Multi-Agent A* for Parallel and Distributed Systems

Raz Nissim    Ronen Brafman

Ben-Gurion University of the Negev, Beer-Sheva, Israel
Contributions

In a nutshell

- A Multi-agent version of A* that can be used for satisficing and optimal search, with a focus on parallel and distributed planning.
- Algorithm is inherently distributed.
- Parallel and distributed versions differ only in information used to compute the heuristics.
Main Idea

Agents perform A* with the following changes:

1. An agent expands a node using *only* its own operators.
2. Agents send newly generated nodes to agents having an applicable operator for the node.
3. A distributed optimal solution detection procedure.
Distributed search

Given:
A distributed system: Each agent has different operators. Some operator are private – interact with the agent’s operators only; known to her only.

Goal:
Fast, distributed optimal search, respecting agent privacy.
Distributed search

Given:
A distributed system: Each agent has different operators. Some operators are private – interact with the agent’s operators only; known to her only.

Goal:
Fast, distributed optimal search, respecting agent privacy.

Performance:
Outperforms best current distributed planner on all IPC benchmarks having MA structure, solves 6 previously unsolved problems.
Distributed search

Given:
A distributed system: Each agent has different operators. Some operator are private – interact with the agent’s operators only; known to her only.

Goal:
Fast, distributed optimal search, respecting agent privacy.

Performance:
Outperforms best current distributed planner on all IPC benchmarks having MA structure, solves 6 previously unsolved problems.

- MA-A* solves problems \textit{optimally}, unlike existing methods.
- Utilizes heuristics developed for forward search.
Parallel search

**Given:**
A regular search/planning problem.

**Goal:**
Exploit parallel hardware.
Parallel search

Given:
A regular search/planning problem.

Goal:
Exploit parallel hardware.

Method:
Factor the problem into a multi-agent problem by partitioning the operators among agents (can be done automatically – see ECAI 12’), then use Multi-agent A*.
## Parallel search

<table>
<thead>
<tr>
<th><strong>Given:</strong></th>
<th>A regular search/planning problem.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Goal:</strong></td>
<td>Exploit parallel hardware.</td>
</tr>
<tr>
<td><strong>Method:</strong></td>
<td>Factor the problem into a multi-agent problem by partitioning the operators among agents (can be done automatically – see ECAI 12’), then use Multi-agent A*.</td>
</tr>
<tr>
<td><strong>Performance:</strong></td>
<td><em>Superlinear</em> speedup (up to x20 faster) in decomposable problems, solving 7 instances unsolved by A*.</td>
</tr>
</tbody>
</table>
The model

Definition (Multi-Agent SAS$^+$)

Standard SAS$^+$ planning problem where the actions are partitioned between agents.
The model

Definition (Multi-Agent SAS$^+$)

Standard SAS$^+$ planning problem where the actions are partitioned between agents.

- In the distributed setting, the partition is induced by the agents of the system, i.e. the problem structure.
- In the parallel setting, the partition is created automatically in order to *artificially factorize* the problem.
### Public vs. private actions

#### Dependencies of actions

The partition induces a distinction of actions and variables as *public* (affecting/affected by more than one agent) and *private* (affecting/affected by a single agent).

#### Example - Logistics

- A vehicle’s *move* actions are private since they affect only the agent performing them.
- *load/unload* actions are public as they may achieve or destroy other agents’ preconditions.
Public vs. private actions

Dependencies of actions

The partition induces a distinction of actions and variables as public (affecting/affected by more than one agent) and private (affecting/affected by a single agent).

- In the parallel setting, agents have complete knowledge of the problem.
- In the distributed setting, agents are unaware of other agents’ private actions and variables. \(\Rightarrow\) each agent has an abstract view of the system.
Our approach – optimal forward search

Each agent runs an A*-like search separately, using its own open/closed list. In each iteration, the agent performs:

**MA-A***

- Receive messages and insert states into open list.
- Retrieve first node $n$ from open list.
- If $n$ is a solution, perform distributed optimality check.
- Expand $n$ using the agent’s own actions only.
- Compute h-value and add to open list all children $n'$.  
  - If $n'$ was obtained by applying a public action
    - then send $n'$ to all agents to which $n'$ is relevant.

Messages contain the full state $n'$, its $g$ and $h$-values, and its creating action.
Main contributions and results
The model & our approach
Properties of MA-A*
Experiments & future work

Running example

Partitioned action graph

A1

a3=<(v2=0),{},(v1=1)>
a4=<(v2=1),{},(v1=0)>
a2=<(v1=1),{},(v1=2)>
a5=<(v1=2),{},(v1=3)>
a8=<(v1=3),{},(v3=2)>
a6=<(v3=0),{},(v3=1)>
a7=<(v3=1),{},(v3=2)>

A2

A* search space

Agent 1

Agent 2

MA-A*

Nissim & Brafman  MA-A* for Parallel and Distributed Systems  10/25
Running example

Partitioned action graph

\( A_1 \)

- \( a_3 = \langle v_2 = 0, \emptyset, \{v_2 = 1\} \rangle \)
- \( a_2 = \langle v_1 = 1, \emptyset, \{v_2 = 2\} \rangle \)
- \( a_5 = \langle v_1 = 2, \{v_2 = 2\}, \{v_1 = 3\} \rangle \)

\( A_2 \)

- \( a_8 = \langle v_1 = 3, \{v_3 = 2\}, \{v_1 = 4\} \rangle \)
- \( a_6 = \langle v_3 = 0, \emptyset, \{v_3 = 1\} \rangle \)
- \( a_7 = \langle v_3 = 1, \emptyset, \{v_3 = 2\} \rangle \)

A* search space

- \( a_1 \)
- \( a_2 \)
- \( a_3 \)
- \( a_4 \)
- \( a_5 \)
- \( a_6 \)
- \( a_7 \)

MA-A*

- Agent 1
- Agent 2
Running example

Partitioned action graph

A^1

a3=<\{v2=0\},\{v1=1\}>
a1=<\{v1=0\},\{v1=1\}>
a4=<\{v2=1\},\{v2=2\}>
a2=<\{v1=1\},\{v1=2\}>
a5=<\{v1=2\},\{v1=3\}>

A^2

a8=<\{v1=3\},\{v3=2\},\{v1=4\}>
a6=<\{v3=0\},\{v3=1\}>
a7=<\{v3=1\},\{v3=2\}>

A* search space

MA-A^*

Nissim & Brafman  MA-A^* for Parallel and Distributed Systems  10/25
Running example
Running example

Partitioned action graph

A1

a3=<v2=0>,{},v2=1>

a1=<v1=0>,{},v1=1>

a4=<v2=1>,{},v2=2>

a2=<v1=1>,{},v1=2>

a5=<v1=2>,v2=2,v1=3>

a8=<v1=3>,v3=2,v1=4>

a6=<v3=0>,{},v3=1>

a7=<v3=1>,{},v3=2>

A2

A* search space

MA-A
Relevancy of messages

- A state $s$ is relevant to an agent if it has a public action for which all public preconditions hold in $s$.
- When some agent performs a private action, other agents’ view of the system relevant to them has not changed!
- Sending only states for which the creating action is public, maintains optimality.
- This effectively prunes many equivalent parts of the search space $\Rightarrow$ may result in fewer expansions than centralized A*.
Termination detection

- Unlike in A*, expansion of goal state does not mean an optimal solution has been found.
- A solution state $s$ is optimal if and only if $g(s) \leq f_{lower-bound}$.
- We use the snapshot algorithm, which detects stable properties of a distributed system.
- $f_{lower-bound} > c$ is stable if the heuristic estimates are globally consistent (pathmax + max operator).
Experimental results: Comparison of centralized A*, MA-A* in its parallel (MAP-A*) and distributed (MAD-A*) settings, and Planning First. Runtime (in sec.), number of expanded nodes and efficiency values (speedup/# of processors) are shown.

<table>
<thead>
<tr>
<th>Problem</th>
<th># agents</th>
<th>LM-cut heuristic</th>
<th>Merge&amp;Shrink heuristic</th>
<th>Planning First</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>A*</td>
<td>time</td>
<td>expands</td>
</tr>
<tr>
<td>logistics7-1(4)</td>
<td>4</td>
<td>55.3</td>
<td>155289</td>
<td>27</td>
</tr>
<tr>
<td>logistics8-1(4)</td>
<td>4</td>
<td>24.1</td>
<td>43665</td>
<td>13</td>
</tr>
<tr>
<td>logistics10-0(4)</td>
<td>5</td>
<td>203</td>
<td>193846</td>
<td>66.6</td>
</tr>
<tr>
<td>Rovers(2)</td>
<td>2</td>
<td>0</td>
<td>50</td>
<td>0</td>
</tr>
<tr>
<td>Rovers(4)</td>
<td>2</td>
<td>0</td>
<td>9</td>
<td>0.04</td>
</tr>
<tr>
<td>Rovers(5)</td>
<td>2</td>
<td>8.8</td>
<td>37397</td>
<td>1.8</td>
</tr>
<tr>
<td>Rovers(6)</td>
<td>2</td>
<td>–</td>
<td>–</td>
<td>236</td>
</tr>
<tr>
<td>Rovers(7)</td>
<td>3</td>
<td>6.7</td>
<td>18315</td>
<td>1</td>
</tr>
<tr>
<td>Rovers(8)</td>
<td>4</td>
<td>–</td>
<td>–</td>
<td>154</td>
</tr>
<tr>
<td>Rovers(12)</td>
<td>4</td>
<td>12.1</td>
<td>15222</td>
<td>0.9</td>
</tr>
<tr>
<td>Rovers(14)</td>
<td>4</td>
<td>–</td>
<td>–</td>
<td>598</td>
</tr>
<tr>
<td>satellites5(3)</td>
<td>3</td>
<td>1.3</td>
<td>1174</td>
<td>0.1</td>
</tr>
<tr>
<td>satellites6(3)</td>
<td>3</td>
<td>3.5</td>
<td>2976</td>
<td>0.2</td>
</tr>
<tr>
<td>satellites7(4)</td>
<td>4</td>
<td>94.5</td>
<td>36652</td>
<td>12.4</td>
</tr>
<tr>
<td>satellites8(4)</td>
<td>4</td>
<td>–</td>
<td>–</td>
<td>94.8</td>
</tr>
<tr>
<td>satellites9(4)</td>
<td>5</td>
<td>–</td>
<td>–</td>
<td>105</td>
</tr>
<tr>
<td>satellites10(4)</td>
<td>5</td>
<td>–</td>
<td>–</td>
<td>61.8</td>
</tr>
<tr>
<td>zenotravel9(3)</td>
<td>3</td>
<td>72.1</td>
<td>15408</td>
<td>20</td>
</tr>
<tr>
<td>zenotravel10(3)</td>
<td>3</td>
<td>16.1</td>
<td>1587</td>
<td>4.3</td>
</tr>
<tr>
<td>zenotravel12(3)</td>
<td>3</td>
<td>458</td>
<td>41311</td>
<td>57</td>
</tr>
<tr>
<td>zenotravel13(3)</td>
<td>3</td>
<td>–</td>
<td>–</td>
<td>382</td>
</tr>
</tbody>
</table>
Related work

Distributed planning

MA-A* dominates the DisCSP based approach *Planning-First* in all IPC problems having MA structure.
Related work

Distributed planning

MA-A* dominates the DisCSP based approach Planning-First in all IPC problems having MA structure.

Parallel planning

MA-A* is the first parallel algorithm, for which the distribution of the search space and the search operators is dependent on the structure of the problem (unlike, e.g. HDA* and PBNF).
Future work

Empirically

- Check how MA-A* scales with > 4 processors (this requires larger, yet solvable problem instances).
- Evaluate communication w.r.t other distributed approaches.
- Compare to other parallel approaches (e.g. PBNF).
Future work

Empirically
- Check how MA-A* scales with > 4 processors (this requires larger, yet solvable problem instances).
- Evaluate communication w.r.t other distributed approaches.
- Compare to other parallel approaches (e.g. PBNF).

Theoretically
- Why does MA-A* achieve superlinear speedup? (in ECAI 12’)
- How should MA-A* be modified when messages take longer to arrive?