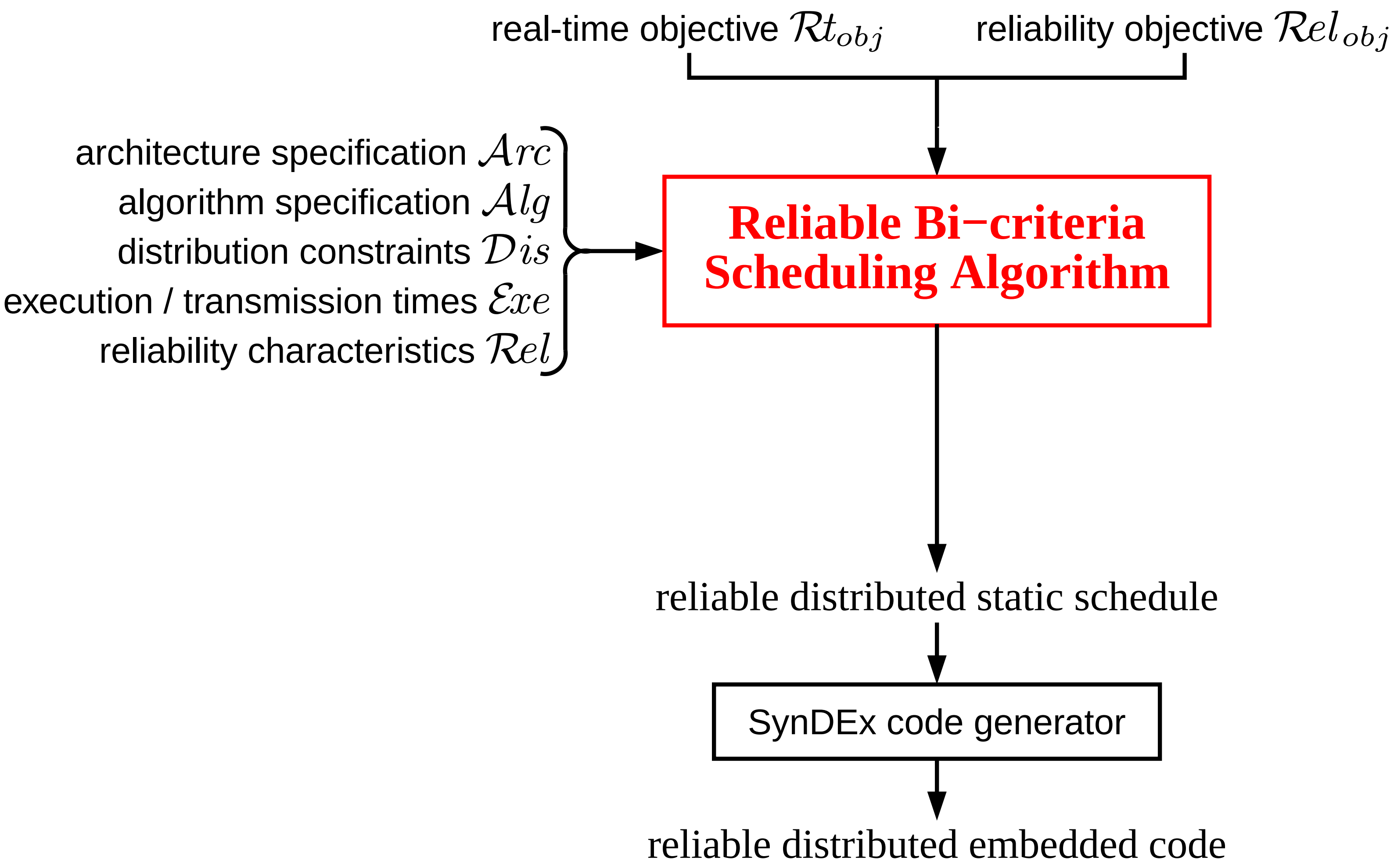
# A bi-criteria scheduling heuristic for distributed embedded systems under reliability and real-time constraints

Ismail ASSAYAD, Alain GIRAULT, Hamoudi Kalla

Verimag/ST Micro, Inria Rhône-Alpes

# The global methodology picture



# Algorithm model

A data-flow graph $\mathcal{A}lg$, where:

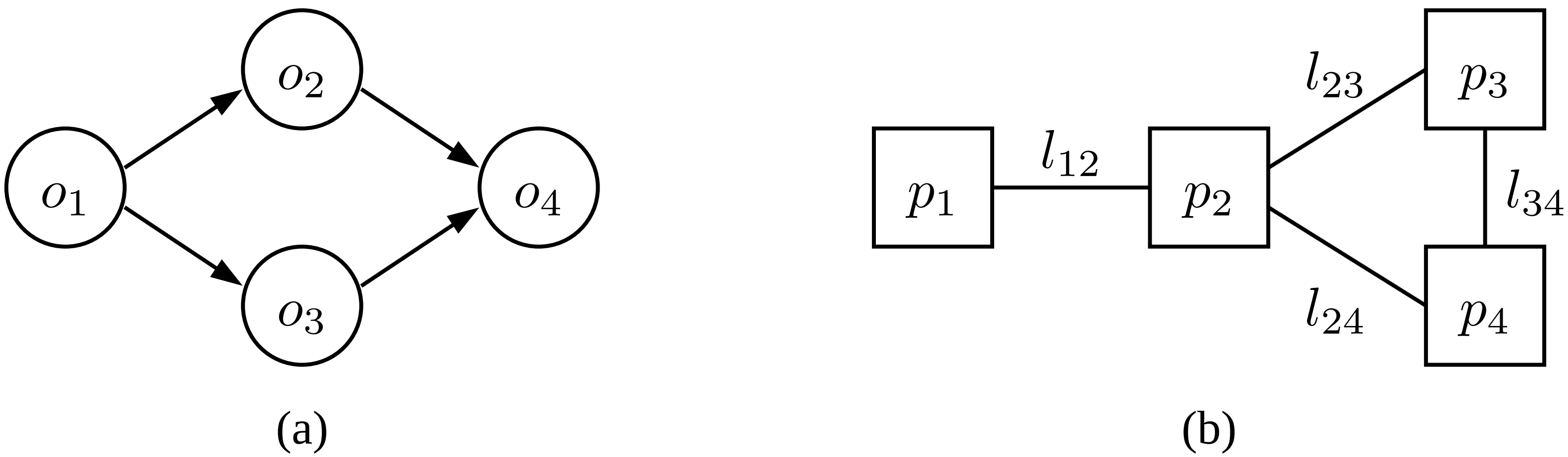
- each vertex is an operation
- each edge is a data-dependency
- a vertex without predescessor is an input operation
- a vertex without successor is an output operation

The graph is executed repeatedly for each input event from the sensors, in order to compute the output events for the actuators

# Architecture model

A graph $\mathcal{A}rc$ of procs linked by point-to-point communication links

Example:



# Worst-case execution times

| | | operation | | | |
|---|---|---|---|---|---|
| | time | $o_1$ | $o_2$ | $o_3$ | $o_4$ |
| processor | $p_1$ | 20 | 3 | 4 | 3 |
| | $p_2$ | 3 | 4 | $\infty$ | 4 |
| | $p_3$ | 5 | 2 | 2 | $\infty$ |
| | $p_4$ | 4 | 4 | 5 | 2 |

| | | data-dependency | | | |
|---|---|---|---|---|---|
| | time | $o_1 \triangleright o_2$ | $o_1 \triangleright o_3$ | $o_2 \triangleright o_4$ | $o_3 \triangleright o_4$ |
| link | $l_{12}$ | 2 | 5 | 3 | 1.5 |
| | $l_{23}$ | 3 | 6 | 5 | 1 |
| | $l_{24}$ | 3 | 6 | 5 | 1 |
| | $l_{34}$ | 2 | 5 | 3 | 1.5 |

# Worst-case execution times

| | | operation | | | |
|---|---|---|---|---|---|
| | time | $o_1$ | $o_2$ | $o_3$ | $o_4$ |
| processor | $p_1$ | 20 | 3 | 4 | 3 |
| | $p_2$ | 3 | 4 | $\infty$ | 4 |
| | $p_3$ | 5 | 2 | 2 | $\infty$ |
| | $p_4$ | 4 | 4 | 5 | 2 |

| | | data-dependency | | | |
|---|---|---|---|---|---|
| | time | $o_1 \triangleright o_2$ | $o_1 \triangleright o_3$ | $o_2 \triangleright o_4$ | $o_3 \triangleright o_4$ |
| link | $l_{12}$ | 2 | 5 | 3 | 1.5 |
| | $l_{23}$ | 3 | 6 | 5 | 1 |
| | $l_{24}$ | 3 | 6 | 5 | 1 |
| | $l_{34}$ | 2 | 5 | 3 | 1.5 |

$\infty$ indicates that the operation cannot be executed on the processor

# Reliability model

We consider only component failures (procs and links)

We assume that component failures are independant

We assume that all the algorithms are correct

Failures have an exponential distribution [Shatz, Wang, & Goto 92]

| | processor | | | |
|---|---|---|---|---|
| | $p_1$ | $p_2$ | $p_3$ | $p_4$ |
| failure rate $\lambda$ | $2 * 10^{-6}$ | $10^{-6}$ | $3 * 10^{-6}$ | $2 * 10^{-6}$ |

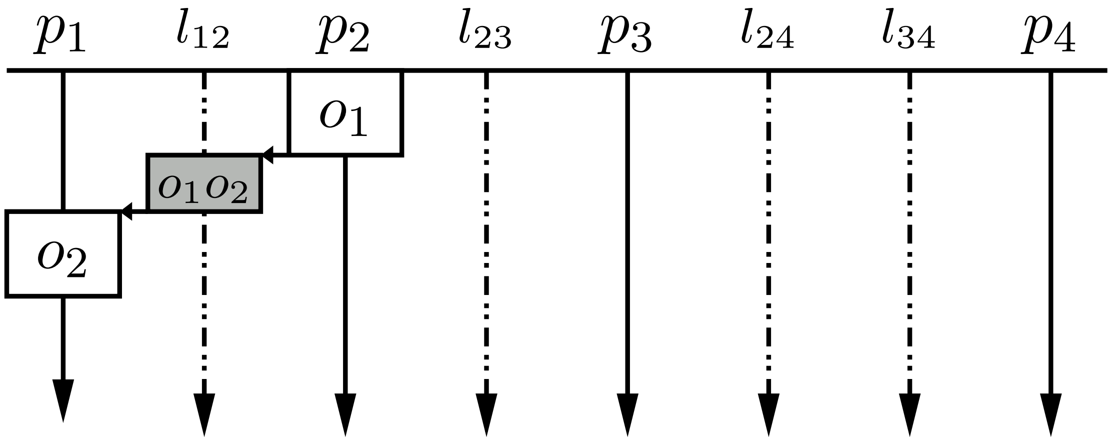
| | communication link | | | |
|---|---|---|---|---|
| | $l_{12}$ | $l_{23}$ | $l_{24}$ | $l_{34}$ |
| failure rate $\lambda$ | $2 * 10^{-5}$ | $4 * 10^{-5}$ | $4 * 10^{-5}$ | $2 * 10^{-5}$ |

# Reliability Block Diagram (RBD)

To each partial schedule of $\mathcal{A}lg$ onto $\mathcal{A}rc$, with or without replication, we associate a RBD [Abd-allah 97] [Figiel & Sule 90]

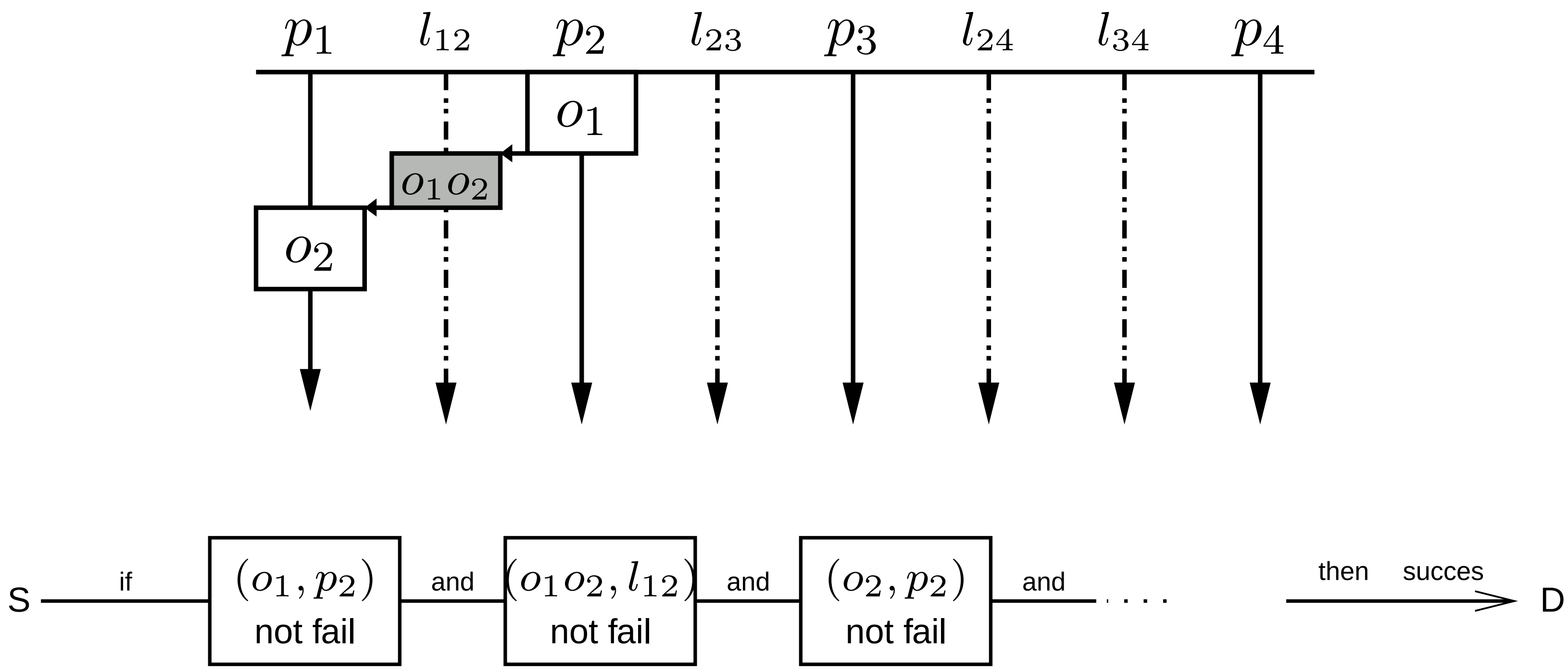
# Reliability Block Diagram (RBD)

To each partial schedule of $\mathcal{A}lg$ onto $\mathcal{A}rc$, with or without replication, we associate a RBD [Abd-allah 97] [Figiel & Sule 90]



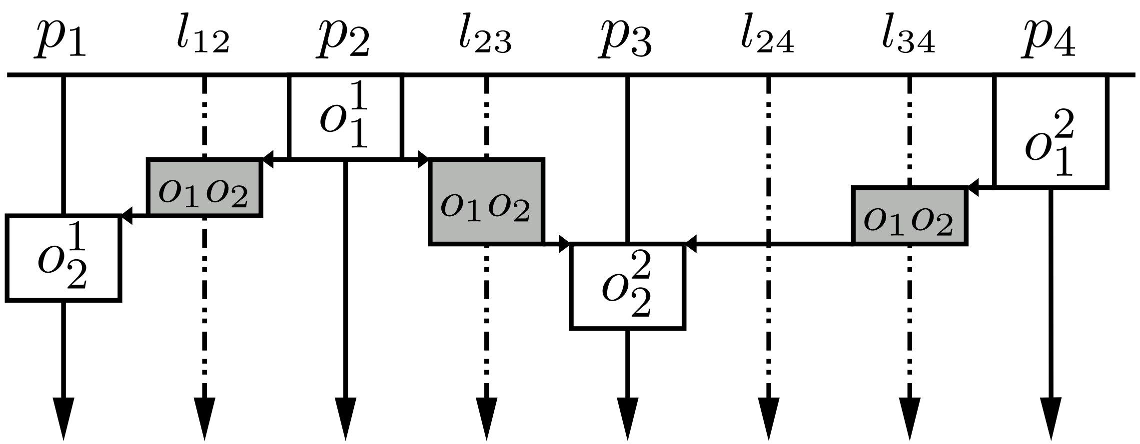
# Reliability Block Diagram (RBD)

To each partial schedule of $\mathcal{A}lg$ onto $\mathcal{A}rc$, with or without replication, we associate a RBD [Abd-allah 97] [Figiel & Sule 90]



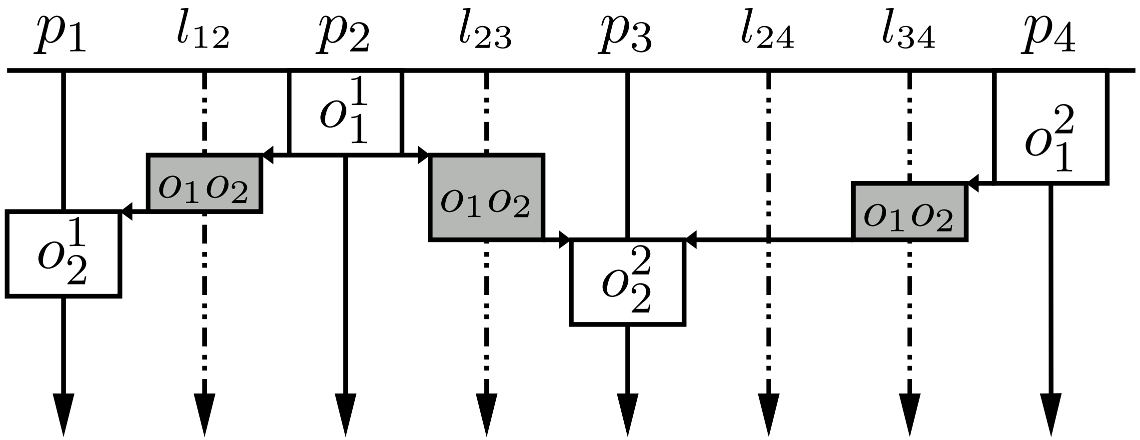
# Reliability Block Diagram (RBD)

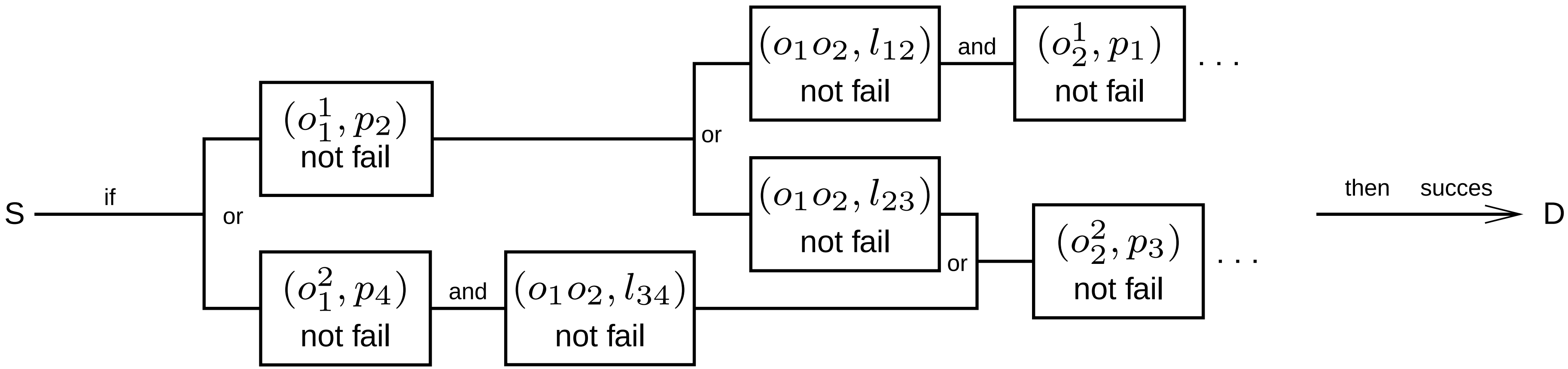
To each partial schedule of $\mathcal{A}lg$ onto $\mathcal{A}rc$, with or without replication, we associate a RBD [Abd-allah 97] [Figiel & Sule 90]



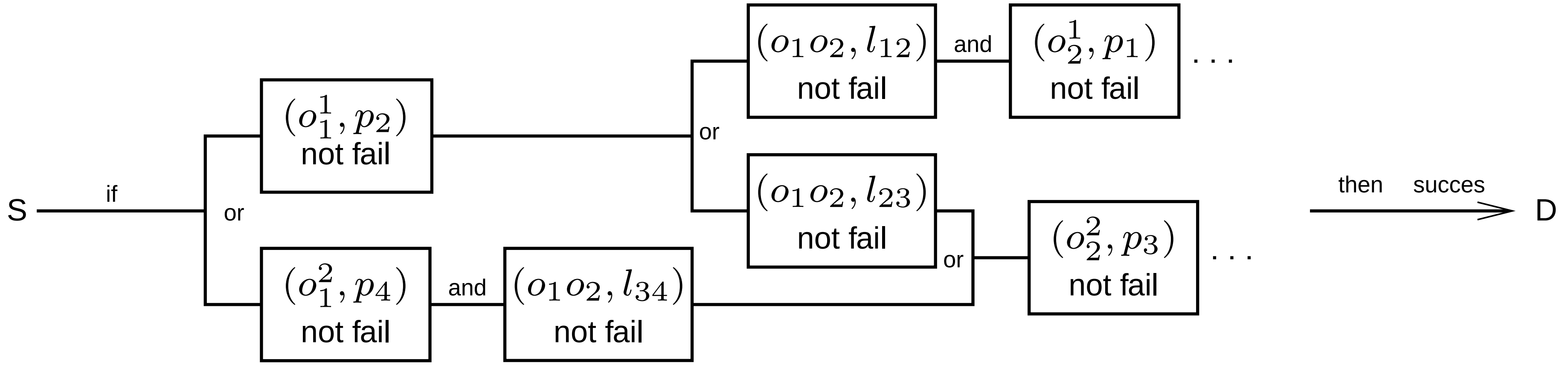
# Reliability Block Diagram (RBD)

To each partial schedule of $\mathcal{A}lg$ onto $\mathcal{A}rc$, with or without replication, we associate a RBD [Abd-allah 97] [Figiel & Sule 90]



# Reliability Block Diagram (RBD)




Then we compute the minimal cut sets of the RDB: the minimum combination of failures that might cause the system to fail

Finally we compute the overall system reliability:

$$\mathcal{R}el^*_{sched} \leq \prod_{i=1}^{k} \Big(1 - \prod_{(o,c)\,\in\,\mathcal{M}cs_i} (1 - \mathcal{R}el_{sched}(o,c))\Big)$$

# The two criteria for scheduling

- $\mathcal{R}t_{obj}$ is the schedule length objective
- $\mathcal{R}el_{obj}$ is the reliability objective

# The two criteria for scheduling

- $\mathcal{R}t_{obj}$ is the schedule length objective
- $\mathcal{R}el_{obj}$ is the reliability objecive

These criteria are antagonistic, because usually:

- introducing replication improves the reliability
- but also increases the schedule length

# The two criteria for scheduling

- $\mathcal{R}t_{obj}$ is the schedule length objective
- $\mathcal{R}el_{obj}$ is the reliability objective

These criteria are antagonistic, because usually:

- introducing replication improves the reliability
- but also increases the schedule length

Here, we also use replication to increase the locality of computations

➯ in some situations, it also reduces the schedule length!

# Reliable Bi-criteria Scheduling Heuristic

Greedy list scheduling [Yang & Gerasoulis 93]

Maintains two lists of operation of $\mathcal{A}lg$:

- $O^{(n)}_{sched}$ = list of scheduled operations; initially empty
- $O^{(n)}_{cand}$ = list of candidate operations; contains initially the operations without predescessors

At the step $(n)$, we compute the compromise function for each pair $\langle$candidate operation,processor set$\rangle$, and we choose the best compromise

If the processor set has more than one element, then the chosen operation is replicated

# Compromise function

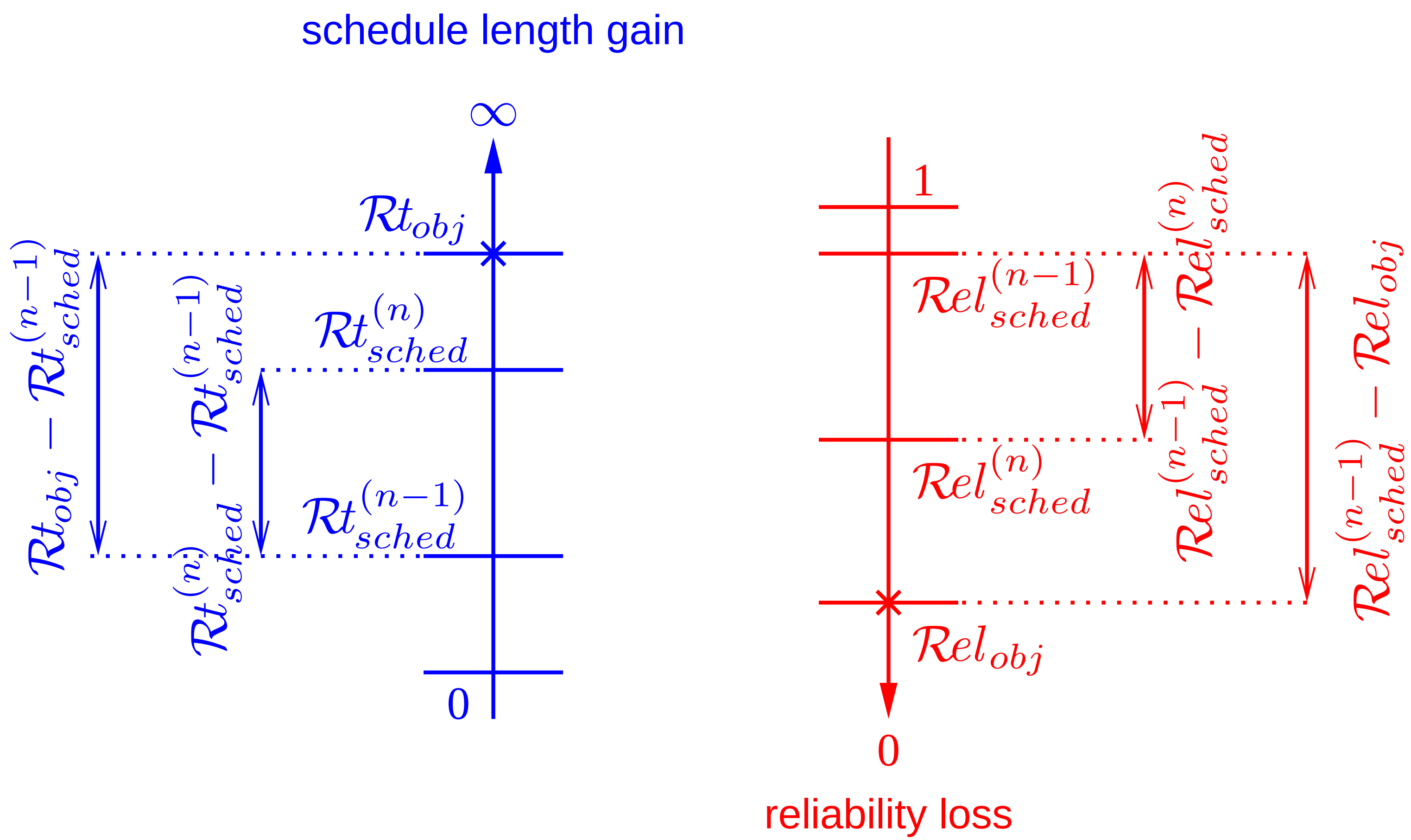
$\mathcal{RBSA}$ uses a compromise function based on the reliability loss and the schedule length gain (both are normalised)

$$\mathcal{L}^{(n)}(o_i, \{p_1, ..., p_j\}) = \frac{\mathcal{R}el^{(n)}_{sched}(o_i, \{p_{,1}, ..., p_j\}) - \mathcal{R}el^{(n-1)}_{sched}}{\mathcal{R}el_{obj} - \mathcal{R}el^{(n-1)}_{sched}}$$

$$\mathcal{S}^{(n)}(o_i, \{p_1, ..., p_j\}) = \frac{\mathcal{R}t^{(n)}_{sched}(o_i, \{p_1, ..., p_j\}) - \mathcal{R}t^{(n-1)}_{sched}}{\mathcal{R}t_{obj} - \mathcal{R}t^{(n-1)}_{sched}}$$

# Compromise function

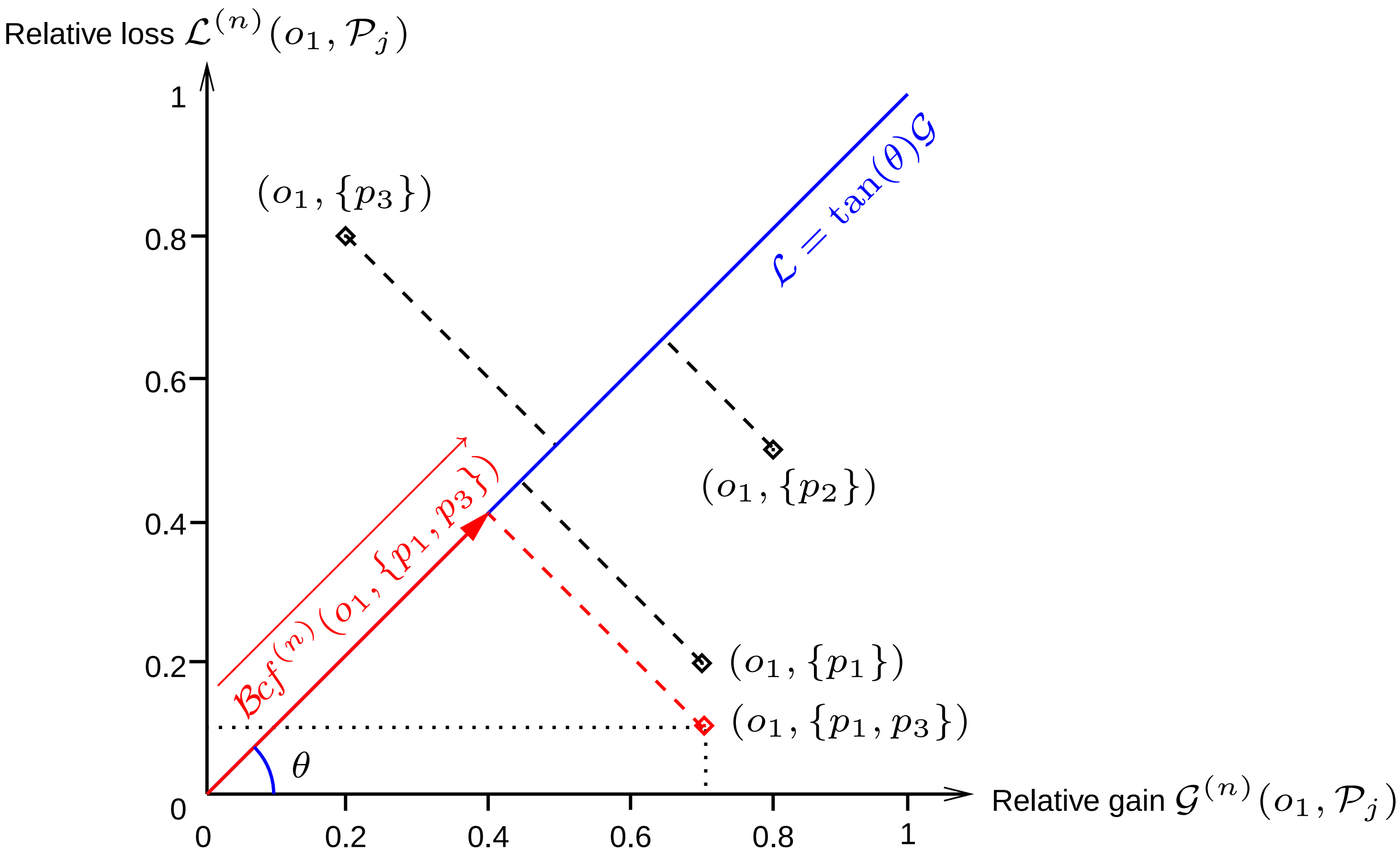
$\mathcal{RBSA}$ uses a compromise function based on the reliability loss and the schedule length gain (both are normalised)



# Selection of the best candidate operation

To each candidate operation corresponds a point in the $\mathcal{L}$-$\mathcal{G}$ plane

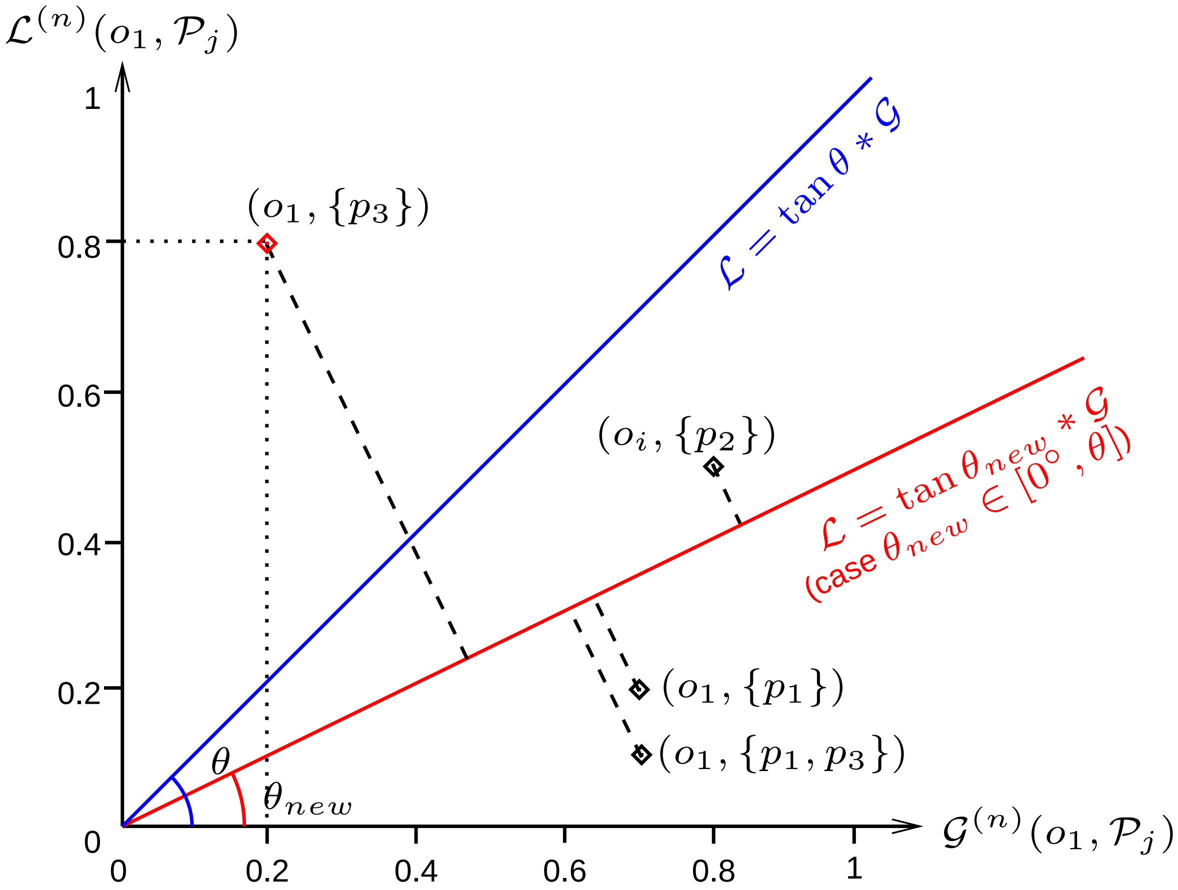
We project all the points onto a line, and select the point which has the best projection ($\theta$ is a parameter of the heuristics)



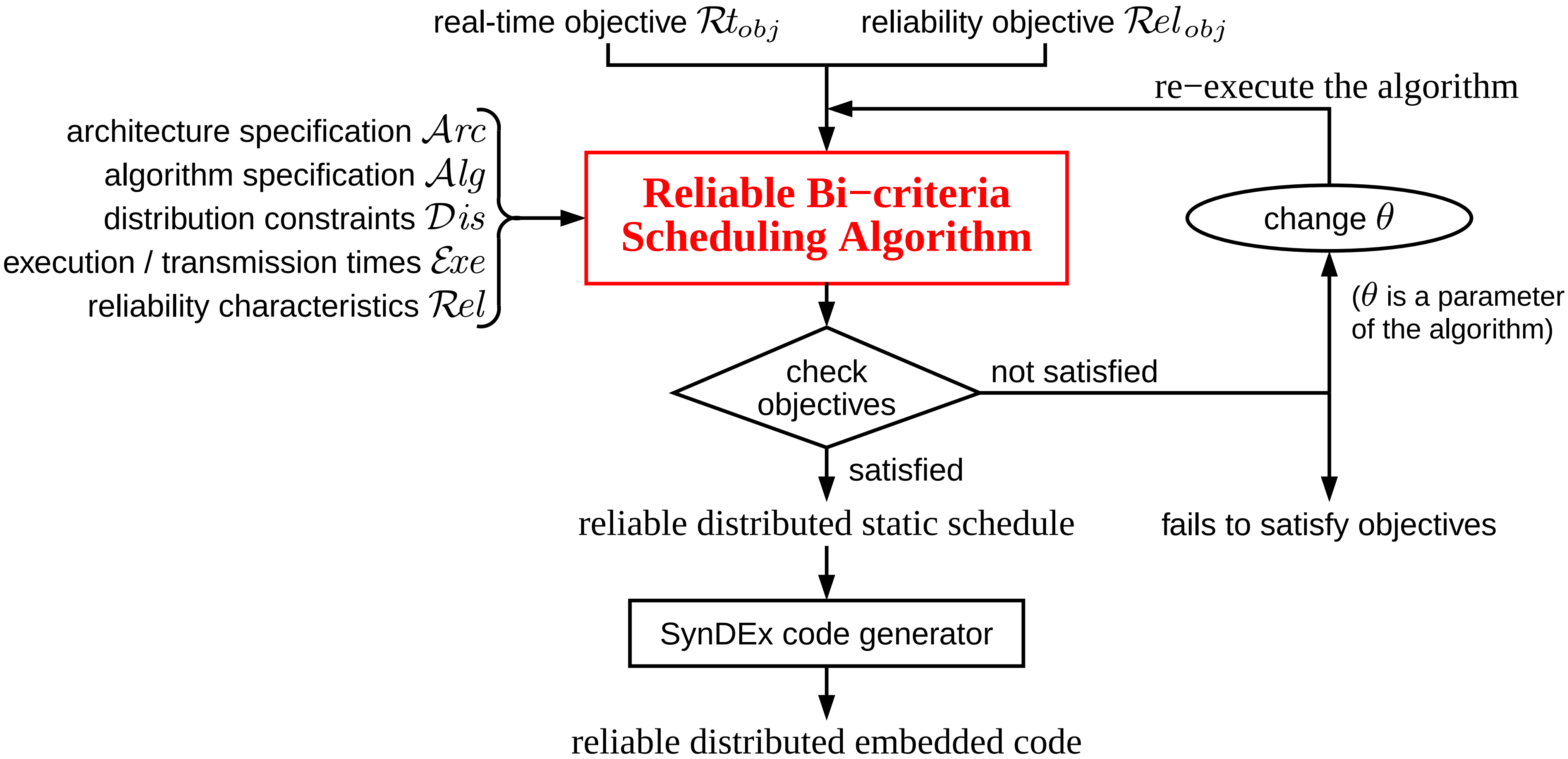
# What if $\mathcal{RBSA}$ fails?

We modify $\theta$ and iterate

The new $\theta$ depends on which objective was not reached



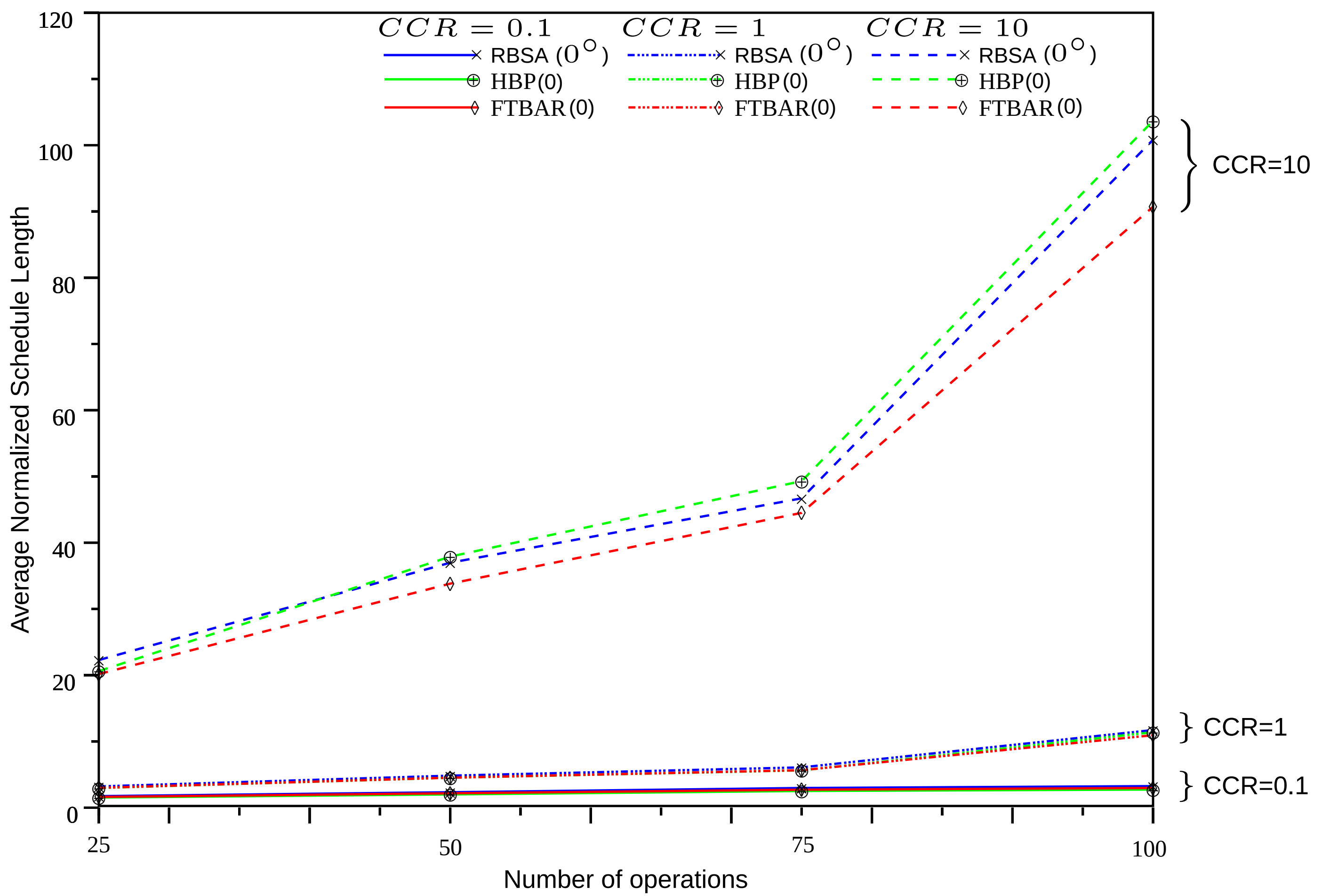
# The final methodology picture



# $\mathcal{RBSA}$ simulations (I)

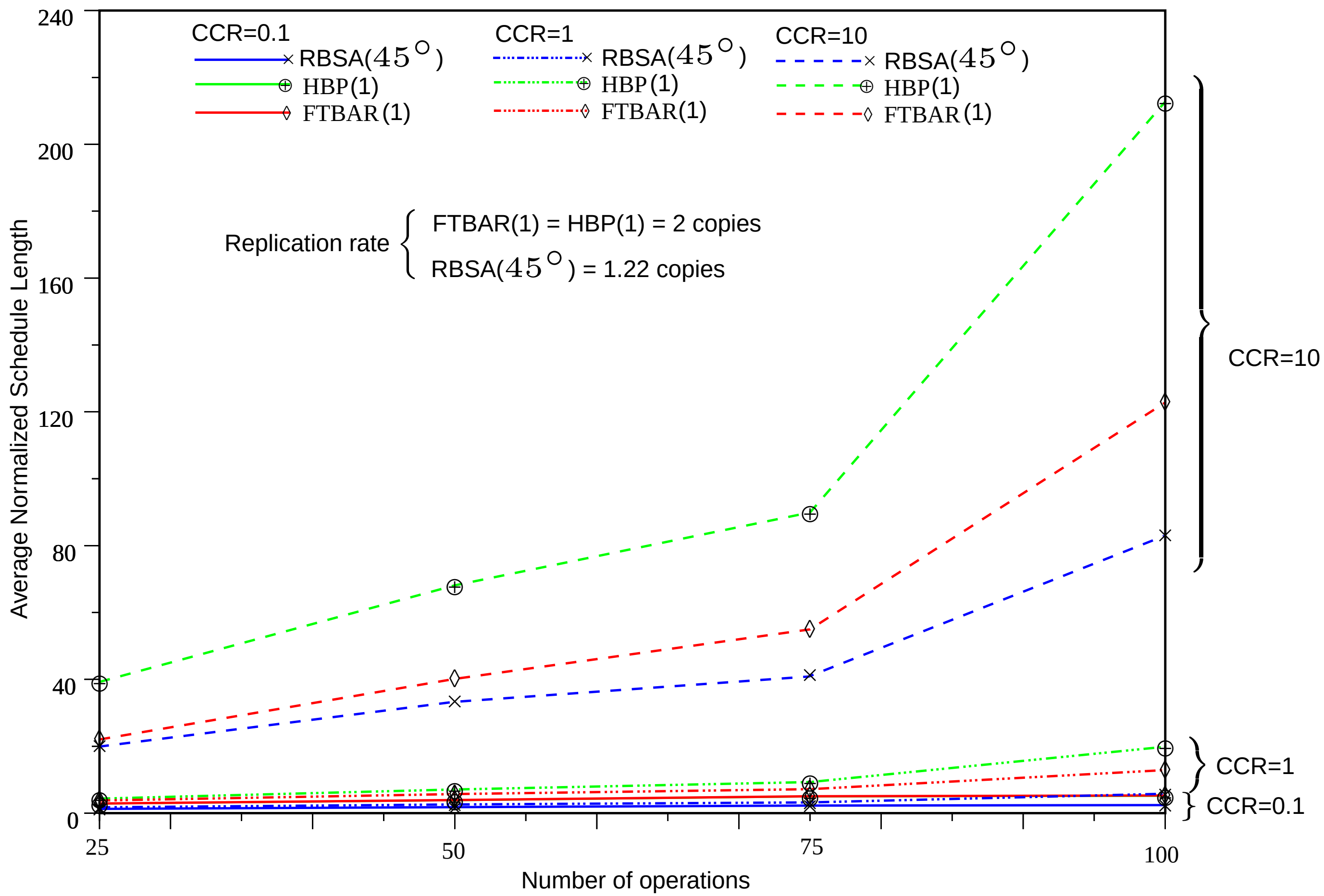
The architecture is assumed to be fully connected

$\mathcal{RBSA}(0^\circ)$ against FTBAR(0) [Girault, Kalla, Sighireanu & Sorel 03] and HBP(0) [Hashimoto, Tsuchiya & Kikuno 02]



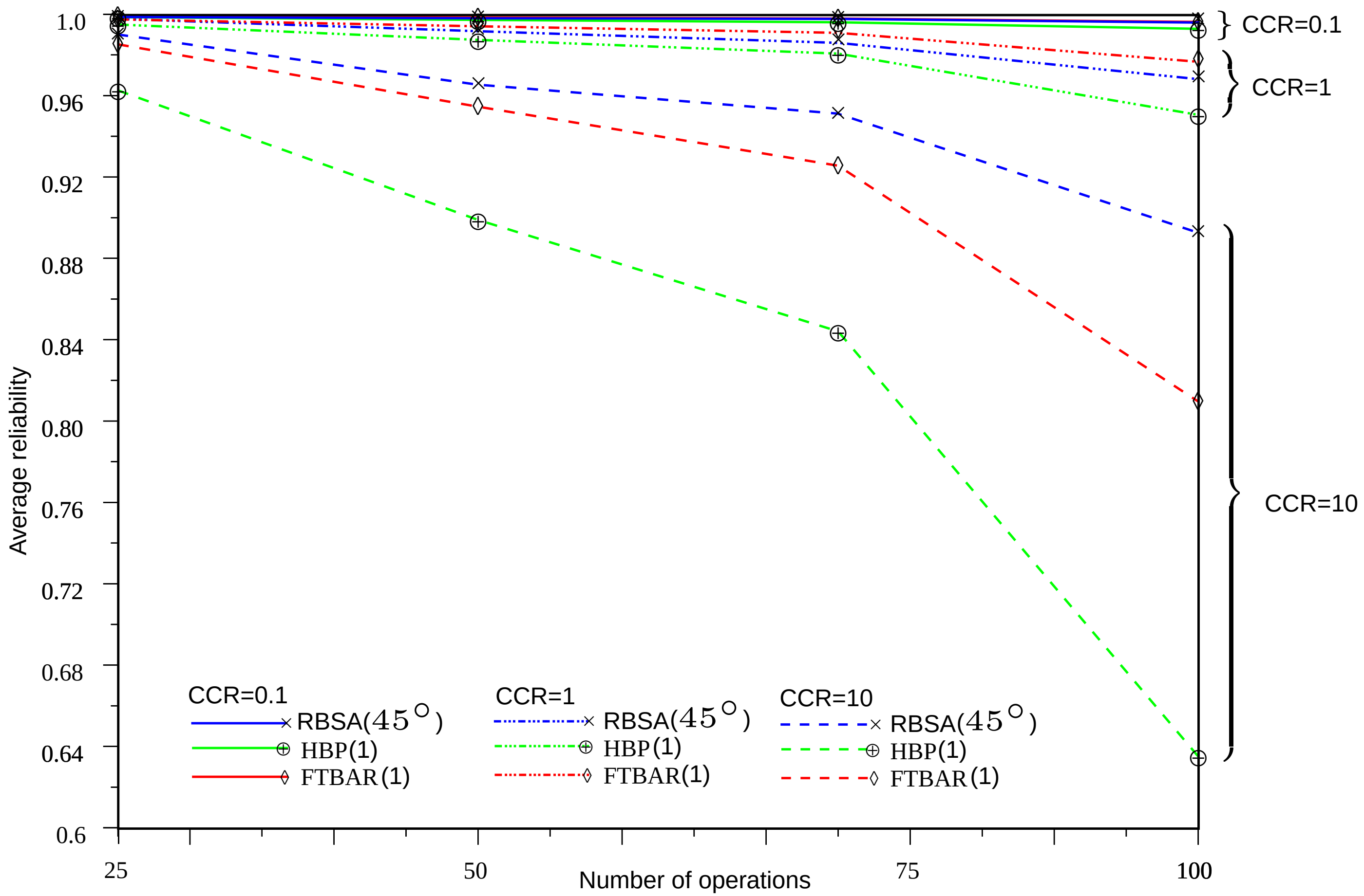
# $\mathcal{RBSA}$ simulations (II)

$\mathcal{RBSA}(45^\circ)$ against FTBAR(1) and HBP(1)



# $\mathcal{RBSA}$ simulations (II)

$\mathcal{RBSA}(45^\circ)$ against FTBAR(1) and HBP(1)



# Analysis

- If only $\mathcal{R}t_{obj}$ is considered, $\mathcal{RBSA}$ is almost as good as FTBAR
- If both objectives are considered, $\mathcal{RBSA}$ is significantly better, and all the more when CCR $\gg$ 1

  ⇨ thanks to our use of replication to improve both the reliability and the locality of computations
- More simulations are required to compare $\mathcal{RBSA}$ with FTBAR(2), on bigger architectures (6 processors)