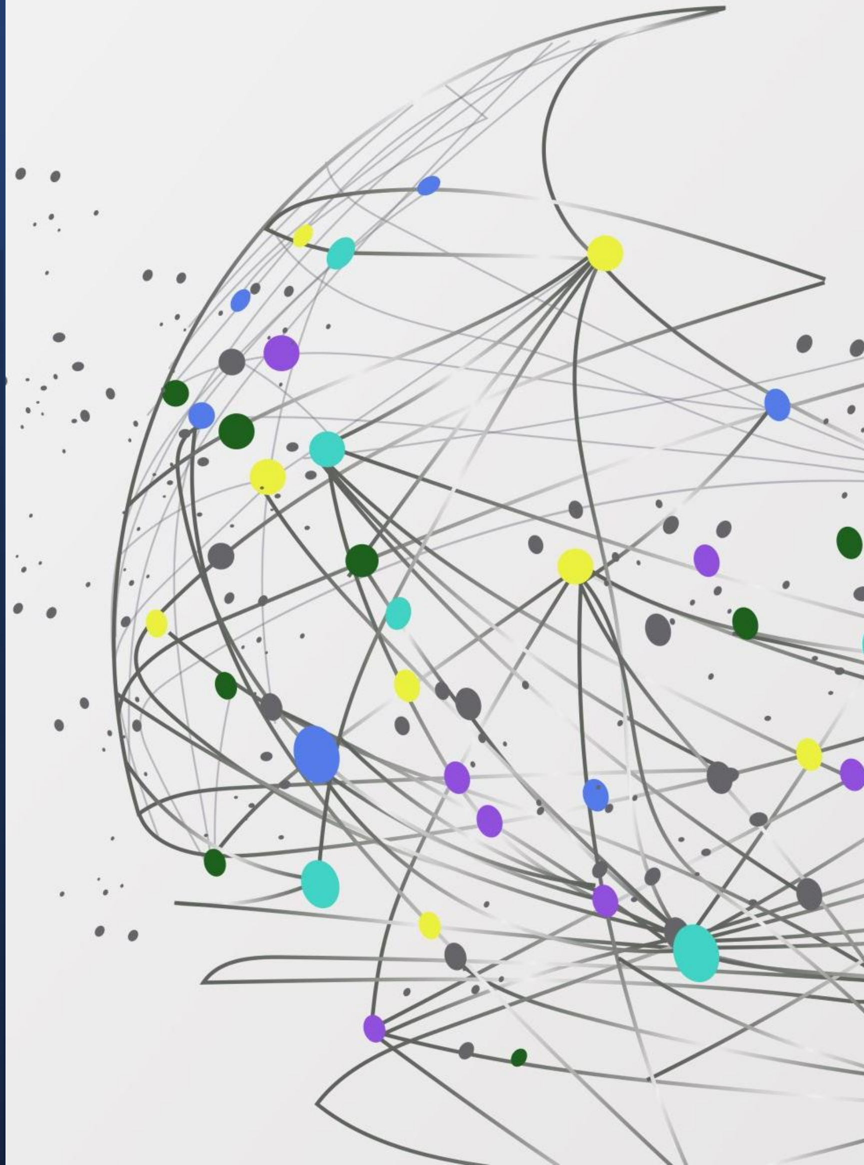
Victor TOPORKOV (National Research University “MPEI”, Russia)

Strategies for Multidisciplinary Workflows Scheduling and Resources Management in Cloud Computing



Outline

Part I. Workflow Scheduling and Resource Management for Running Knowledge-intensive Applications. State of the Art

Part II. Workflows Scheduling

Part III. Multifactor Strategies for Assigning Virtual Resources

Part I. Workflow Scheduling and Resource Management for Running Knowledge-intensive Applications. State of the Art

Clouds and Workflow Scheduling

Cloud technologies are actively used to perform scientific workflows

Infrastructure as a Service (IaaS) enables a Workflow Management System (WMS) to access a virtually unlimited pool of virtualized resources on a “pay-per-use” basis

Problems of heterogeneous works scheduling, the solution of which critically affects the efficiency of resource use in cloud computing

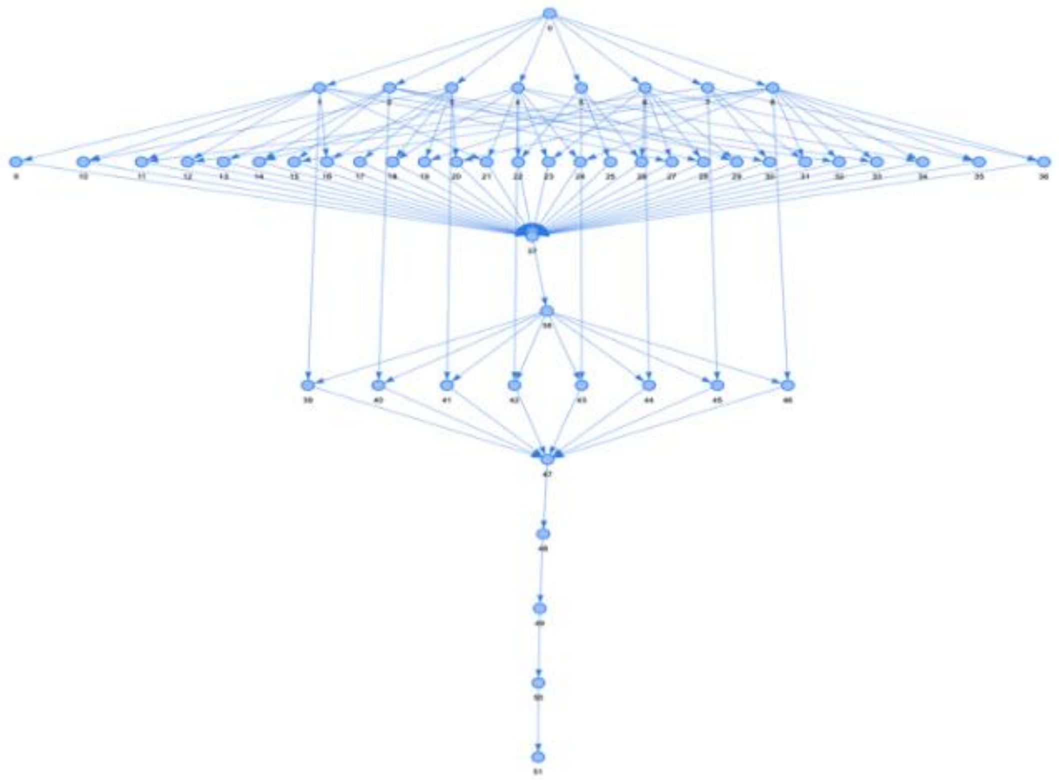
The Workflow as a Service Paradigm - WaaS

- **WaaS - multi-tenant environments that integrate computing, networking, and data storage resources provided by IaaS providers**

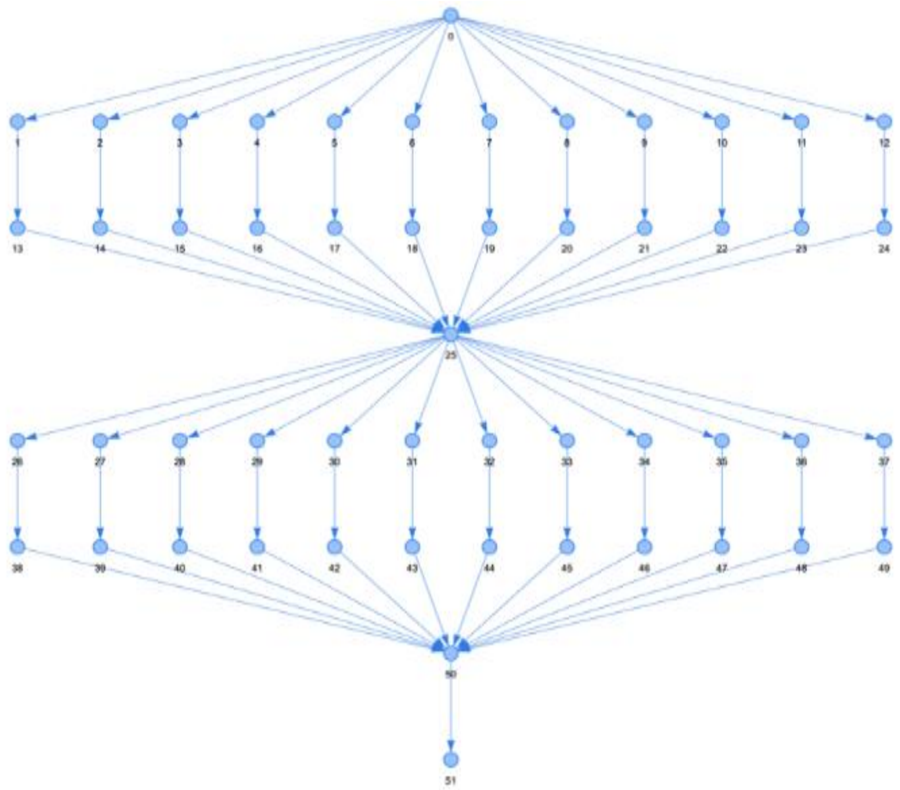
- **Scheduling within a single workflow while maintaining appropriate QoS requirements**

- **The WaaS paradigm allows solving the problem of scheduling for a set of independent jobs**

Examples of DAGs with 50 Nodes

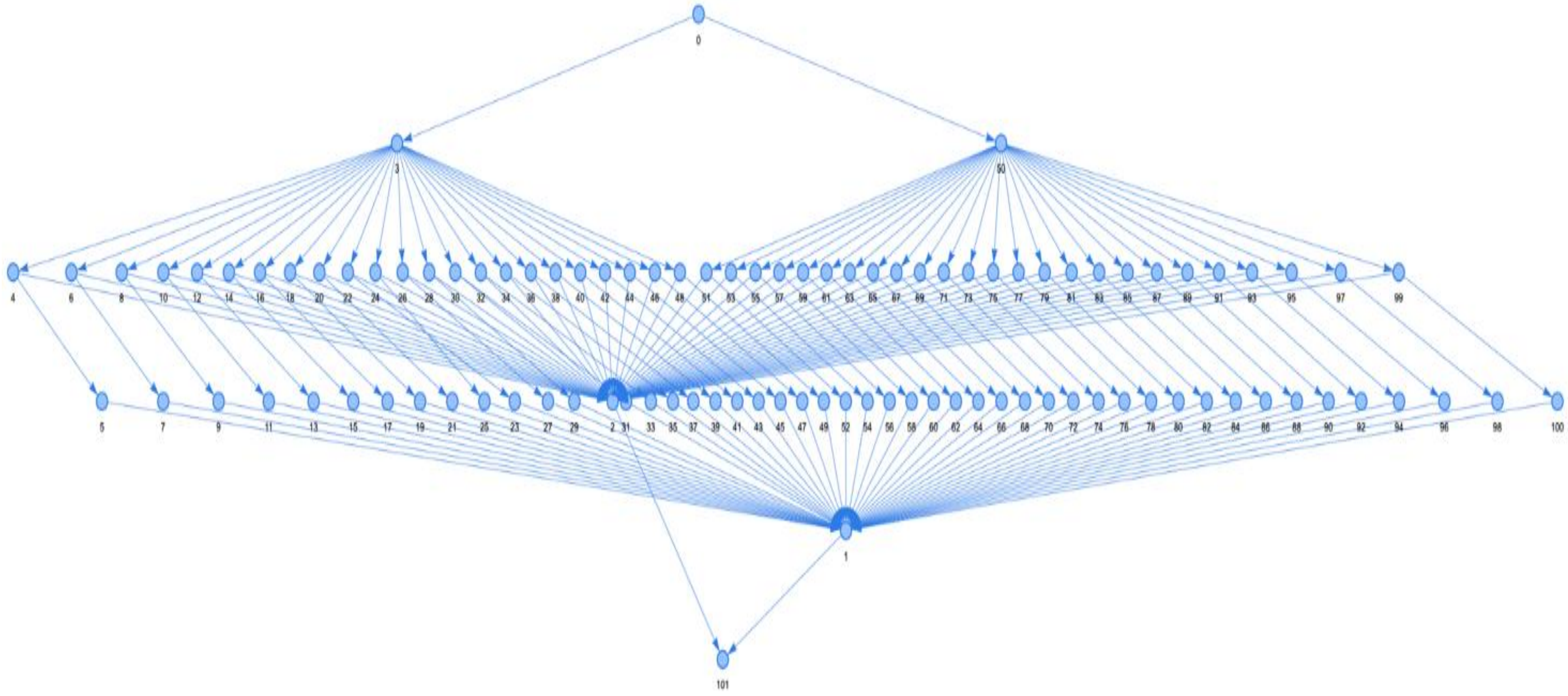


Montage (astronomy)

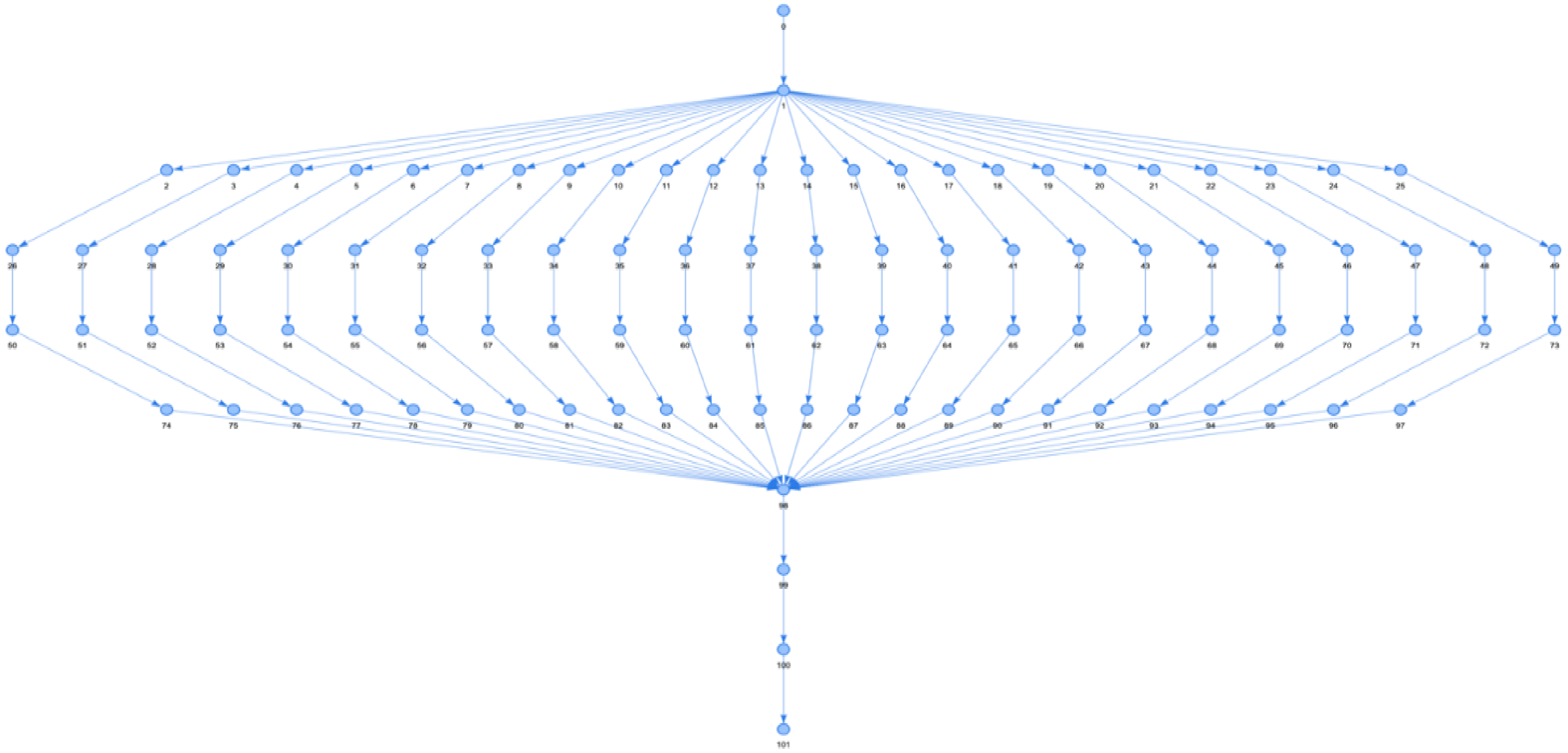


LIGO (gravitational wave physics)

CyberShake (seismology) DAG with 100 Nodes

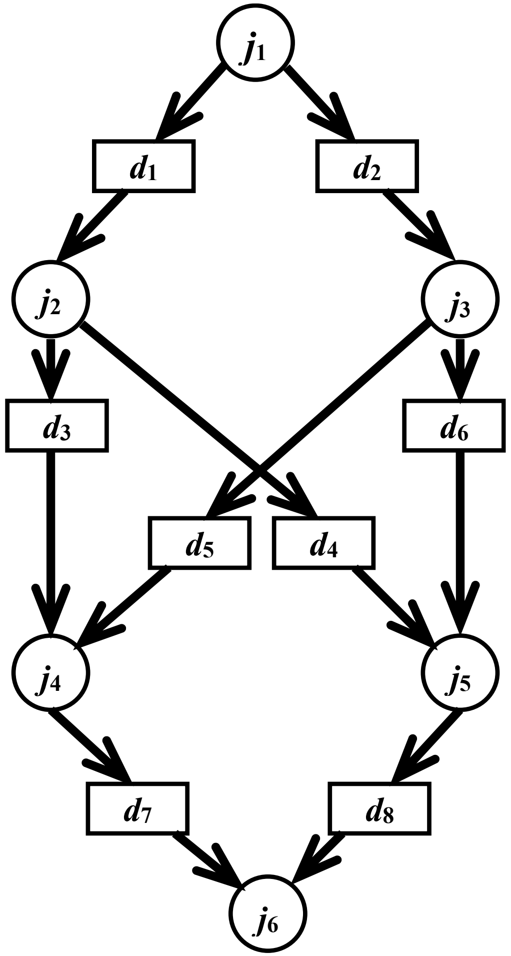


Genome, 1990-2003 (bioinformatics) DAG with 100 Nodes



DAG Parameterization:

examples for tasks $j_1, \ldots, j_6$



Parameters	j_1	j_2	j_3	j_4	j_5	j_6
t_{i1}^0	2	3	1	2	1	2
t_{i2}^0	4	6	2	4	2	4
t_{i3}^0	6	9	3	6	3	6
t_{i4}^0	8	12	4	8	4	8
v_{ik}	20	30	10	20	10	20

Workflow Management Systems

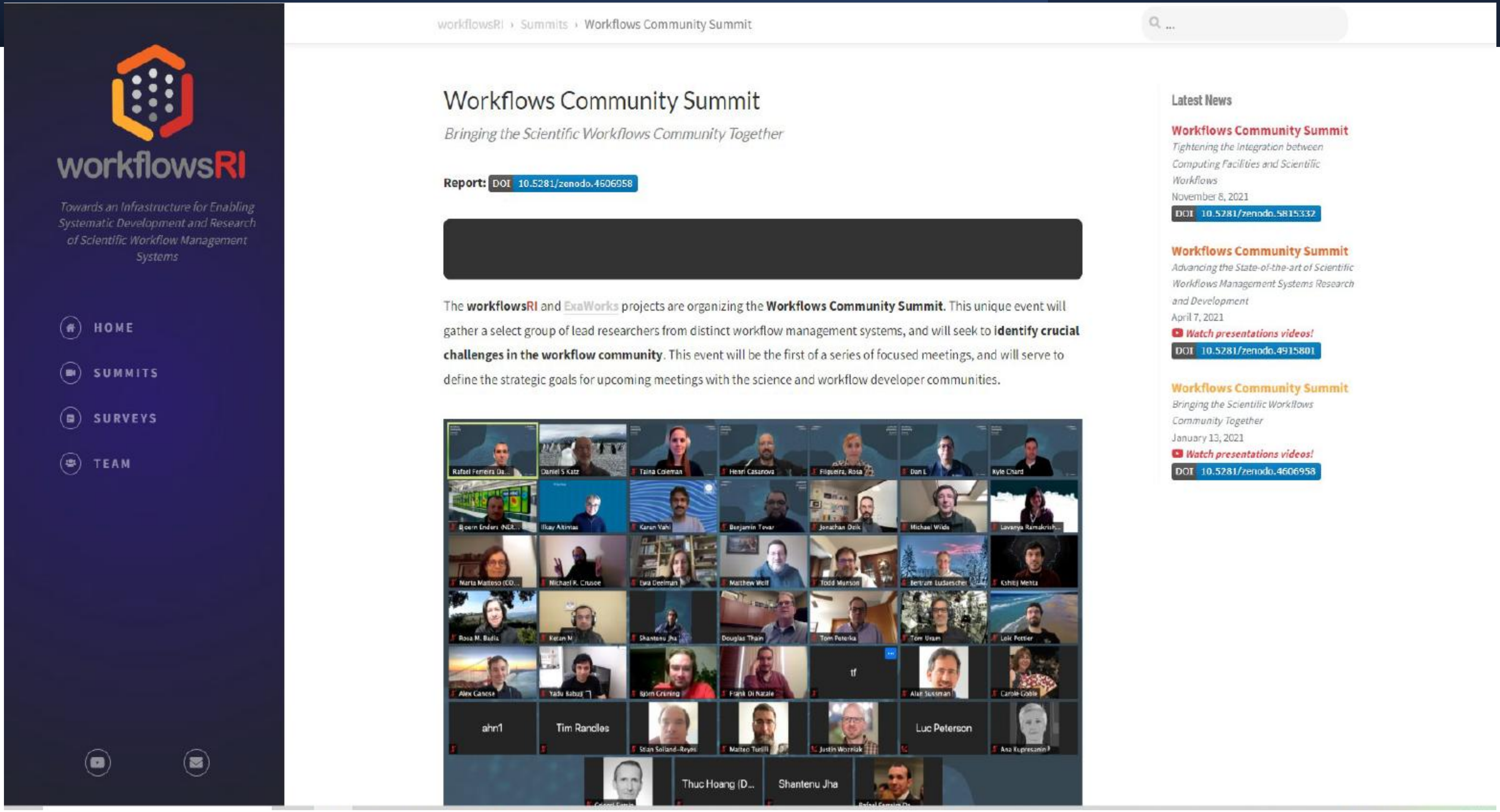
➢ **A huge number of workflow management systems (WMS):**

ASKALON, Galaxy, HyperFlow, Kepler, Pegasus, Taverna, CloudBus and a number of others

(Peter Amstutz, Maxim Mikheev, Michael R. Crusoe, Nebojša Tijanić, Samuel Lampa, et al.

Existing Workflow systems. *Common Workflow Language wiki*, GitHub. https://s.apache.org/existing-workflow-systems) – analysis of more than 360 WMS

https://workflowsri.org/summits/community/



Ewa Deelman · published a preprint
A Terminology for Scientific Workflow Systems

- **Frederic Sutera, Taina Colemanb, Ilkay Altintaș b, Rosa M. Badiac, Bartosz Balisd, Kyle Charde, Iacopo Colonnellif, Ewa Deelmang, Paolo Di Tommasoh, Thomas Fahringeri, Carole Goblej, Shantenu Jhak, Daniel S. Katzl, Johannes K˙osterm,Ulf Leser, Kshitij Mehtaa, Hilary Olivero, J.-Luc Petersonp, Giovanni Pizziq, Lo¨ıc Pottierp, Ra¨ul Sirventc, Eric Suchytaa,Douglas Thainr, Sean R. Wilkinsona,**
- **M. Wozniaks, Rafael Ferreira da Silva**
- **Available from:**

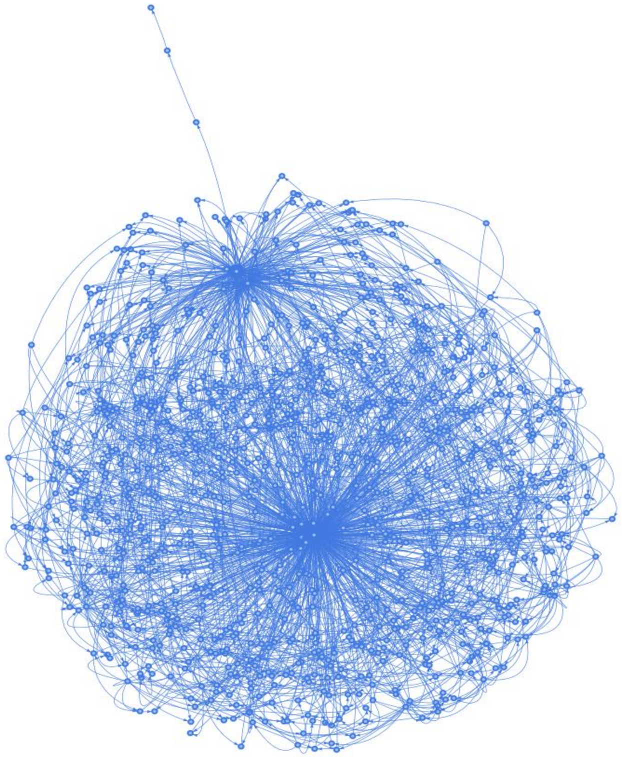
https://www.researchgate.net/publication/392530352_

[accessed June 13 2025]

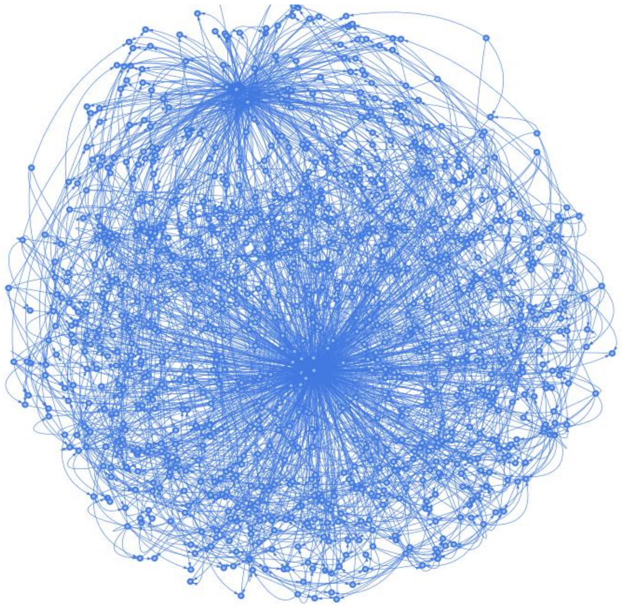
Challenges and Open Questions

- **The presence of multiple IaaS providers and different types of resources**
- **Geographic distribution of data centers**
- **Heterogeneity of workflows entering the WaaS platform**
- **The need to implement the “pay for use” principle for a specific user**
- **Solving the problem of placing virtual machines on physical servers and creating multiple containers in them, each of which can be used by tasks from different workflows**

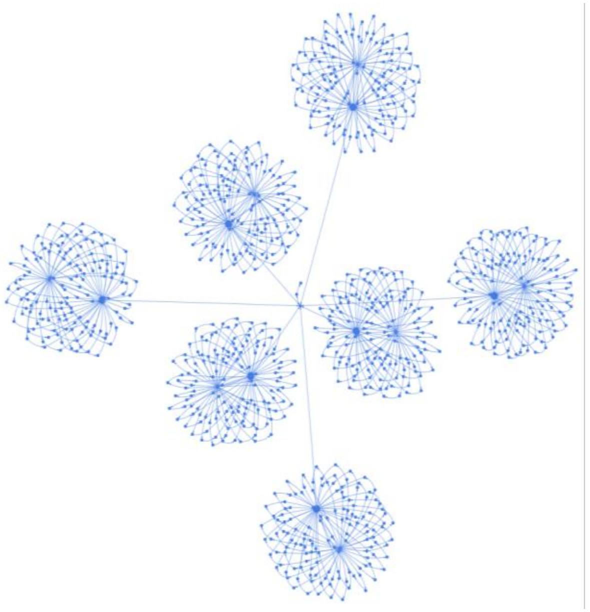
Examples of DAGs with 1000 Nodes



Montage



CyberShake



Epigenomics

Challenges and Open Questions

- **One of the challenges is the workflow model, formalized as a DAG.**
- **In a number of applications, loops are naturally present in workflows.**
- **Palliative techniques generate multiple instances of subflows (Pegasus, Apache, Airflow, Taverna, Kepler).**
- **Dynamic transformation of DAG during application execution, when linear sequences of tasks are generated – process chains.**

Scheduling and Managing Workflows on Cloud Platforms

- **Most of the known scheduling algorithms use the total cost as an optimization criterion given a constraint on the execution time of the workflow: IC-PCP, IC-PCPD2, EIPR, TB и CCA.**

- **Some composite scientific applications consist of interconnected works called ensembles. These algorithms take into account QoS requirements not for each flow, but for the ensemble as a whole. The number of flows in the ensemble is assumed to be known in advance.**

- **Workflows in an ensemble have the same type, that is, they have the same structure and differ only in the volume of calculations and input data.**

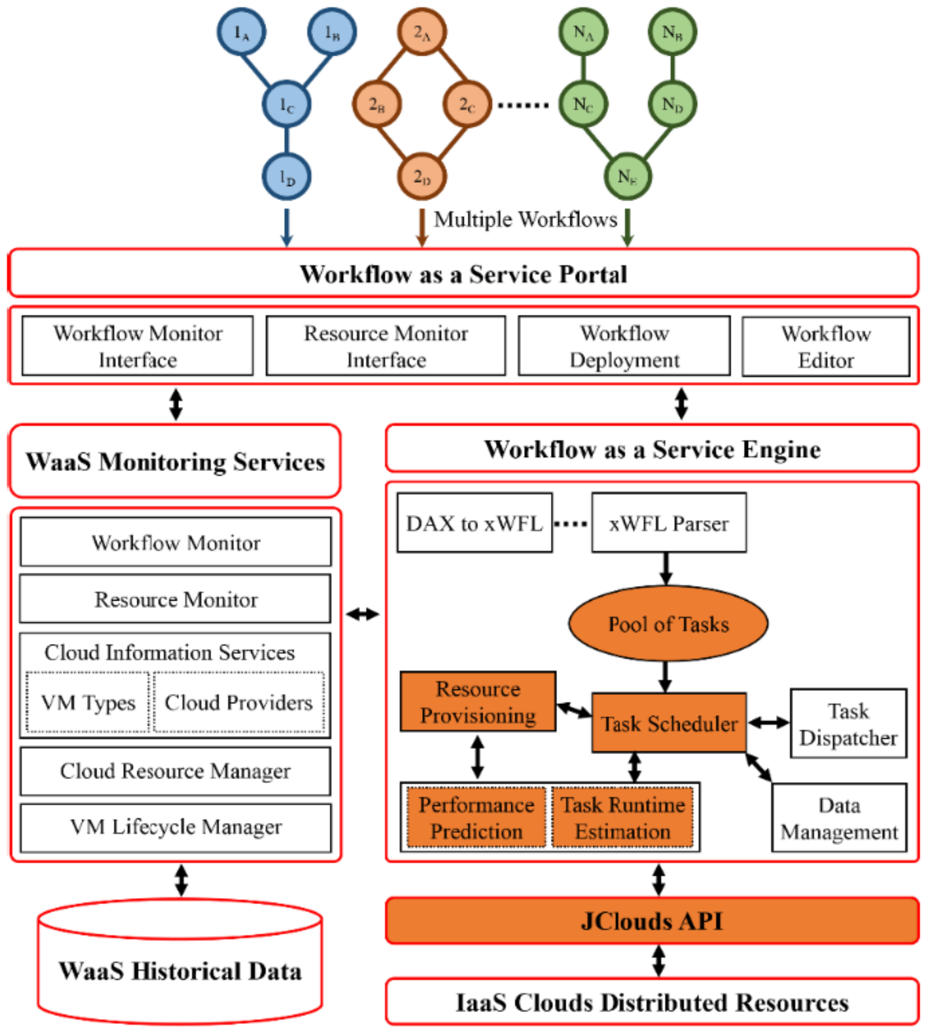
Scheduling and Managing Workflows on Cloud Platforms

- **Scheduling multiple workflows in cloud computing: the number and types of workflows are assumed to be known in advance, with all workflows arriving simultaneously.**

- **Scheduling for a finite set of heterogeneous VMs, the number of which remains unchanged throughout the life cycle of the system.**

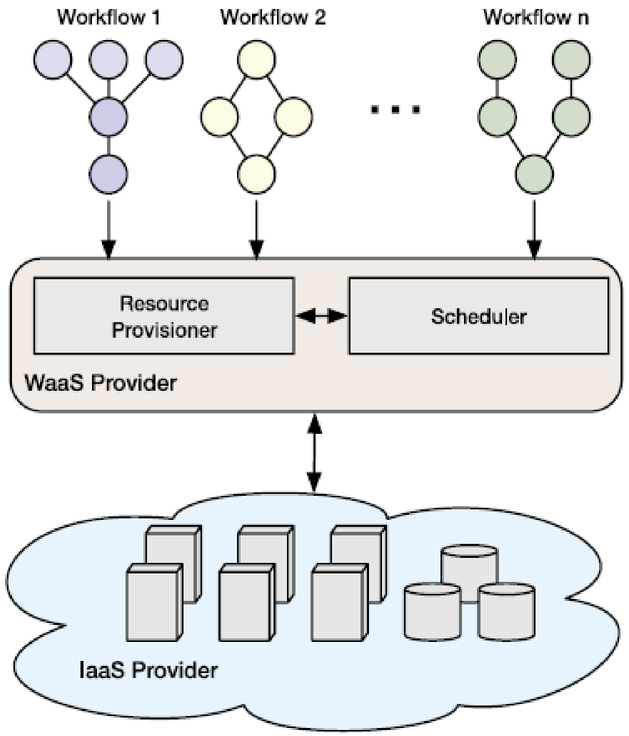
- **Delays in resource provision and data transfer time costs are not taken into account.**

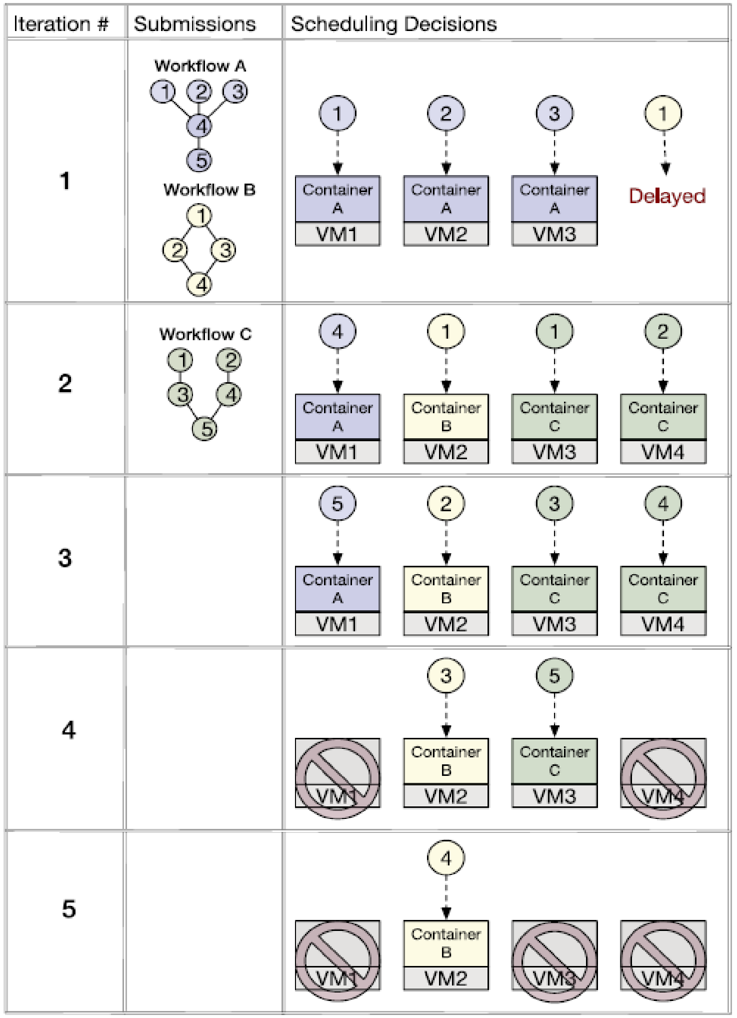
WaaS Platform Based on the Extension of CloudBus Functionality (EBPSM algorithm)



Muhammad Hilman, Rajkumar Buyya. Workflow-as-a-Service Cloud Platform and Deployment of Bioinformatics Workflow Applications. Preprint. June 2020. https://www.researchgate.net/publication/341899292

WaaS Architecture and EPSM Algorithm





Maria A. Rodriguez, Rajkumar Buyya. Scheduling dynamic workloads in multi-tenant scientific workflow as a service platforms // Future Generation Computer Systems 79 (2018) 739–750

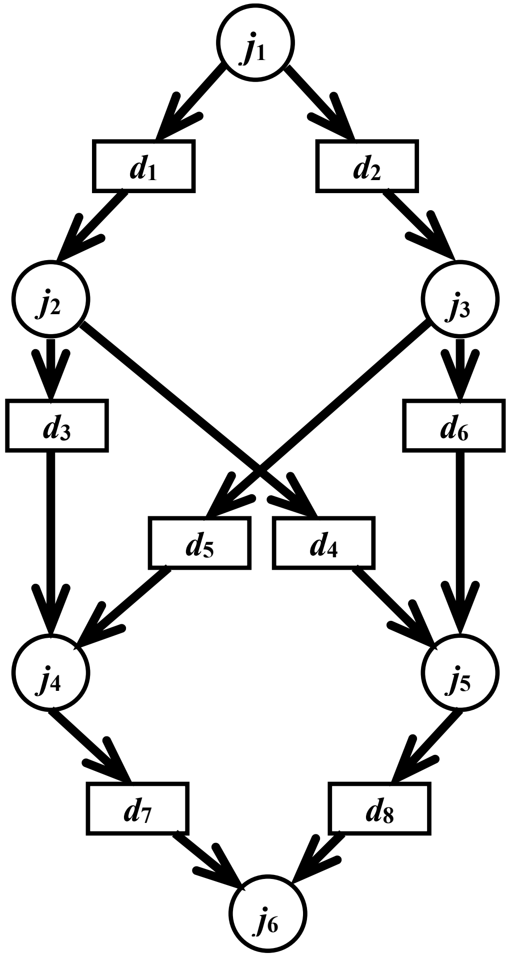
Motivation for Strategies of Management and Scheduling for WaaS Platform

DEVELOPMENT OF A COMPLEX OF MODELS, METHODS AND TOOLS FOR ORGANIZING CLOUD COMPUTING BASED ON A COMBINATION OF PRIORITY SCHEDULING ALGORITHMS FOR BOTH INDIVIDUAL TASKS IN WORKFLOWS AND INDEPENDENT AND DIFFERENT FLOWS OF COMPOSITE APPLICATIONS

DEVELOPMENT OF METHODS AND TOOLS FOR FORECASTING THE STATE OF RESOURCES OF THE WORKFLOW AS A SERVICE (WaaS) PLATFORM IN ORDER TO UPDATED SCHEDULING STRATEGIES

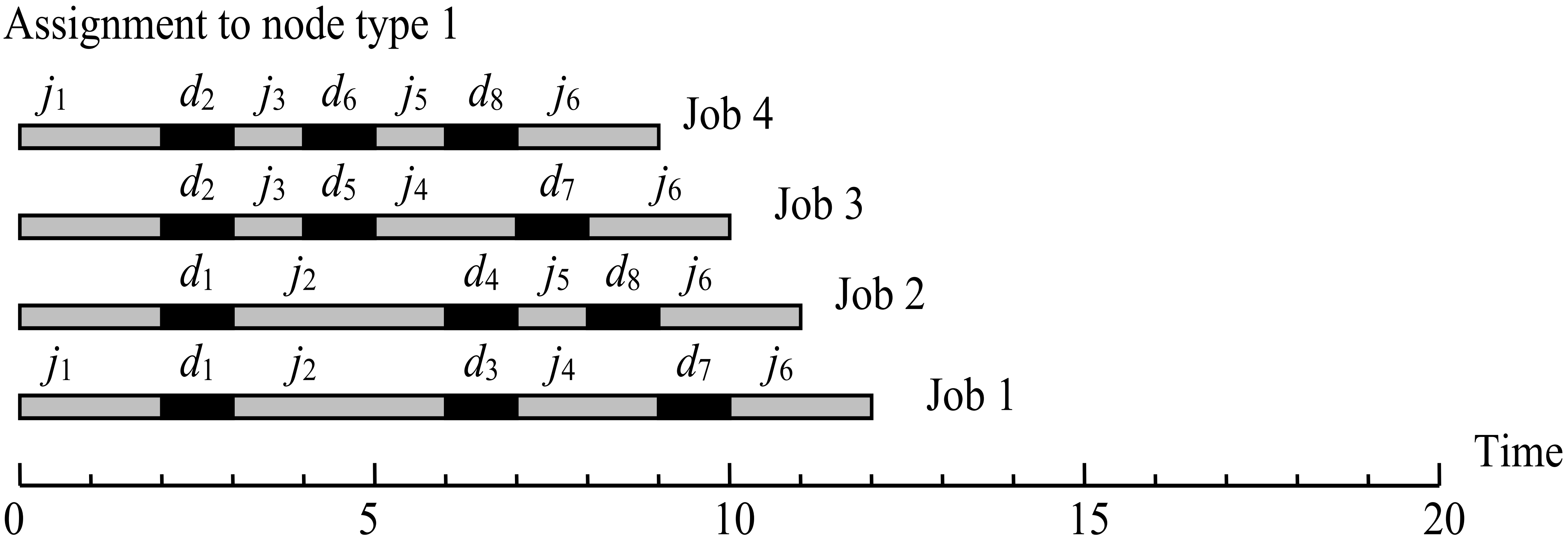
Part II. Workflows Scheduling

Critical Jobs Method: DAG Parameterization

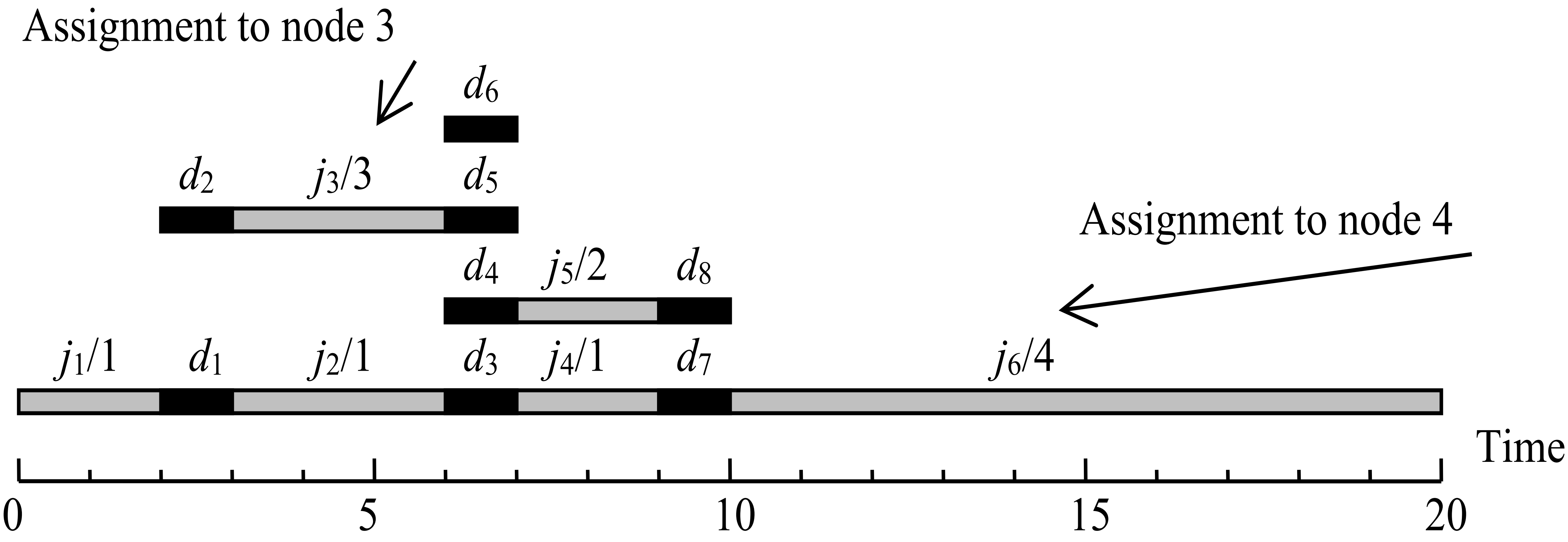


Parameters	j_1	j_2	j_3	j_4	j_5	j_6
t_{i1}^0	2	3	1	2	1	2
t_{i2}^0	4	6	2	4	2	4
t_{i3}^0	6	9	3	6	3	6
t_{i4}^0	8	12	4	8	4	8
v_{ik}	20	30	10	20	10	20

Critical Jobs Method: Ranking

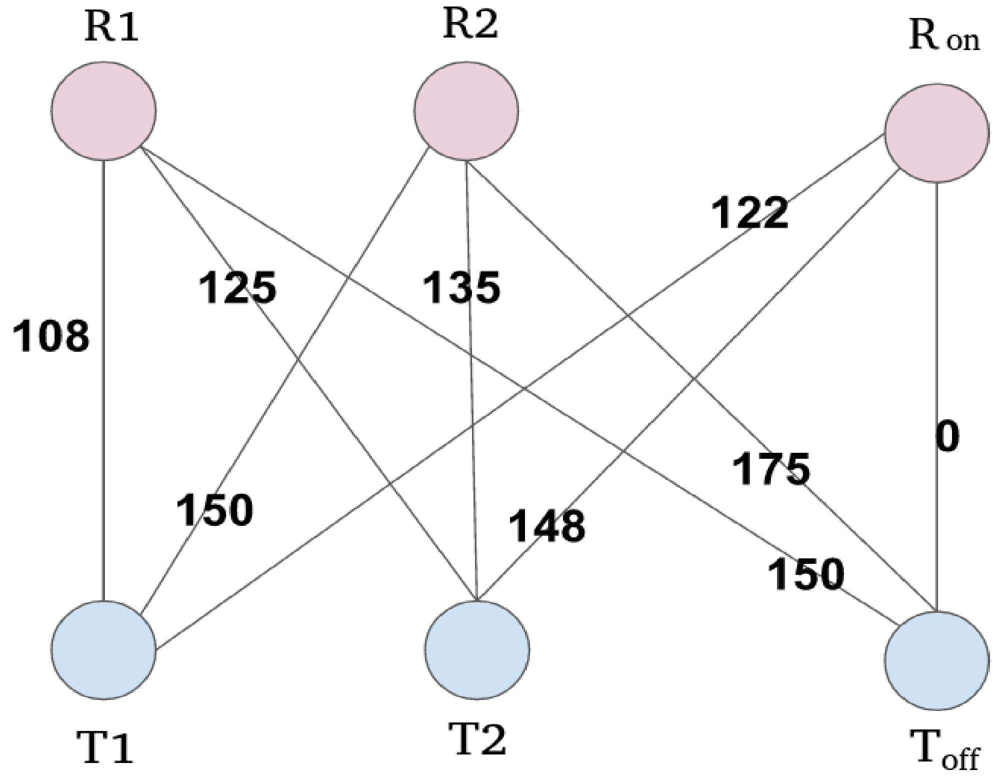


Critical Jobs Method: Result



Critical Jobs' Method Modification

- The conflict resolution process is not included in the modified CJM but is transferred to the stage of assigning tasks to specific instances of VMs.

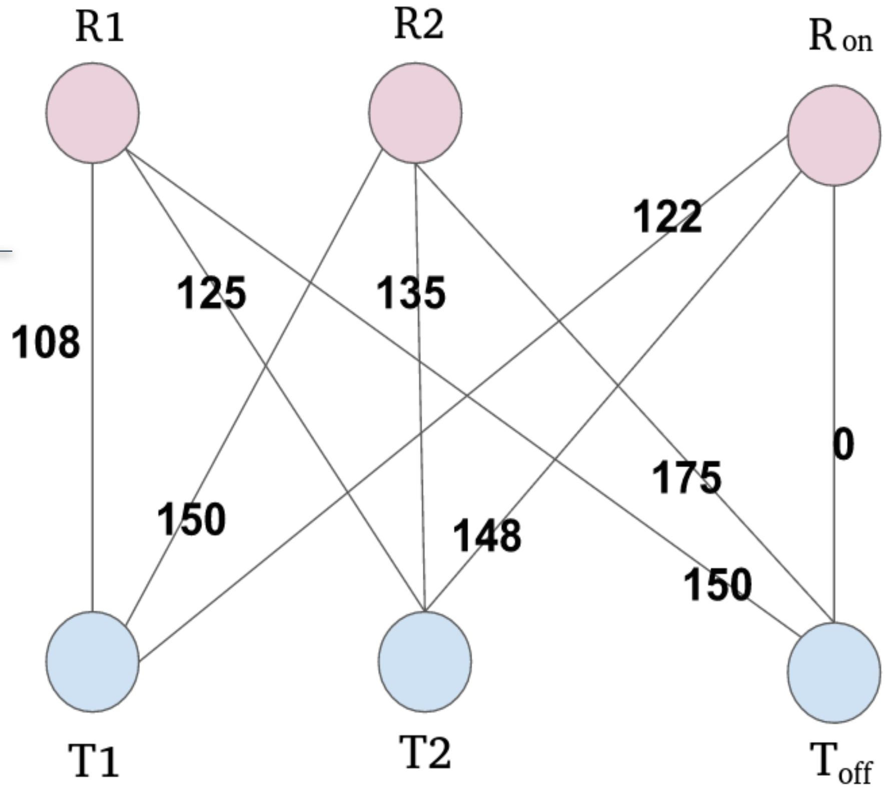


Search for a perfect matching in a bipartite graph *G*=(*T*, *R*, *E*), where *T* represents the set of tasks of batch *B*, *R* corresponds to the set of available resources, and *E* is the set of edges between *T* and *R*.

VM Allocation with Kuhn-Munkres Algorithm

An edge between a task from *T* and a resource from *R* means that the task can be executed on the corresponding VM while meeting all requirements.

The weight of an edge is the value of the efficiency criterion for a given assignment (for example, the total time required to complete a task, taking into account the time it takes to copy data, or the cost of such execution).



Complexity of the VMA Algorithm

- The number of vertices in the graph $G \sim C * N_B$, where C is a constant, N_B is the number of tasks in the parallel execution batch.

- The overall computational complexity of VMA algorithm in this case is

$$O\left(N_B^3\right)$$

- This cubic complexity refers to the number of tasks N_B in the parallel execution batch, not to the total number of tasks in the incoming workflows.

Job Resource Request

The resource requirements are arranged into a resource request containing:

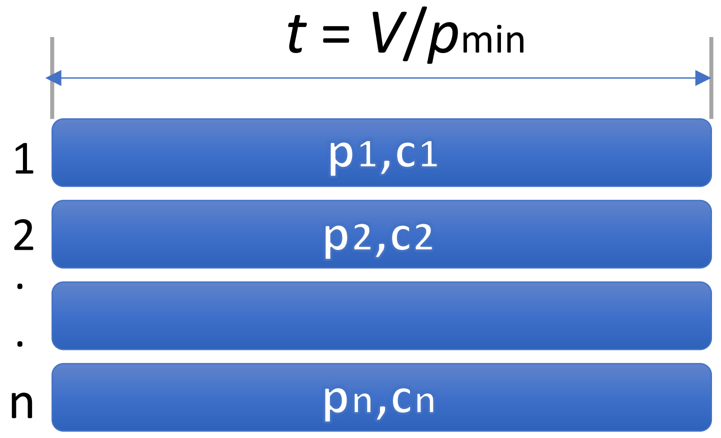
n - number of required VMs

p_{min} - minimal performance requirement for each VM (MIPS), RAM (GB), storage capacity (GB), network bandwidth (GB/s) etc.

V – computational volume for a single task (MI)

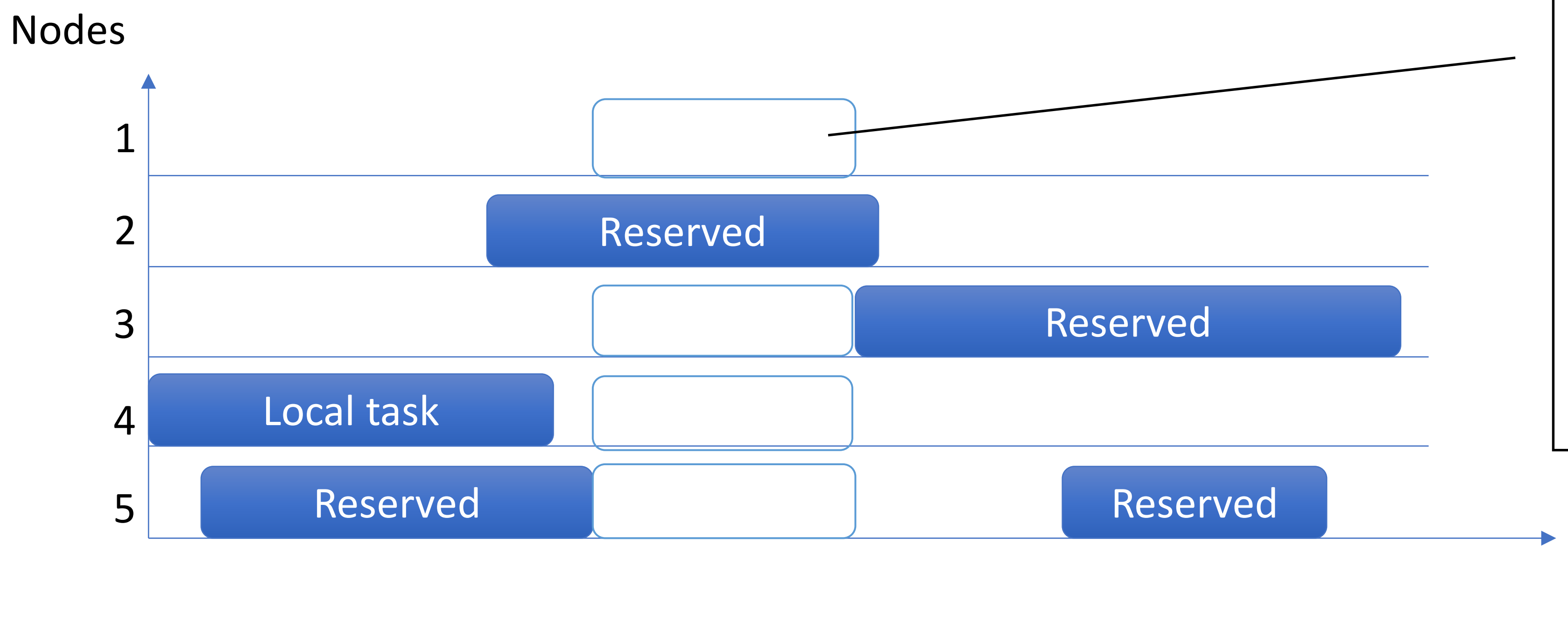
C - maximum total job execution cost (budget)

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:



Slot:

- Node
- Performance
- Cost
- Start Time
- Finish Time

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;

}

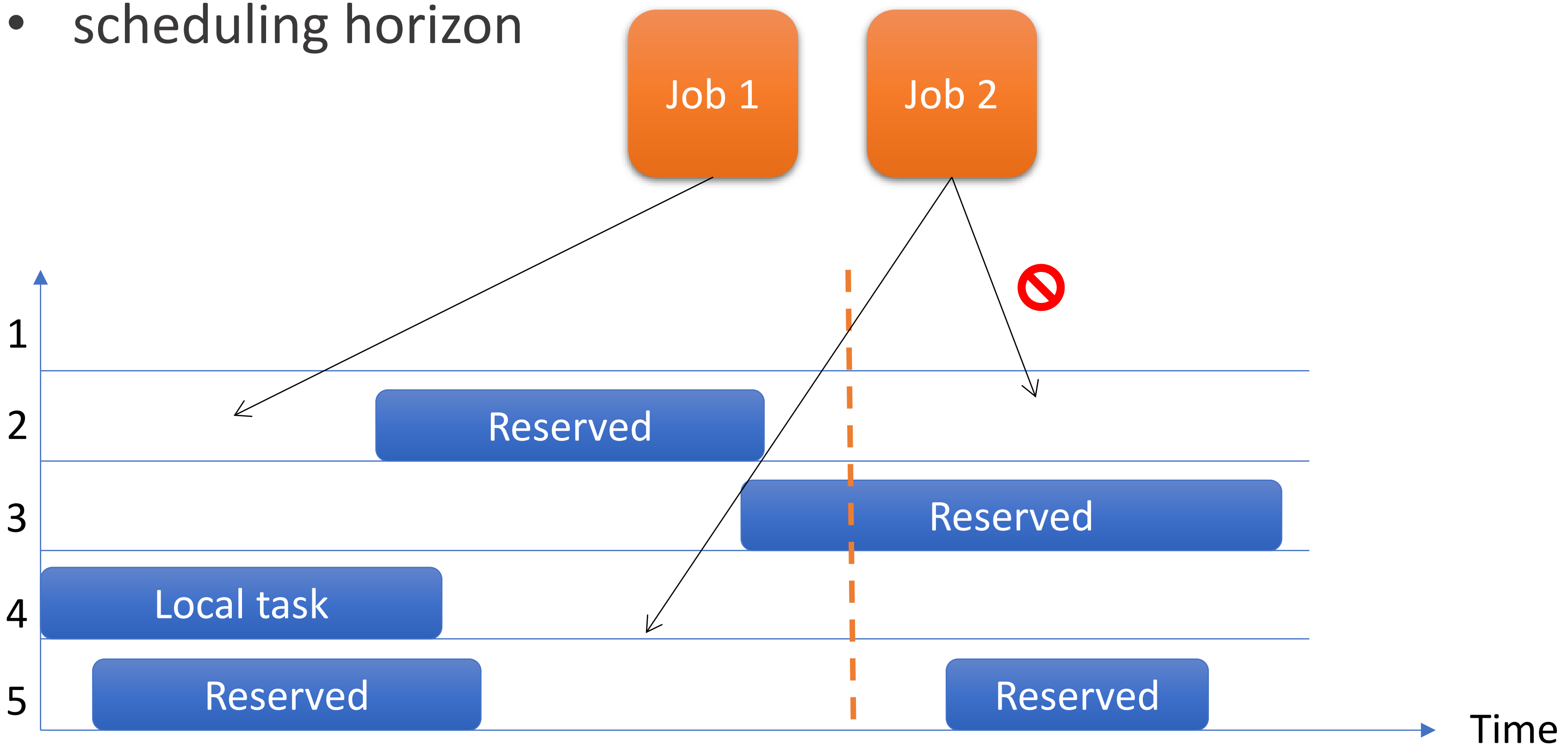
}

return the best of the found interim windows;

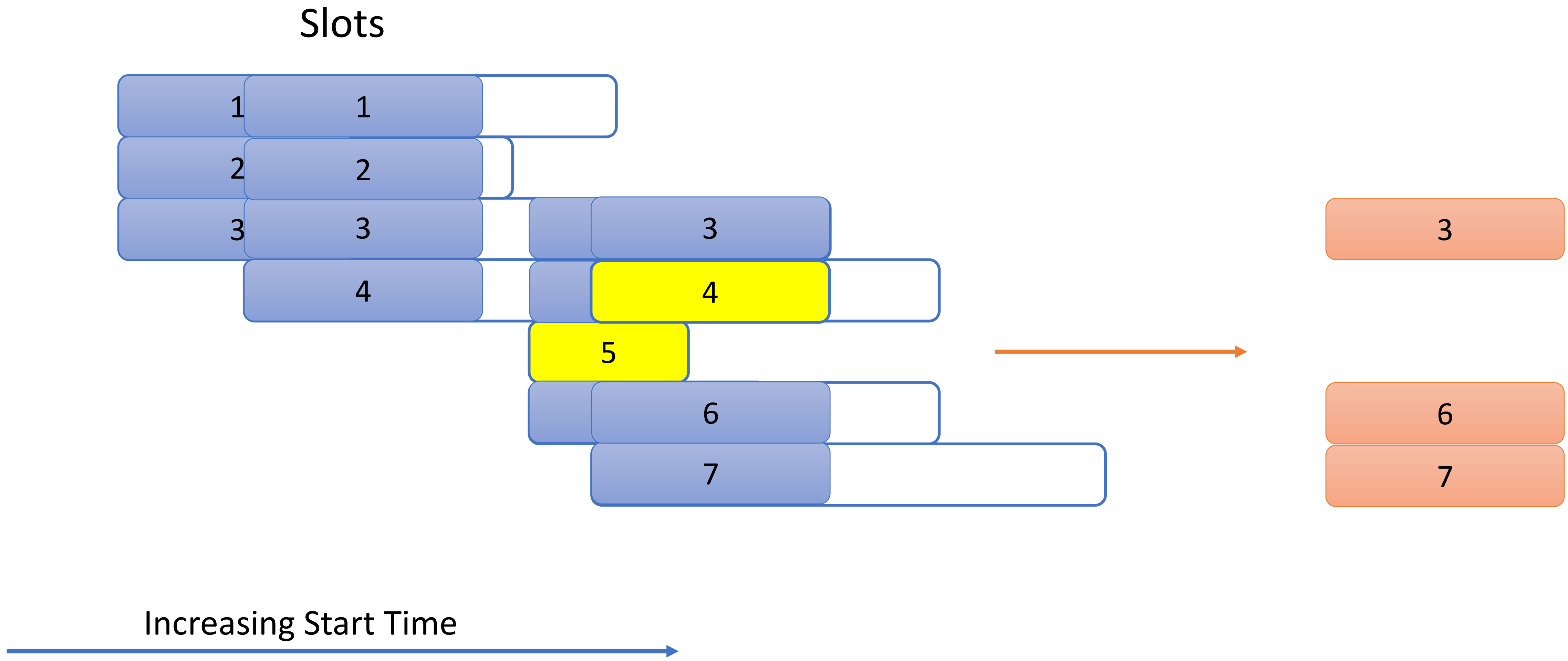
Deadline and Scheduling Horizon

There is a practical limit on the slots availability:

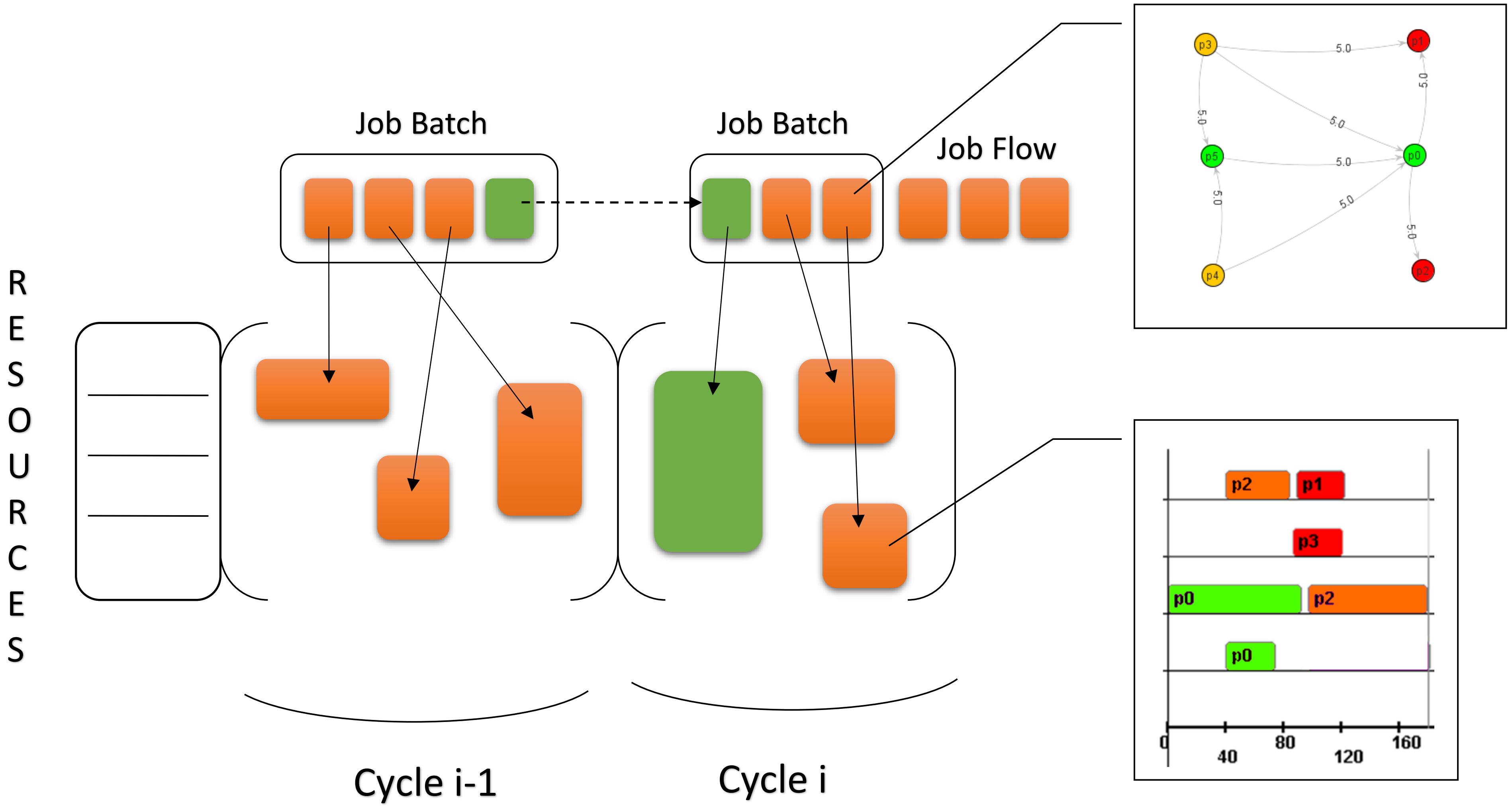
- deadline
- backfilling
- scheduling horizon



Window Search Scheme Visualized



Cyclic Batch Scheduling Scheme



Job Batch Execution Schedule

Set of selected slots, ***batch execution schedule*** – combination of slots

Slot attributes: VM technical characteristics (processor type, network bandwidth), duration, cost of use

The set of available slots is known at the beginning of each scheduling cycle based on the occupancy forecast and the availability of VM containers suitable for tasks

Formation of a Pool of VMs from IaaS Providers

Online scheduling and the knapsack filling problem

Maximizing the overall performance of a resource pool subject to a constraint on the overall cost C_j is represented by the optimization problem below:

$$Z = \sum_{i=1}^{m} z_i x_i \to \max ,$$

$$\sum_{i=1}^{m} c_i x_i \le C_j,$$

$$\sum_{i=1}^{m} x_i = n,$$

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

where z_i is the target value of the characteristic provided by resource i, c_i is the cost of its use, x_i is a variable that determines whether to select resource i ($x_i = 1$) or not ($x_i = 0$).

Number *n* of allocated VMs is not limited: $n \in [0; m]$.

Interval Problem

For the period T, allocate a set of $n \in [n_{min}; n_{max}]$ simultaneously available resources that satisfy the constraints on individual characteristics (type of operating system, minimum VM performance, RAM capacity, etc.) and the general cost constraint C:

$$Z = \sum_{i=1}^{m} z_i x_i \to \max ,$$

$$\sum_{i=1}^{m} c_i x_i \leq C,$$

$$\sum_{i=1}^{m} x_i \geq n_{\min},$$

$$\sum_{i=1}^{m} x_i \leq n_{\max},$$

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

Solution of the Interval Problem

Modification of the 0-1 knapsack problem and application of the dynamic programming scheme:

$$f_i(c,k) = \max\{f_{i-1}(c,k), f_{i-1}(c-c_i, k-1) + z_i\},$$

$$i = 1,..,m, c = 1,..,C_j, k = 1,..,n_{\max},$$

where $f_i(c,k)$ defines the maximum value of criterion Z for pool k of resources allocated from the first i available VMs with budget c.

During the backward induction procedure, the maximum value of the objective criterion is determined as

$$Z_{\max} = \max_n f_m(C,n), n \in [n_{\min}; n_{\max}].$$

The corresponding resulting number n_a of allocated VMs is $n_a = \arg\max_n f_m(C,n), n \in [n_{\min}; n_{\max}]$.

Computational Complexity of the Interval Algorithm

The computational complexity of the interval algorithm according is equal to

$$\boldsymbol{O(m * n_{\max} * C)}$$

Additional calculations associated with the selection of the best solution

$$\boldsymbol{Z_{\max} = \max_{n} f_{m}(C, n)}$$

in accordance with the inequality $\boldsymbol{n_{\max} - n_{\min} \leq n}$, introduce complexity $\boldsymbol{O(n)}$

Workflow Execution Schedule

- ***Execution time***:

- the actual processing time as the ratio of the volume of calculations (in millions of instructions) to the processor performance (million instructions per second) on a VM processor of the corresponding type;

- time for data exchange between workflow tasks (reading and writing to global storage, such as Amazon S3);

- time to deploy a VM and a container in a VM of the corresponding type.

- ***The cost of executing a flow task*** **on a VM of a given type is defined as the ratio of execution time to the billing period of the deployed VMs, multiplied by the cost of one billing period.**
- ***The cost of completing a workflow*** **is the sum of the costs of completing each of the tasks.**

Key Cost Assumptions

If a task can be completed before the next billing period, then the cost of its execution on previously deployed VMs and created containers is assumed to be zero.

If a task is not completed in the current billing period, then the time required to complete it in subsequent periods is paid.

If any VMs are not in demand in the current scheduling cycle, they are terminated.

Part III. Multifactor Strategies for Assigning Virtual Resources

Virtual Resource Assignment Strategies

Greedy strategy: Create a new specialized VM to perform each individual task and stop it when it is finished.

Control strategy for monitoring a dynamically changing pool of constantly active VMs and distributing ready-to-run tasks among them.

A class of mixed strategies where some basic minimum pool of active VMs is maintained when executing workflows, but additional VMs can be created to execute individual tasks, such as those that are not time-consuming to load and save data.

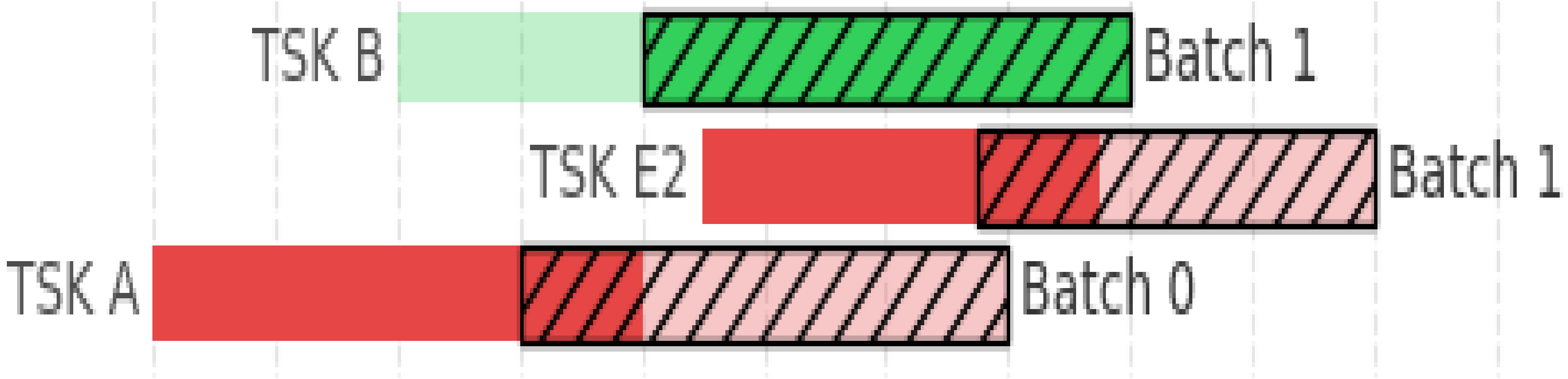
Optimizing Task Assignment in a Parallel Execution Batch

- Batch B contains N_B tasks to be executed in parallel on different resources.
- The main characteristics of the tasks are known in advance: computational volume, input data requirements, deadline.
- The resource pool contains N_{vm} active virtual machines – candidates for running tasks in the considered execution cycle of batch *B*. In general, $N_{vm} \neq N_B$, and furthermore, not all virtual machines may be suitable for executing the tasks in *B*, especially given the need to meet the deadline.
- To fulfill the necessary requirements of the current task batch *B*, or, on the contrary, to save computing resources, the task provides the ability to create new or stop active virtual machines.

FTL Algorithm Scheme

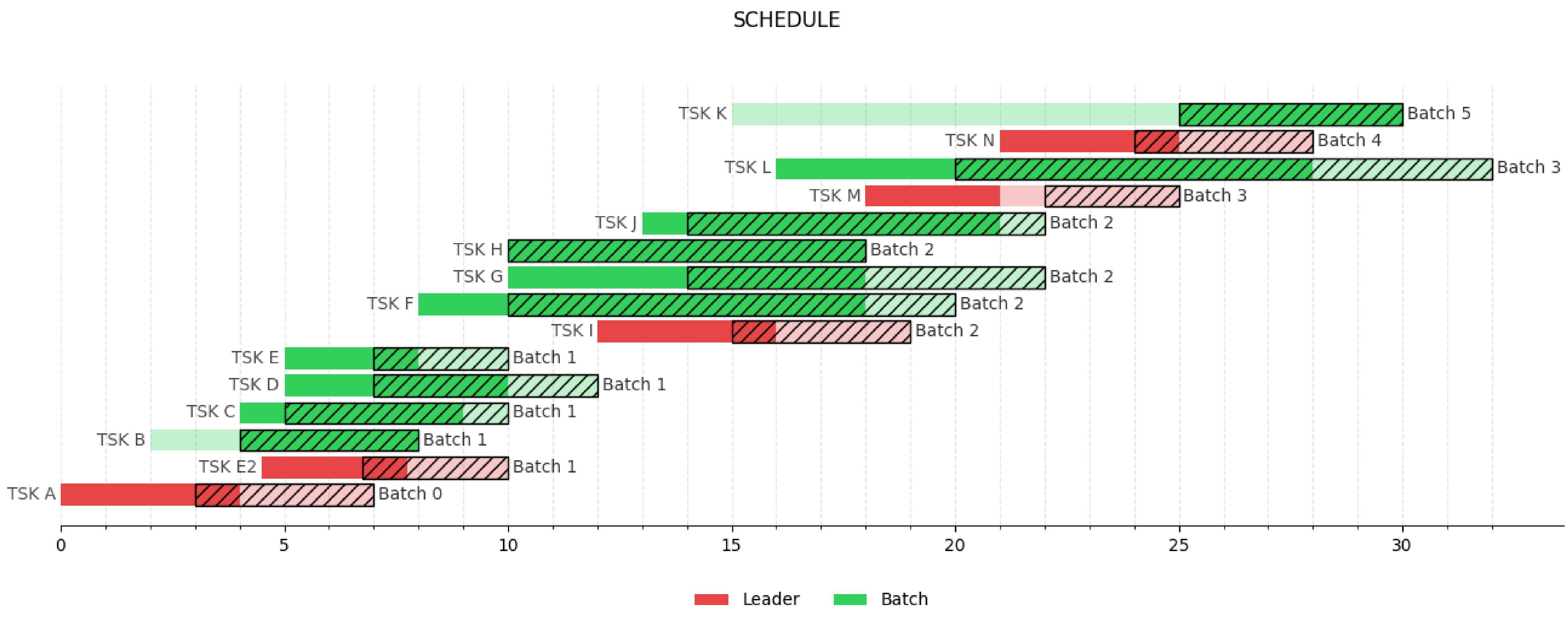
- 1. Select the virtual machine type Vm_t with the highest performance (select the leader).
- 2. For each task, calculate the earliest completion time and the latest start time, taking into account the predicted execution time on the leader virtual machine (type Vm_t) .
- 3. Select the task with the smallest earliest completion time $\min t_f^i$ and assign it to the leader. This task is placed in a new batch for parallel processing.
- 4. All tasks with late start time $\max t_s^i$ less than $\min t_f^i$ are placed in the parallel processing batch from step (3). No two tasks from this batch can be executed sequentially on the same virtual machine. This batch defines the minimum degree of parallelism of a task queue over a period of time.
- 5. For the remaining tasks, recalculate the previously completed time relative to the time $\min t_f^i$, starting from which the leader can continue completing tasks.
- 6. If there are tasks left in the queue that have not been added to parallel processing packages, go to step 3.
- Otherwise, the end of the algorithm.

Part of a Gantt Chart for Three Tasks

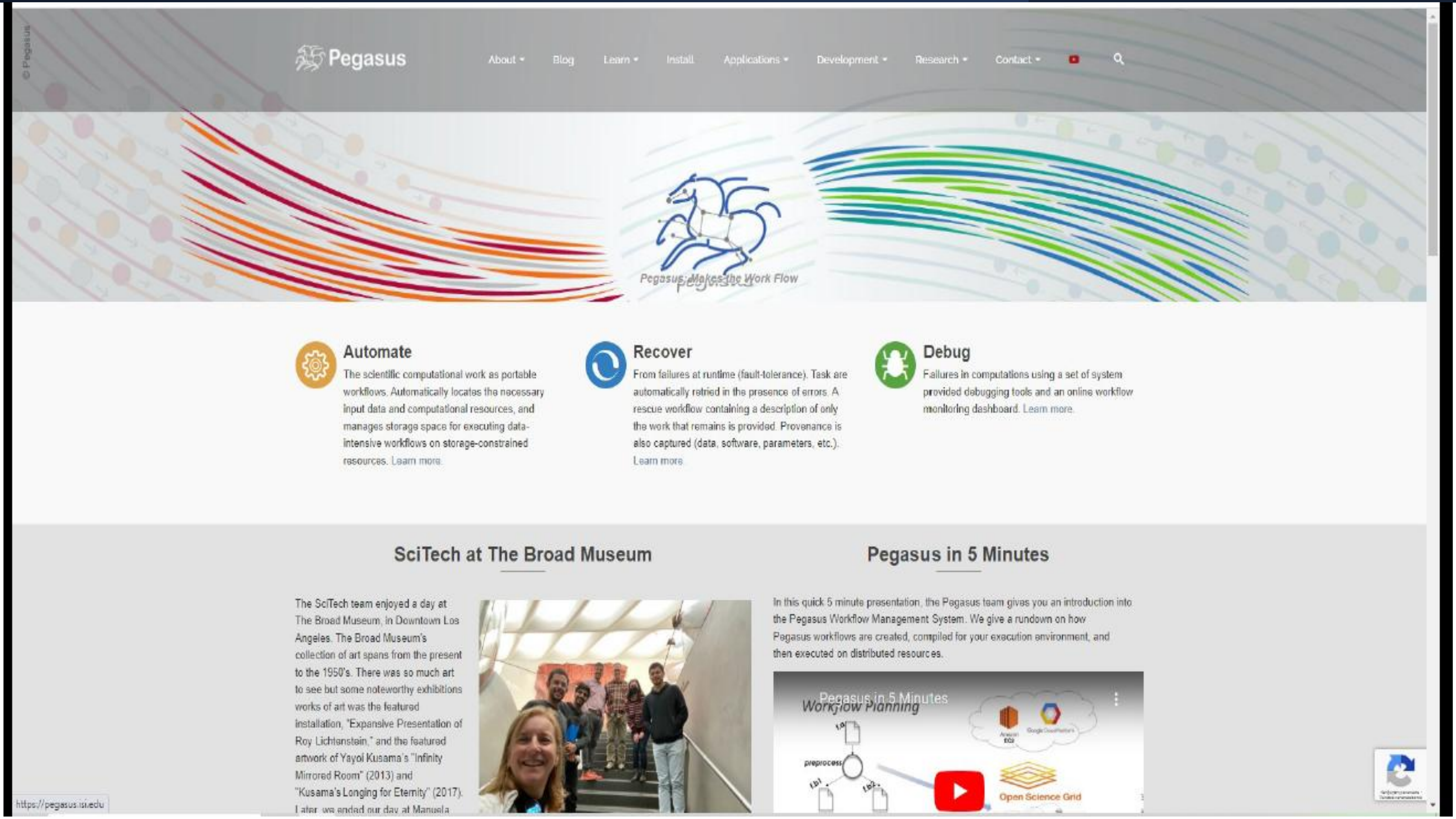


- **The diagonal hatching shows the task's deadline, starting from the late start time.**
- **The bright color shows the planned time of the task's completion.**

Example of FTL Algorithm Execution for a Task Queue (Random Generation)



What's next: experiments to study multifactor scheduling strategies on synthetic datasets and real-world applications: Montage, Epigenomics, CyberShake, Sipht, LIGO (WMS Pegasus / https://pegasus.isi.edu/)



Number of Parent and Child Tasks in Adjacent Batches

Algorithm	LIGO50	GENOME50	CYBERSHAKE50	MONTAGE1000
FTL	31	15	29	834
EBPSM	38	48	45	834

EBPSM (Muhammad H. Hilman, Maria A. Rodriguez, and Rajkumar Buyya. Workflow-as-a-Service Cloud Platform and Deployment of Bioinformatics Workflow Applications. Preprint. June 2020. 30 p. https://www.researchgate.net/scientific-contributions/Maria-A-Rodriguez-2114894132)

LIGO Workflow Optimization Results

Optimization	Total VM Cost	Total Runtime, sec	Total VM Time, sec
Cost minimization	**12740**	4260	4328
Cost maximization	**13057**	4576	4754
Runtime minimization	12929	**4180**	4310
Runtime maximization	12769	**4757**	4840
VM time minimization	12743	4200	**4269**
VM time maximization	12952	4823	**4980**

Python 3 environment, CPU Core i5, 8 GB RAM

Comparison Depending on Workflow Arrival Rate

WORKFLOW ARRIVAL RATE (PER MINUTE)	ALGORITHM	TOTAL TASK EXECUTION TIME, SEC	TOTAL VM COST	# OF CREATED VMS
0.5	VMA	409280	13226	3912
1	VMA	409486	13277	4116
2	VMA	409602	13316	4334
6	VMA	409601	13279	4503
12	VMA	409586	13257	4574
60	VMA	409578	13168	4619
100	VMA	409596	13168	4633
*	**Greedy**	**409650**	**13906**	**4955**

Comparison Depending on VM Initialization and Release Time

VM Init/Release Time	Algorithm	Total Task Execution Time, sec	Total Cost	# of Created VMs
0/0	VMA	391361	10878	4175
0/0	Greedy	391559	10885	4955
10/1	VMA	391297	11009	4127
10/1	Greedy	391559	11053	4955
100/10	VMA	391449	12144	4125
100/10	Greedy	391559	12557	4955
300/30	VMA	391453	14576	4134
300/30	Greedy	391559	15899	4955
500/50	VMA	391413	17076	4118
500/50	Greedy	391559	19242	4955

Conclusion

A set of models, methods and tools for organizing cloud computing on the WaaS platform

Combination of priority scheduling algorithms for individual tasks and independent and heterogeneous workflows of composite applications

The main limiting factor is the high (cubic) computational complexity

Future work will concern problems of scheduling algorithms complexity in scalable WaaS platforms

Thank you for your attention!

{ToporkovVV, YemelyanovDM, BulkhakAN}@mpei.ru

Current links to the GitHub repository:

https://github.com/dmieter/vmallocation/commits/master

https://github.com/Sorran973/Scheduling-in-Workflow-as-a-Service