Multi-Robot Task Scheduling

Yu ("Tony") Zhang  Lynne E. Parker

Distributed Intelligence Laboratory
Electrical Engineering and Computer Science Department
University of Tennessee, Knoxville TN, USA

IEEE International Conference on
Robotics and Automation, 2013
Multi-robot task scheduling

Multi-robot tasks:

Individual robots may not have all the required capabilities

Scheduling:

- A set of robots, $R = \{r_1, \ldots, r_i, \ldots\}$
- A set of tasks, $T = \{t_1, \ldots, t_i, \ldots\}$

Build a schedule to optimize a $func, \{R_i, s_i, p_i\}_i$
Multi-robot task scheduling

To represent a general scheduling problem: $P|T|\text{func}$

Multi-robot task scheduling:

- $P \rightarrow$ Multi-purpose processor
- $T \rightarrow$ Multi-processor task
- Restrictions:
  - Execution is non-preemptive
  - Robots are non-divisible

or the $MPM\ MPT$ problem [Gerkey and Mataric, 2004]
Complexity of $MPM$ $MPT$ 

With $func = \sum_i e_i$: 

- $MPM$: polynomial-time solvable 
- $MPT$: $\mathcal{NP}$-hard 
- $MPT2$: $\mathcal{NP}$-hard 

Two types of multi-robot tasks: 

- Loosely coupled: reducible to single robot tasks ($MPM$ $MPT$ becomes $MPM$) 
- Tightly coupled: ?

Efficient algorithms, preferably with solution bounds, are needed.
Scheduling for tightly coupled multi-robot tasks

Steps:
1. Reduce $MPM_{MPT}$ to $MPM$
2. Solve the $MPM$ problem

When considering a coalition as a robot, $MPM_{MPT}$ becomes $MPM$.

However, coalitions can interfere with each other:

Coalition 1: $\{r_1, r_4, r_5\}$
Coalition 2: $\{r_4, r_6\}$
Contributions

- Considers the scheduling problem for multi-robot tasks at the coalition level
- Proposes four efficient heuristics to address the problem with provable solution bounds
- Provides formal analyses and simulation results to demonstrate and compare their performances
### Notations

**Table: NOTATIONS USED**

<table>
<thead>
<tr>
<th>$R$</th>
<th>Set of robots</th>
<th>$r_i$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$C$</td>
<td>Set of coalitions</td>
<td>$c_j$</td>
</tr>
<tr>
<td>$T$</td>
<td>Set of tasks</td>
<td>$t_l$</td>
</tr>
<tr>
<td>$p_{jl}$</td>
<td>Processing time of $t_l$ by $c_j$</td>
<td></td>
</tr>
<tr>
<td>$e_l$</td>
<td>End time of task $t_l$</td>
<td></td>
</tr>
</tbody>
</table>

We consider $\text{func} = \sum_i e_l$
**Definition (MinProcTime)**

At each step:
1. Find the assignment that has the smallest $p_{jl}$
2. Schedule the task at the earliest possible time

**Theorem**

The MinProcTime heuristic yields a solution quality bounded by $\frac{|T|+1}{2}$.

Tight solution bound
MinStepSum

**Definition (MinStepSum)**

At each step:
- Find the assignment that increases $\sum e_i$ the least

**Theorem**

The MinStepSum heuristic yields a solution quality bounded by $\frac{|T|+1}{2}$.

Tight solution bound
To consider the interference between coalitions:

**Definition (Coalition Interference)**

For any two coalitions $c_j$ and $c_{j'} \ (j \neq j')$, $c_j$ interferes (or conflicts) with $c_{j'}$ if and only if $c_j \cap c_{j'} \neq \emptyset$.

Consider the impact of an assignment $c_j \rightarrow t_i$ on $\sum_i e_i$:

1. The assignment’s processing time $p_{jl}$
2. Tasks that are scheduled on $c_j$ after $t_i$
3. Tasks scheduled on coalitions that interfere with $c_j$
For $c_j \rightarrow t_i$:

1. The assignment’s processing time $p_{jl}$
2. Tasks that are scheduled on $c_j$ after $t_i$

Together, contribute $l_{jl} \cdot p_{jl}$

$l_{jl}$: scheduling position for $t_i$ on $c_j$

For example:

$c_2 : t_2 \Rightarrow t_1 \Rightarrow t_3$

For $t_2$, $l_{22} = 3$ (including influence on $t_1$ and $t_3$)
For $c_j \rightarrow t_l$:

3. Tasks scheduled on coalitions that influence with $c_j$

Upper bound is $|\bigcup_{c \in F_j} N_c| \cdot p_{jl}$

$F_j$: coalitions that interfere with $c_j$

$N_c$: set of tasks that $c$ can accomplish
Convert \textit{MPM MPT} to \textit{MPM} by constructing an assignment problem:

- Create a task node for each task $t_l$
- Create a coalition-position node for each coalition $c_j$ and position pair, with positions ranging from 1 to $N_{c_j}$ for coalition $c_j$
- If a coalition $c_j$ can accomplish a task $t_l$, connect $t_l$ with all coalition-position nodes for $c_j$, and set the weights to be $(| \bigcup_{c \in F_j} N_c | + l_{jl}) \cdot p_{jl}$, respectively, based on $l_{jl}$

Now, solve this problem optimally.
**Lemma**

There exists a schedule that is no worse than the solution of the assignment problem.

**Theorem**

The schedule that is constructed from the solution of the assignment problem yields a solution quality bounded by \( \max_j \left| \bigcup_{c \in F_j} N_c \right| + 1 \).

- Quality dependent on complex structure of the problem instance
- Less coalition interference, better quality
- Optimal solution for single robot tasks
MinInterfere

In *InterfereAssign*:

- $| \bigcup_{c \in F_j} N_c |$ is an overestimation

**Definition (MinInterfere)**

At each step:

1. Compute $\beta_{jl}$ and choose the assignment that minimizes it:

   $$\beta_{jl} = e_{jl} + | \bigcup_{c \in F_j} N_c \setminus M_{jl} | \cdot p_{jl}$$

   $M_{jl}$: $t_l \cup$ the set of tasks that are scheduled before $c_j \rightarrow t_l$
### Table: SUMMARY OF DISCUSSED HEURISTICS

<table>
<thead>
<tr>
<th>Name</th>
<th>Solution Bound</th>
<th>Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Optimal</td>
<td>1</td>
<td>$O((</td>
</tr>
<tr>
<td>MinProcTime</td>
<td>$\frac{</td>
<td>T</td>
</tr>
<tr>
<td>MinStepTime</td>
<td>$\frac{</td>
<td>T</td>
</tr>
<tr>
<td>InterfereAssign</td>
<td>$\max_j \left</td>
<td>\bigcup_{c \in F_j} N_c \right</td>
</tr>
<tr>
<td>MinInterfere</td>
<td>Not Determined</td>
<td>$O(</td>
</tr>
</tbody>
</table>
A simple scenario

<table>
<thead>
<tr>
<th>Task</th>
<th>Robots Required</th>
<th>Process Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>1) Object 1</td>
<td>One gripper, one localizer</td>
<td>6</td>
</tr>
<tr>
<td>2) Object 2</td>
<td>One gripper, one localizer</td>
<td>6</td>
</tr>
<tr>
<td>3) Large Object</td>
<td>Two grippers</td>
<td>5</td>
</tr>
</tbody>
</table>
A simple scenario

Heuristics that consider the interference produce the optimal solution

Figure: Schedules created by our heuristics

Schedule by $MinProcTime$ and $MinStepTime$, with value 27

Schedule by $InterfereAssign$ and $MinInterfere$, with value 23 (optimal)
### Parameters

**Table: PARAMETERS USED IN THE SIMULATIONS**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$n_c$</td>
<td>No. of coalitions</td>
</tr>
<tr>
<td>$n_t$</td>
<td>No. of tasks</td>
</tr>
<tr>
<td>$n_f$</td>
<td>Average no. of conflicting coalitions per coalition</td>
</tr>
<tr>
<td>$n_e$</td>
<td>Average no. of executable tasks per coalition</td>
</tr>
<tr>
<td>$n_{min}$, $n_{max}$</td>
<td>Minimum and maximum processing time</td>
</tr>
</tbody>
</table>
The average solution quality is better than the proven bounds

*MinStepSum, InterfereAssign* and *MinInterfere* perform similarly; *InterfereAssign* is better for smaller $n_c$ (i.e., $3 – 4$)
Varying $n_t$

Similar observations
Since $n_f$ stays as a constant, the curve formed is smoother
Varying $n_f$

Performance decreases as the interference becomes more complex.

Y. Zhang and L.E. Parker
Increase of $n_{\text{max}}$ does not always decrease the performance
Varying $n_{max}$, with large $n_c$ and $n_t$

MinStepSum performs slightly better with large $n_c$ and $n_t$
Has potentials to be applied to large-size problems
Conclusions

- When there is less interference between coalitions, use InterfereAssign

- Otherwise, choose the best
Contributions

- Considers the scheduling problem for multi-robot tasks at the *coalition level*

- Proposes four *efficient* heuristics to address the problem with *provable solution bounds*

- Provides *formal analyses* and *simulation results* to demonstrate and compare their performances
References

A formal analysis and taxonomy of task allocation in multi-robot systems.

IQ-ASyMTRe: Synthesizing coalition formation and execution for tightly-coupled multirobot tasks.