F.L. Lewis
Moncrief-O’Donnell Endowed Chair
Head, Controls & Sensors Group

Automation & Robotics Research Institute
The University of Texas at Arlington

Wireless Sensor Networks

Talk available online at
http://ARRI.uta.edu/acs
II. ARRI Distributed Intelligence & Autonomy Lab

DIAL

Small mobile Sensor- Dan Popa

Unattended Ground Sensors

Testbed containing MICA2 network (circle), Cricket network (triangle), Sentry robots, Garcia Robots & ARRI-bots (Dan Popa)
Deadlock free dynamic resource assignment in multi-robot systems with multiple missions: a matrix-based approach

by

Prasanna Ballal, Vincenzo Giordano, Frank Lewis

Automation & Robotics Research Institute, University of Texas at Arlington

Distributed Intelligence and Autonomy Laboratory (DIAL)
Manufacturing Systems Control Design
A Matrix-based Approach
Discrete event controller

User interface: definition of mission planning, resource allocation, priority rules

Rule-based real time controller

\[ \bar{x} = F_v \otimes \bar{v} \oplus F_r \otimes \bar{r} \oplus F_u \otimes \bar{u} \oplus F_{uc} \otimes \bar{u}_C \]

Dispatching rules

Job start logic

\[ v_s = S_v \otimes x \]

Resource release logic

\[ r_s = S_r \otimes x \]

Task complete logic

\[ y = S_y \otimes x \]

Start tasks \( v_s \)

Start resource release \( r_s \)

Output \( y \)

Plant commands

Plant status

Mission completed

Resource released \( r_c \)

Tasks completed \( v_c \)

Sensor readings

Sensor output \( u \)

Decision-making

Wireless Sensor Network
Matrix Formulation: Definition

multiply = AND & addition = OR
overbar = negation

Logical state equation

\[ \bar{x} = F_v \bar{v}_c + F_r \bar{r}_c + F_u \bar{u} + F_D \bar{u}_D \]

State vector
Task vector
1 = job complete
Resource vector
1 = resource available
Input vector
1 = event detected
Control input
priority sequencing
deadlock avoidance
etc

Compare with \( x_{k+1} = Ax_k + Bu_k \)

Output equations

Job Start Equation:
\[ v_s = S_v x \]
Resource Release Equation:
\[ r_s = S_r x \]
Product Output Equation:
\[ y = S_y x \]
Meaning of Matrices

Task Sequencing Matrix
- Steward
- Prerequisite jobs
- Conditions fulfilled
- Next job
- $F_v$

Resource Requirements Matrix
- Kusiak
- Resources required
- Conditions fulfilled
- Next job
- $F_r$

Task Starting Matrix
- Conditions fulfilled
- Next job
- $S_v$

Resource Releasing Matrix
- Release resource
- Conditions fulfilled
- $S_r$
Construct Job Sequencing Matrix $F_v$

Used by Steward in Manufacturing Task Sequencing

Prerequisite jobs

Next jobs

Contains same information as the Bill of Materials (BOM)
Construct Resource Requirements Matrix $F_r$

Used by Kusiak in Manufacturing Resource Assignment

Contains information about factory resources

Prerequisite resources

Next jobs
OR / AND Matrix Algebra

\[ \bar{x} = F_v \bar{V}_c + F_r \bar{r}_c \]

Example

\[ \bar{x} = \begin{bmatrix} \bar{a} \\ \bar{b} \\ \bar{c} \end{bmatrix} = (1 \land \bar{a}) \lor (0 \land \bar{b}) \lor (1 \land \bar{c}) \]

\[ x = (1 \land \bar{a}) \lor (0 \land \bar{b}) \lor (1 \land \bar{c}) \]

\[ x = (1 \land \bar{a}) \land (0 \land \bar{b}) \land (1 \land \bar{c}) \]

\[ x = (0 \lor a) \land (1 \lor b) \land (0 \lor c) \]

\[ x = a \land c \]
Easy to implement OR/ AND algebra in MATLAB

Matrix multiply

\[ C = AB \]

for \( i = 1, I \)
    for \( j = 1, J \)
        \( c(i,j) = 0 \)
        for \( k = 1, K \)
            \( c(i,j) = c(i,j) \ OR \ ( a(i,k) \ AND \ b(k,j) ) \)
        end
    end
end
Relation to Max-Plus Algebra

\[
\bar{x} = F_v \bar{v}_c + F_r \bar{r}_c + F_u \bar{u} + F_D \bar{u}_D
\]

\(V_s = S_v x\)
\(r_s = S_r x\)
\(y = S_y x\)

State equation
Output equations

Define diagonal timing matrices. Then max plus is

\[
x' = S_v T_v F_v x + S_r T_r F_r r
\]

Can also include nonlinear terms- correspond to decisions
Relation to Petri Nets

Jobs complete

Trans.

Fv

Transition

Next jobs

Sv

Resources available

Trans.

Fr

Transition

Release resource

Sr
\[
\begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1
\end{bmatrix}
\]

\[
\begin{bmatrix}
0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1
\end{bmatrix}
\]

\[
\begin{bmatrix}
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0
\end{bmatrix}
\]
Complete DE Dynamical Description

DE state equation

$$\bar{x} = F_v \bar{v}_c + F_r \bar{r}_c + F_u \bar{u} + F_D \bar{u}_D$$

Activity Completion Matrix

$$F = \begin{bmatrix} F_u & F_v & F_r & F_y \end{bmatrix}$$

Activity Start Matrix

$$S = \begin{bmatrix} S_u^T & S_v^T & S_r^T & S_y^T \end{bmatrix}$$

PN incidence matrix

$$M = S^T - F = \begin{bmatrix} S_u^T - F_u, S_v^T - F_v, S_r^T - F_r, S_y^T - F_y \end{bmatrix}$$

Marking transition equation

$$m(t+1) = m(t) + M^T x = m(t) + [S^T - F]^T x$$

$$p = \begin{bmatrix} u^T & v^T & r^T & y^T \end{bmatrix}^T$$
Include Process Times in Places

Split up marking vector
\[ m(t) = m_a(t) + m_p(t) \]

Add tokens
\[ m_p(t + 1) = m_p(t) + S^T x(t) \]

Wait for process durations
\[ T = [O, vtimes^T, rtimes^T, O]^T \]
\[ T_{pend}(t + 1) = \text{diag}\{m_p(t)\}[T_{pend}(t) - t_{sample}] + \text{diag}\{S^T x(t)\}T \]

Take away tokens
\[ m_a(t + 1) = m_a(t) - F x(t) \]
Allows easy MATLAB simulation of DE systems

1. Rules- fire transitions
\[ \bar{x} = F_v \bar{v}_c + F_r \bar{r}_c + F_u \bar{u} + F_D \bar{u}_D \]

2. Add tokens to places
\[ m_p(t+1) = m_p(t) + S^T x(t) \]

3. Wait until jobs finish

\textit{Duration time counting routine}

4. Take tokens from places
\[ m_a(t+1) = m_a(t) - F x(t) \]

5. Find updated vectors for DE state equation
\[ m(t+1) = m_a(t+1) + m_p(t+1) \]

6. \[ \begin{bmatrix} u^T & v^T & r^T & y^T \end{bmatrix}^T = \overline{m(p)} \]
DEC for WSN

Mission result: False fire alarm

Robot 1 goes to sensor 2

Sensor 1 output: alarm

Robot 1 picks sensor 2

Robot 2 follows robot 1

Sensor 3 measurement

Sensor 4 measurement

Robot 1 places sensor 2

Sensor 2 measurement

Robot 1 goes to sensor 1

Smoke detection

Programmable Missions
## Fast Programming of Missions

### Mission 1 - Task sequence

<table>
<thead>
<tr>
<th>Task</th>
<th>Notation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Task 1</td>
<td>S4m1</td>
<td>UGS4 takes measurement</td>
</tr>
<tr>
<td>Task 2</td>
<td>S5m1</td>
<td>UGS5 takes measurement</td>
</tr>
<tr>
<td>Task 3</td>
<td>R1gS21</td>
<td>R1 goes to UGS2</td>
</tr>
<tr>
<td>Task 4</td>
<td>R2gA1</td>
<td>R2 goes to location A</td>
</tr>
<tr>
<td>Task 5</td>
<td>R1rS21</td>
<td>R1 retrieves UGS2</td>
</tr>
<tr>
<td>Task 6</td>
<td>R1lis1</td>
<td>R1 listens for interrupts</td>
</tr>
<tr>
<td>Task 7</td>
<td>R1gS11</td>
<td>R1 goes to UGS1</td>
</tr>
<tr>
<td>Task 8</td>
<td>R2m1</td>
<td>R2 takes measurement</td>
</tr>
<tr>
<td>Task 9</td>
<td>R1dS21</td>
<td>R1 deploys UGS2</td>
</tr>
<tr>
<td>Task 10</td>
<td>R1m1</td>
<td>R1 takes measurement</td>
</tr>
<tr>
<td>Task 11</td>
<td>S2m1</td>
<td>S2 takes measurement</td>
</tr>
<tr>
<td>Output</td>
<td>y1</td>
<td>Mission 1 completed</td>
</tr>
</tbody>
</table>

### Mission 2 - Task sequence

<table>
<thead>
<tr>
<th>Task</th>
<th>Notation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Task 1</td>
<td>S1m2</td>
<td>UGS1 takes measurement</td>
</tr>
<tr>
<td>Task 2</td>
<td>R1g S32</td>
<td>R1 goes to UGS3</td>
</tr>
<tr>
<td>Task 3</td>
<td>R1cS32</td>
<td>R1 charges UGS3</td>
</tr>
<tr>
<td>Task 4</td>
<td>S3m2</td>
<td>UGS3 takes measurement</td>
</tr>
<tr>
<td>Task 5</td>
<td>R1dC2</td>
<td>R1 docks the charger</td>
</tr>
<tr>
<td>Output</td>
<td>y2</td>
<td>Mission 2 completed</td>
</tr>
</tbody>
</table>
Mission 1 matrices

\[
\begin{align*}
F_v^1 &= x_5^1 \\
F_r^1 &= x_5^1
\end{align*}
\]

Mission 2 matrices

\[
\begin{align*}
F_v^2 &= x_3^2 \\
F_r^2 &= x_3^2
\end{align*}
\]
Simulation Results

Event traces
- Up means task in progress
- Down 😊 means resource in use

Simulation 2 –
change mission priority
Hi

g
Level Controller

Dispatching rules
To Generate

RS232

Wireless Network with Internet connection

Rule Based Real Time Controller

\[ x = F_v \bar{v}_c + F_r \bar{r}_c + F_u \bar{u} + F_D \bar{u}_D \]

Start tasks/jobs

Resource release

Mission result

Sensor output u

Task completed v

Resource released r

Finite state machine for each agent

UC-TDMA MAC protocol

Supervisor control level

Agent control level

Network control level

Agents

DE controller on PC

System model

\[ m_p (t + 1) = m_p (t) + S^T x(t) \]

Duration times

\[ m_a (t + 1) = m_a (t) - F x(t) \]

\[ m(t + 1) = m_a (t + 1) + m_p (t + 1) \]

Replaced by actual WSN

events
LabVIEW diagram of Controller
LabVIEW Controller's interface:
Results of LabVIEW Implementation on Actual Workcell

Compare with MATLAB simulation!

We can now simulate a DE controller and then implement it, Exactly as for continuous state controllers!!
Schematic Event Sequence for Mission Performance
Shared resources $\rightarrow$ circular waits
Simulation results

Event time trace

Deadlock!
One Step Look Ahead Deadlock Avoidance Policy

Some Definitions:

• **Circular Waits**
  - Circular waits (CW) among resources are a set of resources $ra, rb, \ldots rw$ whose wait relationship among them are $ra \xrightarrow{} rb \xrightarrow{} \ldots \xrightarrow{} rw$ and $rw \xrightarrow{} ra$.

• **Simple CW**
  - Simple CWs (sCW) are primitive CWs which do not contain other CWs.

• **Critical siphons, $S_c$**
  - The critical siphon of a CW is the smallest siphon containing the CW. Note that if the critical siphon ever becomes empty, the CW can never again receive any tokens. This is, the CW has become a circular blocking.
One Step Look Ahead Deadlock Avoidance Policy ctd...

• Siphon job set $J_s$
  - Siphon-job set, $J_s$, is the set of jobs which, when added to the set of resources contained in CW $C$, yields the critical siphon.

• Critical Subsystems $J_o$
  - The critical subsystems of the CW $C$, are the **job sets** $J(C)$ from that $C$ not contained in the siphon-job set $J_s$ of $C$. That is $J_o = J(C) \setminus J_s$. 
Deadlock Analysis

The Circular Waits are key to deadlock analysis (Wysk). CW are difficult to find from the Petri Net.

The CW are given directly from our matrices by the Wait Relation Matrix

\[ G_W = S_r F_r \]  

(in AND/OR algebra)

i.e. if \( g_{ij} = 1 \) then resource \( j \) waits for resource \( i \)

Use string algebra to find a matrix \( C \), wherein each column represents a circular wait. When a CW becomes blocked, one has Deadlock. To avoid this, examine the Critical Siphons. CS are difficult to find using Petri Nets.

The CS are given directly from our matrices by the columns of matrix

\[ S_c = \begin{bmatrix} C^T S_r F_v \land C^T F_r^T F_v \land \end{bmatrix} \]

Where vector \( C_s \) is the projection of \( C \) into the shared resources in the CW and \( \land \) denotes the element-by-element matrix ‘and’ operation.
One Step Look Ahead Deadlock Avoidance Policy ctd...

Critical Subsystems $J_o$

Matrix formulation for $J_o$

$$J_o = (d CF_v) \land (C_d F_v) = (C_d S_v^T) \land (C_d F_v)$$

Deadlock Avoidance Strategy- MAXWIP Policy

$$m(J_o(C_i)) < m_o(C_i)$$
Deadlock analysis

Resources

Circular Waits $\mathbf{C_i}$

Gurel’s algorithm

Critical subsystems

Maxwip policy

$m(J_o(C_i)) < m_o(C_i)$

Initial number of tokens in circular wait
$\mathbf{C_i}$ should always be greater than the number of tokens in the corresponding critical subsystem

Critical subsystems Matrix

Matrix operations

Digraph Matrix

$\mathbf{W}$

Tasks

$\mathbf{J_o}$

Resources

$\mathbf{C_o}$
Deadlock avoidance

Circular wait matrix

\[ C_o = \begin{bmatrix}
S1 & S2 & S3 & S4 & R1 & R2 \\
1 & 0 & 0 & 0 & 1 & 0 \\
0 & 1 & 0 & 0 & 0 & 1 \\
0 & 0 & 1 & 1 & 1 & 1 \\
1 & 0 & 1 & 1 & 1 & 1 \\
0 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 \\
\end{bmatrix} \]

Critical subsystem matrix

\[ J_o = \begin{bmatrix}
R1cS1 & R1dC & S3m & S2m & S4m \\
0 & 0 & 1 & 1 & 0 \\
0 & 0 & 1 & 1 & 0 \\
0 & 0 & 1 & 1 & 0 \\
0 & 0 & 1 & 1 & 0 \\
0 & 0 & 1 & 1 & 0 \\
0 & 0 & 1 & 1 & 0 \\
1 & 1 & 0 & 1 & 1 \\
1 & 1 & 0 & 1 & 1 \\
1 & 1 & 0 & 1 & 1 \\
1 & 1 & 0 & 1 & 1 \\
1 & 1 & 0 & 1 & 1 \\
1 & 1 & 0 & 1 & 1 \\
\end{bmatrix} \]
Deadlock avoidance

Sentry robot → Motion with Deadlock Avoidance → Charger

Motion with no Deadlock Avoidance → UGS
Simulation results

Event time trace without deadlock avoidance

Event time trace with deadlock avoidance
New Work—
Modify resource assignments on-line as sensor nodes fail or are added

\[ \overline{x} = F_v \overline{v}_c + F_r \overline{r}_c + F_u \overline{u} + F_D \overline{u}_D \]

Add or modify columns of \( F_r \)

Deadlock avoidance still works if there are no Key resources-1-step look ahead.
New Work --
Deadlock Avoidance for Free-Choice Routing Systems

Choices in routing- select resource for next job

Problems with the old formulae-
need to make critical subsystems include alternate routing resources
New Work-
A new matrix algebra to implement Dempster-Shafer decisions

Make the state equation
\[ \bar{x} = F_v \bar{v}_c + F_r \bar{r}_c + F_u \bar{u} + F_D \bar{u}_D \]

Implement the DS Rule of Combination
\[ m_{1,2}(A) = \frac{\sum_{B_i \cap C_j = A} m_1(B_i)m_2(C_j)}{\sum_{B \cap C \neq \phi} m_1(B_i)m_2(C_j)}. \]

Make the output equations
\[ v_s = S_v x \quad \quad r_s = S_r x \]

Implement
Belief
\[ Bel(A) = \sum_{B \subset A} m(B) \]

Plausibility
\[ Pl(a) = \sum_{B \cap A \neq 0} m(B) \]