basic strategy.
- Nodes are selected from newly generated nodes and the best nodes encountered so far.

Operationalization:

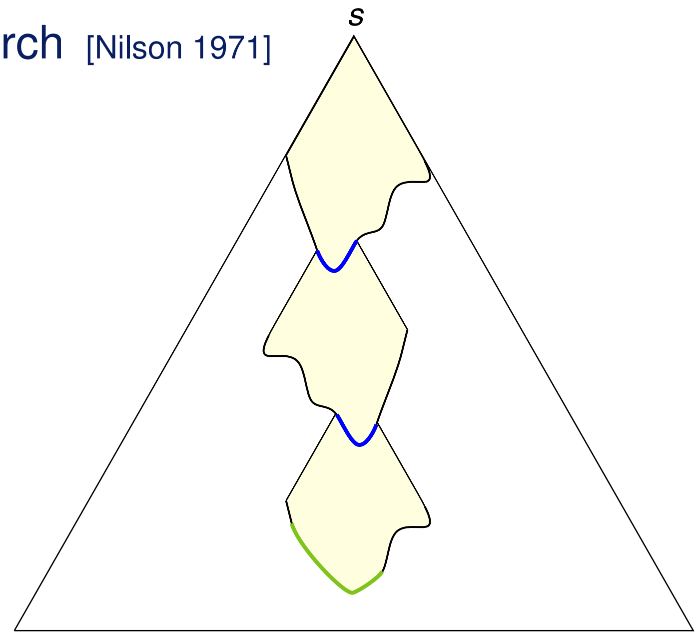
- The informed depth-first search expands the cheapest node $n$ from its list of alternatives.
- For the next expansion, it chooses from the newly generated nodes and the $k$ best nodes (without $n$) from the previous alternatives.

Remarks:

- For $k = 0$ this is identical to an informed depth-first search.
- For $k = \infty$ this is identical to a best-first search.
- Memory consumption (without proof): $O(b \cdot d^{k+1})$, where $b$ denotes the branching degree and $d$ the search depth.
- An advantage of Strategy 5 is that its memory consumption can be controlled via the single parameter $k$.
- Differences to beam search:
  - In focal search no nodes are discarded. Therefore, focal search will never miss a solution.
  - In best-first beam search the OPEN list is of limited size.

# Hybrid Strategies

## Strategy 6: Staged Search [Nilson 1971]




Characteristics:

- Best-first search acts locally to generate a restricted number of promising nodes.
- Hill-climbing acts globally, but by retaining a set of nodes.

Operationalization:

1. Best-first search is applied until a memory allotment of size $M_0$ is exhausted.
2. Then only the cheapest OPEN nodes (and their pointer-paths) are retained.
3. Best-first search continues until Step 1. is reached again.

Remarks:

- Staged search can be considered as a combination of best-first search and hill-climbing. While a pure hill-climbing discards all nodes except one, staged search discards all nodes except a small subset.
- Staged search addresses the needs of extreme memory restrictions and tight runtime bounds.
- Recall that the Strategies 1-5 are complete with regard to recovery, but that Strategy 6, Hill Climbing, and Best-First Beam Search are not.