# Strategies for Replica Placement in Tree Networks

Anne Benoit, Veronika Rehn and Yves Robert

GRAAL team, LIP
École Normale Supérieure de Lyon

30th November 2006

# Outline

1. Rollback
2. Replica Cost problem - No QoS
   - Heuristics
   - Experiments
3. Replica Cost problem - QoS=distance
   - Heuristics
   - Experiments
4. Conclusion

# Outline

1. Rollback
2. Replica Cost problem - No QoS
   - Heuristics
   - Experiments
3. Replica Cost problem - QoS=distance
   - Heuristics
   - Experiments
4. Conclusion

# Introduction and motivation

- Replica placement in tree networks
- Set of clients (tree leaves): requests with QoS constraints, known in advance
- Internal nodes may be provided with a replica;
  in this case they become servers
  and process requests (up to their capacity limit)

How many replicas required?
Which locations?
Total replica cost?

# Introduction and motivation

- Replica placement in tree networks
- Set of clients (tree leaves): requests with QoS constraints, known in advance
- Internal nodes may be provided with a replica;
  in this case they become servers
  and process requests (up to their capacity limit)

How many replicas required?
Which locations?
Total replica cost?

# Introduction and motivation

- Replica placement in tree networks
- Set of clients (tree leaves): requests with QoS constraints, known in advance
- Internal nodes may be provided with a replica;
  in this case they become servers
  and process requests (up to their capacity limit)

How many replicas required?
Which locations?
Total replica cost?

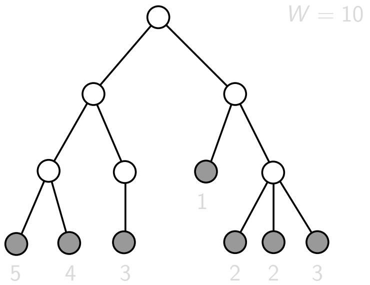
# Introduction and motivation

- Replica placement in tree networks
- Set of clients (tree leaves): requests with QoS constraints, known in advance
- Internal nodes may be provided with a replica;
  in this case they become servers
  and process requests (up to their capacity limit)

How many replicas required?
Which locations?
Total replica cost?

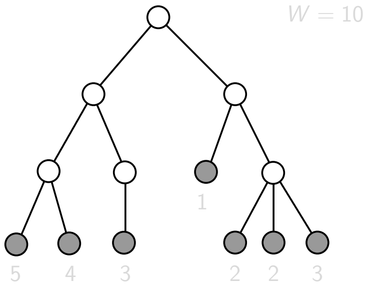
# Rule of the game

- Handle all client requests, and minimize cost of replicas
- → REPLICA PLACEMENT problem
- Several policies to assign replicas



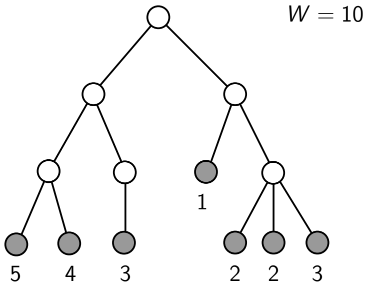
# Rule of the game

- Handle all client requests, and minimize cost of replicas
- → Replica Placement problem
- Several policies to assign replicas



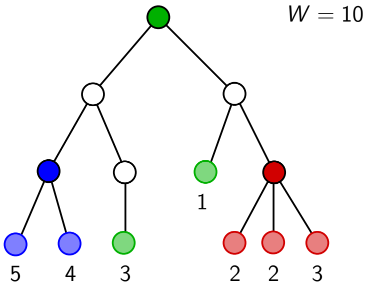
# Rule of the game

- Handle all client requests, and minimize cost of replicas
- → Replica Placement problem
- Several policies to assign replicas



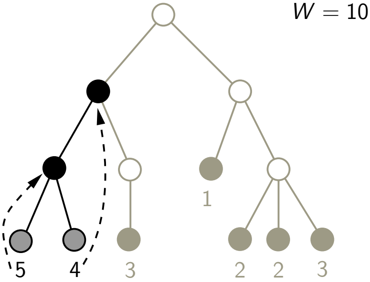
# Rule of the game

- Handle all client requests, and minimize cost of replicas
- → Replica Placement problem
- Several policies to assign replicas



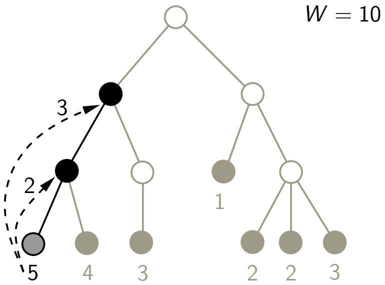
# Rule of the game

- Handle all client requests, and minimize cost of replicas
- → REPLICA PLACEMENT problem
- Several policies to assign replicas



# Rule of the game

- Handle all client requests, and minimize cost of replicas
- → Replica Placement problem
- Several policies to assign replicas



# Major contributions

Theory New access policies
Problem complexity
LP-based lower bound to cost of Replica Placement

Practice Heuristics for each policy
Experiments to assess impact of new policies

# Major contributions

Theory
- New access policies
- Problem complexity
- LP-based lower bound to cost of REPLICA PLACEMENT

Practice
- Heuristics for each policy
- Experiments to assess impact of new policies

# Single server vs Multiple servers

Single server – Each client $i$ is assigned a single server server($i$), that is responsible for processing all its requests.

Multiple servers – A client $i$ may be assigned several servers in a set Servers($i$). Each server $s \in$ Servers($i$) will handle a fraction $r_{i,s}$ of the requests.

In the literature: single server policy with additional constraint.

# Complexity results - Basic problem

| | Replica Counting<br>**Homogeneous** | Replica Cost<br>**Heterogeneous** |
|---|---|---|
| **Closest** | polynomial [Cidon02,Liu06] | NP-complete |
| **Upwards** | NP-complete | NP-complete |
| **Multiple** | polynomial algorithm | NP-complete |

# Complexity results - QoS and Bandwidth

- *Closest*/Homogeneous + QoS: Polynomial (Liu et al.)
- *Closest*/Homogeneous + Bandwidth: ?? (Probably polynomial, Vero still works at it ☺)
- *Multiple*/Homogeneous + QoS: NP-complete (reduction to 2-partition)
- *Multiple*/Homogeneous + Bandwidth: Polynomial? Algorithm quite similar to the case without BW, but proof still to check.

# Linear programming

- General instance of the problem
  - Heterogeneous tree
  - QoS and bandwidth constraints
  - *Closest*, *Upwards* and *Multiple* policies
- Integer linear program: no efficient algorithm
- Absolute lower bound if program solved over the rationals (using the GLPK software)

# Outline

# Heuristics

- Polynomial heuristics for the Replica Cost problem
  - Heterogeneous platforms
  - No QoS nor bandwidth constraints
- Experimental assessment of the relative performance of the three policies
- Traversals of the tree, bottom-up or top-down
- Worst case complexity $O(s^2)$, where $s = |\mathcal{C}| + |\mathcal{N}|$ is problem size

# Heuristics

- Polynomial heuristics for the REPLICA COST problem
  - Heterogeneous platforms
  - No QoS nor bandwidth constraints
- Experimental assessment of the relative performance of the three policies
- Traversals of the tree, bottom-up or top-down
- Worst case complexity $O(s^2)$,
  where $s = |\mathcal{C}| + |\mathcal{N}|$ is problem size

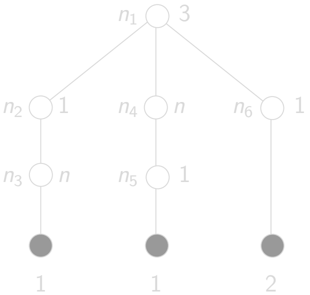
# Heuristics

- Polynomial heuristics for the Replica Cost problem
  - Heterogeneous platforms
  - No QoS nor bandwidth constraints
- Experimental assessment of the relative performance of the three policies
- Traversals of the tree, bottom-up or top-down
- Worst case complexity $O(s^2)$, where $s = |\mathcal{C}| + |\mathcal{N}|$ is problem size

# Heuristics for *Closest*

## Closest Top Down All **CTDA**

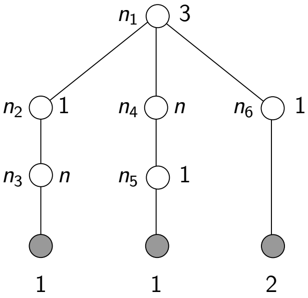
- Breadth-first traversal of the tree
- When a node can process the requests of all the clients in its subtree, node chosen as a server and exploration of the subtree stopped
- Procedure called until no more servers are added
- Choosing $n_2, n_4$ and then $n_1$



# Heuristics for *Closest*

## Closest Top Down All **CTDA**

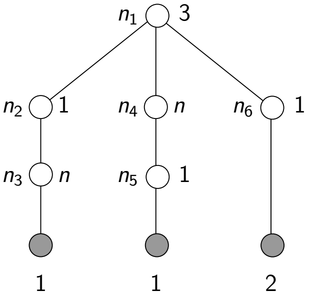
- Breadth-first traversal of the tree
- When a node can process the requests of all the clients in its subtree, node chosen as a server and exploration of the subtree stopped
- Procedure called until no more servers are added
- Choosing $n_2, n_4$ and then $n_1$



# Heuristics for *Closest*

## Closest Top Down Largest First **CTDLF**

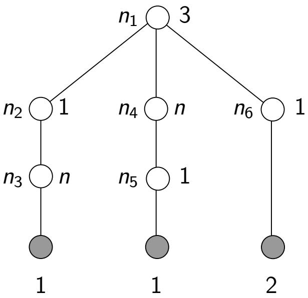
- Traversal of the tree, treating subtrees that contains most requests first
- When a node can process the requests of all the clients in its subtree, node chosen as a server and traversal stopped
- Procedure called until no more servers are added
- Choosing $n_2$ and then $n_1$



# Heuristics for *Closest*

## Closest Bottom Up **CBU**

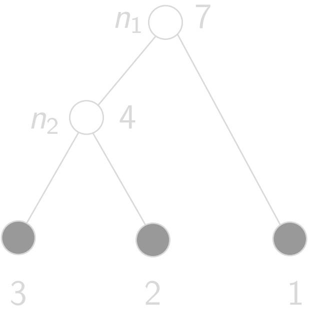
- Bottom-up traversal of the tree
- When a node can process the requests of all the clients in its subtree, node chosen as a server
- Choosing $n_3$, $n_5$, $n_1$



# Heuristics for *Upwards*

## Upwards Top Down **UTD**

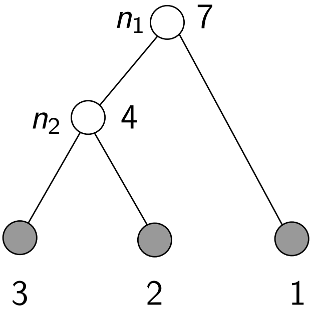
- 2-pass algorithm
- Select first saturating nodes, then extra nodes
- Choosing $n_2$ (for $c_1$) and in second pass $n_1$ (for $c_2, c_3$)



# Heuristics for *Upwards*

## Upwards Top Down **UTD**

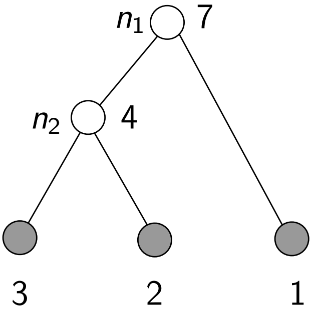
- 2-pass algorithm
- Select first saturating nodes, then extra nodes
- Choosing $n_2$ (for $c_1$) and in second pass $n_1$ (for $c_2, c_3$)



# Heuristics for *Upwards*

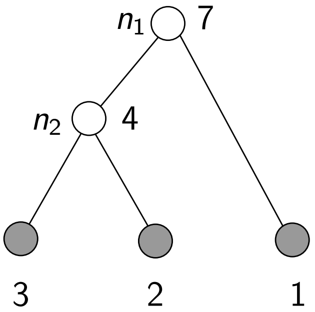
## Upwards Big Client First **UBCF**

- Sorting clients by decreasing request numbers, and finding the server of minimal available capacity to process its requests.
- Choosing $n_2$ for $c_1$, $n_1$ for $c_2$ and $n_1$ for $c_3$



# Heuristics for *Multiple*

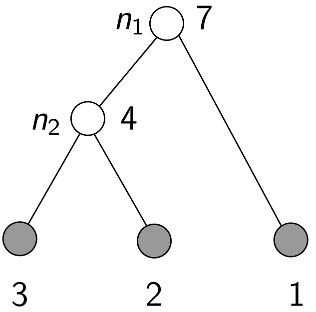
A greedy heuristic **MG**, similar to Pass 3 of the polynomial algorithm for *Multiple*/Homogeneous: fill all servers as much as possible in a bottom-up fashion




- MG affects 4 requests to $n_2$, and then the remaining 2 requests to $n_1$
- CTDLF better on this example: selects $n_1$ only

# Heuristics for *Multiple*

A greedy heuristic **MG**, similar to Pass 3 of the polynomial algorithm for *Multiple*/Homogeneous: fill all servers as much as possible in a bottom-up fashion




- MG affects 4 requests to $n_2$, and then the remaining 2 requests to $n_1$
- CTDLF better on this example: selects $n_1$ only

# Heuristics for *Multiple*

- A top-down and a bottom-up heuristic in 2-passes (**MTD**, **MBU**)
- Heuristic MixedBest **MB** which picks up best result over all heuristics: solution for the *Multiple* policy

# Plan of experiments

- Assess impact of the different access policies
- Assess performance of the polynomial heuristics
- Important parameter:

$$\lambda = \frac{\sum_{i \in \mathcal{C}} r_i}{\sum_{j \in \mathcal{N}} W_j}$$

- 30 trees for each $\lambda = 0.1, 0.2, ..., 0.9$
- Problem size $s = |\mathcal{C}| + |\mathcal{N}|$ such that $15 \leq s \leq 400$
- Computation of the LP lower bound for each tree

# Plan of experiments

- Assess impact of the different access policies
- Assess performance of the polynomial heuristics

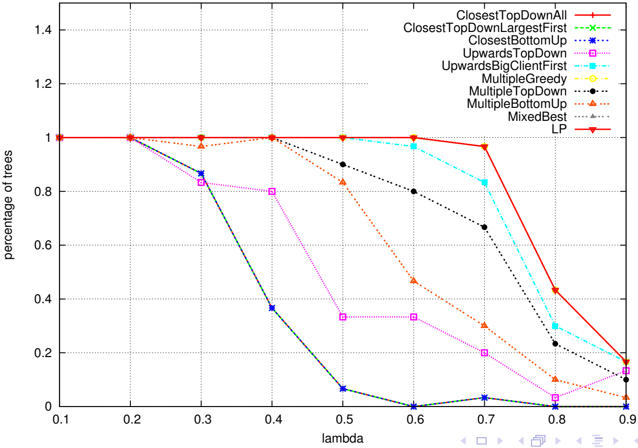
- Important parameter:

$$\lambda = \frac{\sum_{i \in \mathcal{C}} r_i}{\sum_{j \in \mathcal{N}} W_i}$$

- 30 trees for each $\lambda = 0.1, 0.2, ..., 0.9$
- Problem size $s = |\mathcal{C}| + |\mathcal{N}|$ such that $15 \leq s \leq 400$
- Computation of the LP lower bound for each tree

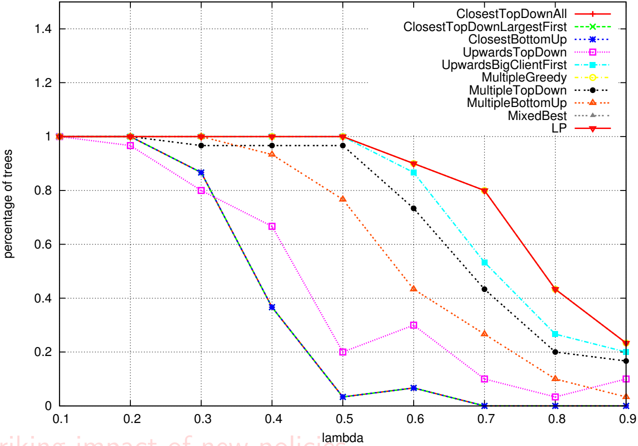
# Results - Percentage of success

- Number of solutions for each lambda and each heuristic
- No LP solution → No solution for any heuristic
- Homogeneous case



# Results - Percentage of success

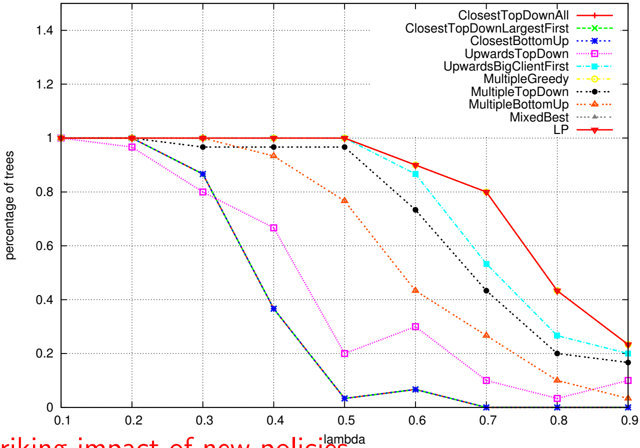
- Heterogeneous trees: similar results




- Striking impact of new policies
- MG and MB always find the solution

# Results - Percentage of success

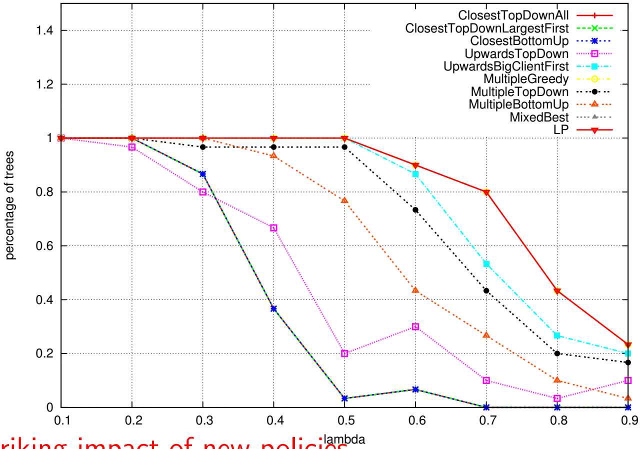
- Heterogeneous trees: similar results




- Striking impact of new policies
- MG and MB always find the solution

# Results - Percentage of success

- Heterogeneous trees: similar results




- Striking impact of new policies
- MG and MB always find the solution

# Results - Solution cost

- Distance of the result (in terms of replica cost) of the heuristic to the lower bound
- $T_\lambda$: subset of trees with a solution
- Relative cost:

$$rcost = \frac{1}{|T_\lambda|} \sum_{t \in T_\lambda} \frac{cost_{LP}(t)}{cost_h(t)}$$

- $cost_{LP(t)}$: lower bound cost on tree $t$
- $cost_h(t)$: heuristic cost on tree $t$; $cost_h(t) = +\infty$ if $h$ did not find any solution

# Results - Solution cost

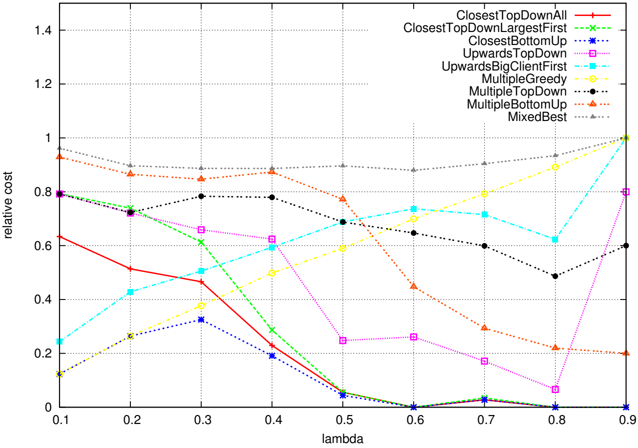
- Distance of the result (in terms of replica cost) of the heuristic to the lower bound
- $T_\lambda$: subset of trees with a solution
- Relative cost:

$$rcost = \frac{1}{|T_\lambda|} \sum_{t \in T_\lambda} \frac{cost_{LP}(t)}{cost_h(t)}$$

- $cost_{LP(t)}$: lower bound cost on tree $t$
- $cost_h(t)$: heuristic cost on tree $t$; $cost_h(t) = +\infty$ if $h$ did not find any solution

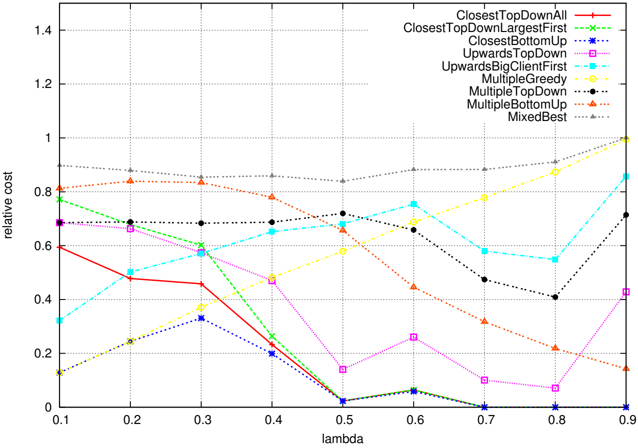
# Results - Solution cost

- Homogeneous results

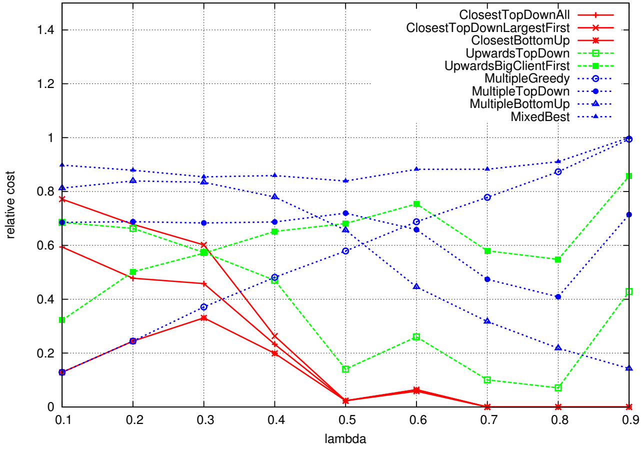


# Results - Solution cost

- Heterogeneous results - similar to the homogeneous case



# Results - Hierarchy



# Summary

- Striking effect of new policies: many more solutions to the REPLICA PLACEMENT problem
- *Multiple* $\geq$ *Upwards* $\geq$ *Closest*: hierarchy observed within our heuristics
- Best *Multiple* heuristic (MB) always at 85% of the lower bound: satisfactory result

# Outline

# Heuristics

- Polynomial heuristics for the REPLICA COST problem
  - Heterogeneous platforms
  - No bandwidth constraints
  - QoS constraints
- Experimental assessment of the impact of QoS constraints on performances
- No more tree traversals, but sorted lists of clients or servers
- Worst case complexity $O(s^2)$, where $s = |\mathcal{C}| + |\mathcal{N}|$ is problem size

# Heuristics

- Polynomial heuristics for the Replica Cost problem
  - Heterogeneous platforms
  - No bandwidth constraints
  - QoS constraints
- Experimental assessment of the impact of QoS constraints on performances
- No more tree traversals, but sorted lists of clients or servers
- Worst case complexity $O(s^2)$, where $s = |\mathcal{C}| + |\mathcal{N}|$ is problem size

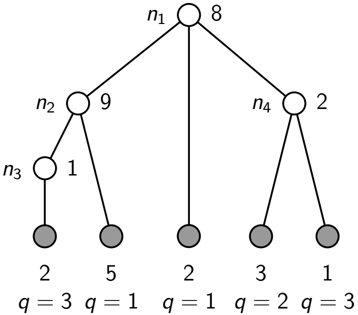
# Heuristics

- Polynomial heuristics for the Replica Cost problem
  - Heterogeneous platforms
  - No bandwidth constraints
  - QoS constraints
- Experimental assessment of the impact of QoS constraints on performances
- No more tree traversals, but sorted lists of clients or servers
- Worst case complexity $O(s^2)$,
  where $s = |\mathcal{C}| + |\mathcal{N}|$ is problem size

# Heuristics for *Closest*

## Closest Big Subtree First **CBSF**

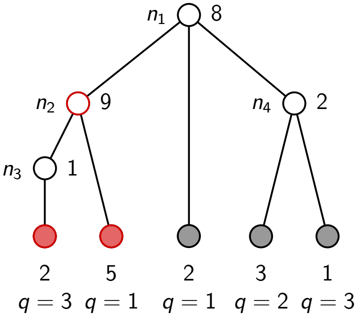
- Traversal of the tree, treating subtrees that contains most requests first
- When a node can process the requests of all the clients in its subtree, node chosen as a server and traversal stopped
- Procedure called until no more servers are added



# Heuristics for *Closest*

## Closest Big Subtree First **CBSF**

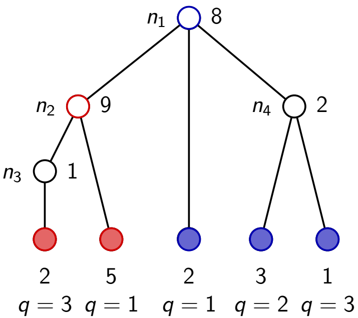
- Traversal of the tree, treating subtrees that contains most requests first
- When a node can process the requests of all the clients in its subtree, node chosen as a server and traversal stopped
- Procedure called until no more servers are added



# Heuristics for *Closest*

## Closest Big Subtree First **CBSF**

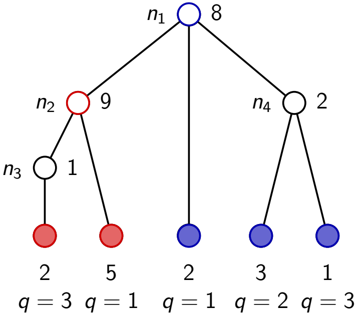
- Traversal of the tree, treating subtrees that contains most requests first
- When a node can process the requests of all the clients in its subtree, node chosen as a server and traversal stopped
- Procedure called until no more servers are added



# Heuristics for *Closest*

## Closest Small QoS First **CSQoS**

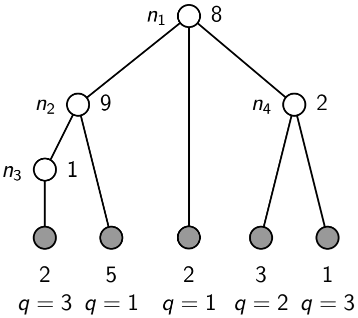
- Treating clients in non-decreasing order of QoS
- Looking for server the next to the root
- When a client is already treated, delete all treated clients
- Procedure called until no more servers are added



# Heuristics for *Upwards*

## Upwards Small QoS **USQoS**

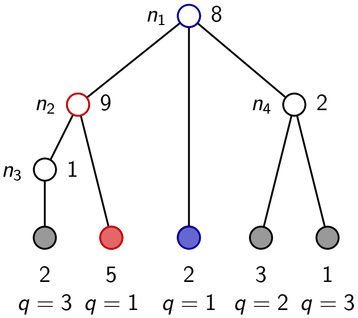
- Treating clients in non-decreasing order of QoS
- Choosing apropriate server
- 2 versions:
  - Started servers first
  - $\min(w_i - inreq_{QoS})$



# Heuristics for *Upwards*

## Upwards Small QoS **USQoS**

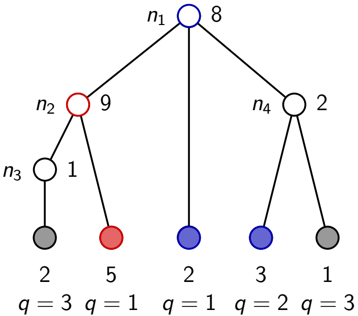
- Treating clients in non-decreasing order of QoS
- Choosing apropriate server
- 2 versions:
  - Started servers first
  - $\min(w_i - inreq_{QoS})$



# Heuristics for *Upwards*

## Upwards Small QoS **USQoS**

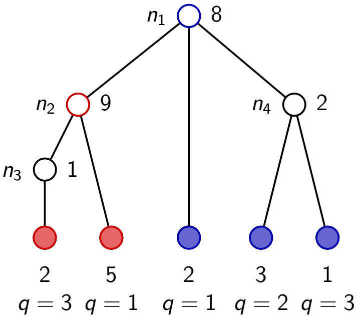
- Treating clients in non-decreasing order of QoS
- Choosing apropriate server
- 2 versions:
  - Started servers first
  - $\min(w_i - inreq_{QoS})$



# Heuristics for *Upwards*

## Upwards Small QoS **USQoS**

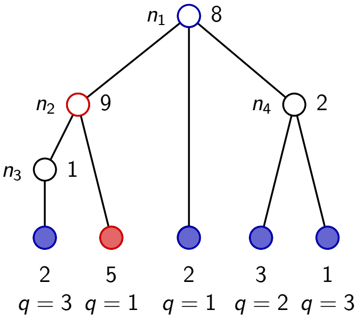
- Treating clients in non-decreasing order of QoS
- Choosing apropriate server
- 2 versions:
  - Started servers first
  - $\min(w_i - inreq_{QoS})$



# Heuristics for *Upwards*

## Upwards Small QoS **USQoS**

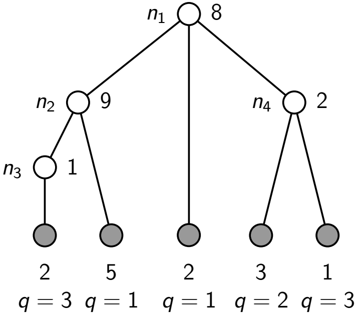
- Treating clients in non-decreasing order of QoS
- Choosing apropriate server
- 2 versions:
  - Started servers first
  - $\min(w_i - inreq_{QoS})$



# Heuristics for *Upwards*

## Upwards Dist Server Indisp **UDI**

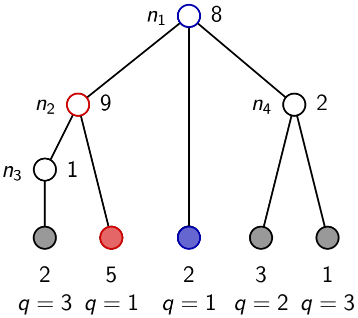
- Choose indispensable servers
- Sorting servers by non-decreasing value of reachable request numbers
- Deleting clients requests by distance to the actual server



# Heuristics for *Upwards*

## Upwards Dist Server Indisp **UDI**

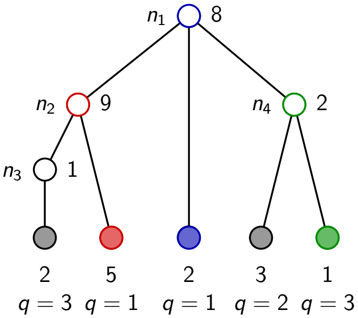
- Choose indispensable servers
- Sorting servers by non-decreasing value of reachable request numbers
- Deleting clients requests by distance to the actual server



# Heuristics for *Upwards*

## Upwards Dist Server Indisp **UDI**

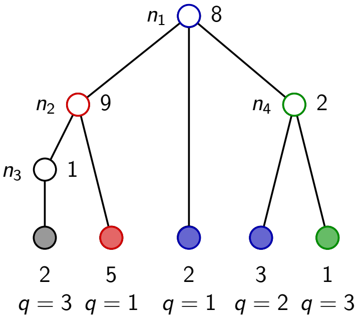
- Choose indispensable servers
- Sorting servers by non-decreasing value of reachable request numbers
- Deleting clients requests by distance to the actual server



# Heuristics for *Upwards*

## Upwards Dist Server Indisp **UDI**

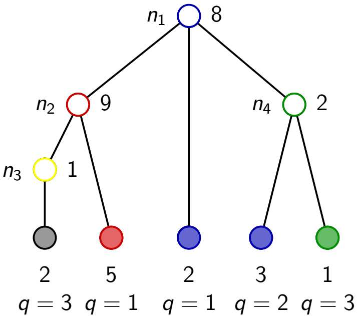
- Choose indispensable servers
- Sorting servers by non-decreasing value of reachable request numbers
- Deleting clients requests by distance to the actual server



# Heuristics for *Upwards*

## Upwards Dist Server Indisp **UDI**

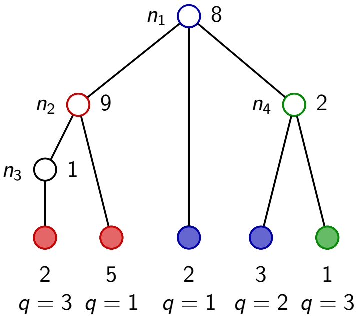
- Choose indispensable servers
- Sorting servers by non-decreasing value of reachable request numbers
- Deleting clients requests by distance to the actual server



# Heuristics for *Upwards*

## Upwards Dist Server Indisp **UDI**

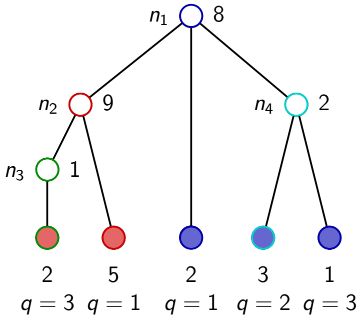
- Choose indispensable servers
- Sorting servers by non-decreasing value of reachable request numbers
- Deleting clients requests by distance to the actual server



# Heuristics for *Multiple*

## Multiple Small QoS First **MSQoS**

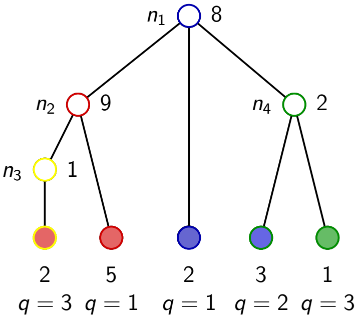
- Treating clients in non-decreasing order of QoS
- Choosing apropriate server
- 2 versions:
  - Close servers first
  - $\min(w_i - inreq_{QoS})$



# Heuristics for *Multiple*

## Multiple MinQoS Indisp **MMQoSI**

- Choose indispensable servers
- Sorting servers by non-decreasing value of reachable request numbers
- Deleting clients requests by $\min(QoS, dist(root))$



# Plan of experiments

- Assess impact of the different access policies
- Assess impact of the QoS constraints on the performance
- Important parameter:

$$\lambda = \frac{\sum_{i \in \mathcal{C}} r_i}{\sum_{j \in \mathcal{N}} W_j}$$

- 30 trees for each $\lambda = 0.1, 0.2, ..., 0.9$
- Problem size $s = |\mathcal{C}| + |\mathcal{N}|$ such that $15 \leq s \leq 400$
- Computation of the LP lower bound for each tree

# Plan of experiments

- Assess impact of the different access policies
- Assess impact of the QoS constraints on the performance

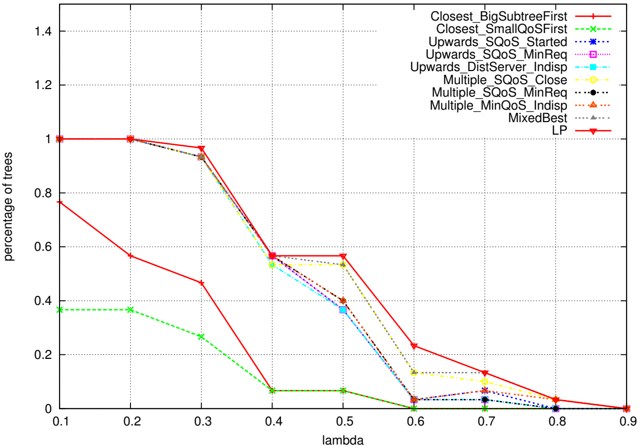
- Important parameter:

$$\lambda = \frac{\sum_{i \in \mathcal{C}} r_i}{\sum_{j \in \mathcal{N}} \mathrm{W}_i}$$

- 30 trees for each $\lambda = 0.1, 0.2, ..., 0.9$
- Problem size $s = |\mathcal{C}| + |\mathcal{N}|$ such that $15 \leq s \leq 400$
- Computation of the LP lower bound for each tree

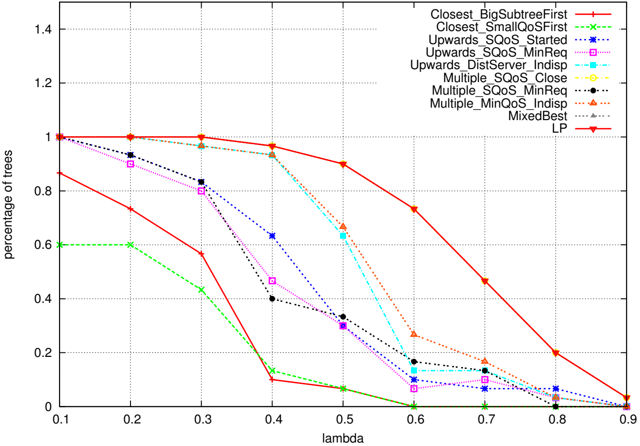
# Results - Percentage of success

- Number of solutions for each lambda and each heuristic
- Small trees, $qos \in \{1, 2\}$



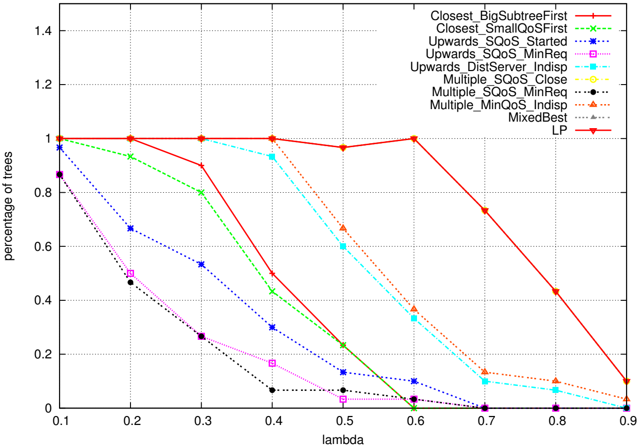
# Results - Percentage of success

- Number of solutions for each lambda and each heuristic
- Small trees, $average(qos) = height/2$



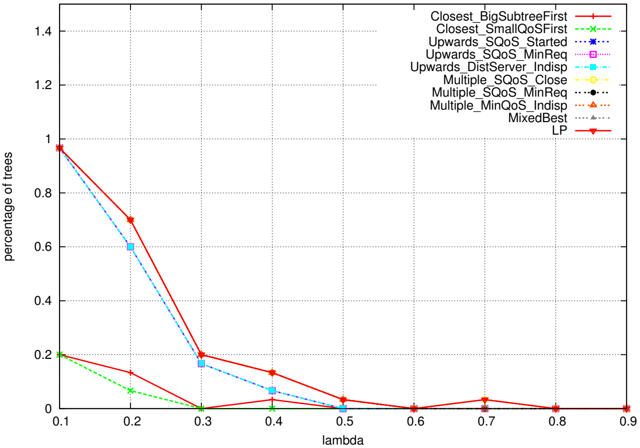
# Results - Percentage of success

- Number of solutions for each lambda and each heuristic
- Small trees, $qos = height + 1 \longrightarrow$ no qos



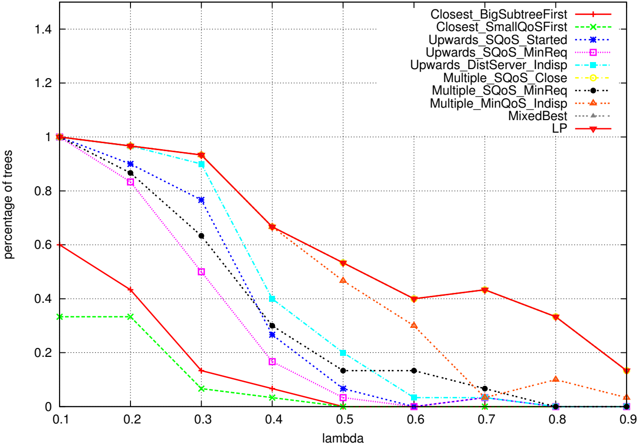
# Results - Percentage of success

- Number of solutions for each lambda and each heuristic
- Big trees, $qos \in \{1, 2\}$



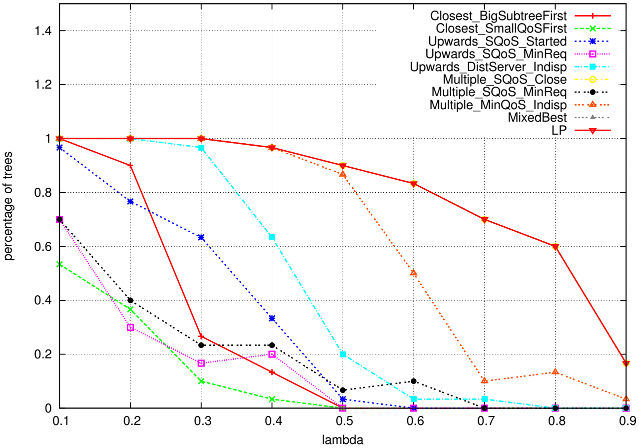
# Results - Percentage of success

- Number of solutions for each lambda and each heuristic
- Big trees, $average(qos) = height/2$



# Results - Percentage of success

- Number of solutions for each lambda and each heuristic
- Big trees, $qos = height + 1 \longrightarrow$ no qos



# Results - Solution cost

- Distance of the result (in terms of replica cost) of the heuristic to the lower bound
- $T_\lambda$: subset of trees with a solution
- Relative cost:

$$rcost = \frac{1}{|T_\lambda|} \sum_{t \in T_\lambda} \frac{cost_{LP}(t)}{cost_h(t)}$$

- $cost_{LP(t)}$: lower bound cost on tree $t$
- $cost_h(t)$: heuristic cost on tree $t$; $cost_h(t) = +\infty$ if $h$ did not find any solution

# Results - Solution cost

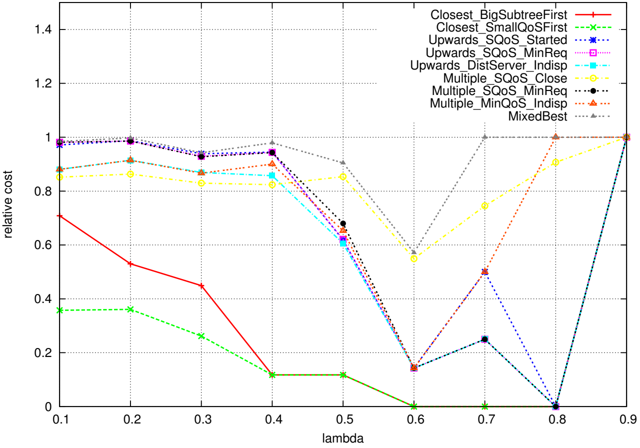
- Distance of the result (in terms of replica cost) of the heuristic to the lower bound
- $T_\lambda$: subset of trees with a solution
- Relative cost:

$$rcost = \frac{1}{|T_\lambda|} \sum_{t \in T_\lambda} \frac{cost_{LP}(t)}{cost_h(t)}$$

- $cost_{LP(t)}$: lower bound cost on tree $t$
- $cost_h(t)$: heuristic cost on tree $t$; $cost_h(t) = +\infty$ if $h$ did not find any solution

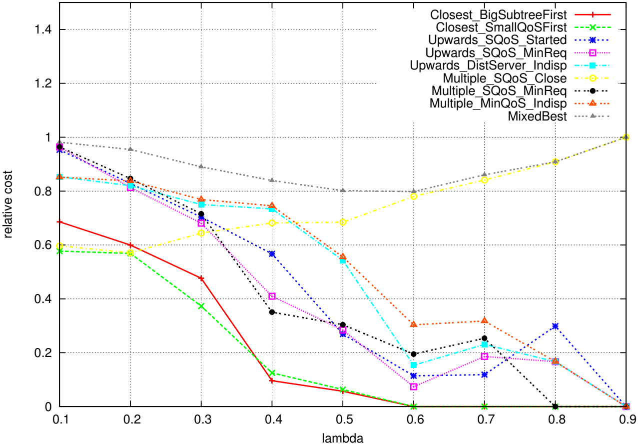
# Results - Solution cost

Small trees, $qos \in \{1, 2\}$



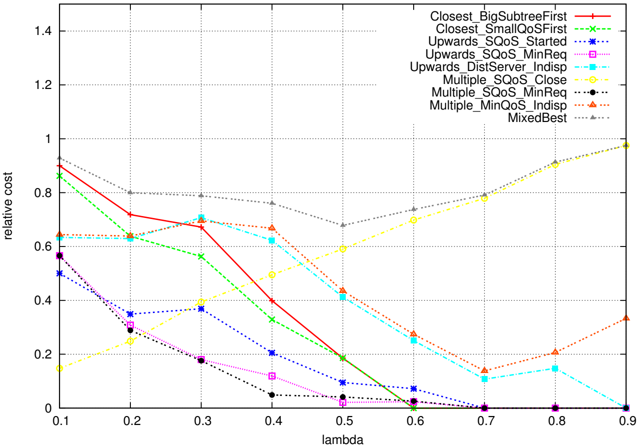
# Results - Solution cost

Small trees, $average(qos) = height/2$



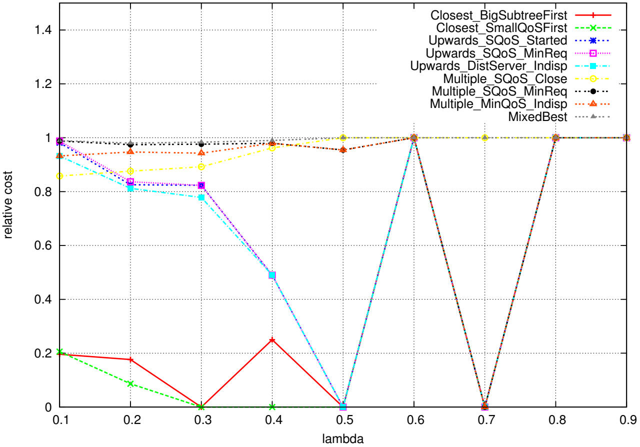
# Results - Solution cost

Small trees, $qos = height + 1 \longrightarrow$ no qos



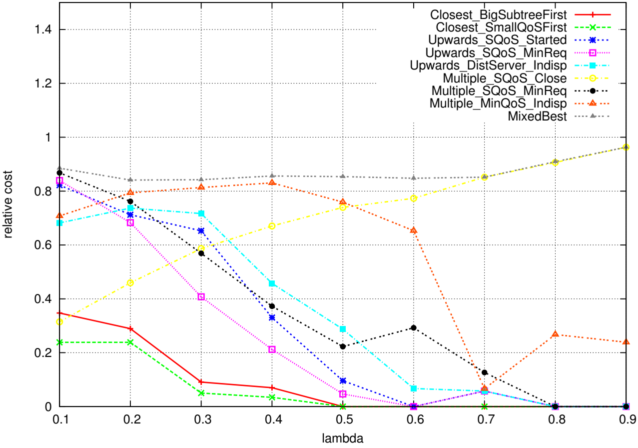
# Results - Solution cost

Big trees, $qos \in \{1, 2\}$



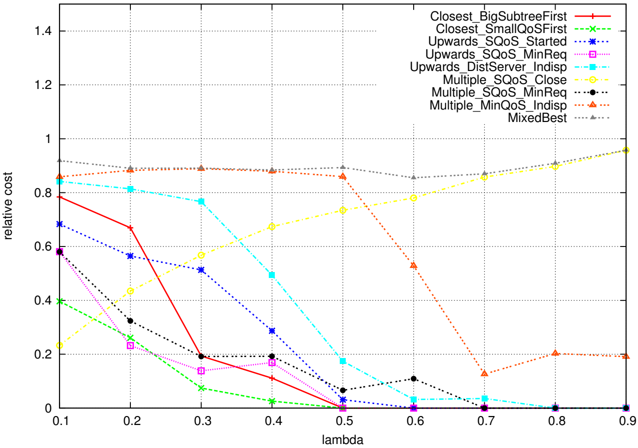
# Results - Solution cost

Big trees, $average(qos) = height/2$



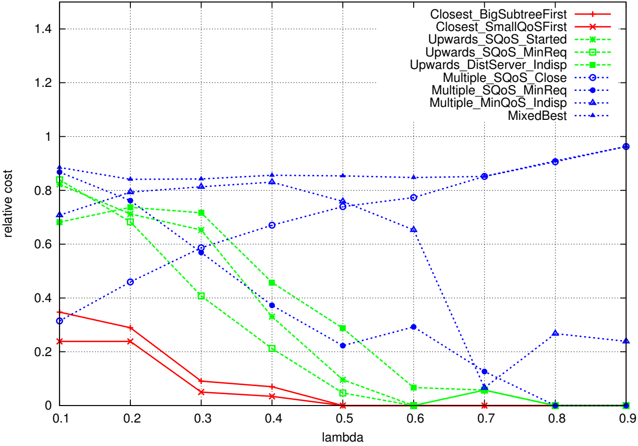
# Results - Solution cost

Big trees, $qos = height + 1 \longrightarrow$ no qos



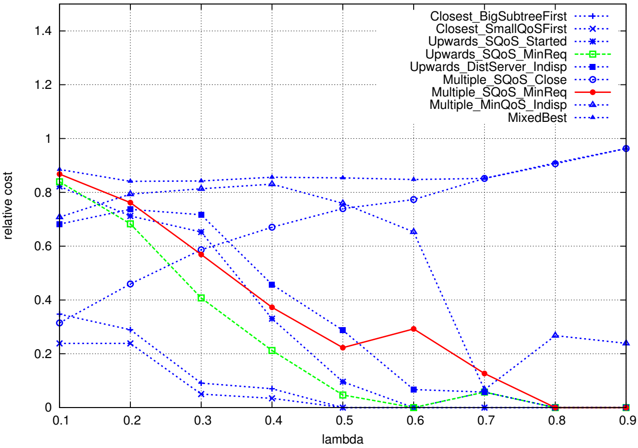
# Results - Hierarchy

Big trees, $average(qos) = height/2$



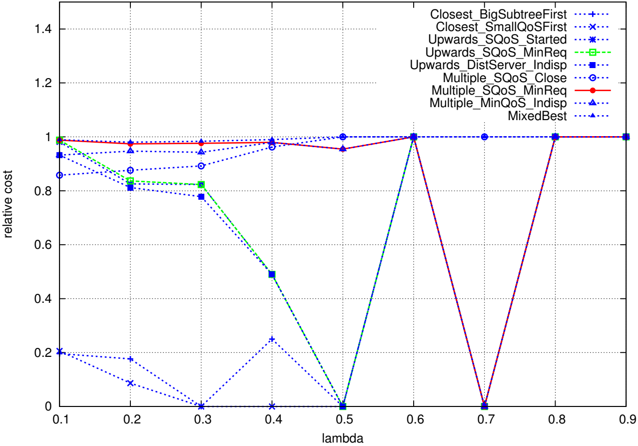
# Results - QoS impact

Big trees, $average(qos) = height/2$



# Results - QoS impact

Big trees, $qos \in \{1, 2\}$



# Summary

- *Multiple* $\geq$ *Upwards* $\geq$ *Closest*: hierarchy also under QoS-constraints
- Best *Multiple* heuristic (MB) at maximal 80% of the lower bound, when QoS constraints have $average(qos) = height/2$
- Big trees achieve better results:

| | |
|---|---|
| $qos \in \{1, 2\}$: | 95% (vs 90% avec une exception) |
| $average(qos) = height/2$: | 85% (vs 80%) |
| no qos: | 85% (vs 70%) |

# Outline

# Conclusion

- Introduction of two new policies for the REPLICA PLACEMENT problem
- *Upwards* and *Multiple*: natural variants of the standard *Closest* approach → surprising they have not already been considered

Theoretical side – Complexity of each policy, for homogeneous and heterogeneous platforms

Practical side

- Design of several heuristics for each policy
- Comparison of their performance
- Striking impact of the policy on the result
- Use of a LP-based lower bound to assess the absolute performance, which turns out to be quite good.

# Conclusion

- Introduction of two new policies for the REPLICA PLACEMENT problem
- *Upwards* and *Multiple*: natural variants of the standard *Closest* approach → surprising they have not already been considered

Theoretical side – Complexity of each policy, for homogeneous and heterogeneous platforms

Practical side

- Design of several heuristics for each policy
- Comparison of their performance
- Striking impact of the policy on the result
- Use of a LP-based lower bound to assess the absolute performance, which turns out to be quite good.

# Future work

## Short term

- More simulations for the Replica Cost problem: shape of the trees, distribution law of the requests, degree of heterogeneity of the platforms
- Designing heuristics for more general instances of the Replica Placement problem (QoS and bandwidth constraints): these constraints may lower the difference between policies

## Longer term

- Consider the problem with several object types
- Extension with more complex objective functions

Still a lot of challenging algorithmic problems ☺

# Future work

## Short term

- More simulations for the Replica Cost problem: shape of the trees, distribution law of the requests, degree of heterogeneity of the platforms
- Designing heuristics for more general instances of the Replica Placement problem (QoS and bandwidth constraints): these constraints may lower the difference between policies

## Longer term

- Consider the problem with several object types
- Extension with more complex objective functions

Still a lot of challenging algorithmic problems ☺