Dependable and Coordinated Resources Allocation Algorithms for Distributed Computing

Victor Toporkov and Dmitry Yemelyanov

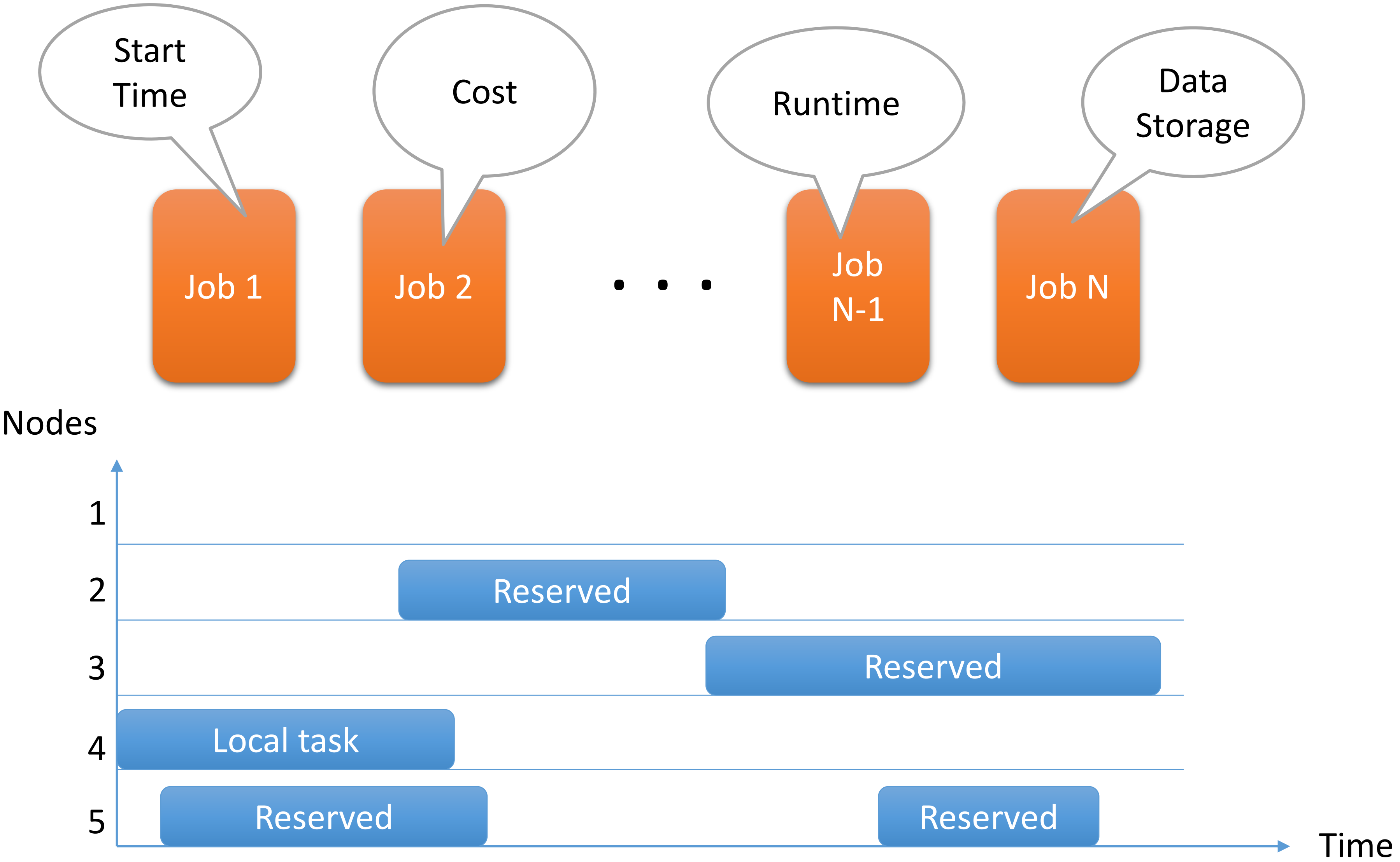
National Research University “Moscow Power Engineering Institute”

RSCDays 2018

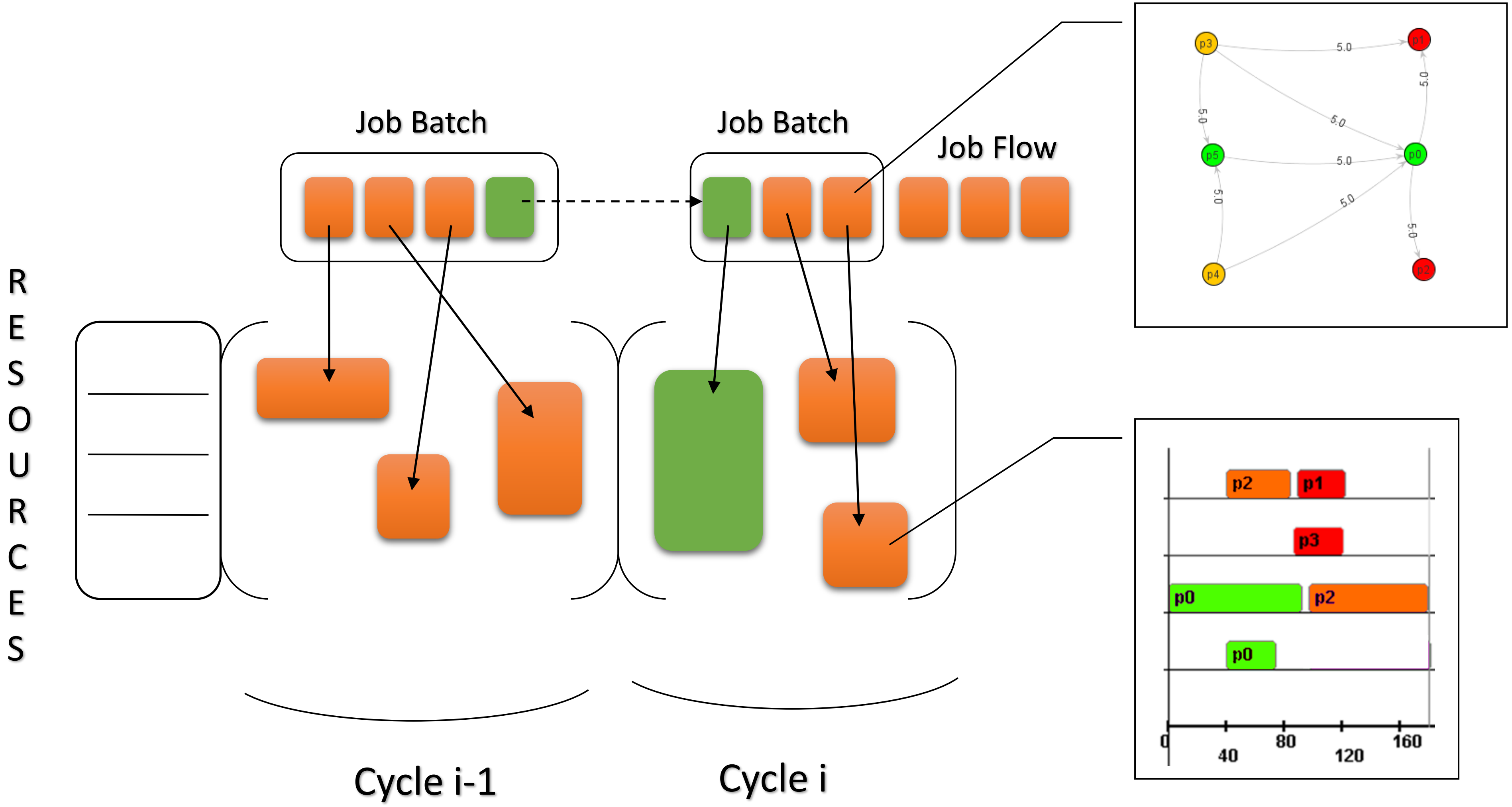
Introduction

- Resource co-allocation and scheduling of complex sets of parallel jobs is an essential issue in distributed environments with non-dedicated resources
- Economic models are used to solve resource sharing and scheduling problems in a transparent and an efficient way in cloud computing, utility Grids, and multi-agent systems
- The main aspect of job-flow scheduling is its efficiency in terms of QoS and resources utilization

Job Flow Scheduling Problem



Job Flow Scheduling: *V. Toporkov et al. 2017. Cyclic Anticipation Scheduling in Grid VOs with Stakeholders Preferences. PaCT 2017, LNCS 10421, pp. 372–383.*



Job Resource Request

The resource requirements are arranged into a resource request containing:

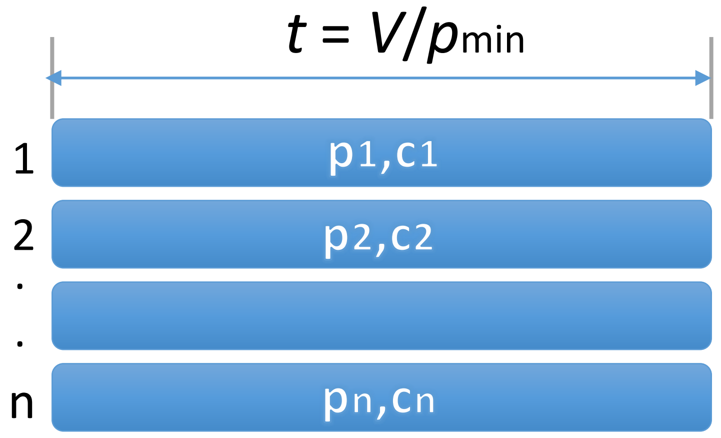
n - number of simultaneously reserved computational nodes

p - minimal performance requirement for each computational node

V – computational volume for a single node task

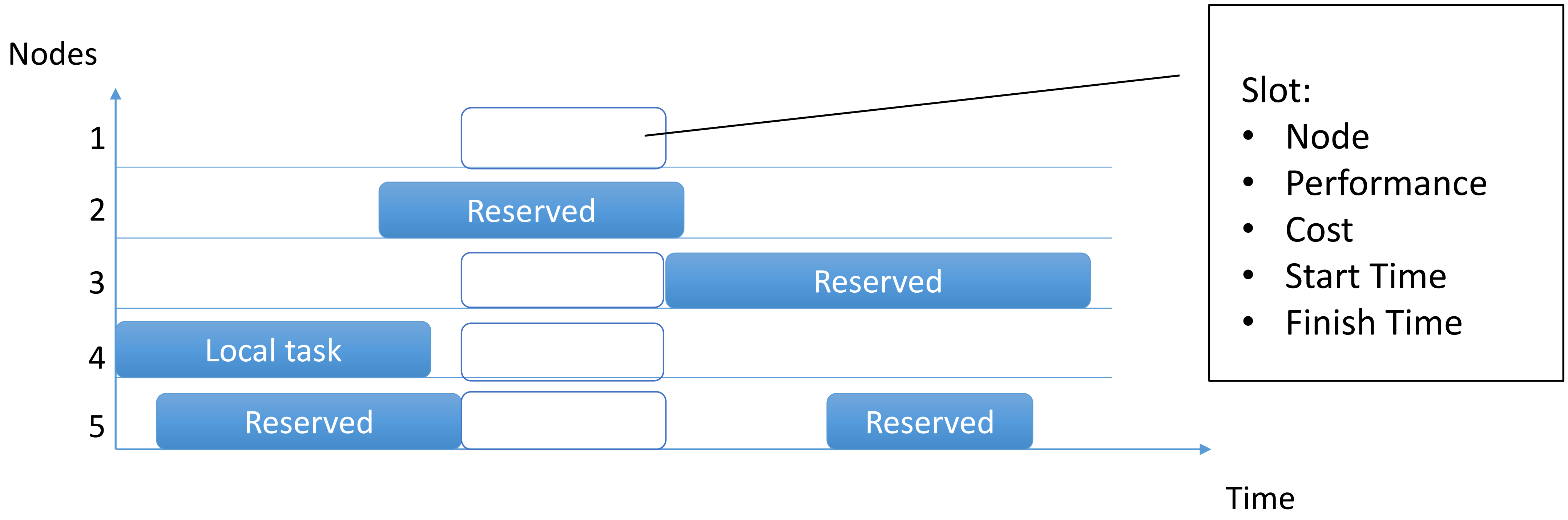
C - maximum total job execution cost (budget)

z – preferred job optimization criterion



Window Search Problem

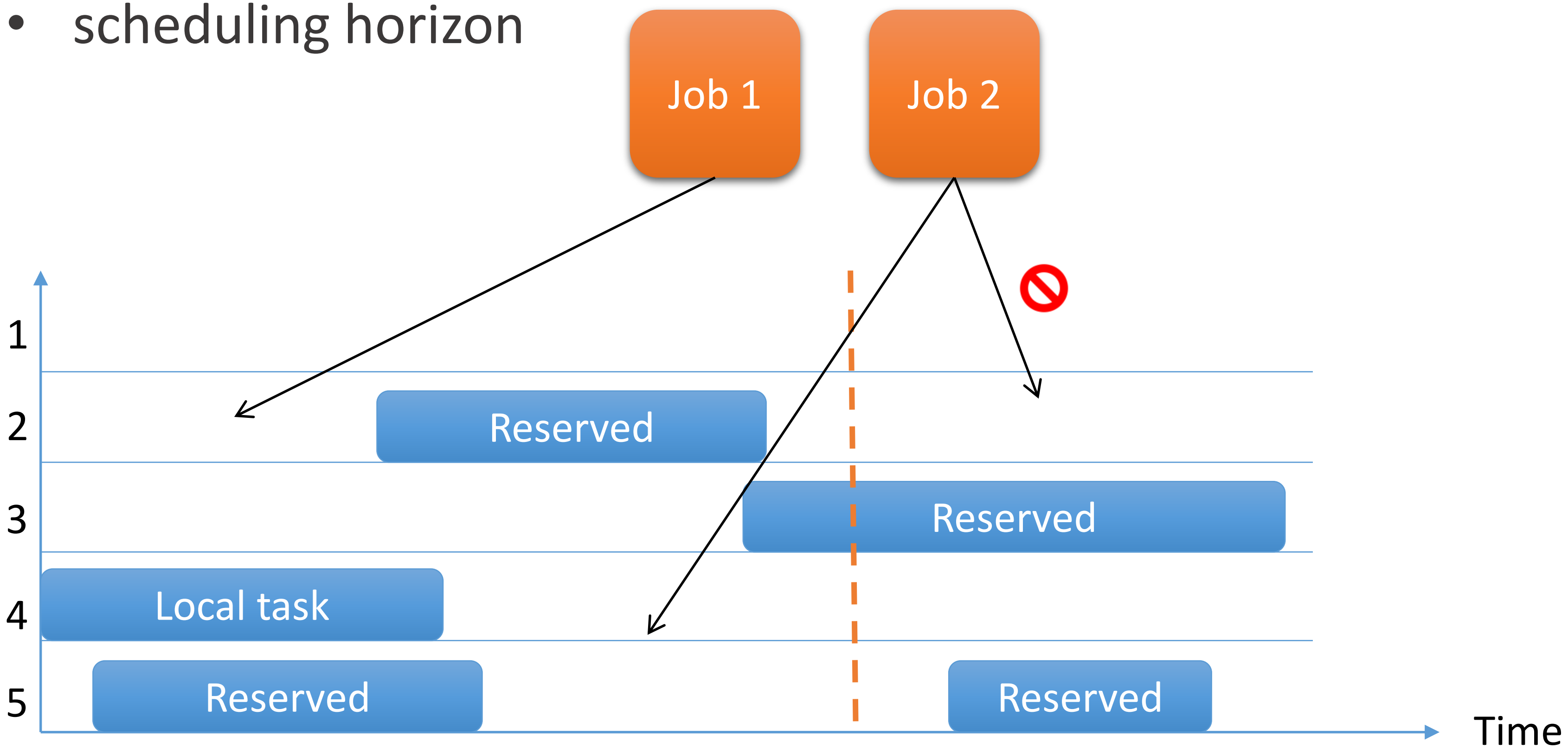
Allocate a window of four nodes for a time *T*, with requirements on nodes performance and total cost. Minimize window start time:



Deadline and Scheduling Horizon

There's a practical limit on the slots availability:

- deadline
- backfilling
- scheduling horizon



General Window Search Scheme

All available time-slots are ordered by the start time;

for each p_i in (p_{min}; p_{max}) {

while there is at least one slot available {

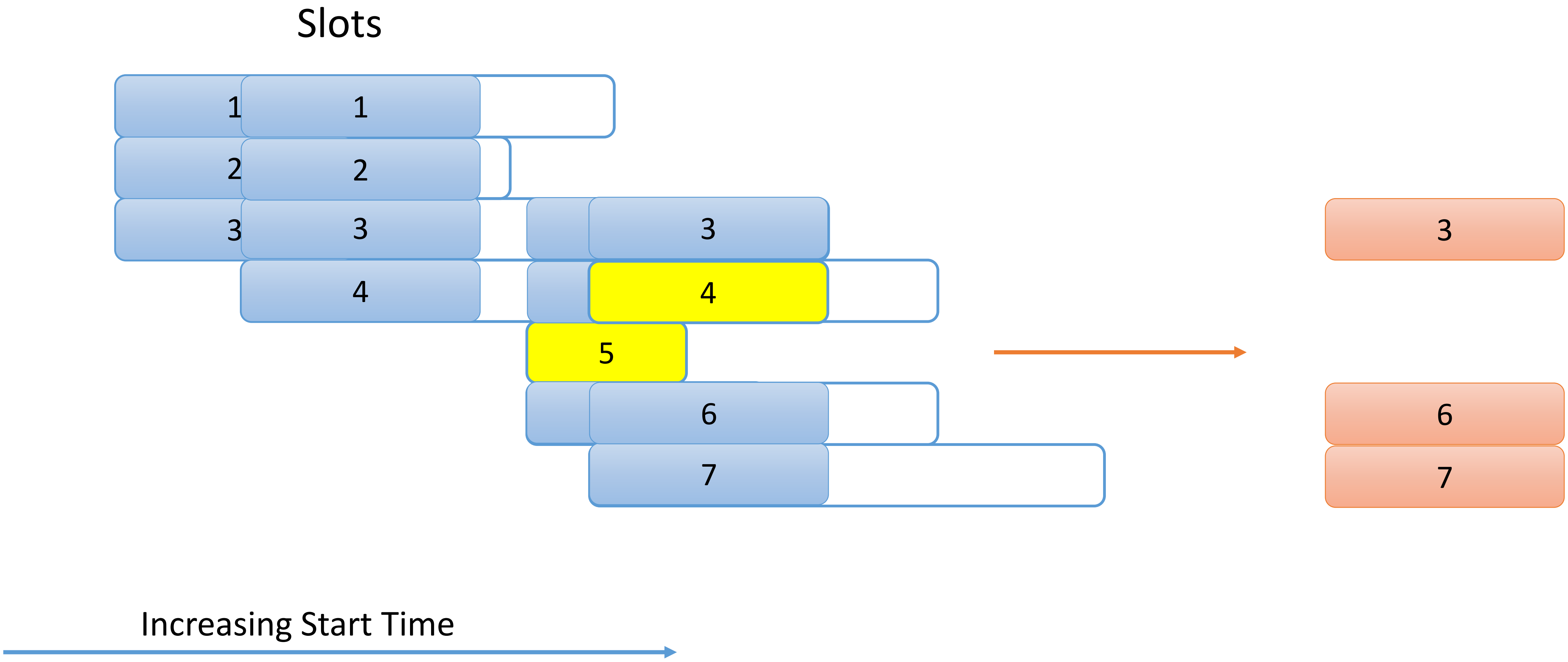
- Add next available slot to the window list;
- Check all slots in the window considering required length $t = V/p_{min}$ and remove the slots being late;
- Select *n*-slot window best by the given user criterion Z;

}

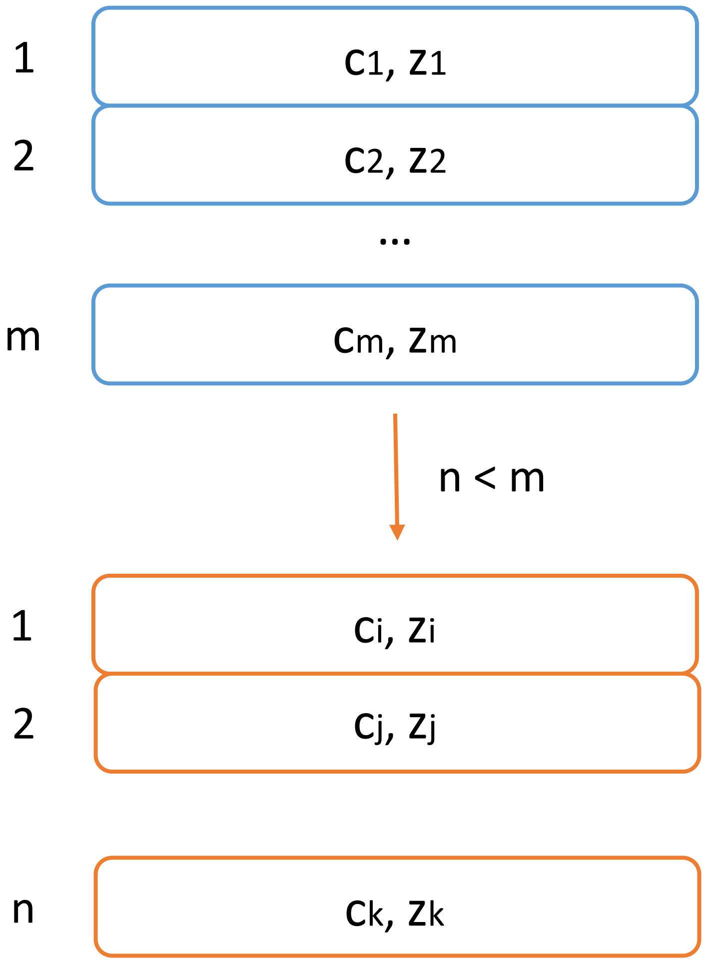
}

return the best of the found interim windows;

Window Search Scheme Visualized



Optimal Slots Subset Allocation



Maximize:

$$Z = x_1 z_1 + x_2 z_2 + \ldots + x_m z_m,$$

$$x_1 c_1 + x_2 c_2 + \ldots + x_m c_m \leq C$$

$$x_1 + x_2 + \ldots + x_m = n$$

$$x_i \in \{0, 1\}, i = 1, \ldots, m,$$

Optimization Scheme
0-1 Knapsack Problem with $O(m*C)$

$$f_i(C_j, n_k) = \max\{f_{i-1}(C_j, n_k), f_{i-1}(C_j - c_i, n_k - 1) + z_i\},$$
$$i = 1, \ldots, m, j = 1, \ldots, C, k = 1, \ldots, n,$$

i →

Cj ↓

k=1	c=3 z=3	c=2 z=1	c=4 z=6
0	-	-	-
1	-	-	-
2	-	1	1
3	3	3	3
4	3	3	6
5	3	3	6
6	3	3	6

k=2	c=3 z=3	c=2 z=1	c=4 z=6
0	-	-	-
1	-	-	-
2	-	-	-
3	-	-	-
4	-	-	-
5	-	4	4
6	-	4	7

$O(m * n * C)$

n << m

n << C

Simulation Study https://github.com/dmieter/mimapr

Intel Core i3-4130,16GB RAM

Simulation environment:

- heterogeneous resource domain model

  N =100 computing nodes

  L = 1200 scheduling horizon

- non-linear probabilistic model for the computing environment parameters generation: cost, performance, initial load
- a space-shared resources allocation policy

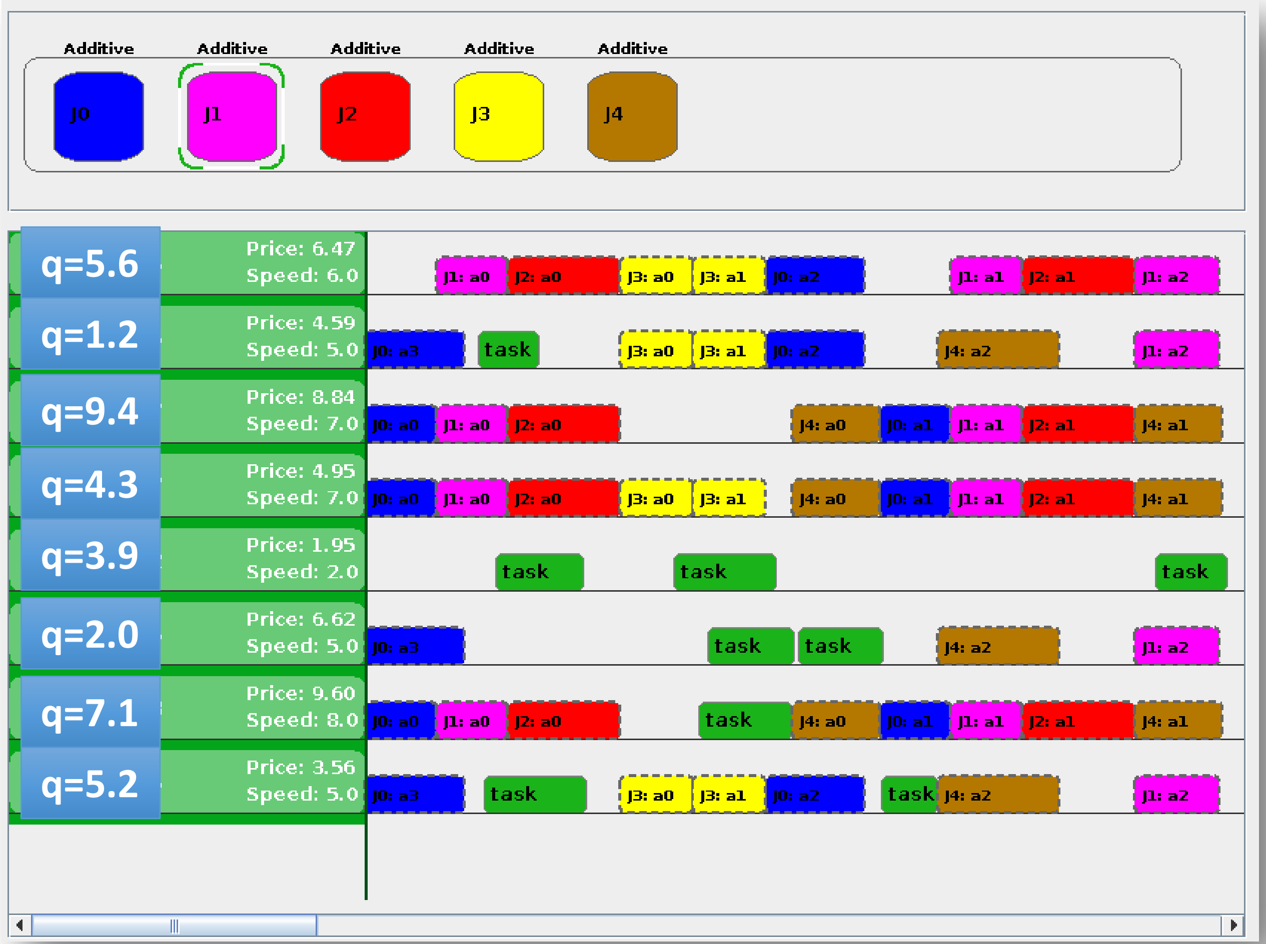
Job request:

n = 7 nodes

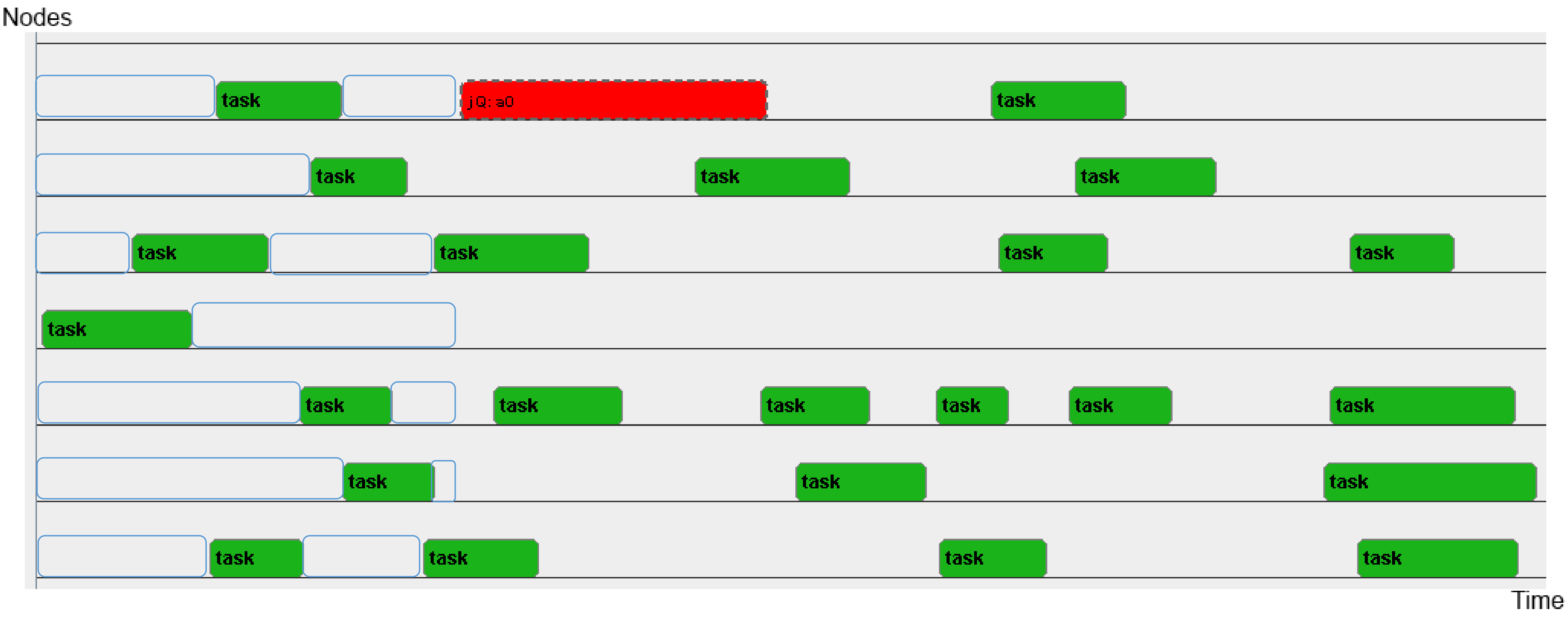
V = 800 computational volume

C = 644 maximum budget

qi in [0;10] randomly generated for each node to compare algorithms against Q (sum of qi, i = 1,…,n)



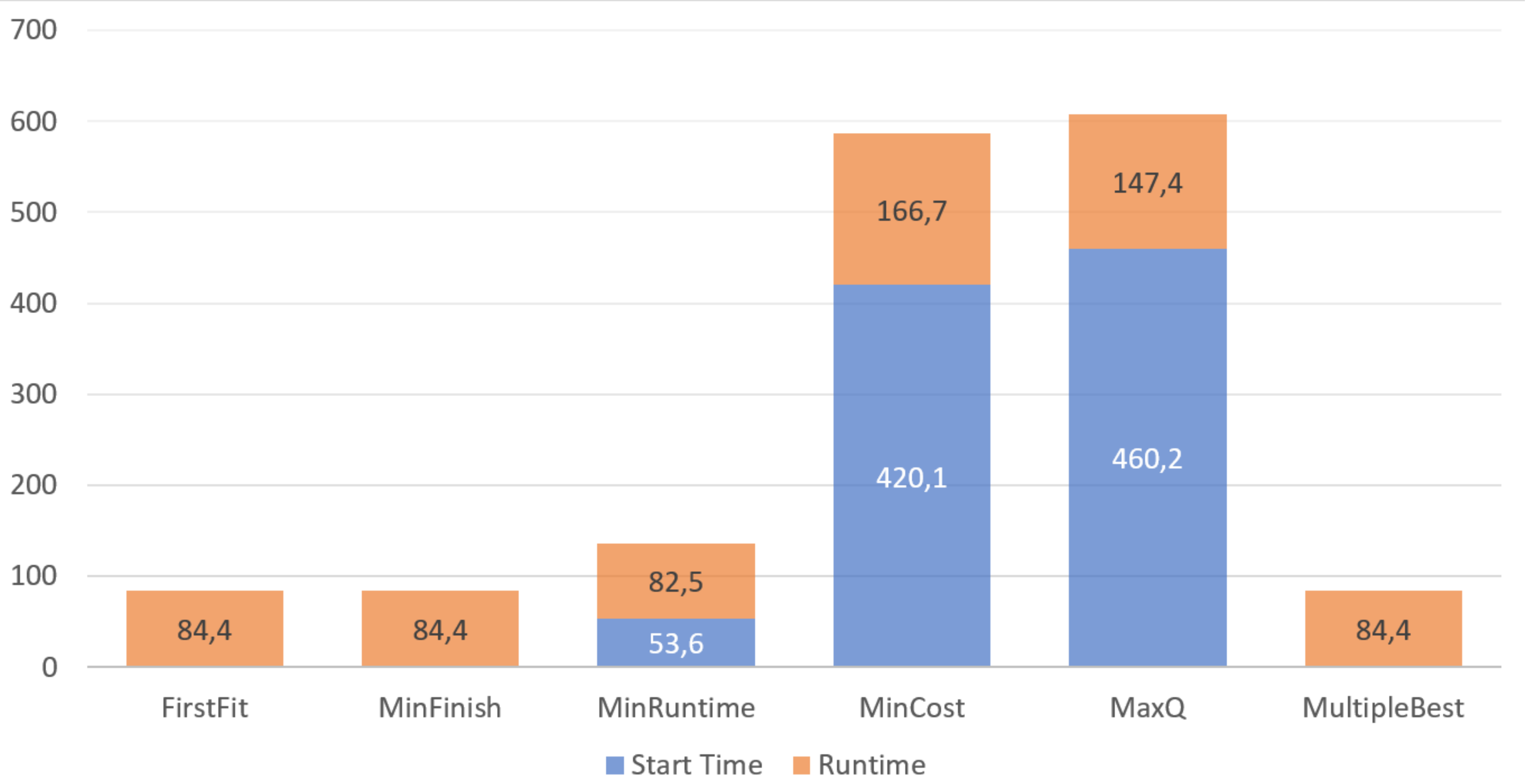
Resource Domain Utilization Example



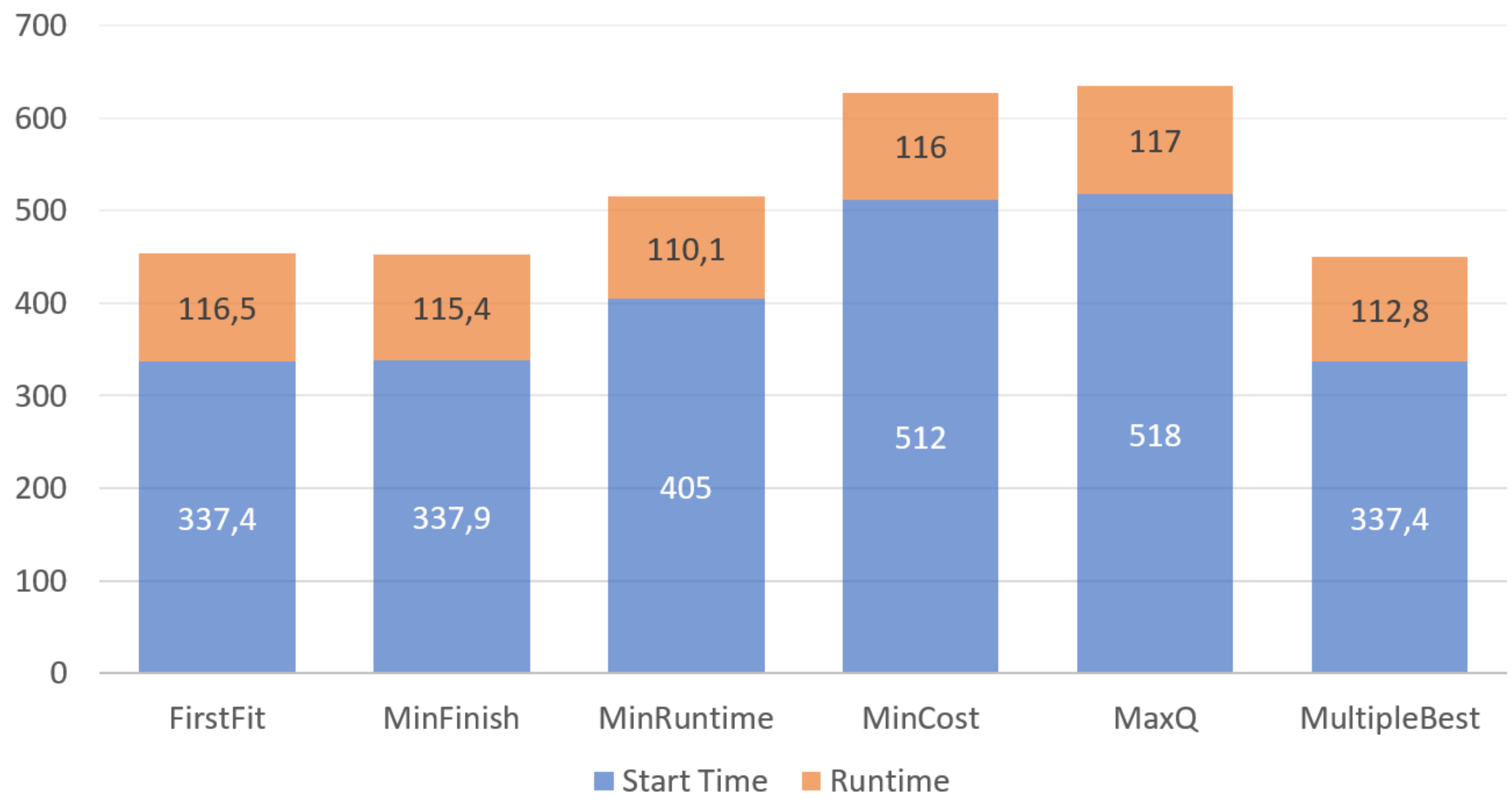
Algorithms Comparison

- First Fit returns first suitable and affordable window found — Kovalenko et al., 2007; Buyya et al., 2007
- Multiple Best returns the best window over a set of execution alternatives found — Ernemann et al., 2002
- General Window Search Scheme: — Toporkov et al., 2012
  - Min Finish Time
  - Min Runtime
  - Min Cost
- Max Q additionally implements an optimal slots subset allocation — Toporkov, Yemelyanov, 2018

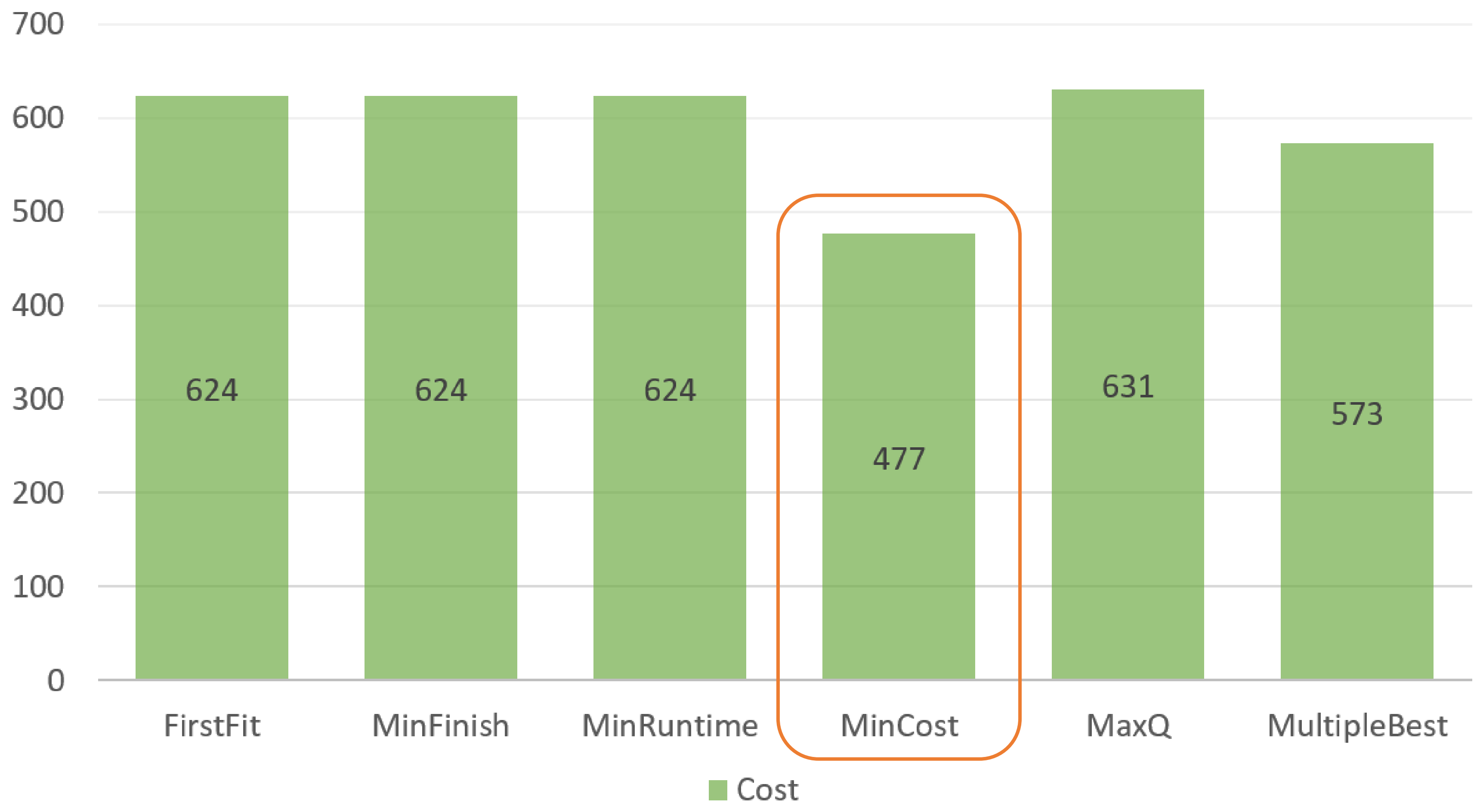
Simulation Study: Time I (7 nodes out of 100)



Simulation Study: Time II (7 nodes out of 40)

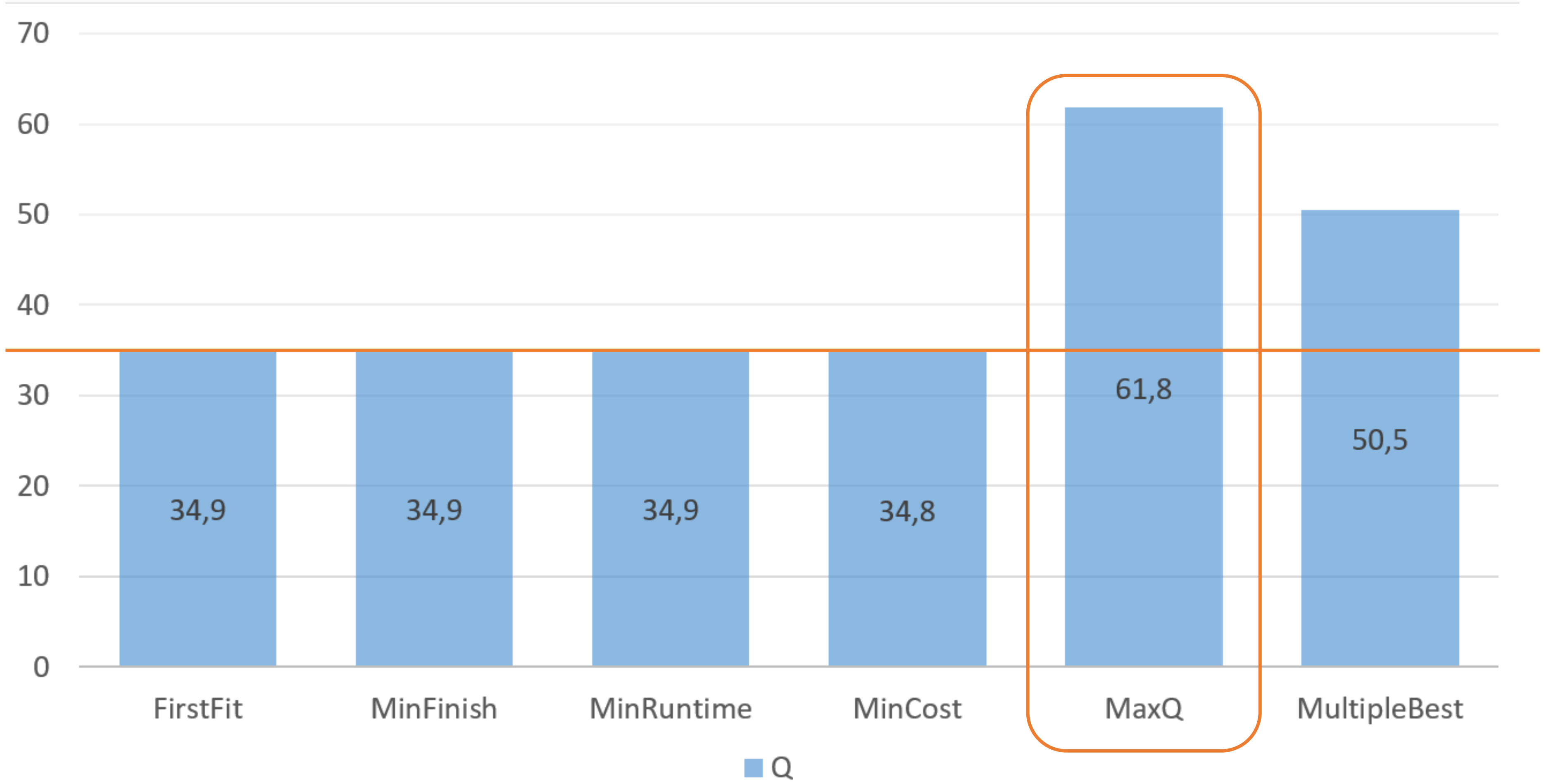


Simulation Study: Execution Cost

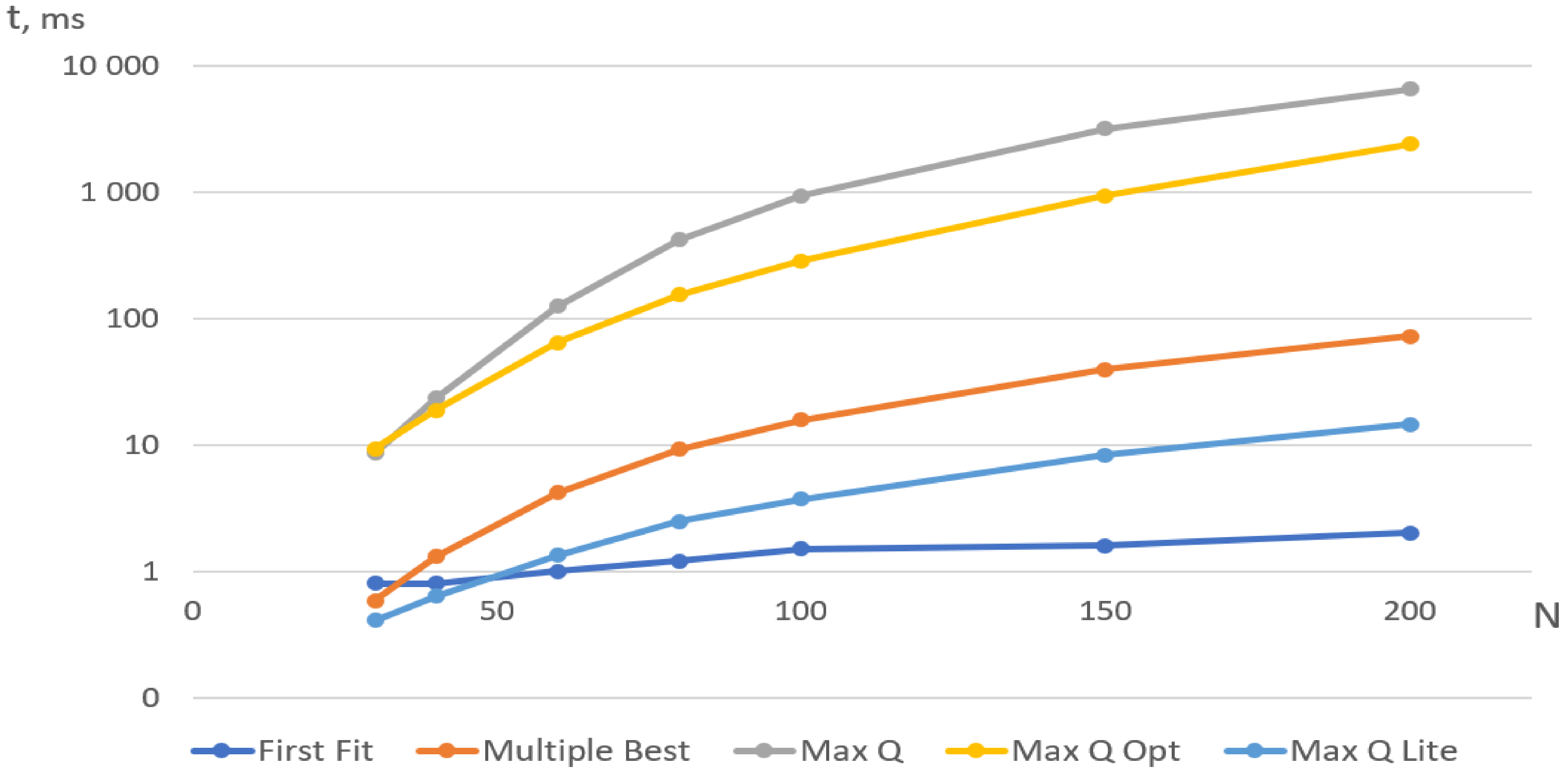


Simulation Study: Custom Criterion

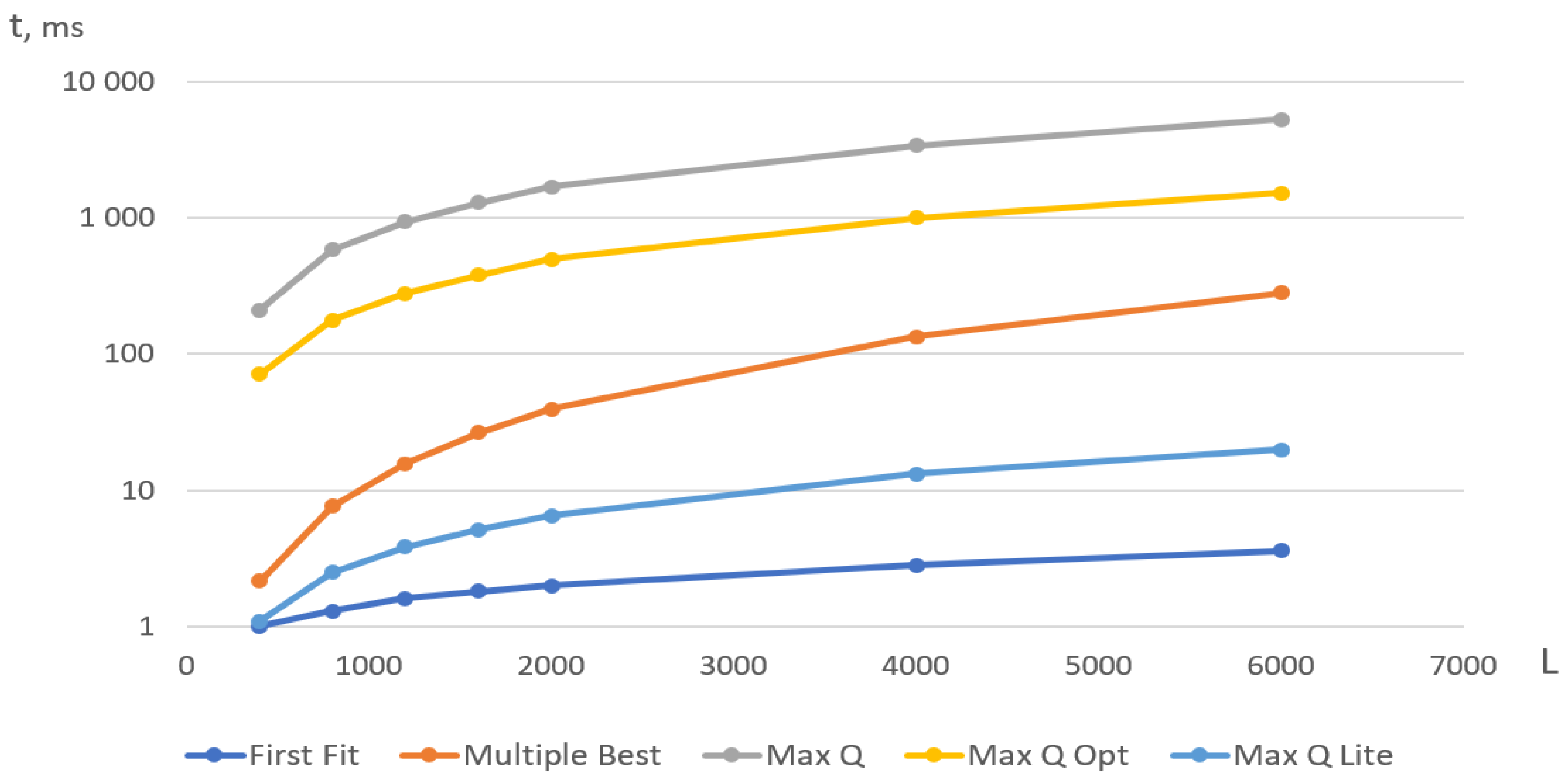
Q = sum of *qi*, *i* = 1,…,*n*, *qi* in [0;10], *Q* in [0;70]



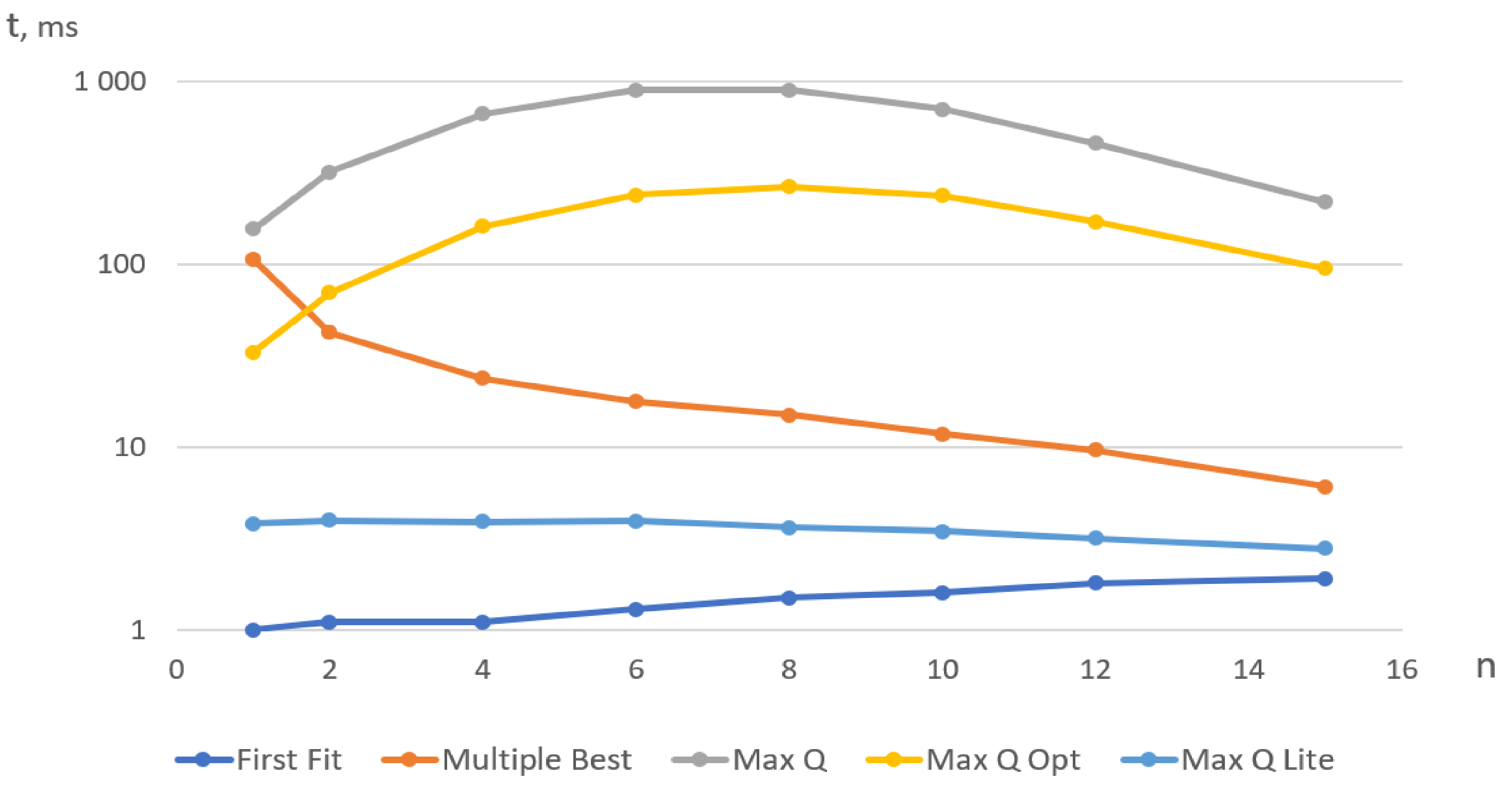
Running Time and Complexity $O(m * n * C)$



Running Time Depending on the Scheduling Interval Length *L*



Running Time Depending on a Number *n* of Nodes Required for a Job Execution



Conclusions and Further Work

- The general square window search algorithm with the special slots subset allocation procedure is proposed for the resources co-allocation problem
- Simulation study proved 44% advantage over the First Fit algorithm and 18% over the MultipleBest optimization heuristic
- As a drawback, the general case algorithm requires 600 times longer realization time compared to FirstFit
- In our further work, we will be focused on a practical resources allocation tasks implementation based on the proposed general case approach

Acknowledgments

This work was partially supported by:

- Ministry on Education and Science of the Russian Federation (project no. 2.9606.2017/8.9)
- Council on Grants of the President of the Russian Federation for State Support of Young Scientists (grant YPhD-2297.2017.9)
- RFBR (grants 18-07-00456 and 18-07-00534)

References

- https://github.com/dmieter/mimapr

CloudOfferFinder

SquareWindowFinder

- https://srmpds.github.io/

International Workshop on SRMPDS (Scheduling and Resource Management for Parallel and Distributed Systems) 2018, University of Oregon, USA

- https://www.youtube.com/watch?v=h6KfBKdVHOU

Victor Toporkov and Dmitry Yemelyanov. Resources Co-Allocation Optimization Algorithms for Distributed Computing Environments

Thank You!