Bounded Suboptimal Search for Classical Planning

Caroline Steiblin <caroline.steiblin@stud.unibas.ch>

Artificial Intelligence Group, University of Basel

November 06, 2020

Contents

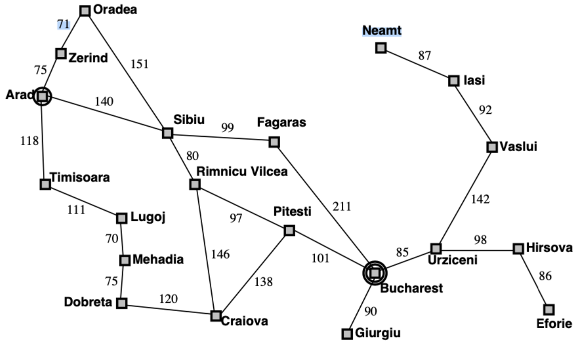
State Spaces

Definition (State Space)

A **state space** or **transition system** is defined as $\mathcal{S} = \langle S, A, cost, T, s_0, S_* \rangle$ with

- S, the finite set of states
- A, the finite set of actions
- $cost$, $A \to \mathbb{R}_0^+$, the action costs
- $T \subseteq S \times A \times S$, the transition relation
- $s_0 \in S$, the initial state
- $S_* \subseteq S$, the set of goal state(s)

Example of a State Space



Route Planning in Romania:
S are all the city names shown,
A are the routes between two states with associated *cost* for each,
s_0 here as Arad,
S_* here as Bucharest.

Classical Planning

Goal: Find plans (sequences of actions) that lead from an initial state to a goal state, for general search problems that are *static*, *deterministic*, *fully observable*.

Given: A state space description (planning formalism)

Required: Either a plan (solution) or proof that no plan exists

Difference between *optimal planning* (returned plans are optimal, minimal total cost) and *suboptimal planning* (satisficing, suboptimal plans allowed)

Optimal Search, A*

Definition (A*)

A* is a variant of best-first search that aims to find an optimal solution through

$$f(n) = g(n) + h(n.state)$$

with $f(n)$ as the evaluation function, $g(n)$ as the path cost, and $h(n)$ as the heuristic cost of the given state

A* is optimal by expanding nodes with minimal $f(n)$ values.
A* search is complete with safe heuristics.
If reopening is allowed, A* is optimal with admissible heuristics.
Without reopening, A* is only optimal with admissible and consistent heuristics.

Suboptimal Search, Weighted A*

Definition (WA*)

Weighted A* (WA*) is a suboptimal variant of A* with

$$f(n) = g(n) + w \cdot h(n.state)$$

with weight $w \geq 1$, guaranteeing a solution at most w times as expensive compared to A* when reopening is used.

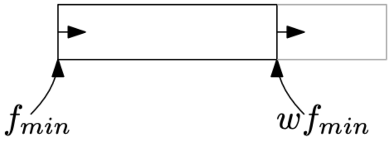
WA* carries the same properties as A*, so will provide w-suboptimal solutions if heuristics are admissible (with reopening) or admissible and consistent (without reopening).

Suboptimal Search, Focal Search

Definition (Focal Search)

Focal search is a variant of bounded suboptimal search (BSS), using both an *open list* and *focal list* in parallel to separately find a solution and guarantee its w-suboptimality.

Open list like A*, contains all states in increasing order of $f(n)$. The focal list contains states from the open list, but only those whose $f(n)$ values are smaller than the $w \cdot f_{min}$, where f_{min} is the smallest $f(n)$ value on the open list.



BSS's FOCAL and OPEN

Suboptimal Search, Optimistic Search

Definition (OS)

Optimistic search (OS) is a focal search variation of WA* that searches with

$$f(n) = g(n) + (2w - 1) \cdot h(n.state)$$

and can find solutions that are even w-suboptimal

OS uses WA*'s tendency to often find solutions that are better than w-suboptimal
OS uses weight bound of $2w - 1$ on focal list, which improves total search time and validates solution in parallel through A* search.
OS reopens states if shorter path is found.

Evaluation Functions

Definition (Evaluation functions)

Evaluation functions guide informed search algorithms to a solution, improving solution quality while avoiding node re-openings:

$$f(n) = \Phi(h(n), g(n))$$

where $\Phi : \mathbb{R} \times \mathbb{R} \rightarrow \mathbb{R}$

Linear Evaluation Functions

Definition (Linear Evaluation Functions)

Linear evaluation functions keep the maximum level of suboptimality constant at the weight. For WA*, this is:

$$\Phi_{WA*}(h, g) = \frac{1}{w} \cdot g + h.$$

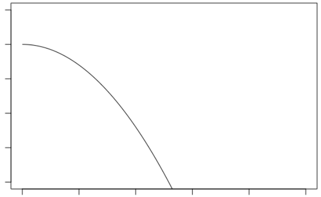
where $\Phi : \mathbb{R} \times \mathbb{R} \rightarrow \mathbb{R}$.

Nonlinear Evaluation Functions

Nonlinear evaluation functions allow tolerance for suboptimality to vary during search.

Nonlinear functions can strongly limit degree of suboptimality at beginning of search to ensure suboptimality focused on critical parts of search.

More tolerance available after beginning of path.



Experimentally, would imply that nonlinear functions would perform better with suboptimality than linear functions.

XDP

Definition (XDP)

Convex Downward Parabola (XDP) evaluation function is defined through:

$$\Phi_{XDP}(h, g) = \frac{1}{2w}(g + (2w - 1)h + \sqrt{((g + h)^2 + 4wgh})$$

where w is the weight, h is the heuristic function $h(n)$ and g is the path cost $g(n)$

Paths with low g (path cost) should be near optimal.

XUP

Definition (XUP)

Convex Upward Parabola (XUP) evaluation function is defined through:

$$\Phi_{XUP}(h, g) = \frac{1}{2w}(g + h + \sqrt{((g + h)^2 + 4w(w - 1)h^2})$$

where w is the weight, h is the heuristic function $h(n)$ and g is the path cost $g(n)$

XUP parabola is constructed such that the path found near the goal will be near-optimal.

PWXDP

Definition (PWXDP)

Piece-Wise Convex Downward Parabola (PWXDP) piece-wise is a linear function evaluation function defined through:

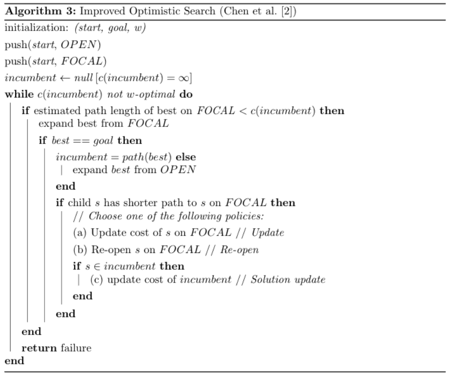
$$f(n) = \begin{cases} g(n) + h(n) & \text{if } h(n) > g(n) \\ \frac{g(n)+(2w-1)h(n)}{w} & \text{otherwise} \end{cases}$$

where w is the weight, h is the heuristic function $h(n)$ and g is the path cost $g(n)$

PWXDP is constructed through concatenation of two linear functions, and has not yet been published.

Improved Optimistic Search (theoretical)

The recently developed *Improved Optimistic Search* (IOS) algorithm adapts the OS algorithm.



Algorithm 3: Improved Optimistic Search (Chen et al. [2])

```
initialization: $(start, goal, w)$
push($start$, $OPEN$)
push($start$, $FOCAL$)
$incumbent \leftarrow null\ [c(incumbent) = \infty]$
while $c(incumbent)$ not $w$-optimal do
    if estimated path length of best on $FOCAL < c(incumbent)$ then
        expand best from $FOCAL$
        if $best == goal$ then
            $incumbent = path(best)$ else
                expand $best$ from $OPEN$
            end
            if child $s$ has shorter path to $s$ on $FOCAL$ then
                // Choose one of the following policies:
                (a) Update cost of $s$ on $FOCAL$ // Update
                (b) Re-open $s$ on $FOCAL$ // Re-open
                if $s \in incumbent$ then
                    (c) update cost of $incumbent$ // Solution update
                end
            end
    end
    return failure
end
```

Open list search runs an A* search with $f(n) = g(n) + h(n)$, whereas the focal list uses an *evaluation function*, $f'(n)$.

Open list search runs an A* search with $f(n) = g(n) + h(n)$, whereas the focal list uses an *evaluation function*, $f'(n)$.
First termination condition is

$$c(I) \leq wf_{min}$$

to prove the w-suboptimality of a solution found, where $c(I)$ is the cost of I, the incumbent plan.
Through $\Phi(h, g)$, $f(n)$ cost is not estimate of experimental solution cost, but optimal solution cost, so like f_{min} on open list.
Second termination condition with $\Phi(h, g)$, or $f'(n)$, for focal list:

$$c(I) \leq wf'_{max}$$

with f'_{max} the maximum $f'(n)$ (evaluation function) value of state on focal list.
Dual termination conditions should result in solutions being even closer to optimal cost.

Improved Optimistic Search (realized)

Due to a lack of any practical application of the published IOS code means IOS remains theoretical. Implementation here modified open and focal lists to be run sequentially versus in parallel.

Both individual searches reopen nodes in own search space, but no information is shared between searches.

Second part of algorithm, where algorithm switches between both search spaces, and path costs are updated across search spaces, was not able to be implemented at this point.

Improved Optimistic Search Algorithm (realized)

Algorithm 4: Improved Optimistic Search (modified)

```
initialization: (start, goal, w)
push(start, OPEN)
push(start, FOCAL)
incumbent ← null [$c(I) = \infty$]
while $c(I)$ not w-optimal do
    if incumbent is null then
        expand best from FOCAL
        if best is goal state then
            incumbent = path(best)
        end
    else
        expand best from OPEN
        if best is goal state then
            incumbent = path(best)
            break
        end
    end
end
if incumbent is null then
    return failure
else
    return incumbent
end
```

Overview

Two main experiments were created to run over *optimal-strips* benchmark suite:

- Four relevant evaluation functions, WA*, XDP, XUP, and PWXDP, in eager BFS with A* control search,
- Modified IOS algorithm with each evaluation function

Weight used in all experiments held at $w = 2$ to maximize efficiency and coverage, although this is higher than weights typically used in practice.

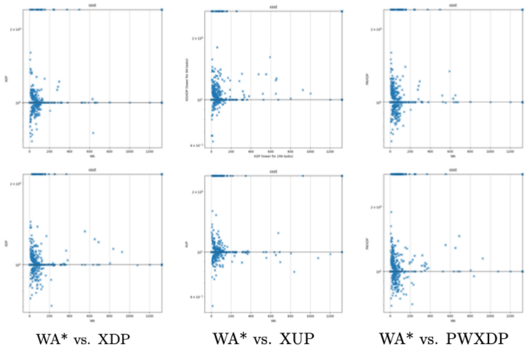
Weight bound set at standard $wf = 2w \;\; 1$ ($wf = 3$).

CEGAR heuristic used in all experiments as is admissible, safe, and consistent.

Effective Weight and Coverage for Algorithm Pairs

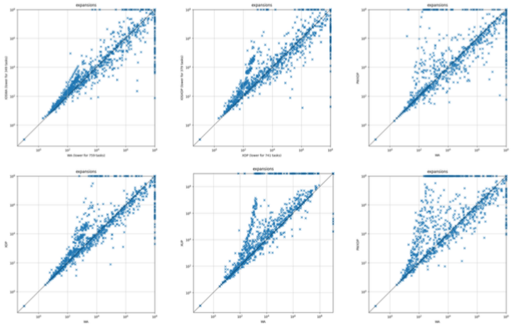
Algorithm	Cost (Weight)	Coverage (%)
A*	1.00x	48.5
WA*	1.08x	62.3
XDP	1.08x	63.2
XUP	1.07x	56.3
PWXDP	1.10x	63.3
IOS-WA*	1.11x	63.7
IOS-XPD	1.12x	64.5
IOS-XUP	1.10x	59.2
IOS-PWXDP	1.13x	56.2

Relative Cost



Relative cost comparison of linear (WA*) and nonlinear evaluation functions in eager BFS algorithm (top) and IOS algorithm (bottom)

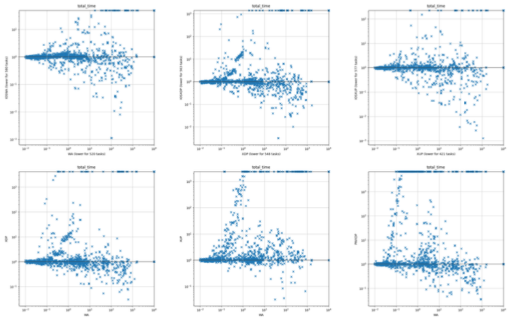
Number of expansions



WA* vs. XPD WA* vs. XUP WA* vs. PWXDP

Number of expansions comparison of linear (WA*) and nonlinear evaluation functions in eager BFS algorithm (top) and IOS algorithm (bottom)

Relative Time



WA* vs. XPD WA* vs. XUP WA* vs. PWXDP

Relative time comparison of linear (WA*) and nonlinear evaluation functions in eager BFS algorithm (top) and IOS algorithm (bottom)

Final Remarks

- Average suboptimal solution was 1.1x optimal cost, with weight 2
- No clear "best" evaluation function
  - XDP slightly better overall performance
  - XUP showed lowest costs in all suboptimal implementations
  - No significant advantage to implementing nonlinear evaluation functions over WA* in Fast Downward
- Increase in coverage compared to optimal search - from 48.6% in A* to around 61.0% in suboptimal implementations
- Further testing and full IOS implementation required to yield clear results

Questions?

caroline.steiblin@stud.unibas.ch

Optimistic Search Algorithm

Algorithm 2: Optimistic Search (Thayer and Ruml [10])

```
initialization: $initial$, $bound$
$OPEN_f \leftarrow \{initial\}$
$OPEN_{\hat{f}} \leftarrow \{initial\}$
$incumbent \leftarrow \infty$
repeat until $bound \cdot f(\text{first on } OPEN_f) \geq f(incumbent)$
    if $\hat{f}(\text{first on } OPEN_{\hat{f}}) < \hat{f}(incumbent)$ then
        $n \leftarrow$ remove first on $OPEN_{\hat{f}}$
        remove $n$ from $OPEN_f$
    else
        $n \leftarrow$ remove first on $OPEN_f$
        remove $n$ from $OPEN_{\hat{f}}$
    end
    add $n$ to $CLOSED$
    if $n$ is a goal then
        $incumbent \leftarrow n$
    else
        foreach child $c$ of $n$ do
            if $c$ is duplicated in $OPEN_f$ then
                if $c$ is better than the duplicate then
                    replace copies in $OPEN_f$ and $OPEN_{\hat{f}}$
                end
            else if $c$ is duplicated in $CLOSED$ then
                if $c$ is better than the duplicate then
                    add $c$ to $OPEN_f$ and $OPEN_{\hat{f}}$
                end
            else
                add $c$ to $OPEN_f$ and $OPEN_{\hat{f}}$
            end
        end
    end
end
```