Sequencing Operator Counts with State-Space Search

Wesley L. Kaizer\textsuperscript{1} \hspace{1cm} André G. Pereira\textsuperscript{1} \hspace{1cm} Marcus Ritt\textsuperscript{1}

\textsuperscript{1}Federal University of Rio Grande do Sul, Brazil
Introduction
[Davies et al., 2015] decomposes the process of solving planning tasks into a master problem and a subproblem;
Introduction

- [Davies et al., 2015] decomposes the process of solving planning tasks into a **master problem** and a **subproblem**;
- The master problem solves an **operator-counting integer program**;
Introduction

- [Davies et al., 2015] decomposes the process of solving planning tasks into a master problem and a subproblem;
- The master problem solves an operator-counting integer program;
- The subproblem tries to sequence the operator counts;
[Davies et al., 2015] decomposes the process of solving planning tasks into a master problem and a subproblem;

The master problem solves an operator-counting integer program;

The subproblem tries to sequence the operator counts;

Primal solution: contains more information than the objective function value.
Introduction

- [Davies et al., 2015] decomposes the process of solving planning tasks into a **master problem** and a **subproblem**;
- The master problem solves an **operator-counting integer program**;
- The subproblem tries to sequence the **operator counts**;
- **Primal solution**: contains more information than the objective function value.
OpSeq and SAT

- *OpSeq* solves the subproblem using a **SAT solver**;
OpSeq and SAT

- **OpSeq** solves the subproblem using a **SAT solver**;
- Encodes the planning task and the operator counts in a SAT formula;
OpSeq and SAT

- **OpSeq** solves the subproblem using a **SAT solver**;
- Encodes the planning task and the operator counts in a SAT formula;
- If the formula is satisfiable, **OpSeq** can directly extract a plan;
OpSeq and SAT

- *OpSeq* solves the subproblem using a **SAT solver**;
- Encodes the planning task and the operator counts in a SAT formula;
- If the formula is satisfiable, *OpSeq* can directly extract a plan;
- Otherwise, *OpSeq* uses **assumptions** to generate a constraint.
OpSearch
Heuristic search is the most common approach to solve classical planning tasks optimally;
OpSearch

- Heuristic search is the most common approach to solve classical planning tasks optimally;
- OpSearch is a new algorithm based on heuristic search to solve the operator counts sequencing subproblem;
Heuristic search is the most common approach to solve classical planning tasks optimally;
OpSearch is a new algorithm based on heuristic search to solve the operator counts sequencing subproblem;
It uses information from the search graph, such as the $f$-values;
OpSearch

- Heuristic search is the most common approach to solve classical planning tasks optimally;
- OpSearch is a new algorithm based on heuristic search to solve the operator counts sequencing subproblem;
- It uses information from the search graph, such as the $f$-values;
- This approach generates smaller and more informed constraints;
OpSearch

- Heuristic search is the most common approach to solve classical planning tasks optimally;
- OpSearch is a new algorithm based on heuristic search to solve the operator counts sequencing subproblem;
- It uses information from the search graph, such as the $f$-values;
- This approach generates smaller and more informed constraints;
- Improves from advancements in planning research.
Constraint Generation Rule
Constraint Generation Rule

\[ L = \{ [Y_o \geq C(o) + 1] \mid \exists s \xrightarrow{o} s' : f(s') \leq f_{\text{max}} \land ((v_o \notin \text{vars}(s) \land c(o) > 0) \lor s(v_o) = 0) \} \].
Constraint Generation Rule

\[ L = \{ [Y_o \geq C(o) + 1] \mid \exists s \xrightarrow{o} s' : f(s') \leq f_{\text{max}} \land \]
\[ ((v_o \notin \text{vars}(s) \land c(o) > 0) \lor s(v_o) = 0) \} \].
Constraint Generation Rule

\[ L = \{ [Y_o \geq C(o) + 1] \mid \exists s \xrightarrow{o} s' : f(s') \leq f_{\text{max}} \land ((v_o \notin \text{vars}(s) \land c(o) > 0) \lor s(v_o) = 0) \}. \]
Constraint Generation Rule

\[ L = \{ [Y_o \geq C(o) + 1] \mid \exists s \xrightarrow{o} s' : f(s') \leq f_{\text{max}} \land \\
( (v_o \notin \text{vars}(s) \land c(o) > 0) \lor s(v_o) = 0) \} \].
Constraint Generation Rule

\[ L = \{ [Y_o \geq C(o) + 1] | \exists s \xrightarrow{o} s' : f(s') \leq f_{\text{max}} \land
(\left( v_o \not\in \text{vars}(s) \land c(o) > 0 \right) \lor s(v_o) = 0) \}. \]
Constraint Generation Rule

\[ L = \{ [Y_o \geq C(o) + 1] \mid \exists s \xrightarrow{o} s' : f(s') \leq f_{\text{max}} \land \\
((v_o \notin \text{vars}(s) \land c(o) > 0) \lor s(v_o) = 0)\}. \]
Constraint Generation Strategy
Example
Example: First Iteration

\[ C = \{ o_1 \mapsto 1 \} \text{ and } f_{\text{max}} = 1: \]
Example: First Iteration

$$\mathcal{C} = \{ o_1 \mapsto 1 \} \text{ and } f_{\text{max}} = 1:$$
Example: First Iteration

\[ C = \{ o_1 \mapsto 1 \} \text{ and } f_{\text{max}} = 1: \]

GLC: \[ [Y_f \geq 3] \geq 1. \]
Example: First Iteration

\[ \mathcal{C} = \{ o_1 \mapsto 1 \} \text{ and } f_{\text{max}} = 1: \]

\[ n_0 \langle s_0, o_1 \mapsto 1 \rangle f = 3 \]

\[ n_1 \langle s_1 \rangle f = 3 \rightarrow o_2 \rightarrow n_4 \langle s_4 \rangle f = 4 \rightarrow o_4 \rightarrow n_7 \langle s_7 \rangle f = 4 \]

\[ n_2 \langle s_2 \rangle f = 3 \rightarrow o_3 \rightarrow n_5 \langle s_5 \rangle f = 3 \rightarrow o_2 \rightarrow n_8 \langle s_8 \rangle f = 3 \]

\[ n_3 \langle s_1 \rangle f = 5 \rightarrow o_2 \rightarrow n_6 \langle s_6 \rangle f = 5 \rightarrow o_4 \rightarrow n_9 \langle s_9 \rangle f = 5 \]

GLC: \[ [Y_f \geq 3] \geq 1. \]
Example: Second Iteration

\[ C = \{o_1 \mapsto 3\} \text{ and } f_{\text{max}} = 3: \]
Example: Second Iteration

\[ C = \{ o_1 \mapsto 3 \} \text{ and } f_{\text{max}} = 3: \]

\[ n_0 \langle s_0, o_1 \mapsto 3 \rangle f=3 \]

\[ n_1 \langle s_1, o_1 \mapsto 3 \rangle f=3 \]

\[ n_2 \langle s_2, o_1 \mapsto 2 \rangle f=3 \]

\[ n_3 \langle s_1, o_1 \mapsto 2 \rangle f=5 \]

\[ n_4 \langle s_4 \rangle f=4 \]

\[ n_5 \langle s_5 \rangle f=3 \]

\[ n_6 \langle s_6 \rangle f=5 \]

\[ n_7 \langle s_7 \rangle f=4 \]

\[ n_8 \langle s_8 \rangle f=3 \]

\[ n_9 \langle s_9 \rangle f=5 \]
Example: Second Iteration

\[ \mathcal{C} = \{ o_1 \mapsto 3 \} \text{ and } f_{\text{max}} = 3: \]

\[ n_0 \langle s_0, o_1 \mapsto 3 \rangle f=3 \]

\[ n_1 \langle s_1, o_1 \mapsto 3 \rangle f=3 \]

\[ n_2 \langle s_2, o_1 \mapsto 2 \rangle f=3 \]

\[ n_3 \langle s_1, o_1 \mapsto 2 \rangle f=5 \]

\[ n_4 \langle s_4 \rangle f=3 \]

\[ n_5 \langle s_5 \rangle f=3 \]

\[ n_6 \langle s_6 \rangle f=5 \]

\[ n_7 \langle s_7 \rangle f=4 \]

\[ n_8 \langle s_8 \rangle f=3 \]

\[ n_9 \langle s_9 \rangle f=5 \]

**GLC:** \[ [Y_{o_3} \geq 1] + [Y_f \geq 4] \geq 1. \]
Example: Second Iteration

\[ \mathcal{C} = \{ o_1 \mapsto 3 \} \text{ and } f_{\text{max}} = 3: \]

GLC: \[ Y_{o_3} \geq 1 \] + \[ Y_f \geq 4 \] \geq 1.
Example: Second Iteration

$C = \{ o_1 \mapsto 3 \} \text{ and } f_{\text{max}} = 3:

GLC: $[Y_{o_3} \geq 1] + [Y_f \geq 4] \geq 1.$
Example: Third Iteration

\[ C = \{ o_1 \mapsto 2, o_3 \mapsto 1 \} \] and \( f_{\text{max}} = 3 \):
Example: Third Iteration

\[ C = \{ o_1 \mapsto 2, o_3 \mapsto 1 \} \text{ and } f_{\text{max}} = 3: \]
Example: Third Iteration

\[ C = \{ o_1 \mapsto 2, \, o_3 \mapsto 1 \} \text{ and } f_{\text{max}} = 3: \]

GLC: \[ [Y_{o_2} \geq 1] + [Y_f \geq 4] \geq 1. \]

Wesley Luciano Kaizer
Example: Third Iteration

\[ C = \{ o_1 \mapsto 2, o_3 \mapsto 1 \} \text{ and } f_{\text{max}} = 3: \]

\[ n_0 \langle s_0, o_1 \mapsto 1, o_3 \mapsto 1 \rangle f=3 \]

\[ n_1 \langle s_1, o_1 \mapsto 1, o_3 \mapsto 1 \rangle f=3 \]

\[ n_2 \langle s_2, o_3 \mapsto 1 \rangle f=3 \]

\[ n_3 \langle s_1, o_3 \mapsto 1 \rangle f=5 \]

\[ n_4 \langle s_4 \rangle f=4 \]

\[ n_5 \langle s_5 \rangle f=3 \]

\[ n_6 \langle s_6 \rangle f=5 \]

\[ n_7 \langle s_7 \rangle f=4 \]

\[ n_8 \langle s_8 \rangle f=3 \]

\[ n_9 \langle s_9 \rangle f=5 \]

GLC: \[ Y_{o_2} \geq 1 \] + \[ Y_f \geq 4 \] \geq 1.

Wesley Luciano Kaizer
Example: Third Iteration

\[ C = \{ o_1 \mapsto 2, o_3 \mapsto 1 \} \text{ and } f_{\text{max}} = 3: \]

GLC: \[ [Y_{o_2} \geq 1] + [Y_f \geq 4] \geq 1. \]
Example: Fourth Iteration

\[ C = \{ o_1 \mapsto 1, o_2 \mapsto 1, o_3 \mapsto 1 \} \text{ and } f_{\text{max}} = 3: \]
Example: Fourth Iteration

\[ C = \{ o_1 \mapsto 1, o_2 \mapsto 1, o_3 \mapsto 1 \} \text{ and } f_{\text{max}} = 3: \]
Example: Fourth Iteration

\[ C = \{ o_1 \mapsto 1, o_2 \mapsto 1, o_3 \mapsto 1 \} \] and \( f_{\text{max}} = 3 \):

Plan: \( \langle o_1, o_3, o_2 \rangle \).

Wesley Luciano Kaizer
Theorem
Theorem 1. For a solvable SAS$^+$ planning task $\Pi$, an operator counts $C_s$ with an associated $f$-bound value $f_{\text{max}}$, such that OpSearch’s modified $A^*$ with an admissible heuristic function $h$ cannot sequence $C_s$, OpSearch always returns an admissible constraint to the master integer program.
Results
OpSearch is Better than OpSeq
OpSearch is better than OpSeq: solves more tasks, solves less subproblems, uses less memory and generates smaller constraints.
<table>
<thead>
<tr>
<th></th>
<th>$OpSeq$</th>
<th>$OpSearch$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$C$</td>
<td>63</td>
<td>73</td>
</tr>
<tr>
<td>$S$</td>
<td>121202</td>
<td>99437</td>
</tr>
</tbody>
</table>

OpSearch is better than OpSeq: solves more tasks, solves less subproblems, uses less memory and generates smaller constraints.
OpSearch is better than OpSeq: solves more tasks, solves less subproblems, uses less memory and generate smaller constraints.

<table>
<thead>
<tr>
<th></th>
<th>OpSeq</th>
<th>OpSearch</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td>63</td>
<td>73</td>
</tr>
<tr>
<td>S</td>
<td>121202</td>
<td>99437</td>
</tr>
<tr>
<td>$\bar{T}_t$</td>
<td>1783</td>
<td>1720</td>
</tr>
</tbody>
</table>
OpSearch is better than OpSeq: solves more tasks, solves less subproblems, uses less memory and generates smaller constraints.

<table>
<thead>
<tr>
<th></th>
<th>OpSeq</th>
<th>OpSearch</th>
</tr>
</thead>
<tbody>
<tr>
<td>$C$</td>
<td>63</td>
<td>73</td>
</tr>
<tr>
<td>$S$</td>
<td>121202</td>
<td>99437</td>
</tr>
<tr>
<td>$\bar{T}_t$</td>
<td>1783</td>
<td>1720</td>
</tr>
<tr>
<td>$\bar{M}$</td>
<td>865</td>
<td>367</td>
</tr>
</tbody>
</table>
OpSearch is better than OpSeq: solves more tasks, solves less subproblems, uses less memory and generates smaller constraints.

<table>
<thead>
<tr>
<th></th>
<th>OpSeq</th>
<th>OpSearch</th>
</tr>
</thead>
<tbody>
<tr>
<td>$C$</td>
<td>63</td>
<td>73</td>
</tr>
<tr>
<td>$S$</td>
<td>121202</td>
<td>99437</td>
</tr>
<tr>
<td>$\bar{T}_t$</td>
<td>1783</td>
<td>1720</td>
</tr>
<tr>
<td>$\bar{M}$</td>
<td>865</td>
<td>367</td>
</tr>
<tr>
<td>$\bar{u}$</td>
<td>20</td>
<td>6</td>
</tr>
</tbody>
</table>
OpSearch is better than OpSeq: solves more tasks, solves less subproblems, uses less memory and generate smaller constraints.

\begin{tabular}{|c|c|c|}
\hline
 & OpSeq & OpSearch \\
\hline
$C$ & 63 & 73 \\
$S$ & 121202 & 99437 \\
$\overline{T}_t$ & 1783 & 1720 \\
$\overline{M}$ & 865 & 367 \\
$\bar{u}$ & 20 & 6 \\
\hline
\end{tabular}
OpSearch is better than OpSeq: solves more tasks, solves less subproblems, uses less memory and generates smaller constraints.

<table>
<thead>
<tr>
<th></th>
<th>OpSeq</th>
<th>OpSearch</th>
</tr>
</thead>
<tbody>
<tr>
<td>S</td>
<td>2738</td>
<td>2169</td>
</tr>
</tbody>
</table>
OpSearch is better than OpSeq: solves more tasks, solves less subproblems, uses less memory and generate smaller constraints.

<table>
<thead>
<tr>
<th></th>
<th>$S$</th>
<th>$\bar{T}_t$</th>
</tr>
</thead>
<tbody>
<tr>
<td>OpSeq</td>
<td>2738</td>
<td>92</td>
</tr>
<tr>
<td>OpSearch</td>
<td>2169</td>
<td>191</td>
</tr>
</tbody>
</table>
OpSearch is better than OpSeq: solves more tasks, solves less subproblems, uses less memory and generates smaller constraints.

<table>
<thead>
<tr>
<th></th>
<th>OpSeq</th>
<th>OpSearch</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S$</td>
<td>2738</td>
<td>2169</td>
</tr>
<tr>
<td>$\bar{T}_t$</td>
<td>92</td>
<td>191</td>
</tr>
<tr>
<td>$\bar{M}$</td>
<td>122</td>
<td>118</td>
</tr>
</tbody>
</table>
OpSearch is better than OpSeq: solves more tasks, solves less subproblems, uses less memory and generates smaller constraints.

<table>
<thead>
<tr>
<th></th>
<th>OpSeq</th>
<th>OpSearch</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S$</td>
<td>2738</td>
<td>2169</td>
</tr>
<tr>
<td>$\bar{T}_t$</td>
<td>92</td>
<td>191</td>
</tr>
<tr>
<td>$\bar{M}$</td>
<td>122</td>
<td>118</td>
</tr>
<tr>
<td>$\bar{u}$</td>
<td>15</td>
<td>9</td>
</tr>
</tbody>
</table>
OpSearch is better than OpSeq: solves more tasks, solves less subproblems, uses less memory and generate smaller constraints.
OpSearch Improves with Better Heuristics
As a more informed heuristic is used by OpSearch, the number of subproblems solved, the memory usage and the size of the generated constraints decrease and the number of solved tasks increases.

<table>
<thead>
<tr>
<th></th>
<th>$OpSeq$</th>
<th>$h^{\text{blind}}$</th>
<th>$h^{\text{LMCut}}$</th>
<th>$h^{\text{OC}}$</th>
<th>$h^*$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S$</td>
<td>29106</td>
<td>25059</td>
<td>13304</td>
<td>7215</td>
<td>3214</td>
</tr>
</tbody>
</table>
As a more informed heuristic is used by OpSearch, the number of subproblems solved, the memory usage and the size of the generated constraints decrease and the number of solved tasks increases.
As a more informed heuristic is used by OpSearch, the number of subproblems solved, the memory usage and the size of the generated constraints decrease and the number of solved tasks increases.
As a more informed heuristic is used by OpSearch, the number of subproblems solved, the memory usage and the size of the generated constraints decrease and the number of solved tasks increases.

<table>
<thead>
<tr>
<th></th>
<th>OpSeq</th>
<th>$h^{\text{blind}}$</th>
<th>$h^{\text{LMCut}}$</th>
<th>$h^{\text{OC}}$</th>
<th>$h^{*}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S$</td>
<td>29106</td>
<td>25059</td>
<td>13304</td>
<td>7215</td>
<td>3214</td>
</tr>
<tr>
<td>$\bar{T}_t$</td>
<td>37</td>
<td>10</td>
<td>11</td>
<td>39</td>
<td>13</td>
</tr>
<tr>
<td>$\bar{M}$</td>
<td>95</td>
<td>82</td>
<td>82</td>
<td>81</td>
<td>234</td>
</tr>
<tr>
<td>$\bar{u}$</td>
<td>18</td>
<td>18</td>
<td>11</td>
<td>10</td>
<td>8</td>
</tr>
</tbody>
</table>
As a more informed heuristic is used by OpSearch, the number of subproblems solved, the memory usage and the size of the generated constraints decrease and the number of solved tasks increases.

<table>
<thead>
<tr>
<th></th>
<th>OpSeq</th>
<th>$h^{\text{blind}}$</th>
<th>$h^{\text{LMCut}}$</th>
<th>$h^{\text{OC}}$</th>
<th>$h^*$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S$</td>
<td>29106</td>
<td>25059</td>
<td>13304</td>
<td>7215</td>
<td>3214</td>
</tr>
<tr>
<td>$\bar{T}_t$</td>
<td>37</td>
<td>10</td>
<td>11</td>
<td>39</td>
<td>13</td>
</tr>
<tr>
<td>$\bar{M}$</td>
<td>95</td>
<td>82</td>
<td>82</td>
<td>81</td>
<td>234</td>
</tr>
<tr>
<td>$\bar{u}$</td>
<td>18</td>
<td>18</td>
<td>11</td>
<td>10</td>
<td>8</td>
</tr>
<tr>
<td>$C$</td>
<td>169</td>
<td>191</td>
<td>195</td>
<td>200</td>
<td>241</td>
</tr>
</tbody>
</table>
As a more informed heuristic is used by *OpSearch*, the number of subproblems solved, the memory usage and the size of the generated constraints decrease and the number of solved tasks increases.
Applications
Applications

- Our research results are relevant in practical applications besides the further development of automated planning;
Applications

- Our research results are relevant in practical applications besides the further development of automated planning;
- *OpSearch* can be used as an anytime method to obtain lower-bounds on plan costs;
Applications

- Our research results are relevant in practical applications besides the further development of automated planning;
- *OpSearch* can be used as an anytime method to obtain lower-bounds on plan costs;
- Also in *agile planning* to solve planning tasks for which informative heuristics are already known;
Applications

- Our research results are relevant in practical applications besides the further development of automated planning;
- *OpSearch* can be used as an anytime method to obtain lower-bounds on plan costs;
- Also in *agile planning* to solve planning tasks for which informative heuristics are already known;
- Another practical application of our approach is for *diverse planning*, used for example by IBM, that aims to find several plans while guaranteeing diversity.
Take Home Messages
Take Home Messages

- The operator counts sequencing problem can be efficiently solved using heuristic search;
The operator counts sequencing problem can be efficiently solved using heuristic search;
Our approach opens new research directions towards specialized methods or heuristics to this problem;
Take Home Messages

- The operator counts sequencing problem can be efficiently solved using heuristic search;
- Our approach opens new research directions towards specialized methods or heuristics to this problem;
- It is a novel research problem with great potential of development in both areas of operations research and artificial intelligence.
Thanks!

Wesley Luciano Kaizer
kaizerwesley@gmail.com
References I

Sequencing operator counts.
In International Conference on Automated Planning and Scheduling, pages 61–69.