Dynamic Task Allocation Algorithms within Intelligent Operator Support Concepts for Shore Control Centres

Tycho Brug, Jasper van der Waa, Valentina Maccatrozzo, Hans van den Broek
How to support remote operators in their supervision of dozens of autonomous operations in the maritime industry?
The future we contribute to...
and the problem we tackle.
The future we contribute to...

Robotic Container Handling System
Intelligent Operator Support
Support functions

Live 3D scanning

Continuous Interpretable Risk Assessment

Progressive Disclosure

Dynamic Allocation of Operations to Operators
Dynamic Operation Allocation: Who does what?

Operator expertise & workload

Operation difficulty

3x
7x
4x
The framework: How is an operation defined?

Each operation is composed of multiple tasks, with each a default workload $w_o(t)$. Tasks are linked to each other and sequential in a given operation.
The workload function $w_{o,p}(t)$ is defined as:

$$w_{o,p}(t) = \begin{cases} 
  w_o(t), & e_p \geq s_o \\
  w_o(t) \cdot \left(\frac{s_o}{e_p}\right), & e_p < s_o 
\end{cases}$$
Dynamic Operation Allocation: Aggregated workload

\[ w_{p=1}(t) = w_{o=1,p=1}(t) + w_{o=2,p=1}(t) + w_{o=3,p=1}(t) \]
Dynamic Operation Allocation; Aggregated workload

Operation difficulty

Operator expertise & workload
What is the most optimal division?

Cost function

Capacity of an operator

\[ w_p(t) < 100\% \text{ for every } t \text{ and every } p \]

Average mean workload

\[ \overline{W} = \frac{1}{M} \sum_{p=1}^{M} \overline{W}_p \]

Standard deviation of mean workload

\[ \sigma_w = \sqrt{\frac{1}{M} \sum_{p=1}^{M} (w_p - \overline{w})} \]

Average time in a critical workload zone

\[ \overline{c_w} = \frac{1}{MN} \sum_{p=1}^{M} \sum_{t=1}^{N} w_p(t) > 0.85 \]

Ability for the operator to take a break

\[ \overline{b_w} \]

Cost function

\[ C = \alpha \overline{w} + \beta \sigma_w + \gamma \overline{c_w} + \delta \overline{b_w} \]
Dynamic Operation Allocation; Aggregated workload

Operation difficulty

- C=2
- C=0.5
- C=1.5

Operator expertise & workload

- 🌟🌟🌟
- 🌟🌟
- 🌟
Optimization needed

[Diagram showing three ships with ratings and question marks.]
Optimization needed

**Brute force**

<table>
<thead>
<tr>
<th>Operator</th>
<th>Alternative 1</th>
<th>Alternative 2</th>
<th>Alternative 3</th>
<th>Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>$p_1$</td>
<td>$[o_1, o_2, o_3]$</td>
<td>$[o_1, o_2]$</td>
<td>$[o_1, o_3]$</td>
<td>4.5</td>
</tr>
<tr>
<td>$p_2$</td>
<td>$-$</td>
<td>$[o_3]$</td>
<td>$[o_2]$</td>
<td>4.3</td>
</tr>
<tr>
<td>$p_3$</td>
<td>$-$</td>
<td>$-$</td>
<td>$[o_3]$</td>
<td>4.0</td>
</tr>
<tr>
<td>$p_1$</td>
<td>$-$</td>
<td>$-$</td>
<td>$[o_1, o_2, o_3]$</td>
<td>4.2</td>
</tr>
<tr>
<td>$p_2$</td>
<td>$-$</td>
<td>$[o_2]$</td>
<td>$[o_1, o_3]$</td>
<td>4.1</td>
</tr>
<tr>
<td>$p_3$</td>
<td>$-$</td>
<td>$-$</td>
<td>$[o_1, o_2]$</td>
<td>5.0</td>
</tr>
</tbody>
</table>

\[ N = p^o \]

**Iterative deepening depth-first search (IDDFS) algorithm**

<table>
<thead>
<tr>
<th>Operator</th>
<th>Operands</th>
<th>Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>$p_1$</td>
<td>$[o_1]$</td>
<td>0.4</td>
</tr>
<tr>
<td>$p_2$</td>
<td>$[o_1, o_2]$</td>
<td>0.1</td>
</tr>
<tr>
<td>$p_3$</td>
<td>$[o_1]$</td>
<td>0.3</td>
</tr>
</tbody>
</table>

Operations to allocate:

<table>
<thead>
<tr>
<th>Operator</th>
<th>Candidates</th>
</tr>
</thead>
<tbody>
<tr>
<td>$p_1$</td>
<td>$[o_1]$</td>
</tr>
<tr>
<td>$p_2$</td>
<td>$[o_1, o_2]$</td>
</tr>
<tr>
<td>$p_3$</td>
<td>$[o_1]$</td>
</tr>
</tbody>
</table>

\[ N = \text{repetitions} \times p \times o, \]
Re-Optimization needed?

What if:
- An operation is delayed
- A new operation is added
- An operation takes longer than expected...

Making changes to an already existing allocation is not preferred!

Solution:

\[ |C - C'| > \varepsilon \times \Delta o \]
Results – Simple Example

Operators: 3 ($e_p = [1, 2, 3]$)

Operations:

<table>
<thead>
<tr>
<th>o</th>
<th>$t_{o,\text{start}}$</th>
<th>$s_o$</th>
<th>Work profile</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0:00</td>
<td>1</td>
<td>30% (0-1h)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>60% (1-2h)</td>
</tr>
<tr>
<td>2</td>
<td>1:00</td>
<td>2</td>
<td>30% (0-1h)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>60% (1-2h)</td>
</tr>
<tr>
<td>3</td>
<td>2:00</td>
<td>3</td>
<td>30% (0-1h)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>60% (1-2h)</td>
</tr>
<tr>
<td>4</td>
<td>4:00</td>
<td>1</td>
<td>30% (0-1h)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>60% (1-2h)</td>
</tr>
<tr>
<td>5</td>
<td>5:00</td>
<td>2</td>
<td>30% (0-1h)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>60% (1-2h)</td>
</tr>
<tr>
<td>6</td>
<td>6:00</td>
<td>3</td>
<td>30% (0-1h)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>60% (1-2h)</td>
</tr>
</tbody>
</table>

Evaluations: 729
Results – Simple Example

Operators: 3 (\(e_p = \{1, 2, 3\}\))

Operations:

<table>
<thead>
<tr>
<th>(o)</th>
<th>(t_{o, \text{start}})</th>
<th>(s_o)</th>
<th>Work profile</th>
</tr>
</thead>
</table>
| 1    | 0:00           | 3     | 30% (0-1h)  
          |                 |       | 60% (1-2h)  |
| 2    | 1:00           | 3     | 30% (0-1h)  
          |                 |       | 60% (1-2h)  |
| 3    | 2:00           | 3     | 30% (0-1h)  
          |                 |       | 60% (1-2h)  |
| 4    | 4:00           | 3     | 30% (0-1h)  
          |                 |       | 60% (1-2h)  |
| 5    | 5:00           | 3     | 30% (0-1h)  
          |                 |       | 60% (1-2h)  |
| 6    | 6:00           | 3     | 30% (0-1h)  
          |                 |       | 60% (1-2h)  |

Evaluations: 729
Results – Extended Example

Operators:
3 \((e_p = [1, 2, 3])\)

Operations:
12 operations
Varying durations
[1.5h – 4h]
Varying start times:
[0h – 6h]
Varying difficulty:
[2.5-6]
Varying workloads
Results – Extended Example

Brute force quickly becomes unfeasible
IDDFS is the solution!

However, getting the global minimum is dependent on amount of repetitions

With 560 repetitions and \(560 \times 3 \times 12 = 20.160\) (6 sec) evaluations we reliably get to the global minimum.

But more important: It is scalable!

Evaluations (BF): 531.441 (5 min)
Results – Difficult Example

Operators:
6 ($e_p = [6, 6, 3, 3, 3, 2.5]$)

Operations:
16 operations
Varying durations
[1.5h – 4h]
Varying start times:
[0h – 6h]
Varying difficulty:
[2.5-6]
Varying workloads

Only possible with IDDFS
BF: $2.8 \times 10^{12}$ (65 years!)

Evaluations (IDDFS): 33.600 (10s)
Results – Difficult Example Extended

Operators:
6 ($e_p = [6, 6, 3, 3, 3, 2.5]$)

Operations:
3 Changes (1 update, 2 new)!

<table>
<thead>
<tr>
<th>$o$</th>
<th>$t_{o, start}$</th>
<th>$S_o$</th>
<th>Work profile</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>0:00 (3:00)</td>
<td>3</td>
<td>25% (0-0:45h)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>40% (0:45-1:15h)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>10% (1:15-1:30h)</td>
</tr>
<tr>
<td>17</td>
<td>4:00</td>
<td>6</td>
<td>20% (0-4h)</td>
</tr>
<tr>
<td>18</td>
<td>4:30</td>
<td>3</td>
<td>25% (0-0:45h)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>40% (0:45-1:15h)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>10% (1:15-1:30h)</td>
</tr>
</tbody>
</table>
Results – Difficult Example Extended

Operators:
6 ($e_p = [6, 6, 3, 3, 3, 2.5]$)

Re-optimization:
$O_3$ (from $P_4 \rightarrow P_1$)
$O_6$ (from $P_1 \rightarrow P_3$)

<table>
<thead>
<tr>
<th>$o$</th>
<th>$t_{0, \text{start}}$</th>
<th>$s_o$</th>
<th>Work profile</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>0:00</td>
<td>3</td>
<td>25% (0-0:45h)</td>
</tr>
<tr>
<td></td>
<td>(3:00)</td>
<td>(6)</td>
<td>40% (0:45-1:15h)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>10% (1:15-1:30h)</td>
</tr>
<tr>
<td>17</td>
<td>4:00</td>
<td>6</td>
<td>20% (0-4h)</td>
</tr>
<tr>
<td>18</td>
<td>4:30</td>
<td>3</td>
<td>25% (0-0:45h)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>40% (0:45-1:15h)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>10% (1:15-1:30h)</td>
</tr>
</tbody>
</table>
Conclusions

The results show that the proposed dynamic task allocation algorithm can distribute operations over operators in a way that optimizes the workload following human factors knowledge.

IDDFS makes the dynamic task allocation scalable
Re-optimization makes the task allocation useful (a plan never goes to plan)

But also – Still a lot of work to do:
- The proposed cost function is an approximation of reality.
- The workload of supervising one or more operations might be hard to define as a predetermined percentage.
- Modelling an operator’s expertise or operation’s complexity in a single score, can be too simplistic
- Weighting of costs needs to be elicited (maybe dynamically)
- Most important: Evaluation in real world scenario needs to wait on further developments in autonomy
Dynamic Allocation of Operations to Operators

Progressive Disclosure Dynamic Allocation of Operations to Operators?

Continuous Interpretable Risk Assessment

Live 3D scanning
Thank you for your attention!

Tycho Brug, TNO

tycho.brug@tno.nl

This project has received funding from the European Union’s horizon 2020 research and innovation programme under grant agreement No. 861678.