Energy-aware mappings of series-parallel workflows onto chip multiprocessors

Anne Benoit, Paul Renaud-Goud and Yves Robert
LIP, École Normale Supérieure de Lyon, France

Rami Melhem, *University of Pittsburgh, USA*

International Research Workshop on
Advanced High Performance Computing Systems
Cetraro (Italy), June 27-29, 2011

Motivations

- Mapping streaming applications onto parallel platforms: practical applications (image processing, astrophysics, meteorology, neuroscience, ...), but difficult problems (NP-hard)
- Objective: maximize the throughput, i.e., minimize the period of the application
- Energy saving is becoming a crucial problem (economic and environmental reasons)
- M. P. Mills, The internet begins with coal, Environment and Climate News (1999)
- Objective of a mapping: minimize energy consumption while maintaining a given level of performance (bound on period)

Motivations

- Mapping streaming applications onto parallel platforms: practical applications (image processing, astrophysics, meteorology, neuroscience, ...), but difficult problems (NP-hard)
- Objective: maximize the throughput, i.e., minimize the period of the application

- Energy saving is becoming a crucial problem (economic and environmental reasons)
- M. P. Mills, The internet begins with coal, Environment and Climate News (1999)

- Objective of a mapping: minimize energy consumption while maintaining a given level of performance (bound on period)

Motivations

- Mapping streaming applications onto parallel platforms: practical applications (image processing, astrophysics, meteorology, neuroscience, ...), but difficult problems (NP-hard)
- Objective: maximize the throughput, i.e., minimize the period of the application
- Energy saving is becoming a crucial problem (economic and environmental reasons)
- M. P. Mills, The internet begins with coal, Environment and Climate News (1999)
- Objective of a mapping: minimize energy consumption while maintaining a given level of performance (bound on period)

Our contribution

- Applications: most task graphs of streaming applications are series-parallel graphs (SPGs), see for instance the *StreamIt* suite from MIT
- Platforms: Chip MultiProcessors (CMPs)
  $\rightarrow$ $p \times q$ homogeneous cores arranged along a 2D grid
- Trend: increase the number of cores on single chips
- Increasing number of cores rather than processor's complexity: slower growth in power consumption
- This work: energy-aware mappings of SPG streaming applications onto CMPs

Our contribution

- Applications: most task graphs of streaming applications are series-parallel graphs (SPGs), see for instance the *StreamIt* suite from MIT
- Platforms: Chip MultiProcessors (CMPs)
  $\rightarrow$ $p \times q$ homogeneous cores arranged along a 2D grid
- Trend: increase the number of cores on single chips
- Increasing number of cores rather than processor's complexity: slower growth in power consumption
- This work: energy-aware mappings of SPG streaming applications onto CMPs

Our contribution

- Applications: most task graphs of streaming applications are series-parallel graphs (SPGs), see for instance the *StreamIt* suite from MIT
- Platforms: Chip MultiProcessors (CMPs)
  $\rightarrow$ $p \times q$ homogeneous cores arranged along a 2D grid
- Trend: increase the number of cores on single chips
- Increasing number of cores rather than processor's complexity: slower growth in power consumption
- This work: energy-aware mappings of SPG streaming applications onto CMPs

Outline of the talk

1. Framework
   - Application model
   - Platform
   - Mapping strategies
   - Objective functions
2. Complexity results
   - Uni-directional uni-line CMP
   - Bi-directional uni-line CMP
   - Bi-directional square CMP
3. Heuristics
4. Simulations

Outline of the talk

Application model

- Series-parallel graph (SPG) streaming application
- Nodes: n application stages
  - w_i: computation requirement of stage S_i
- Edges: precedence constraints
  - $\delta_{i,j}$: volume of communication between S_i and S_j
- G is a SPG if G is a composition of two SPGs
- Elementary SPG:  (two stages $S_1 \rightarrow S_2$)
- Two kind of compositions:

Application model

- Series-parallel graph (SPG) streaming application
- Nodes: n application stages

  w_i: computation requirement of stage S_i
- Edges: precedence constraints

  $\delta_{i,j}$: volume of communication between S_i and S_j

- G is a SPG if G is a composition of two SPGs
- Elementary SPG:  (two stages $S_1 \rightarrow S_2$)

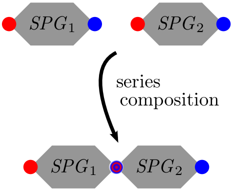
- Two kind of compositions:

Application model

- Series-parallel graph (SPG) streaming application
- Nodes: n application stages
  w_i: computation requirement of stage S_i
- Edges: precedence constraints
  $\delta_{i,j}$: volume of communication between S_i and S_j

- G is a SPG if G is a composition of two SPGs
- Elementary SPG:  (two stages $S_1 \rightarrow S_2$)

- Two kind of compositions:

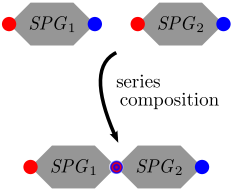


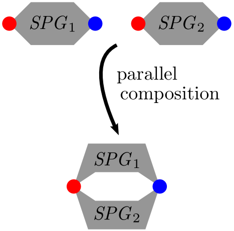
Application model

- Series-parallel graph (SPG) streaming application
- Nodes: n application stages
  w_i: computation requirement of stage S_i
- Edges: precedence constraints
  $\delta_{i,j}$: volume of communication between S_i and S_j

- G is a SPG if G is a composition of two SPGs
- Elementary SPG:  (two stages $S_1 \rightarrow S_2$)

- Two kind of compositions:



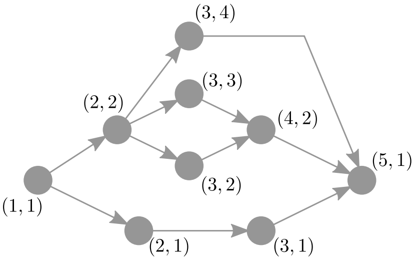


Application model

- Recursive definition of the label of stage S_i, (x_i, y_i): coordinates along a 2D grid in the recursive construction

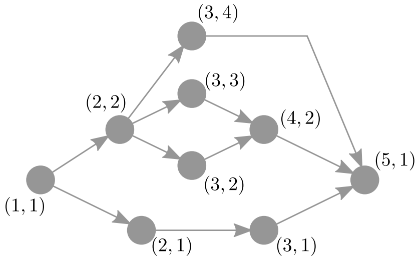
Application model

- Recursive definition of the label of stage S_i, (x_i, y_i): coordinates along a 2D grid in the recursive construction



Application model

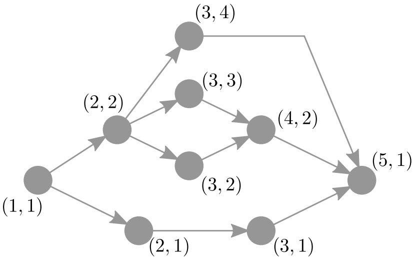
- Recursive definition of the label of stage S_i, (x_i, y_i): coordinates along a 2D grid in the recursive construction



- Source node: label $(1, 1)$; Sink node: label $(x_n, 1)$
- $x_n = \max_{1 \leq i \leq n} x_i$, $y_{\max} = \max_{1 \leq i \leq n} y_i$

Application model

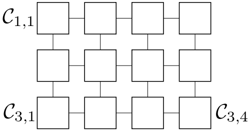
- Recursive definition of the label of stage S_i, (x_i, y_i): coordinates along a 2D grid in the recursive construction



- Source node: label $(1, 1)$; Sink node: label $(x_n, 1)$
- $x_n = \max_{1 \le i \le n} x_i$, $y_{\max} = \max_{1 \le i \le n} y_i$
- $y_{\max}$ is the maximum elevation; special case of bounded-elevation SPGs

Target platform

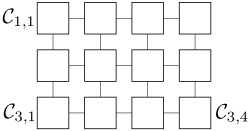
- Chip Multiprocessor: cores $\mathcal{C}_{u,v}$ on a $p \times q$ grid



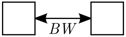
- Bidirectional links of bandwidth BW:
- Time $\frac{\delta}{BW}$ to send δ bytes to a neighboring core
- $\mathcal{C}_{u,v}$ running at speed $s_{u,v} \in \{s^{(1)}, \ldots, s^{(M)}\}$ (M possible voltage/frequency, leading to different speeds, identical on each core)
- Time $\frac{w_i}{s_{u,v}}$ to compute one data set for stage S_i on core $\mathcal{C}_{u,v}$

Target platform

- Chip Multiprocessor: cores $\mathcal{C}_{u,v}$ on a $p \times q$ grid



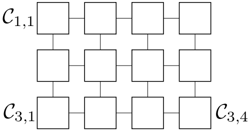
- Bidirectional links of bandwidth BW:



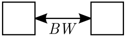
- Time $\frac{\delta}{BW}$ to send δ bytes to a neighboring core
- $\mathcal{C}_{u,v}$ running at speed $s_{u,v} \in \{s^{(1)}, \ldots, s^{(M)}\}$ (M possible voltage/frequency, leading to different speeds, identical on each core)
- Time $\frac{w_i}{s_{u,v}}$ to compute one data set for stage S_i on core $\mathcal{C}_{u,v}$

Target platform

- Chip Multiprocessor: cores $\mathcal{C}_{u,v}$ on a $p \times q$ grid



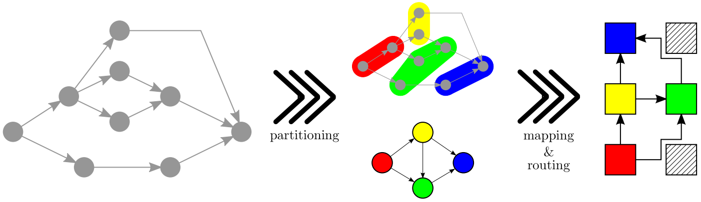
- Bidirectional links of bandwidth BW:



- Time $\frac{\delta}{BW}$ to send δ bytes to a neighboring core
- $\mathcal{C}_{u,v}$ running at speed $s_{u,v} \in \{s^{(1)}, \dots, s^{(M)}\}$ (M possible voltage/frequency, leading to different speeds, identical on each core)
- Time $\frac{w_i}{s_{u,v}}$ to compute one data set for stage S_i on core $\mathcal{C}_{u,v}$

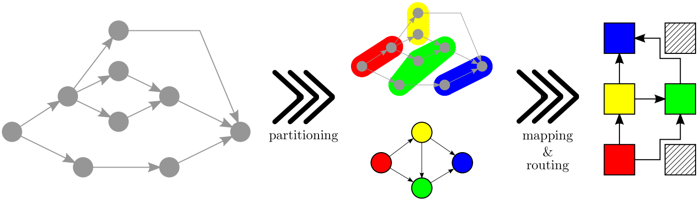
Mapping strategies

- Trade-off between one-to-one and general mappings
  - One-to-one mappings: each stage is mapped on a distinct core; unduly restrictive, high communication costs
  - General mappings: no restriction; arbitrary number of communications between two cores, and NP-complete
  - DAG-partition mappings: first partition the SPG into acyclic clusters, and then perform one-to-one mapping
- Allocation function: $alloc(i) = (u, v)$ if S_i is mapped on $\mathcal{C}_{u,v}$
  Routes to communicate between two cores: $path_{i,j}$



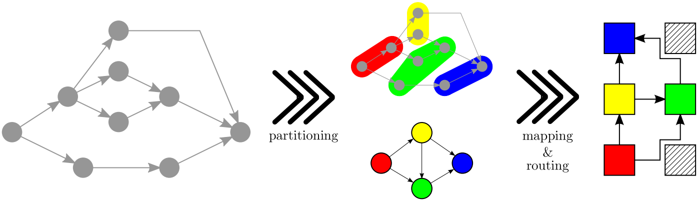
Mapping strategies

- Trade-off between one-to-one and general mappings
  - One-to-one mappings: each stage is mapped on a distinct core; unduly restrictive, high communication costs
  - General mappings: no restriction; arbitrary number of communications between two cores, and NP-complete
  - DAG-partition mappings: first partition the SPG into acyclic clusters, and then perform one-to-one mapping
- Allocation function: $alloc(i) = (u, v)$ if S_i is mapped on $\mathcal{C}_{u,v}$
  Routes to communicate between two cores: $path_{i,j}$



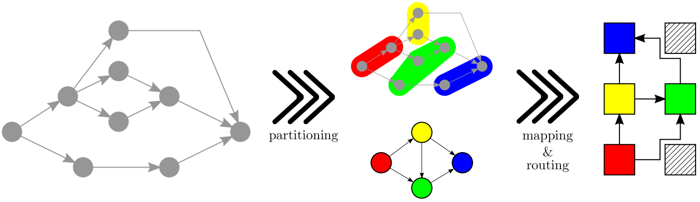
Mapping strategies

- Trade-off between one-to-one and general mappings
  - One-to-one mappings: each stage is mapped on a distinct core; unduly restrictive, high communication costs
  - General mappings: no restriction; arbitrary number of communications between two cores, and NP-complete
  - DAG-partition mappings: first partition the SPG into acyclic clusters, and then perform one-to-one mapping
- Allocation function: $alloc(i) = (u, v)$ if S_i is mapped on $\mathcal{C}_{u,v}$
  Routes to communicate between two cores: $path_{i,j}$



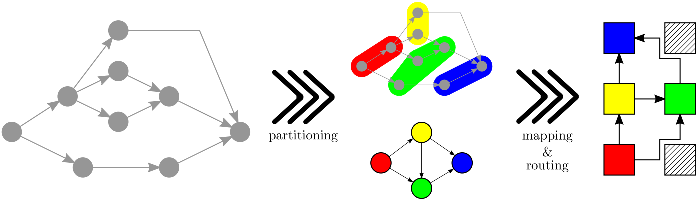
Mapping strategies

- Trade-off between one-to-one and general mappings
  - One-to-one mappings: each stage is mapped on a distinct core; unduly restrictive, high communication costs
  - General mappings: no restriction; arbitrary number of communications between two cores, and NP-complete
  - DAG-partition mappings: first partition the SPG into acyclic clusters, and then perform one-to-one mapping
- Allocation function: $alloc(i) = (u, v)$ if S_i is mapped on $\mathcal{C}_{u,v}$
  Routes to communicate between two cores: $path_{i,j}$



Mapping strategies

- Trade-off between one-to-one and general mappings
  - One-to-one mappings: each stage is mapped on a distinct core; unduly restrictive, high communication costs
  - General mappings: no restriction; arbitrary number of communications between two cores, and NP-complete
  - DAG-partition mappings: first partition the SPG into acyclic clusters, and then perform one-to-one mapping
- Allocation function: $alloc(i) = (u, v)$ if S_i is mapped on $\mathcal{C}_{u,v}$
  Routes to communicate between two cores: $path_{i,j}$



Objective functions: period of the application

- Data sets arrive at regular time intervals: period T
- Given a mapping and an execution speed for each core, check whether the period can be respected, i.e., the cycle-time of each core does not exceed T
- Computations: $w_{u,v} = \sum_{1 \leq i \leq n | alloc(i)=(u,v)} w_i$
  (work assigned to $\mathcal{C}_{u,v}$, running at speed $s_{u,v}$)
  $\rightarrow$ check that $\frac{w_{u,v}}{s_{u,v}} \leq T$
- Communications: $((u' = u + 1 \text{ and } v' = v) \text{ or } (u' = u \text{ and } v' = v + 1))$
  $b_{(u,v) \leftrightarrow (u',v')} = \sum_{1 \leq i,j \leq n | (u,v) \leftrightarrow (u',v') \in path_{i,j}} \delta_{i,j}$
  (communication on link $(u, v) \leftrightarrow (u', v')$)
  $\rightarrow$ check that $\frac{b_{(u,v) \leftrightarrow (u',v')}}{BW} \leq T$

Objective functions: period of the application

- Data sets arrive at regular time intervals: period T
- Given a mapping and an execution speed for each core, check whether the period can be respected, i.e., the cycle-time of each core does not exceed T
- Computations: $w_{u,v} = \sum_{1 \leq i \leq n | alloc(i)=(u,v)} w_i$
  (work assigned to $\mathcal{C}_{u,v}$, running at speed $s_{u,v}$)
  $\rightarrow$ check that $\frac{w_{u,v}}{s_{u,v}} \leq T$
- Communications: $((u' = u+1 \text{ and } v' = v) \text{ or } (u' = u \text{ and } v' = v+1))$
  $b_{(u,v)\leftrightarrow(u',v')} = \sum_{1 \leq i,j \leq n | (u,v)\leftrightarrow(u',v') \in path_{i,j}} \delta_{i,j}$
  (communication on link $(u,v) \leftrightarrow (u',v')$)
  $\rightarrow$ check that $\frac{b_{(u,v)\leftrightarrow(u',v')}}{BW} \leq T$

Objective functions: period of the application

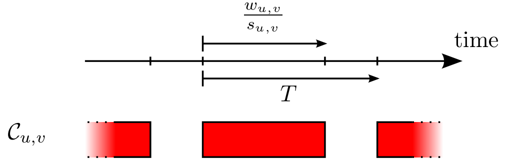
- Data sets arrive at regular time intervals: period T
- Given a mapping and an execution speed for each core, check whether the period can be respected, i.e., the cycle-time of each core does not exceed T
- Computations: $w_{u,v} = \sum_{1 \leq i \leq n | alloc(i)=(u,v)} w_i$
  (work assigned to $\mathcal{C}_{u,v}$, running at speed $s_{u,v}$)
  $\rightarrow$ check that $\frac{w_{u,v}}{s_{u,v}} \leq T$
- Communications: $((u' = u+1 \text{ and } v' = v) \text{ or } (u' = u \text{ and } v' = v+1))$
  $b_{(u,v) \leftrightarrow (u',v')} = \sum_{1 \leq i,j \leq n | (u,v) \leftrightarrow (u',v') \in path_{i,j}} \delta_{i,j}$
  (communication on link $(u,v) \leftrightarrow (u',v')$)
  $\rightarrow$ check that $\frac{b_{(u,v) \leftrightarrow (u',v')}}{BW} \leq T$

Objective functions: period of the application

- Data sets arrive at regular time intervals: period T
- Given a mapping and an execution speed for each core, check whether the period can be respected, i.e., the cycle-time of each core does not exceed T
- Computations: $w_{u,v} = \sum_{1 \leq i \leq n | alloc(i)=(u,v)} w_i$
  (work assigned to $\mathcal{C}_{u,v}$, running at speed $s_{u,v}$)
  $\rightarrow$ check that $\frac{w_{u,v}}{s_{u,v}} \leq T$
- Communications: $((u' = u+1 \text{ and } v' = v) \text{ or } (u' = u \text{ and } v' = v+1))$
  $b_{(u,v)\leftrightarrow(u',v')} = \sum_{1 \leq i,j \leq n | (u,v)\leftrightarrow(u',v') \in path_{i,j}} \delta_{i,j}$
  (communication on link $(u,v) \leftrightarrow (u',v')$)
  $\rightarrow$ check that $\frac{b_{(u,v)\leftrightarrow(u',v')}}{BW} \leq T$

Objective functions: energy consumption

- Energy consumed by computations



$$E^{(\mathrm{comp})} = |\mathcal{A}| \times P_{\mathrm{leak}}^{(\mathrm{comp})} \times T + \sum_{\mathcal{C}_{u,v} \in \mathcal{A}} \frac{w_{u,v}}{s_{u,v}} \times P_{s_{u,v}}^{(\mathrm{comp})},$$

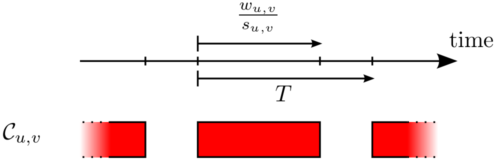
where $\mathcal{A}$ is the set of active cores

- Energy consumed by communications

$$E^{(\mathrm{comm})} = P_{\mathrm{leak}}^{(\mathrm{comm})} \times T + \left(\sum_{u,v} \sum_{u',v'} b_{(u,v)\leftrightarrow(u',v')} \right) \times E^{(\mathrm{bit})}$$

Objective functions: energy consumption

- Energy consumed by computations



$$E^{(\mathrm{comp})} = |\mathcal{A}| \times P_{\mathrm{leak}}^{(\mathrm{comp})} \times T + \sum_{\mathcal{C}_{u,v} \in \mathcal{A}} \frac{w_{u,v}}{s_{u,v}} \times P_{s_{u,v}}^{(\mathrm{comp})},$$

where $\mathcal{A}$ is the set of active cores

- Energy consumed by communications

$$E^{(\mathrm{comm})} = P_{\mathrm{leak}}^{(\mathrm{comm})} \times T + \left(\sum_{u,v} \sum_{u',v'} b_{(u,v)\leftrightarrow(u',v')} \right) \times E^{(\mathrm{bit})}$$

Optimization problem

MinEnergy(T)

- Given
  - a *(bounded-elevation)* SPG
  - a $p \times q$ CMP
  - a period threshold T
- Find a mapping such that
  - the maximal cycle-time does not exceed T
  - the energy $E = E^{(\mathrm{comp})} + E^{(\mathrm{comm})}$ is minimum

Optimization problem

MinEnergy(T)

- Given
  - a *(bounded-elevation)* SPG
  - a $p \times q$ CMP
  - a period threshold T
- Find a mapping such that
  - the maximal cycle-time does not exceed T
  - the energy $E = E^{(\mathrm{comp})} + E^{(\mathrm{comm})}$ is minimum

Optimization problem

MinEnergy(T)

- Given
  - a *(bounded-elevation)* SPG
  - a $p \times q$ CMP
  - a period threshold T
- Find a mapping such that
  - the maximal cycle-time does not exceed T
  - the energy $E = E^{(\mathrm{comp})} + E^{(\mathrm{comm})}$ is minimum

Outline of the talk

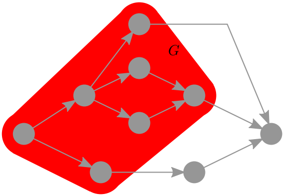
Uni-directional uni-line CMP ($1 \times q$)

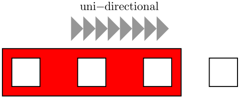
- Polynomial with bounded elevation:

  dynamic programming algorithm

$$\mathcal{E}(G,k) = \min_{G' \subseteq G} \left(\mathcal{E}(G', k-1) \oplus \mathcal{E}^{\mathrm{cal}}(G \setminus G') \right) ,$$

where
- G' is admissible: no more than $n^{y_{\max}}$ such graphs
- outgoing communications of G' do not exceed BW
- energy of communications accounted in the $\oplus$





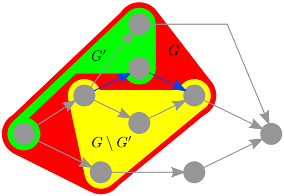
Uni-directional uni-line CMP ($1 \times q$)

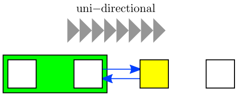
- Polynomial with bounded elevation:

  dynamic programming algorithm

$$\mathcal{E}(G,k) = \min_{G' \subseteq G} \left(\mathcal{E}(G', k-1) \oplus \mathcal{E}^{\text{cal}}(G \setminus G') \right) ,$$

where
- G' is admissible: no more than $n^{y_{\max}}$ such graphs
- outgoing communications of G' do not exceed BW
- energy of communications accounted in the $\oplus$





G' is not admissible

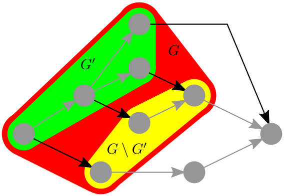
Uni-directional uni-line CMP ($1 \times q$)

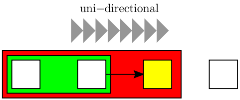
- Polynomial with bounded elevation:

dynamic programming algorithm

$$\mathcal{E}(G,k) = \min_{G' \subseteq G} \left(\mathcal{E}(G', k-1) \oplus \mathcal{E}^{\mathrm{cal}}(G \setminus G') \right) ,$$

where
- G' is admissible: no more than $n^{y_{\max}}$ such graphs
- outgoing communications of G' do not exceed BW
- energy of communications accounted in the $\oplus$





G' is admissible

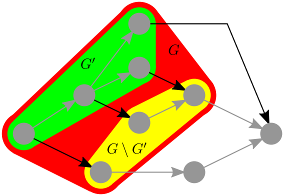
Uni-directional uni-line CMP ($1 \times q$)

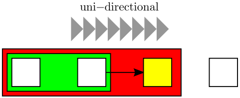
- Polynomial with bounded elevation:

  dynamic programming algorithm

$$\mathcal{E}(G,k) = \min_{G' \subseteq G} \left(\mathcal{E}(G', k-1) \oplus \mathcal{E}^{\mathrm{cal}}(G \setminus G') \right) ,$$

where
- G' is admissible: no more than $n^{y_{\max}}$ such graphs
- outgoing communications of G' do not exceed BW
- energy of communications accounted in the $\oplus$

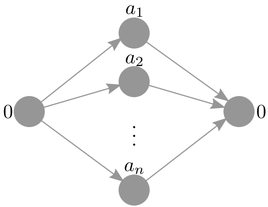


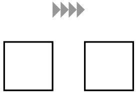


Polynomial: $O(q \times n^{2y_{\max}+1})$

Uni-directional uni-line CMP $(1 \times q)$

- NP-complete with unbounded elevation:
  reduction from 2-PARTITION



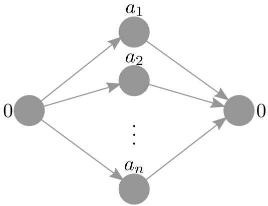


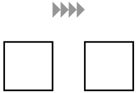
Single speed $\frac{\sum_{i=1}^{n} a_i}{2}$

- Previous algorithm: exponential complexity

Uni-directional uni-line CMP ($1 \times q$)

- NP-complete with unbounded elevation:
  reduction from 2-PARTITION



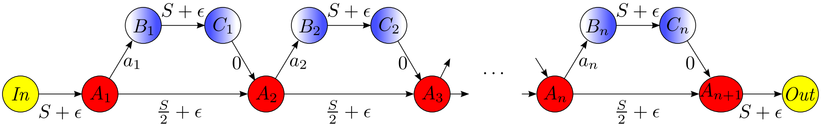


Single speed $\frac{\sum_{i=1}^{n} a_i}{2}$

- Previous algorithm: exponential complexity

Bi-directional uni-line CMP ($1 \times q$)

- NP-complete with bounded elevation:
  reduction from 2-PARTITION
- We enforce $In, A_1, \dots, A_{n+1}, Out$ to be mapped consecutively
- 2-partition of the blue nodes on both sides

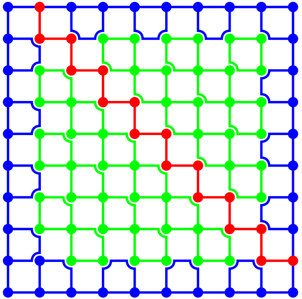


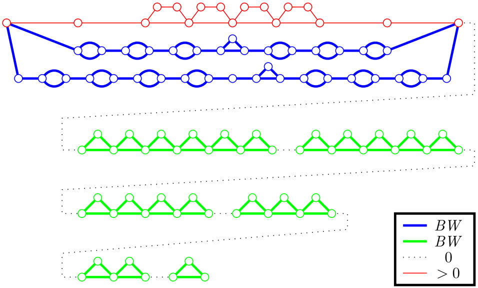
C_{i_1} B_{i_1} C_{i_2} B_{i_2} $\cdots$ C_{i_k} B_{i_k} In A_1 A_2 $\cdots$ A_{n+1} Out $B_{\bar{i}_1}$ $C_{\bar{i}_1}$ $B_{\bar{i}_2}$ $C_{\bar{i}_2}$ $\cdots$ $B_{\bar{i}_{n-k}}$ $C_{\bar{i}_{n-k}}$

$$BW = \frac{3S}{2} + \epsilon$$

Bi-directional square CMP ($p \times p$)

- The previous result implies the NP-completeness for $1 \times q$ CMPs, and hence CMPs of arbitrary shapes ($p \times q$)
- Square: not a direct consequence, but still NP-complete; reuse the uni-line proof by enforcing a line in the square
- Surprisingly involved proof





Outline of the talk

Heuristic summary

- **Random** heuristic: random speeds for each cores; random assignments preserving a DAG-partition and matching period for computations; comm. always following an XY routing
- **Greedy** heuristic: given a speed s, starting from $\mathcal{C}_{1,1}$, process as many stages as possible, partition following stages between right and down cores, iterate on those cores
  Try all possible speed values and keep the best solution
- 2D dynamic programming algorithm, **DPA2D**: map the SPG onto an $x_{max} \times y_{max}$ grid, following labels, and then map the grid onto the CMP thanks to a double nested DP algorithm
- 1D heuristics (2D CMP configured as a snake):
  - **DPA1D**: Optimal solution on uni-directional uni-line CMP
  - **DPA2D1D**: Previous 2D DP heuristic on the snake

Heuristic summary

- **Random** heuristic: random speeds for each cores; random assignments preserving a DAG-partition and matching period for computations; comm. always following an XY routing
- **Greedy** heuristic: given a speed s, starting from $\mathcal{C}_{1,1}$, process as many stages as possible, partition following stages between right and down cores, iterate on those cores
  Try all possible speed values and keep the best solution
- 2D dynamic programming algorithm, **DPA2D**: map the SPG onto an $x_{max} \times y_{max}$ grid, following labels, and then map the grid onto the CMP thanks to a double nested DP algorithm
- 1D heuristics (2D CMP configured as a snake):
  - **DPA1D**: Optimal solution on uni-directional uni-line CMP
  - **DPA2D1D**: Previous 2D DP heuristic on the snake

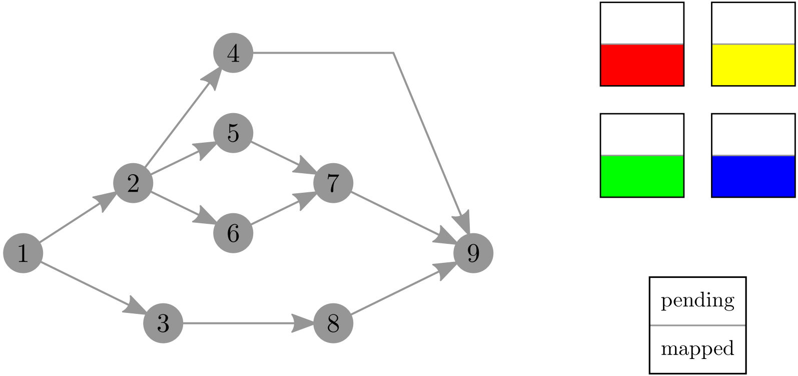
Heuristic summary

- **Random** heuristic: random speeds for each cores; random assignments preserving a DAG-partition and matching period for computations; comm. always following an XY routing
- **Greedy** heuristic: given a speed s, starting from $\mathcal{C}_{1,1}$, process as many stages as possible, partition following stages between right and down cores, iterate on those cores
  Try all possible speed values and keep the best solution
- 2D dynamic programming algorithm, **DPA2D**: map the SPG onto an $x_{max} \times y_{max}$ grid, following labels, and then map the grid onto the CMP thanks to a double nested DP algorithm
- 1D heuristics (2D CMP configured as a snake):
  - **DPA1D**: Optimal solution on uni-directional uni-line CMP
  - **DPA2D1D**: Previous 2D DP heuristic on the snake

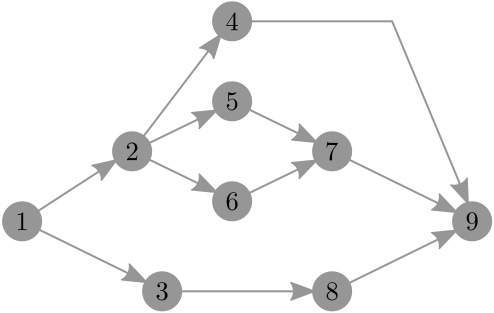
Heuristic summary

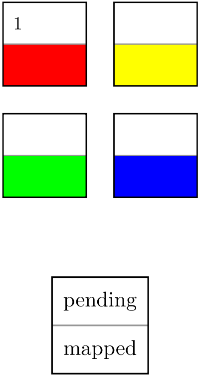
- **Random** heuristic: random speeds for each cores; random assignments preserving a DAG-partition and matching period for computations; comm. always following an XY routing
- **Greedy** heuristic: given a speed s, starting from $\mathcal{C}_{1,1}$, process as many stages as possible, partition following stages between right and down cores, iterate on those cores
  Try all possible speed values and keep the best solution
- 2D dynamic programming algorithm, **DPA2D**: map the SPG onto an $x_{max} \times y_{max}$ grid, following labels, and then map the grid onto the CMP thanks to a double nested DP algorithm
- 1D heuristics (2D CMP configured as a snake):
  - **DPA1D**: Optimal solution on uni-directional uni-line CMP
  - **DPA2D1D**: Previous 2D DP heuristic on the snake

Greedy

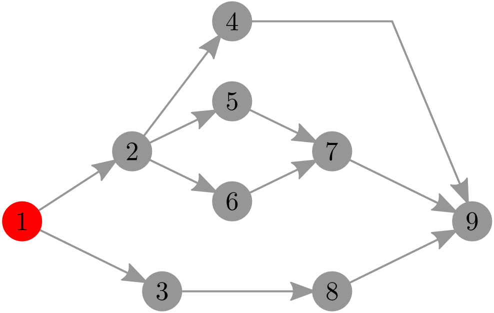


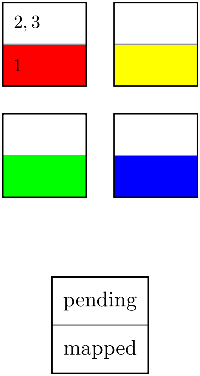
Greedy



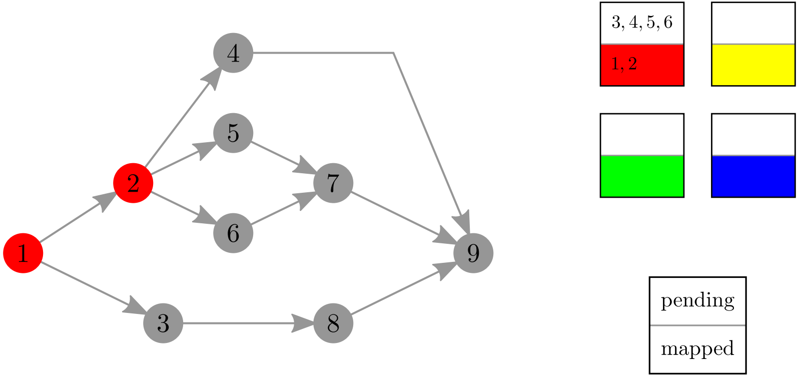


Greedy

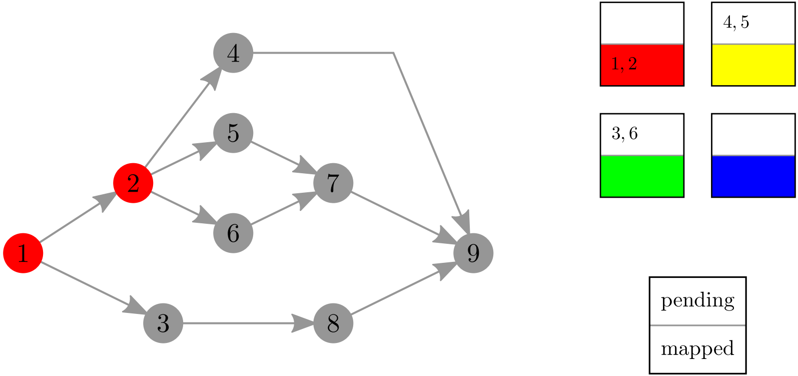




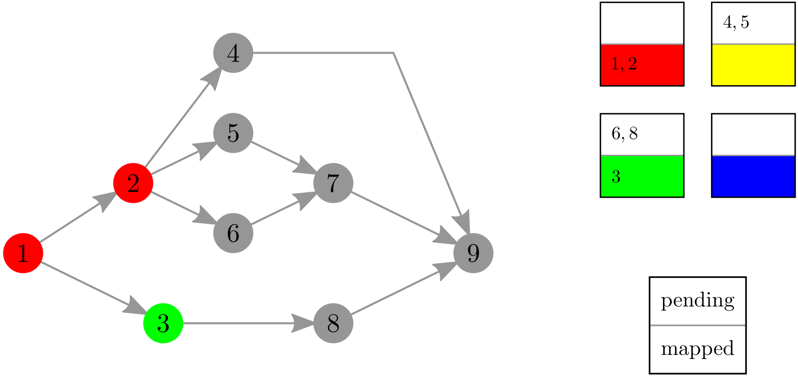
Greedy



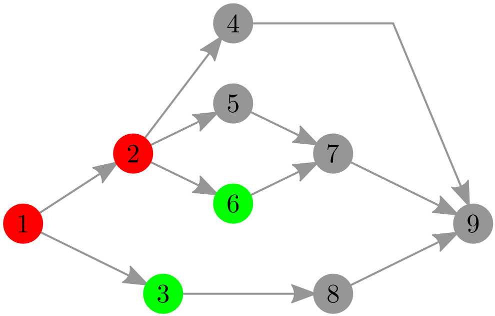
Greedy

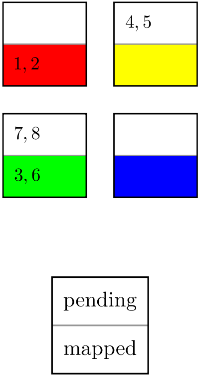


Greedy

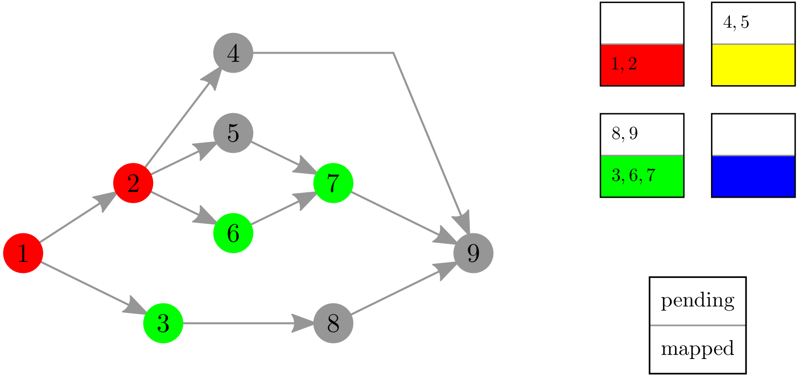


Greedy

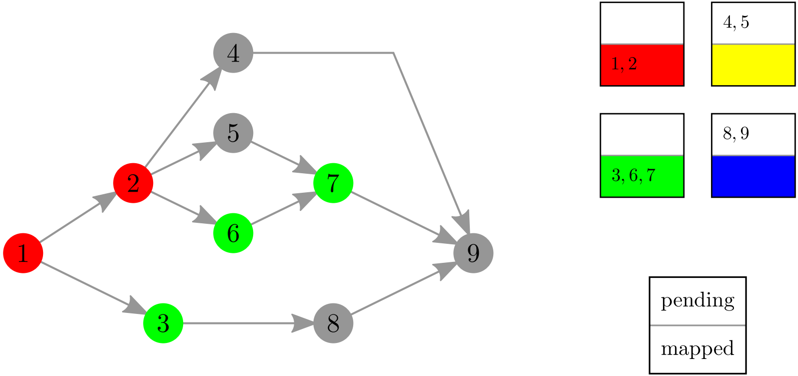




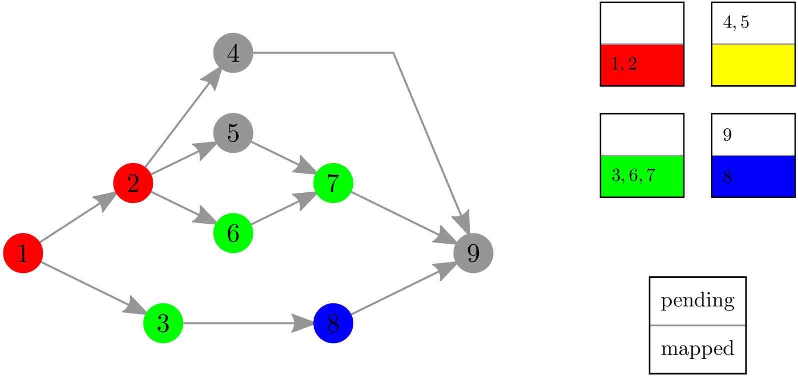
Greedy



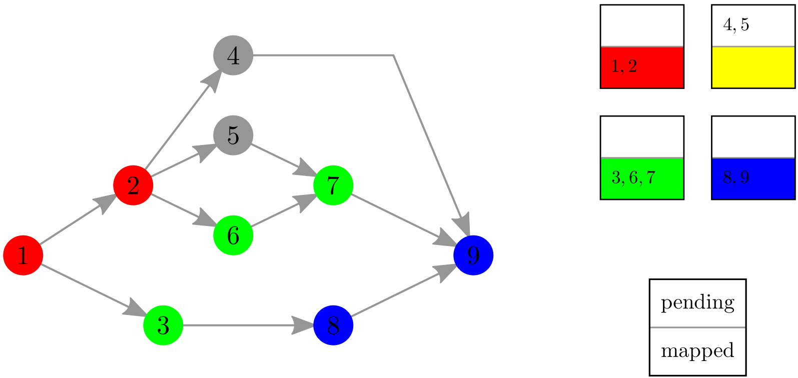
Greedy



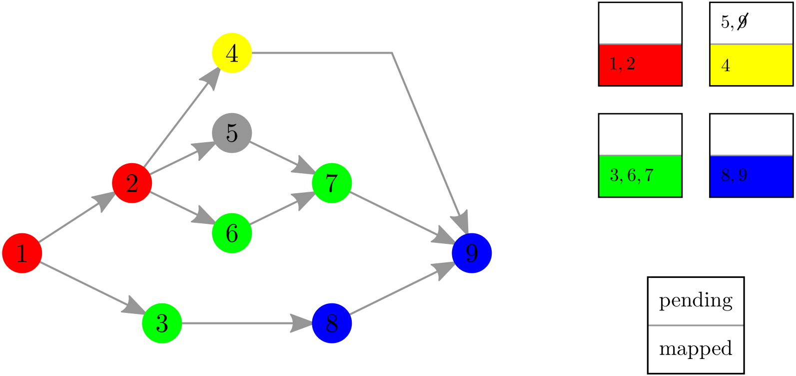
Greedy



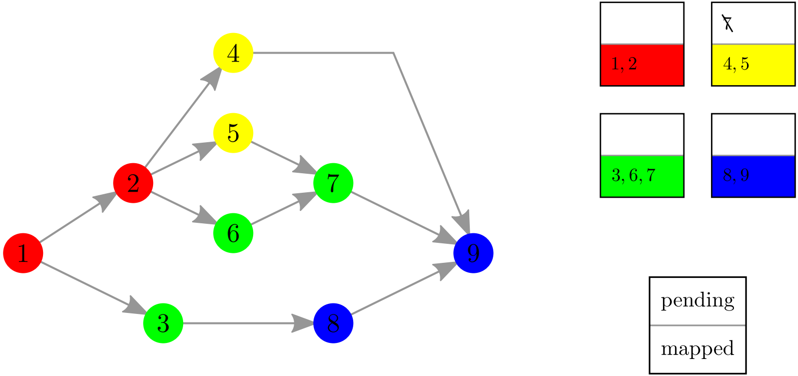
Greedy



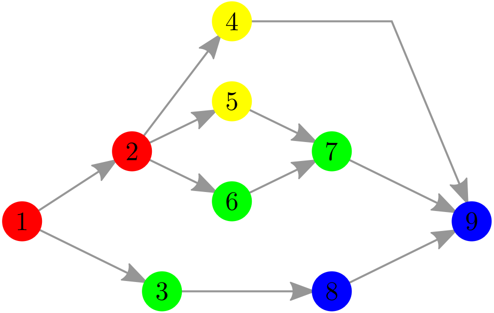
Greedy

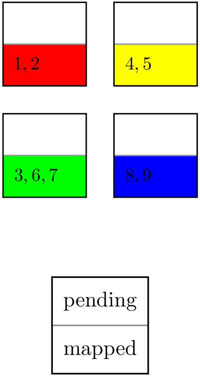


Greedy

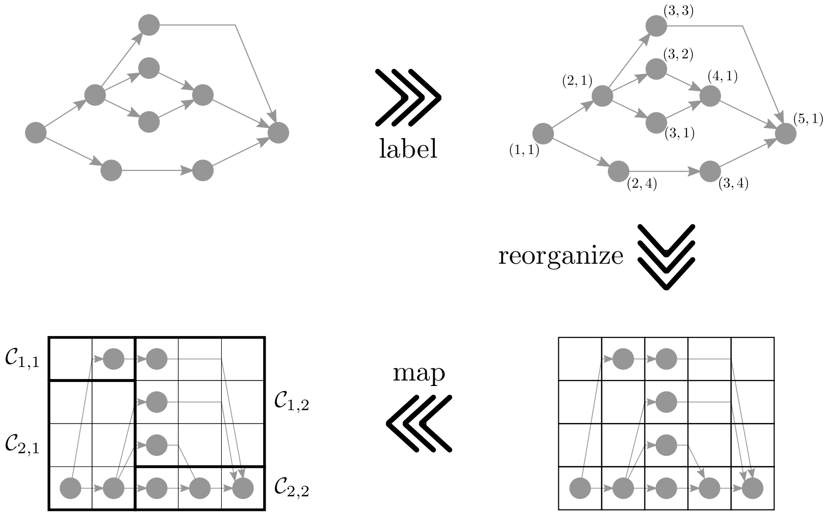


Greedy

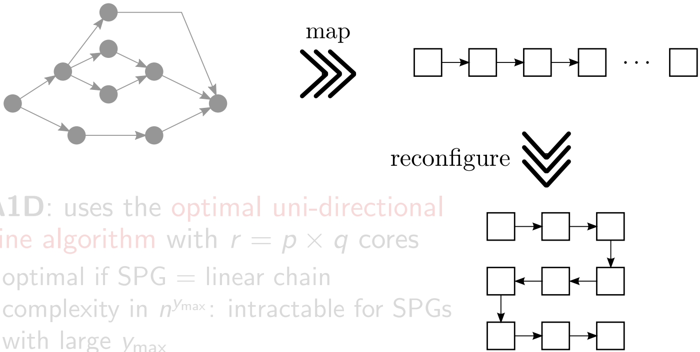




DPA2D

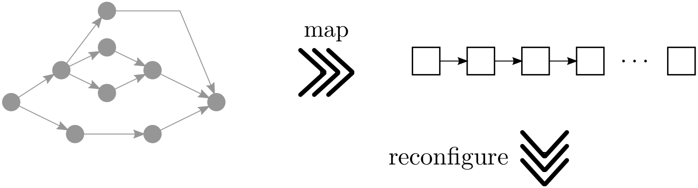

label
(3, 3)
(3, 2)
(2, 1)
(4, 1)
(5, 1)
(3, 1)
(1, 1)
(2, 4)
(3, 4)
reorganize
map
$\mathcal{C}_{1,1}$
$\mathcal{C}_{1,2}$
$\mathcal{C}_{2,1}$
$\mathcal{C}_{2,2}$

DPA1D, DPA2D1D

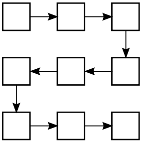


- **DPA1D**: uses the optimal uni-directional uni-line algorithm with $r = p \times q$ cores
  - optimal if SPG = linear chain
  - complexity in $n^{y_{\max}}$: intractable for SPGs with large $y_{\max}$
- **DPA2D1D**: uses **DPA2D** on the $1 \times r$ CMP
  - efficient with little communication
  - more tractable than **DPA1D**

DPA1D, DPA2D1D

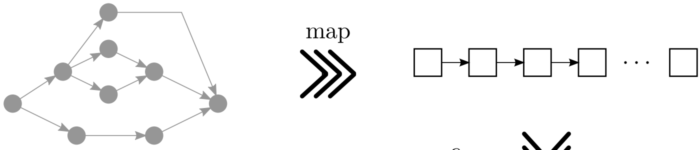


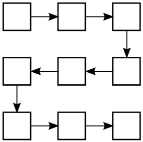
- **DPA1D**: uses the optimal uni-directional uni-line algorithm with $r = p \times q$ cores
  - optimal if SPG = linear chain
  - complexity in $n^{y_{\max}}$: intractable for SPGs with large $y_{\max}$



- **DPA2D1D**: uses **DPA2D** on the $1 \times r$ CMP
  - efficient with little communication
  - more tractable than **DPA1D**

DPA1D, DPA2D1D





- **DPA1D**: uses the optimal uni-directional uni-line algorithm with $r = p \times q$ cores
  - optimal if SPG = linear chain
  - complexity in $n^{y_{\max}}$: intractable for SPGs with large $y_{\max}$
- **DPA2D1D**: uses **DPA2D** on the $1 \times r$ CMP
  - efficient with little communication
  - more tractable than **DPA1D**

Outline of the talk

1. Framework
   - Application model
   - Platform
   - Mapping strategies
   - Objective functions
2. Complexity results
   - Uni-directional uni-line CMP
   - Bi-directional uni-line CMP
   - Bi-directional square CMP
3. Heuristics
4. Simulations

Simulation settings

- Random SPGs
  - Average over 100 applications
  - SPGs with 150 nodes
  - Elevation: from 1 to 30
- Real-life SPGs: the *StreamIt* suite
  - 12 different streaming applications
  - From 8 to 120 nodes
  - Elevation: from 1 to 17
- CMP configuration
  - 4×4 CMP following the Intel Xscale model
  - Five possible speeds per core
- Impact of the computation-to-communication ratio (CCR)

Simulation settings

- Random SPGs
  - Average over 100 applications
  - SPGs with 150 nodes
  - Elevation: from 1 to 30
- Real-life SPGs: the *StreamIt* suite
  - 12 different streaming applications
  - From 8 to 120 nodes
  - Elevation: from 1 to 17
- CMP configuration
  - 4×4 CMP following the Intel Xscale model
  - Five possible speeds per core
- Impact of the computation-to-communication ratio (CCR)

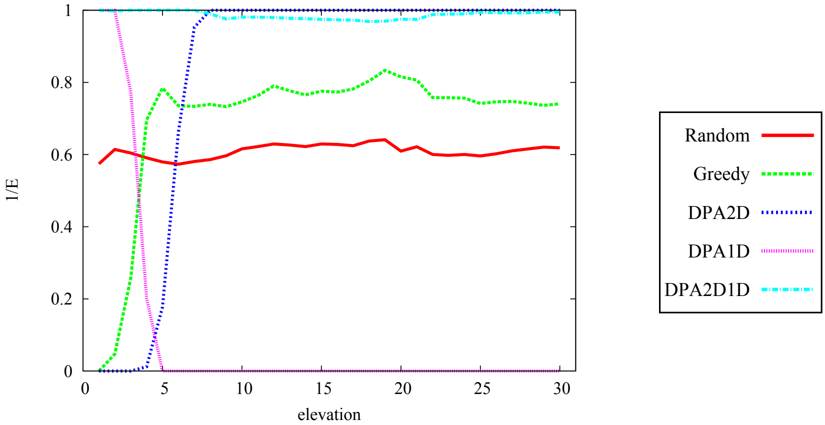
Simulation settings

- Random SPGs
  - Average over 100 applications
  - SPGs with 150 nodes
  - Elevation: from 1 to 30
- Real-life SPGs: the *StreamIt* suite
  - 12 different streaming applications
  - From 8 to 120 nodes
  - Elevation: from 1 to 17
- CMP configuration
  - 4×4 CMP following the Intel Xscale model
  - Five possible speeds per core
- Impact of the computation-to-communication ratio (CCR)

Simulation settings

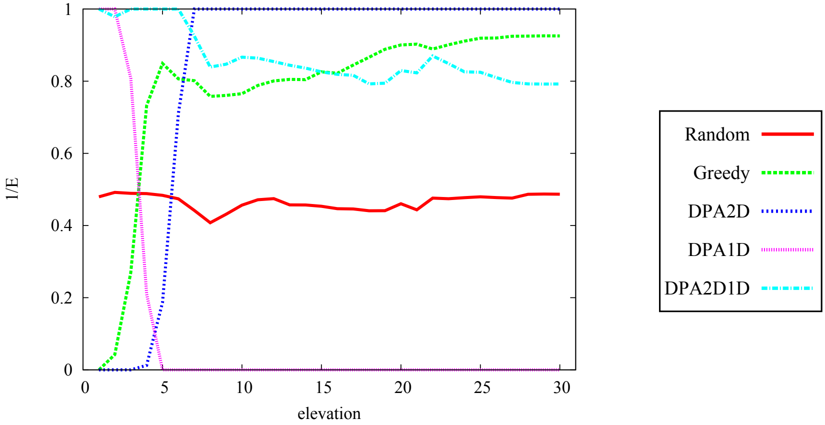
- Random SPGs
  - Average over 100 applications
  - SPGs with 150 nodes
  - Elevation: from 1 to 30
- Real-life SPGs: the *StreamIt* suite
  - 12 different streaming applications
  - From 8 to 120 nodes
  - Elevation: from 1 to 17
- CMP configuration
  - 4×4 CMP following the Intel Xscale model
  - Five possible speeds per core
- Impact of the computation-to-communication ratio (CCR)

Random SPGs; computation intensive (CCR=10)



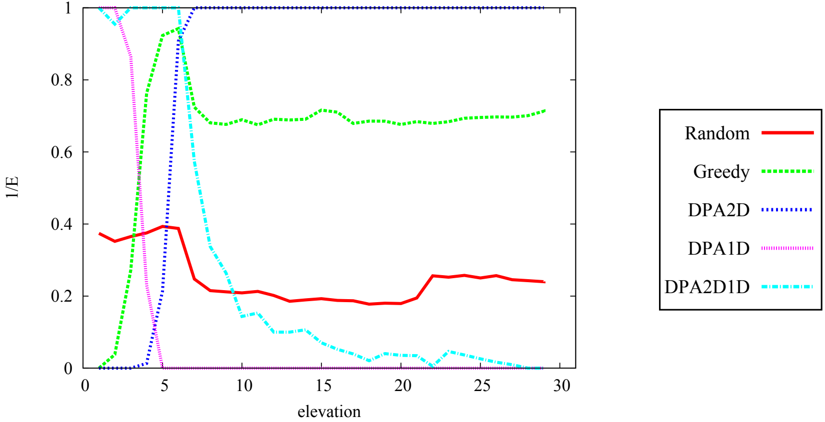
- **DPA1D** best for $1 \leq y_{\max} \leq 3$, then it fails
- **DPA2D** best for $y_{\max} \geq 6$
- **DPA2D1D** always efficient, whatever $y_{\max}$
- **Greedy** intermediate

Random SPGs; balanced (CCR=1)



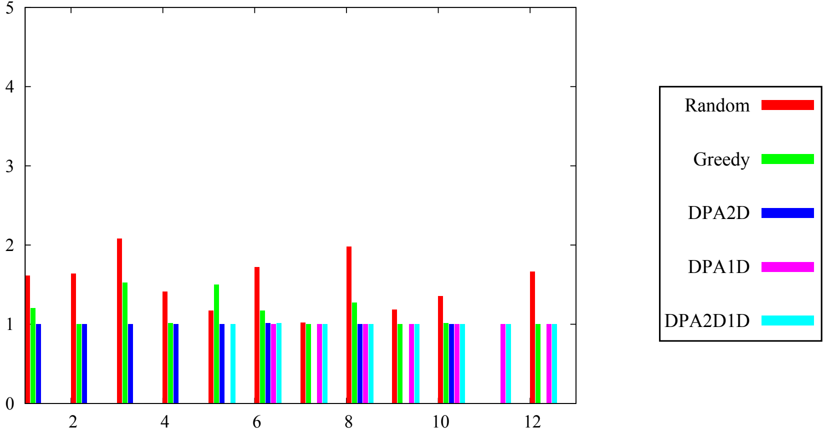
- Almost similar
- **DPA2D1D** is further from the best heuristic: cannot use all communication links

Random SPGs; communication intensive (CCR=0.1)

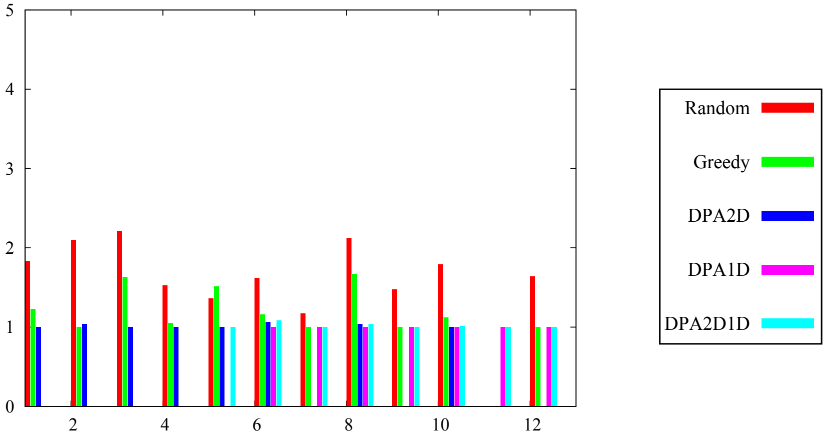


- **Random** and the 1D heuristics do not perform well for large $y_{\max}$
- **DPA2D** remains the best for large $y_{\max}$

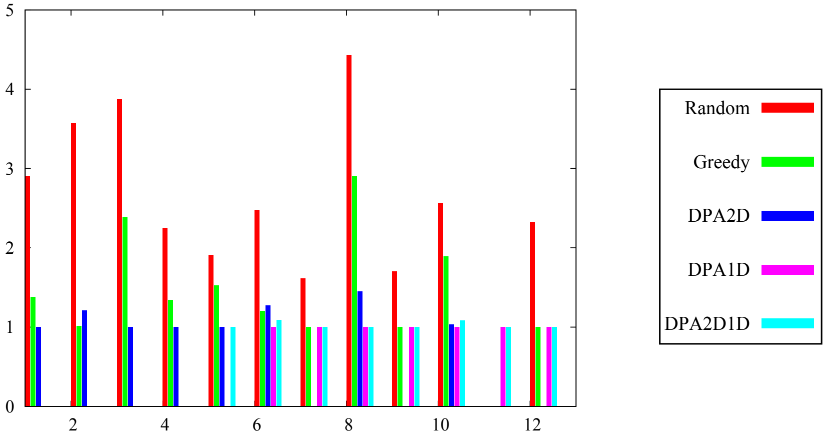
StreamIt; computation intensive (CCR=10)



StreamIt; balanced (CCR=1)



StreamIt; communication intensive (CCR=0.1)



Summary of simulations

- Further simulations on larger applications (up to 200 stages), larger CMPs (6×6), which confirm the results
- Number of failures (out of 1000 instances per CCR value)

CCR	Random	Greedy	DPA2D	DPA1D	DPA2D1D
10	29	28	85	758	1
1	29	28	78	760	3
0.1	300	287	348	670	458

- Execution times: 1ms for **Random** and **Greedy**, 50ms for **DPA2D** and **DPA2D1D**, 10s for **DPA1D**
- **Greedy**: general-purpose heuristic, fast and succeeds on most graphs; **DPA1D**: best for small elevation, optimal with no communication, but very costly; **DPA2D1D**: useful when the elevation gets higher; **DPA2D**: most efficient when communication increases, judiciously handles 2D comms

Summary of simulations

- Further simulations on larger applications (up to 200 stages), larger CMPs (6×6), which confirm the results
- Number of failures (out of 1000 instances per CCR value)

CCR	**Random**	**Greedy**	**DPA2D**	**DPA1D**	**DPA2D1D**
10	29	28	85	758	1
1	29	28	78	760	3
0.1	300	287	348	670	458

- Execution times: 1ms for **Random** and **Greedy**, 50ms for **DPA2D** and **DPA2D1D**, 10s for **DPA1D**
- **Greedy**: general-purpose heuristic, fast and succeeds on most graphs; **DPA1D**: best for small elevation, optimal with no communication, but very costly; **DPA2D1D**: useful when the elevation gets higher; **DPA2D**: most efficient when communication increases, judiciously handles 2D comms

Summary of simulations

- Further simulations on larger applications (up to 200 stages), larger CMPs (6×6), which confirm the results
- Number of failures (out of 1000 instances per CCR value)

CCR	**Random**	**Greedy**	**DPA2D**	**DPA1D**	**DPA2D1D**
10	29	28	85	758	1
1	29	28	78	760	3
0.1	300	287	348	670	458

- Execution times: 1ms for **Random** and **Greedy**, 50ms for **DPA2D** and **DPA2D1D**, 10s for **DPA1D**
- **Greedy**: general-purpose heuristic, fast and succeeds on most graphs; **DPA1D**: best for small elevation, optimal with no communication, but very costly; **DPA2D1D**: useful when the elevation gets higher; **DPA2D**: most efficient when communication increases, judiciously handles 2D comms

Summary of simulations

- Further simulations on larger applications (up to 200 stages), larger CMPs (6×6), which confirm the results
- Number of failures (out of 1000 instances per CCR value)

CCR	**Random**	**Greedy**	**DPA2D**	**DPA1D**	**DPA2D1D**
10	29	28	85	758	1
1	29	28	78	760	3
0.1	300	287	348	670	458

- Execution times: 1ms for **Random** and **Greedy**, 50ms for **DPA2D** and **DPA2D1D**, 10s for **DPA1D**
- **Greedy**: general-purpose heuristic, fast and succeeds on most graphs; **DPA1D**: best for small elevation, optimal with no communication, but very costly; **DPA2D1D**: useful when the elevation gets higher; **DPA2D**: most efficient when communication increases, judiciously handles 2D comms

Conclusion

- Exhaustive complexity study
- Efficient heuristics, from general-purpose to more specialized ones
- Simulations on both randomly generated and real-life SPGs
- Integer linear program (ILP) to solve the problem, intractable for CMPs larger than 2×2 (large number of variables to express communication paths)

Conclusion

- Exhaustive complexity study
- Efficient heuristics, from general-purpose to more specialized ones
- Simulations on both randomly generated and real-life SPGs
- Integer linear program (ILP) to solve the problem, intractable for CMPs larger than 2×2 (large number of variables to express communication paths)

Conclusion

- Exhaustive complexity study
- Efficient heuristics, from general-purpose to more specialized ones
- Simulations on both randomly generated and real-life SPGs
- Integer linear program (ILP) to solve the problem, intractable for CMPs larger than 2×2 (large number of variables to express communication paths)

Conclusion

- Exhaustive complexity study
- Efficient heuristics, from general-purpose to more specialized ones
- Simulations on both randomly generated and real-life SPGs
- Integer linear program (ILP) to solve the problem, intractable for CMPs larger than 2×2 (large number of variables to express communication paths)

Future work

- Investigate general mappings, and assess the difference with DAG-partition mappings (in theory and in practice)
- Simplify the ILP to assess the absolute performance of the heuristics
- Propose a more accurate power consumption model for communications: allow for bandwidth scaling, similarly to the frequency scaling of cores
- Study some multi-path routing policies, and compare with single-path or XY routing
- Mappings that make a trade-off between performance, energy consumption, and also reliability (failures, variations)

Future work

- Investigate general mappings, and assess the difference with DAG-partition mappings (in theory and in practice)
- Simplify the ILP to assess the absolute performance of the heuristics
- Propose a more accurate power consumption model for communications: allow for bandwidth scaling, similarly to the frequency scaling of cores
- Study some multi-path routing policies, and compare with single-path or XY routing
- Mappings that make a trade-off between performance, energy consumption, and also reliability (failures, variations)

Future work

- Investigate general mappings, and assess the difference with DAG-partition mappings (in theory and in practice)
- Simplify the ILP to assess the absolute performance of the heuristics
- Propose a more accurate power consumption model for communications: allow for bandwidth scaling, similarly to the frequency scaling of cores
- Study some multi-path routing policies, and compare with single-path or XY routing
- Mappings that make a trade-off between performance, energy consumption, and also reliability (failures, variations)

Future work

- Investigate general mappings, and assess the difference with DAG-partition mappings (in theory and in practice)
- Simplify the ILP to assess the absolute performance of the heuristics
- Propose a more accurate power consumption model for communications: allow for bandwidth scaling, similarly to the frequency scaling of cores
- Study some multi-path routing policies, and compare with single-path or XY routing
- Mappings that make a trade-off between performance, energy consumption, and also reliability (failures, variations)

Future work

- Investigate general mappings, and assess the difference with DAG-partition mappings (in theory and in practice)
- Simplify the ILP to assess the absolute performance of the heuristics
- Propose a more accurate power consumption model for communications: allow for bandwidth scaling, similarly to the frequency scaling of cores
- Study some multi-path routing policies, and compare with single-path or XY routing
- Mappings that make a trade-off between performance, energy consumption, and also reliability (failures, variations)