Data-aware Scheduling in Massive Heterogeneous Systems

Magdalena Szmajduch, Joanna Kołodziej
Department of Computer Science
Cracow University of Technology,
ul. Warszawska 24, 34-155 Cracow, Poland
E-mail: mszmajduch@pk.edu.pl, jokolodziej@pk.edu.pl

KEYWORDS
Data cloud, data grid, data processing, data scheduling, ETC Matrix.

ABSTRACT

Data-aware scheduling in large-scale heterogeneous computing systems remains a challenging research issue, especially in the era of Big Data. Design of all data-related components of the popular distributed environments, such as Data Clouds (DCs), Data Grids (DGs) and Data Centers supports the processing, analysis and monitoring of the big data generated by various sources at computing centers by the end-users, devices and services. The above facts leave no doubts that data scheduling must be integrated in a single joint process together with the scheduling of computer tasks and applications. Therefore, many of the current optimization issues need to be changed and new requirements have to be considered in the scheduling process. This includes data transmission times, data processing times, availability of the data servers, safety and authentication in the data access processes. This paper presents a new version of the Expected Time to Compute Matrix model (ETC Matrix) for the case of data-aware independent batch scheduling in physical network in DGs and DCs environments. Simple genetic-based schedulers have been developed for experimental justification of the significance of the presented problem.

INTRODUCTION

Today's distributed systems, such as grid and cloud environments, are the large-scale global infrastructures enabling the remote access to variety of data types and applications, and large amount of data bases. Data in such systems can be generated also by multiple highly distributed users, different types of services and sources such mobile devices, computing applications, social networks, enterprise, cameras etc. The researchers must to face the problem of scheduling such big data, but also they must develop new methodologies and models for an effective management of large volumes of data and information. Common scheduling issues in distributed environments are mainly concerned with the task processing CPU-related requirements which are makespan, flowtime, resource usage, energy utilization etc. (Kolodziej 2012). In all similar approaches, typical data-related scheduling criteria, such as data processing and transmission time, data availability, data access and security requirements, are not considered.

The reality, where data facilities can be located anywhere, with different access rights and administrative domains, is far more different from the current assumptions.

Scheduling with data-awareness has been considered in many research works on cluster computing, DGs infrastructures and also recently in DCs (Buyya et al. 2005).Most of the provided surveys concentrates on data processing optimization issues along with data servers reliability in the data centers. Other approaches focused on data transmission scheduling and data allocation (Kosar and Balman 2009) for effective resource/storage utilization or energy-aware scheduling in large-scale data centers (Kliazovich et al. 2010; Kolodziej et al. 2011).

GridBatch (Liu and Orban 2008) can be a good example for large-scale data-intensive issues in cloud environments. The significant survey challenge is to efficiently process the huge amount of data in such infrastructures and the major issue is the scheduling process with the data transmission criteria.

In this work, a new version of the Expected Time to Compute Matrix (ETC Matrix) model is defined for Computational Grids (CGs) and the physical layers of the cloud environments, which are considered with new requirements like the data transmission and separation data from transformation (Zeadally et al. 2011; Kołodziej and Xhafa 2010).

The main aim of this paper is to define the scheduling process with the criteria mentioned above, as a multi-objective global optimization problem, similarly to the classical grid scheduling with ETC Matrix model (Ali et al. 2000a).Grid schedulers in the proposed model have both DGs and CGs features to meet the required performance of grid-enabled applications (Kołodziej and Xhafa 2011a; Kołodziej and Xhafa 2011b).

This work is a simple extension of our previous results presented in (Szmajduch 2014). We implemented the developed model in the dynamic grid scenarios for three types of grid environments: small (nb tasks/nb of hosts), medium (the same) and large (the same) grids.

The remainder of the paper is structured as follows. In the next section, the modified data-aware ETC Matrix model for independent batch scheduling and major scheduling requirements are defined. The analysis of the empirical results is then described. The last section is the paper summary and conclusions.

Proceedings 29th European Conference on Modelling and Simulation ©ECMS Valeri M. Mladenov, Petia Georgieva, Grisha Spasov, Galidiya Petrova (Editors)
ISBN: 978-0-9932440-0-1 / ISBN: 978-0-9932440-1-8 (CD)

DATA-AWARE EXPECTED TIME TO COMPUTE (ETC) MATRIX MODEL

We consider a batch scheduling problem of independent processed tasks, which need for their execution multiple data packages located at various heterogeneous data hosts, in physical computational infrastructures such as large-scale cluster, grid or the Infrastructure as a Service (IaaS) layer of the cloud system. The required data collection can be replicated at different servers, databases and can be delivered to the computational grid by the different capabilities networks (see Fig. 1).
Such data-aware grid system may be composed of elements denoted as follows:

- a meta-tasks $N = \{t_1,...,t_n\}$defined as a batch of independent tasks,
- a set of computing grid nodes $M = \{m_1,...,m_m\}$ available for a given batch;
- a set of data-files $F = \{ f_1,..., f_r\}$ needed for the batch execution,
- a set of data-hosts $D = \{dh_1,...,dh_s\}$ dedicated for the data storage purposes, having the necessary data services capabilities.

The *computing capacity vector* $CC_{batch} = [cc_1, \dots, cc_m]$ is used to define the performance efficiency of the available computational server for a given set. The cc_i element of the vector denotes the computing capacity of the server $i$and is expressed in a Million of Instructions Per Second (MIPS).
The *ready times vector* $ready_{times\ (batch)} = [ready_1, \dots, ready_m]$ characterize the calculation of the prior load of every machine from the M_{batch} set.
To estimate the completion times of tasks allocated at a specific computational server an Expected Time to Compute (ETC) matrix model (Ali et al. 2000a) is adapted.
The particular elements of the ETC matrix are estimated as the proportion of the vectors WL and CC coordinates, which are:

$$ETC[i][j] = \frac{wl_j}{cc_i} \quad (1)$$

For every single pair machine m_i and task t_j in Eq. (1) the value of the matrix element $ETC[i][j]$ primarily depend on the computing speeds of the machines. However also the diversity of tasks and sources in the system has to be reflected and taken into account.

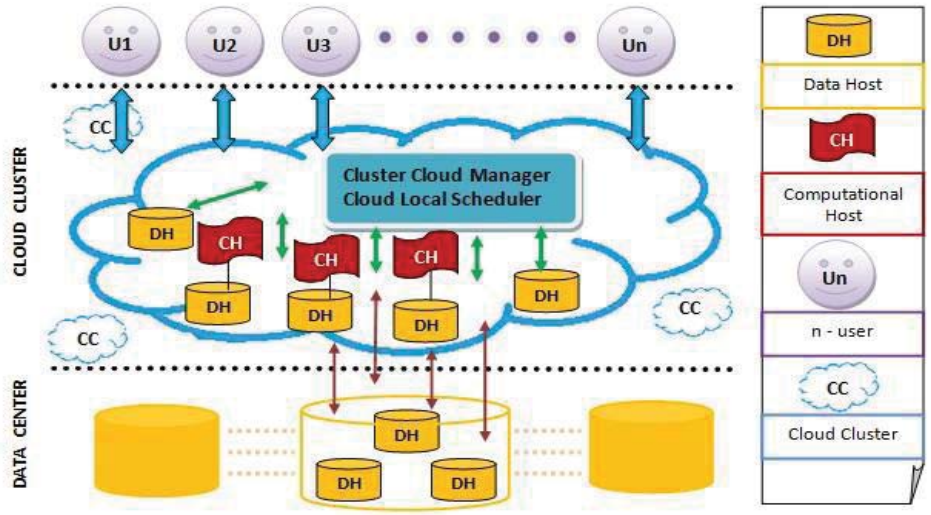

Fig. 1 Data-aware meta-task grid scheduling problem.

The *tasks workload vector* $WL_{batch} = [wl_1, \dots, wl_n]$ is used todefine the computational load of the meta-task, where wl_j is the evaluation of the computational load of task t_j (measured in Millions of Instructions (MI)). Each task $t_j$needs a batch of data files $F_j = \{f_{(1,j)}, \dots, f_{(r,j)}\}(F_j \subseteq F_{batch})$ for its correct computation. Such batch is copied and located at the following data serversdh_d. The dh_d is a part of the D. Each file $F_{(p,j)} \in F_j\ (p \in \{1, \dots, r\})$ is replicated on and available from the set of data hosts dh_d. Each data host is assumed to be able to serve multiple data files at a time and data replication is a priori defined as a separate replication process.

For that reason, this model use the Gaussian distribution to produce the elements of both vectors WL and CC. What is more when considering data-aware scheduling is the estimation of the data transfer time. The time needed to transfer each, necessary for the execution of the task t_j, data file $f_{c,j} \in F\ (c \in \{1, \dots, r\})$ from the data host $dh_d \in D$ to the server m_i is marked as $TT_{ij}[c][d]$ and can be computed as follows:

$$TT_{ij}[c][d] = RES[c][d] + \frac{Size[f_{cj}]}{B[dh_d,i]} \quad (2)$$

The $RES[c][d]$ stands for the response time of the dh_d data server and is evaluated as a difference between the time of the demand send to dh_d and the time when the first byte of the data file f_c reached the machine m_i for processing the task t_j. The size of the data file f_c required for the execution of the task t_j is defined by $Size\left[f_{jc}\right]$ and is expressed in Mbits. Where the bandwidth of the logical link connecting dh_d and m_i is denoted by $B\left[dh_d, i\right]$ and expressed with Mbits/time unit.

The $RES[c][d]$ are the elements which form the Data Response Times matrix denoted as $RES_{s\times r}$. Similarly to the vectors WL and CC, the data response times are generated using the standard Gaussian distribution.

The major scheduling factors in the ETC matrix model are the resources completion times. The $completion[i][j]$ defines the calculated completion time of the task t_j on machine m_i as the wall-clock time measured from the task submission till its completion. In data-aware approach it highly depends on the computing and transmission times specified in Eq. (1) and Eq. (2).

The data transfer time can have different influence on the task completion time depending on the method which is used to process the data file by the task. Two possible scenarios are presented in Figure 2.

The first mode, "a", assumes that the data files required for the computation of the t_j task are delivered to the machine before the execution of all the tasks, from the tasks batch, assigned to his machine, including task t_j.

Every transfer bandwidth is calculated due to the number of possible synchronized data transfers. In such case the completion time on machine m_i of the task t_j is expressed by:

$$completion[i][j] = \max_{f_{cj}\in F; dh_d \in D} TT_{ij}[c][d] + ETC[i][j] \quad (3)$$

The second mode, scenario "b", assumes that some of the data files are transferred as in the first mode ("a"), with the difference that the rest of data required for the execution of every task on this particular machine m_i (including task t_j) is delivered while executing the tasks. In this case, the delivery times of the streamed data files are concealed by execution times of the tasks, thus the completion time of the task t_j on machine m_i is calculated with a different, following equation:

$$completion[i][j] = \max_{f_{cj}\in \hat{F}_j} TT_{ij}[c][d] + \sum_{F_{lj}\in[F\setminus\hat{F}_j]}\left(TT_{ij}[l][d] + ETC[i][j]\right) \quad (4)$$

where $\hat{F}_j$ represents the data files batch which is delivered before the execution of the task t_j and obviously all other tasks belonged to this machine.

This survey considered the data hosts as, separated from the computing resources, data storage centers.

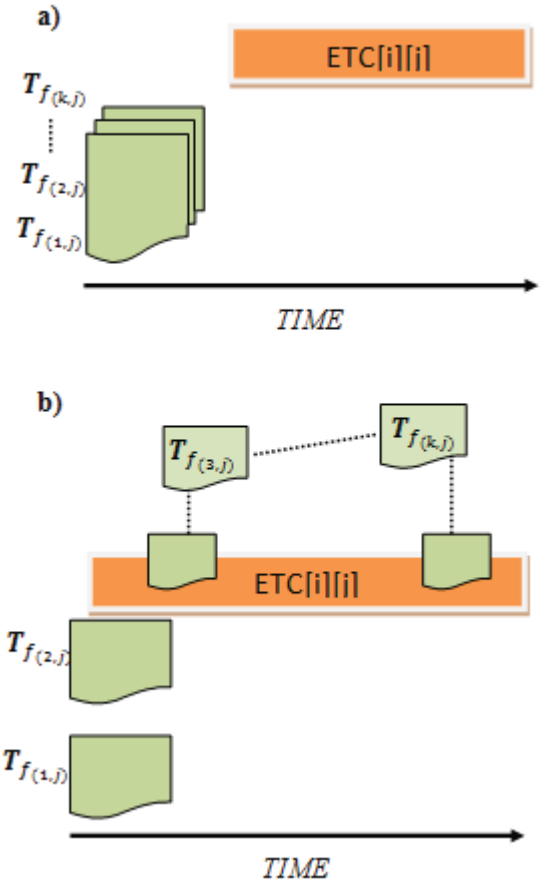

Fig.2. Two variants of task completion times estimation assigned to the machine mi with k data files needed for the task execution.

Scheduling criteria

The overall data-aware batch scheduling procedure is performed in the following steps:

- obtain the information about resources that are available in the system,
- obtain the information about unsettled tasks,
- establish the location of data hosts where the data files needed for the tasks completion are placed,
- prepare a set of tasks and calculate a schedule for this set on available machines and data hosts,
- allocate the tasks,
- monitor the process and re-scheduled the tasks which failed.

This process has been presented graphically in Figure 3 below.

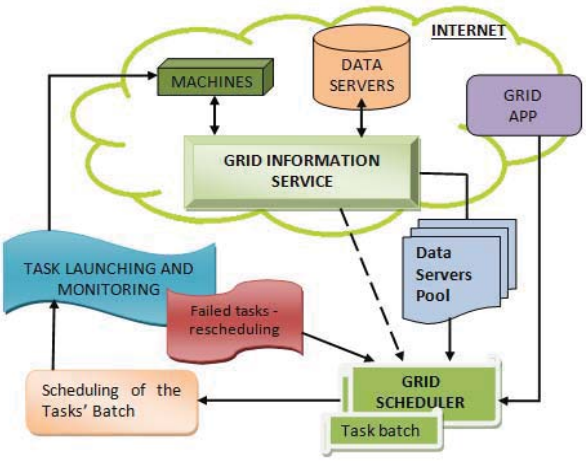

Fig.3. Phases of the data-aware batch scheduling.

The main data-aware scheduling criteria are very similar to those desired in common scheduling systems where data file transfers are not considered. It includes minimization of the completion time, makespan and average flowtime, defined as follows:

- the minimizing completion time of the set of tasks is defined as follows:

$$completion_{batch} = \sum_{t_j \in N_{batch}; m_i \in M_{batch}} completion[i][j], \quad (5)$$

where $completion[i][j]$ is calculated in Eq. 3 or Eq. 4according to data transfer mode;

- minimizing makespan C_{max} computed as:

$$C_{max} = \max_{m_i \in M_{batch}} completion[i] \quad (6)$$

where $completion[i]$ is calculated as the sum of completion times of all the tasks assigned to machine m_i by using either Eq. 3or Eq. 4.

- minimizing average flowtime $\tilde{F}$. For a machine $m_i$the flowtime can be computed as a workflow of the tasks chain on this machine, specifically:

$$F[i] = completion[i] \quad (7)$$

The cumulative flowtime for the entire system is denoted as the sum of $F[i]$ factors, namely:

$$F = \sum_{i \in M} F[i]. \quad (8)$$

In the end, the scheduling aim is to minimize the average flowtime $\tilde{F}$ for single machine, which is defined as below:

$$\tilde{F} = \frac{F}{m}. \quad (9)$$

The above formal definitions of the major scheduling criteria are based on the ETC matrix model, which is very helpful in formulating such equations.The $completion[i]$parameters form the completion vector$completion[i] = [completion[1], \dots, completion[m]]^T$.The full list of the major scheduling criteria defined in terms of completion times and ETC matrix, is presented in (Kolodziej 2012).

EXPERIMENTS

The aim of a simple experimental analysis is to show, how much the data access and transfer can possibly delay the whole scheduling process. The scheduling

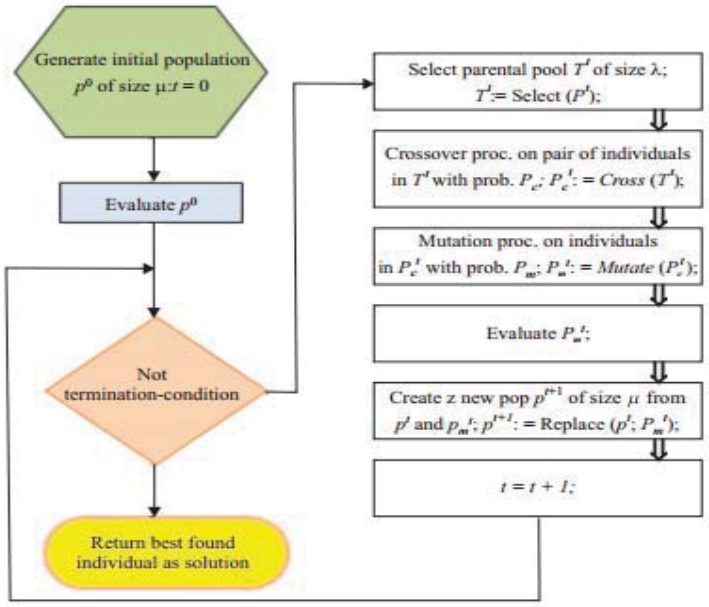

Fig 4. General template of the GA-scheduler implementation.

criteria (and schedulers' performance metrics) considered in the experiments were the makespan and average flowtime calculated by using Eqs. 6 and 9. The results of data-aware scheduling were compared with the results achieved in the conventional scheduling, where data transfer times are ignored. In such a case it is assumed that all necessary data is stored at computational nodes and ready for use, which is not the realistic scenario. For the analysis both data transfer scenarios (scenario "a" and scenario "b") specified in Section II are considered. Therefore, the completion times in Eq. 5 are estimated using Eq. 3in the first scenario, and Eq. 4in the second scenario.

We used as the scheduler in our experiments a simple genetic-based scheduler presented in Fig. 4. This is a simple (μ +λ) evolutionary strategy often used for solving classical combinatorial optimization problems (see Xhafa et al. 2007; Pinel et al. 2011; Michalewicz 1992).We configured the genetic operators in the following way:

- selection – Linear Ranking,
- crossover – Cycle Crossover,
- mutation – Rebalancing,
- replacement – Steady State.

All those genetic operators are commonly used in solving the large-scale combinatorial problems. The detailed definition of those operators and schedule representation can be found in ().

The input parameters for the scheduler are presented in Table 1

TABLE 1. Settings of the genetic scheduler.

Parameter	**Value**
μ	$4 \cdot (\log_2 n - 1)$
λ	μ / 3
mut_prob	0.15
cross_prob	0.9
nb_of_epochs	$20 \cdot n$
max_time_to_spend	$25\ s$

The number of individuals in base populations shown asP^t and P^{t+1} in Fig. 5 is denoted by μ, λ is the number of individuals in offspring populations T^t, P_c^t and P_m^t. The parameters *cross_prob*, *mut_prob* are used for the notation of the crossover and mutation probabilities. The *nb_of_epochs* denotes the maximal number of main loop executions of the algorithm. Each loop execution is interpreted as genetic epoch. The maximal number of such epochs is defined as the main global stopping criterion for the scheduler. However, if the execution of those epochs will take much time, the algorithm is stopped after 25 s (*max_time_to_spend*).

The main reason of our choice of such a simple scheduler was to demonstrate the impact of the data transfer and access on the optimization of the scheduling criteria. Therefore, we wanted to use a simple method, easy for the implementation in the performed analysis. However, this is just an early stage of our research in the domain and of course we plan to conduct a comprehensive analysis of the effectiveness of various heuristic-based schedulers in data-aware scheduling.

The experiments have been conducted by using the Sim-G-Batch data grid simulator defined in (Kolodziej et al. 2012). The main input data for the simulator is:

- the workload vector of tasks,
- the computing capacity vector of machines,
- the vector of prior loads of machines, and
- the ETC matrix of estimated execution times of tasks on machines
- the data host response times.

The parameters of the simulator are presented in Table 2. We consider in our experiments three grid size scenarios are defined: small (64 hosts/1024 tasks), medium (128 hosts/2048 tasks), and large (256 hosts/4096 tasks). The capacities of the resources, data transmission times and the workloads of tasks are randomly generated by the Gaussian distributions. This is the dynamic case, so the number of hosts and tasks can be different in the different time units (*add_host, delet_host, add_task, delete_task* parameters) It is assumed that all tasks submitted to the system must be scheduled and all machines in the system can be used. The sizes of data files and the bandwidth are generated by the uniform distributions defined for the following intervals [2;1600] and [10;100] respectively.

TABLE 2. Settings of the simulator.

	Small	Medium	Large
Init. hosts	64	128	256
Max. hosts	70	135	264
Min. hosts	58	121	248
MIPS		*N* (1000, 175)	
Add host	*N* (562500, 84375)	*N* (500000, 75000)	*N*(437500, 65625)
Delete host		*N*(625000, 93750)	
Total tasks	1024	2048	4096
Init. tasks	768	1536	3072
Workload		*N* (250000000, 43750000)	
Interarrival	*E* (3906.25)	*E* (1953.125)	*E* (976.5625)
Data Respond Time		*N* (100, 35)	
Host select		ALL	
Task select		ALL	
Number of runs		30	

Results

The results of the experiments achieved in the scenarios "a" and "b" (see Section II) and No Data Transfer (NDT) case are presented in Table 3 (makespan) and 4. (average flowtime).The results were averaged over 30 independent runs of the simulator with [±s.d.] s.d-standard deviation values. Both makespan and average flowtime are expressed in arbitrary (but not concrete) time units.

In both makespan and average flowtime optimizations, a big differences in the achieved results is observed in the additional data transfer and no data transfer cases. In a data-aware scheduling, scenario "b" is the case, where the achieved results (for makespan and flowtime) in Medium grid and Large grid infrastructures are better than for the prior load of all data files before the task execution (scenario "a"). In Small grid the results are similar for both scenarios.

TABLE 3. Makespan results (±S.D.) (inarbitrary time unites (S.D. – standard deviation)).

Scenario	Small	Medium	Large
Scenario „a"	**751166581.648** (±44117322.872)	860885586.314 (±49641822.213)	927657348.425 (±38411457.341)
Scenario „b"	762942252.985 (±42989403.987)	**820238735.873** (±39103304.032)	**889227303.643** (±21393320.672)
NDT	4534354074.674 (±432550682.566)	489261762.387 (±41873795.873)	491155422.873 (±39721132.235)

TABLE 4. Average flowtime results (±S.D.) (inarbitrary time unites (S.D. –standard deviation)).

Scenario	Small	Medium	Large
Scenario „a"	**1589661820.816** (±436299327.723)	3190329034.728 (±827910218.673)	6398508188.267 (±546372083.478)
Scenario „b"	1597014554.934 (±718565549.521)	**3009072850.369** (±276260267.376)	**5908237732.186** (±895036487.371)
NDT	1087665145.384 (±103328184. 256)	2141758180.791 (±17796699.367)	4241955274.638 (±799690481.526)

CONCLUSIONS AND RESEARCH DIRECTIONS

This paper presents the new version of ETC Matrix model for batch scheduling in the physical clusters, where separate computing and data servers are located. In this model, the completion times of all tasks assigned to the computing nodes of the network have included the data transmission times. Two data transmission scenarios were considered with prior load of all files necessary for the execution of assigned tasks, and with the ad-hoc delivery of just requested (necessary) data files during the task execution. The implementation of this model and further experimental analysis were performed in the case of dynamic grid infrastructure, were number of network nodes and assigned tasks may vary in different time intervals. The results of the performed experiments show that omitting the data transfer phase in the scheduling process may lead to the bad estimations of the scheduling times, and more general scheduling costs.

The performed analysis in its early stage. The presented work is a simple extension of the previous analysis published in (Szmajduch 2014).The author plans to extend it to the virtual resources and databases and the extended cloud infrastructures, where the mobile devices (smartphones, tablets, laptops, etc.) are considered as the computational nodes of the physical cloud layer and can additionally store and generate the data. This will allow to validate proposed model in much more realistic cloud scheduling scenarios, but also will increase the complexity of the scheduling problem.

ACKNOLEDGMENT

Joanna Kołodziej's research presented in the paper was partially supported by the European Commission FP7 through the project ParaPhrase: Parallel Patterns for Adaptive Heterogeneous Multicore Systems, under contract no. 288570 (http://paraphrase-ict.eu).

REFERENCES

J. Kołodziej, Evolutionary Hierarchical Multi - Criteria Metaheuristics for Scheduling in Large-Scale Grid Systems. Studies in Computational Intelligence Serie, vol. 419. Berlin-Heidelberg: Springer, 2012.

R. Buyya, M. Murshed, D. Abramson, and S. Venugopal, "Scheduling parameter sweep applications on global Grids: a deadline and budget constrained cost-time optimization algorithm", Softw. Pract. Exper., vol. 35, no. 5, pp. 491–512, 2005.

T. Kosar and M. Balman, "A new paradigm: Data-aware scheduling in grid computing", Future Gener. Comp. Syst., vol. 25, no. 4, pp. 406–413, 2009

Kliazovich, D., Bouvry, P., Audzevich, Y., and Khan, S.U.: "GreenCloud: A PacketlevelSimulator of Energy-aware Cloud Computing Data Centers", in Proc. of the 53rd IEEE GlobalCommunications Conference (Globecom), Miami, FL, USA, December 2010.

Kolodziej, J., Khan, S. U., and Xhafa, F.: "Genetic Algorithms for Energy-aware Scheduling in Computational Grids", in Proc. of the 6th IEEE International Conference on P2P, Parallel, Grid, Cloud, and Internet Computing (3PGCIC), Barcelona,Spain, October 2011.

H. Liu and D. Orban, "GridBatch: Cloud Computing for LargeScale Data-Intensive Batch Applications", in Proc. 8th IEEE Int. Symp. Cluster Comput. and the Grid CCGRID 2008, Lyon, France, 2008, pp. 295–305.

Zeadally, S., Khan, S.U., and Chilamkurti, N.: "Energy-Efficient Networking: Past, Present, and Future", *Journal of Supercomputing*, pp. 1–26.doi:10.1007 /s11227-011-0632-2, 2011.

J. Kołodziej and F. Xhafa, "A game-theoretic and hybrid genetic meta-heuristic model for security-assured scheduling of independent jobs in computational grids", in Proc. Int. Conf. Complex, Intell. Softw. Inten. Syst. CISIS 2010, Krakow, Poland, 2010, pp. 93–100.

S. Ali, H. J. Siegel, M. Maheswaran, and D. Hensgen, "Task execution time modeling for heterogeneous computing systems", in Proc. 9th Heterogen. Comput. Worksh. HCW 2000, Cancun, Mexico, 2000, pp. 185–199.

J. Kołodziej and F. Xhafa, "Meeting security and user behaviour requirements in grid scheduling", Simul. Model. Pract. Theory, vol. 19, no. 1, pp. 213–226, 2011.

J. Kołodziej and F. Xhafa, "Integration of task abortion and security requirements in GA-based meta-heuristics for independent batch grid scheduling", Comp. Mathem. Appl., vol. 63, no. 2, pp. 350–364, 2011.

F. Xhafa, L. Barolli, and A. Durresi, "Batch mode schedulers for grid systems", Int. J. Web and Grid Serv., vol. 3, no. 1, pp. 19–37, 2007

F. Pinel, J. E. Pecero, P. Bouvry, and S. U. Khan, "A two-phase heuristic for the scheduling of independent tasks on computational grids", in Proc. of ACM/IEEE/IFIP Int. Conf. High Perform. Comput. Simul. HPCS 2011, Istanbul, Turkey, 2011, pp. 471–477

Z. Michalewicz, Genetic Algorithms + Data Structures = Evolution Programs. Berlin: Springer, 1992.

M.Szmajduch, "Data and Task Scheduling in Distributed Computing Environments", Journal Of Telecommunications and Information Technology, Vol. 4, 71-78,2014.

MAGDALENA SZMAJDCUH is a Ph.D. student of computer science in the interdisciplinary Ph.D. Programme managed jointly by the Jagiellonian University in Cracow, Polish Academy of Science in Warsaw and Cracow University of Technology. She is also the assistant professor at the Department of Computer Science of Cracow University of Technology. The main topic of her interest is data processing in large scale distributed dynamic systems.

JOANNA KOŁODZIEJ is an associate professor in Department of Computer Science of Cracow University of Technology. She is a vice Head of the Department for Sciences and Development. She serves also as the President of the Polish Chapter of IEEE Computational Intelligence Society. She published over 150 papers in the international journals and conference proceedings. She is also a Honorary Chair of the HIPMOS track of ECMS. The main topics of here research is artificial Intelligence, grid and cloud computing, multiagent systems. The detailed information is available at www.joannakolodziej.org