UNIVERSITY OF TWENTE.

Improving the Performance of Periodic Real-time Processes: a Graph Theoretical Approach

Ton Boode
Hajo Broersma
Jan Broenink

Faculty of Electrical Engineering, Mathematics and Computer Science, University of Twente, The Netherlands

August 26, 2013

Overview

- **Periodic real-time processes represented by graphs**

Overview

- Periodic real-time processes represented by graphs
- **Cartesian product $H_i \square H_j$**

Overview

- Periodic real-time processes represented by graphs
- Cartesian product $H_i \square H_j$
- **Weak synchronised product** $H_i \boxminus H_j$

Overview

- Periodic real-time processes represented by graphs
- Cartesian product $H_i \square H_j$
- Weak synchronised product $H_i \boxminus H_j$
- **Reduced weak synchronised product $H_i \boxdot H_j$**

Overview

- Periodic real-time processes represented by graphs
- Cartesian product $H_i \Box H_j$
- Weak synchronised product $H_i \boxminus H_j$
- Reduced weak synchronised product $H_i \boxdot H_j$
- **Synchronised product** $H_i \boxbslash H_j$

Overview

- Periodic real-time processes represented by graphs
- Cartesian product $H_i \square H_j$
- Weak synchronised product $H_i \boxminus H_j$
- Reduced weak synchronised product $H_i \boxdot H_j$
- Synchronised product $H_i \boxbslash H_j$
- **Performance gain, necessary and sufficient conditions**

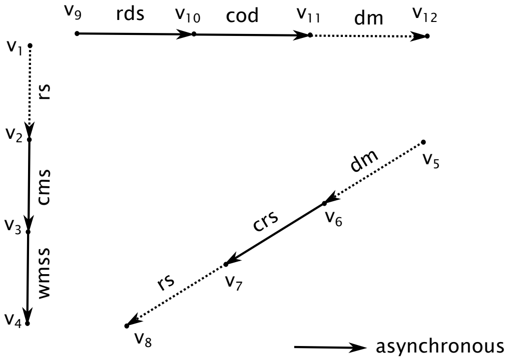
Overview

- Periodic real-time processes represented by graphs
- Cartesian product $H_i \square H_j$
- Weak synchronised product $H_i \boxminus H_j$
- Reduced weak synchronised product $H_i \boxdot H_j$
- Synchronised product $H_i \boxbslash H_j$
- Performance gain, necessary and sufficient conditions
- **Future work**

Parallel processes represented by graphs

$$
\begin{array}{lll}
\text{OBJECT\_ DISTANCE} = & \text{read\_distance\_sensors} & \rightarrow \\
 & \text{compute\_object\_distance} & \rightarrow \\
 & \text{distance\_meas} & \rightarrow \text{SKIP} \\
\text{ROBOT\_SPEED} = & \text{distance\_meas} & \rightarrow \\
 & \text{compute\_robot\_speed} & \rightarrow \\
 & \text{robot\_speed} & \rightarrow \text{SKIP} \\
\text{MOTOR\_SPEED} = & \text{robot\_speed} & \rightarrow \\
 & \text{compute\_motor\_speed} & \rightarrow \\
 & \text{write\_motor\_speed\_setpoint} & \rightarrow \text{SKIP} \\
\text{SEQUENCE\_CONTROL} = & \text{(OBJECT\_DISTANCE} & \| \\
 & \text{ROBOT\_SPEED} & \| \\
 & \text{MOTOR\_SPEED);} & \\
 & \text{SEQUENCE\_CONTROL;} &
\end{array}
$$

Parallel processes represented by graphs



SQ = MS + RS + OD

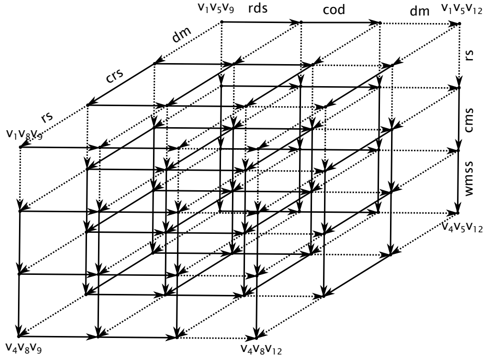
Parallel processes represented by graphs

$$
\begin{aligned}
MS = \ & (V(H_1), A(H_1), \{\lambda(a) | a \in A(H_1)\}) \\
= \ & (\{v_1, v_2, v_3, v_4\}, \{v_1 v_2, v_2 v_3, v_3 v_4\}, \\
& \{(v_1 v_2, rs), (v_2 v_3, cms), (v_3 v_4, wmss)\})
\end{aligned}
$$

$$
\begin{aligned}
RS = \ & (V(H_2), A(H_2), \{\lambda(a) | a \in A(H_2)\}) \\
= \ & (\{v_5, v_6, v_7, v_8\}, \{v_5 v_6, v_6 v_7, v_7 v_8\}, \\
& \{(v_5 v_6, dm), (v_6 v_7, crs), (v_7 v_8, rs)\})
\end{aligned}
$$

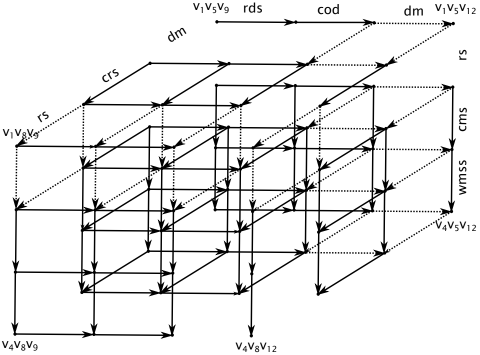
$$
\begin{aligned}
OD = \ & (V(H_3), A(H_3), \{\lambda(a) | a \in A(H_3)\}) \\
= \ & (\{v_9, v_{10}, v_{11}, v_{12}\}, \{v_9 v_{10}, v_{10} v_{11}, v_{11} v_{12}, \}, \\
& \{(v_9 v_{10}, rds), (v_{10} v_{11}, cod), (v_{11} v_{12}, dm)\})
\end{aligned}
$$

Cartesian product



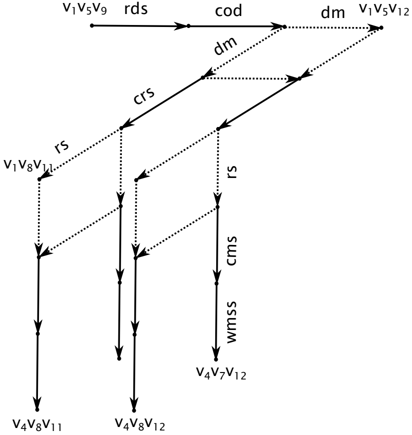
MS □ RS □ OD

Weak synchronised product



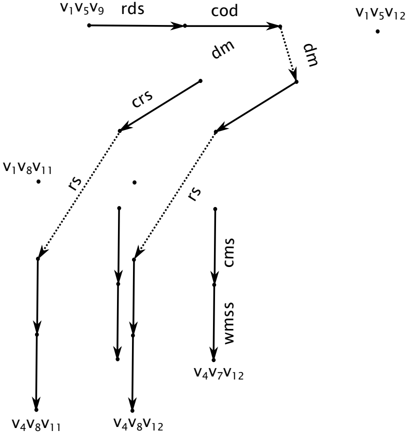
MS ⊟ RS ⊟ OD

Reduced weak synchronised product



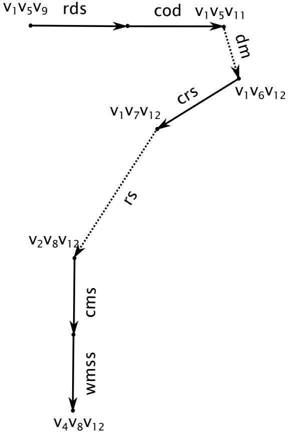
MS ⊡ RS ⊡ OD

Synchronised product intermediate stage



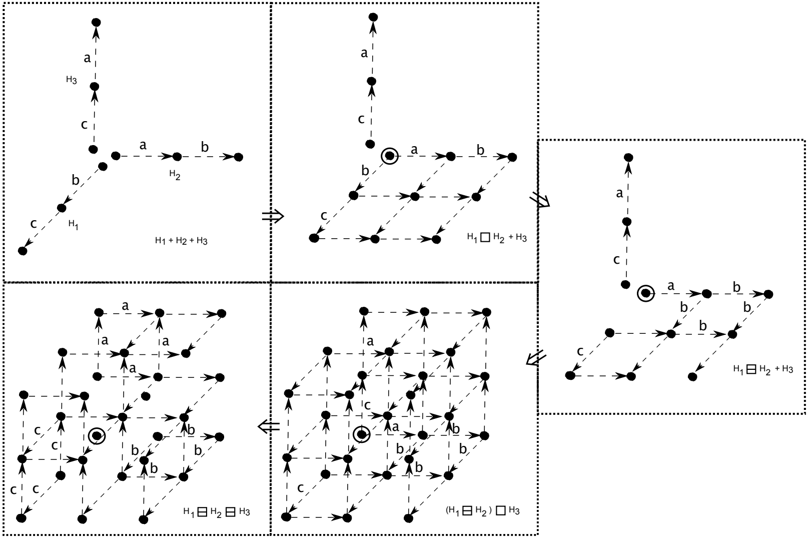
Intermediate stage

Synchronised product



MS ⧅ RS ⧅ OD

Multi dimensional pathological example



Lemma 5

Lemma

Let H_i be an acyclic graph for $i = 1, 2, \ldots, k$, where $k \geqslant 2$. Then $\ell(\boxbslash H_i) = \ell(H_1) + \ell(H_2) + \ldots + \ell(H_k)$ if and only if every H_i has at least one longest path without synchronising arcs.

Lemma 6

Lemma

Let H_i be an acyclic graph for $i = 1, 2, \ldots, k$, where $k \geqslant 2$. Then $\ell(\boxbslash H_i) < \ell(\boxdot H_i)$ if there exists H_n, H_m, $n \neq m, 1 \leqslant n, m \leqslant k$, such that each longest path in H_n, H_m, contains at least one same labelled synchronising arc.

Theorem 1

Theorem

Let H_i be an acyclic graph for $i = 1, 2, \ldots, k$, where $k \geqslant 2$. Then $\ell(\boxtimes H_i) < \ell(\boxdot H_i)$ if there exists H_n, H_m, $n \neq m, 1 \leqslant n, m \leqslant k$, such that each longest path in H_n, contains at least one synchronising arc and there is at least one longest path with a same labelled synchronisation arc in H_m.

Future work

- **Algorithms for optimising the performance gain**

Future work

- Algorithms for optimising the performance gain
- **The number of longest paths in a graph is exponential**

Future work

- Algorithms for optimising the performance gain
- The number of longest paths in a graph is exponential
- **Scheduling of the synchronised product with internal deadlines**

Future work

- Algorithms for optimising the performance gain
- The number of longest paths in a graph is exponential
- Scheduling of the synchronised product with internal deadlines
- **Memory usage**

Future work

- Algorithms for optimising the performance gain
- The number of longest paths in a graph is exponential
- Scheduling of the synchronised product with internal deadlines
- Memory usage
- **Synchronised product, associativity and commutativity**

Future work

- Algorithms for optimising the performance gain
- The number of longest paths in a graph is exponential
- Scheduling of the synchronised product with internal deadlines
- Memory usage
- Synchronised product, associativity and commutativity
- **Decomposition of a component into its prime factors**

Future work

- Algorithms for optimising the performance gain
- The number of longest paths in a graph is exponential
- Scheduling of the synchronised product with internal deadlines
- Memory usage
- Synchronised product, associativity and commutativity
- Decomposition of a component into its prime factors
- **Constraints for the prime factors of the synchronised product**

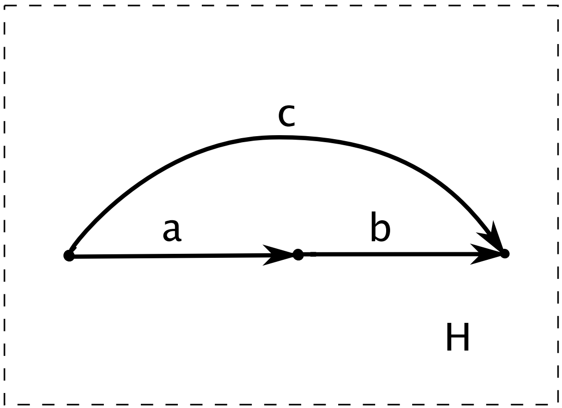
Future work

- Algorithms for optimising the performance gain
- The number of longest paths in a graph is exponential
- Scheduling of the synchronised product with internal deadlines
- Memory usage
- Synchronised product, associativity and commutativity
- Decomposition of a component into its prime factors
- Constraints for the prime factors of the synchronised product
- **Algorithm that calculates prime factors.**

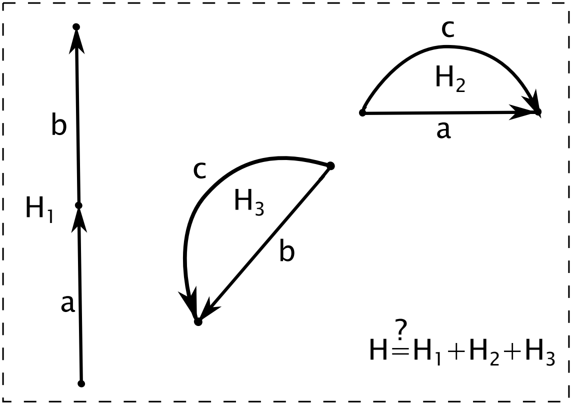
Future work

- Algorithms for optimising the performance gain
- The number of longest paths in a graph is exponential
- Scheduling of the synchronised product with internal deadlines
- Memory usage
- Synchronised product, associativity and commutativity
- Decomposition of a component into its prime factors
- Constraints for the prime factors of the synchronised product
- Algorithm that calculates prime factors.
- **An example of the decomposition of a graph**

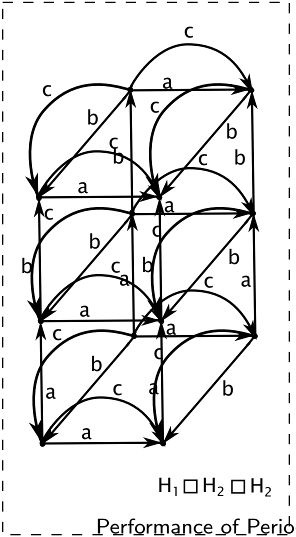
Decomposition of the original graph into its prime factors



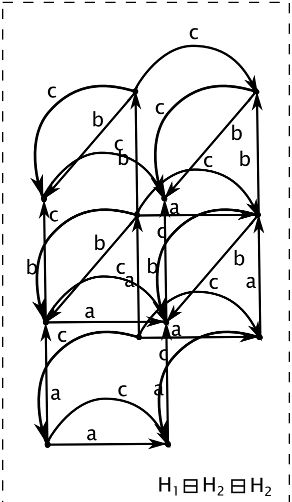
Decomposition of the original graph into its prime factors



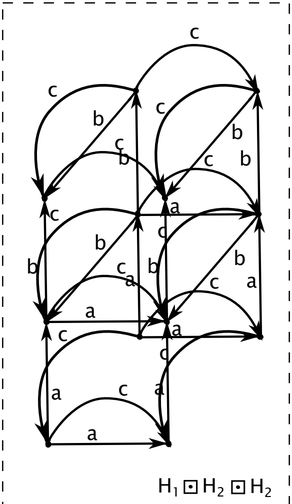
Decomposition of the original graph into its prime factors



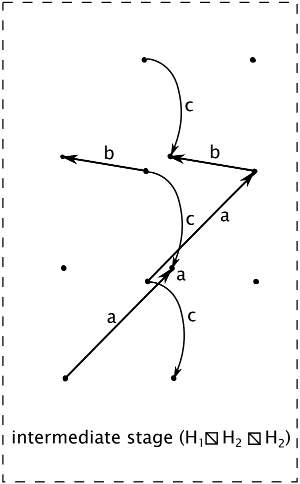
Decomposition of the original graph into its prime factors



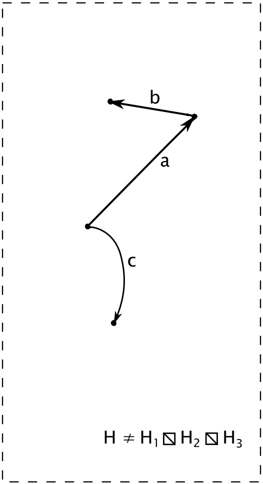
Decomposition of the original graph into its prime factors



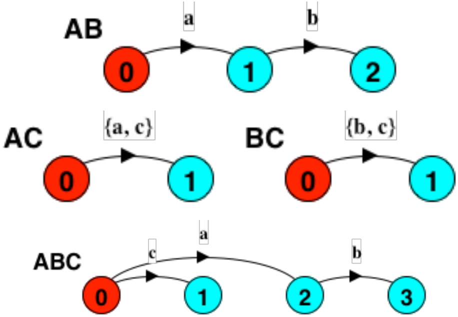
Decomposition of the original graph into its prime factors



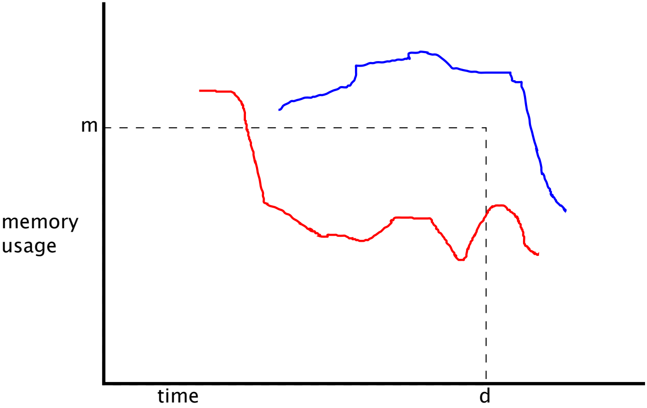
Decomposition of the original graph into its prime factors



Decomposition of a component into its prime factors



Memory usage versus performance using decomposition



Thank you for listening!

UNIVERSITY OF TWENTE.

Improving the Performance of Periodic Real-time Processes: a Graph Theoretical Approach

Ton Boode
Hajo Broersma
Jan Broenink

Faculty of Electrical Engineering, Mathematics and Computer Science, University of Twente, The Netherlands

August 26, 2013