

# Harmony Search for Self-configuration of Fault-Tolerant and Intelligent Grids

Jerzy Balicki<sup>1(✉)</sup>, Waldemar Korlub<sup>2</sup>, Jacek Paluszak<sup>2</sup>,  
and Maciej Tyszką<sup>2</sup>

<sup>1</sup> Faculty of Mathematics and Information Science,  
Warsaw University of Technology, Koszykowa St. 75, 00-662 Warsaw, Poland  
balicki@eti.pg.gda.pl

<sup>2</sup> Faculty of Telecommunications, Electronics and Informatics,  
Gdańsk University of Technology, Narutowicza St. 11/12,  
80-233 Gdańsk, Poland  
waldemar.korlub@pg.gda.pl,  
jpaluszak@gmail.com, tyszka.maciej@gmail.com

**Abstract.** In this paper, harmony search algorithms have been proposed to self-configuration of intelligent grids for big data processing. Self-configuration of computer grids lies in the fact that new computer nodes are automatically configured by software agents and then integrated into the grid. A base node works due to several configuration parameters that define some aspects of data communications and energy power consumption. We propose some optimization agents that are based on harmony search to find a suboptimal configuration of fault-tolerant grids processing big data. Criteria such as probability that all tasks meet their deadlines and also a reliability of grid are considered. Finally, some experimental results have been considered.

## 1 Introduction

An intelligent grid is supposed to manage its resources to meet the task requirements on the way to achieving the common objective. Self-configuration of computer grids lies in the fact that new computer nodes are automatically configured by software agents and then integrated into the grid. The whole process of self-configuration is similar to the “plug-and-play” rule for some operating systems. However, configuring agents launch connectivity and download some configuration parameters. If a new computer node is added to the middleware layer and powered on, it is instantly identified and registered by configuration agents.

A base node works due to several configuration parameters that define some aspects of data communications and energy power consumption. These parameters can be improved to change grid behavior, based on some administrator observations. Another way is to delegate this competences to optimization agents they find the most adjusted configuration to the workload and resource using. One of the most commonly used criterion of grid behavior is its reliability that should be maximized. The main dilemma is the fact that this problem is NP-hard and it is impossible to find an optimal configuration for hundreds of nodes.

In the presented model, we propose some optimization agents that are based on harmony search to find a suboptimal configuration of fault-tolerant grids processing big data. A fault-tolerant grid deals with failures of its nodes and software where each node has some duplicated servers associated with its [38]. One node is the primary, and some associated nodes are dedicated for backup [18]. Tasks are performed by primary and backup servers, concurrently. Another model of grid is based on assumption that there are no fault-tolerant nodes. A grid node cooperates with other nodes as backups. In case of a node failing, all tasks allocated to this server are re-allocated to one of its backups. Some algorithms of resource using take into account the failure/repair rates and the fault-tolerant overheads. These algorithms can improve the grid performance meaningfully, but the quality of configuration and delay for its founding are still under construction [20, 42].

In this paper, an outlook of harmony search metaheuristics is discussed in Sect. 2. Moreover, specific aspects for big data are presented in Sect. 3. Especially, *Map-Reduce* model for BD processing is studied in Sect. 4. Then, intelligent agents based on harmony search for improvement of fault-tolerant measure are described in Sect. 5. Moreover, some outcomes from numerical experiments are interpreted in Sect. 6.

## 2 Outlook of Harmony Search Metaheuristics

Harmony search can be applied for self-configuration support of some fault-tolerant grids. Harmony search metaheuristics HS models phenomena related to the process of playing on musical instruments [41]. An optimization process can be compared to a process of selection the best sound while improvising jazz musicians. Similarly, a conductor of orchestra searches the best harmony of several instruments or a composer creates the best melody for different music lanes [1]. The HM concept was suggested by Geem [15, 45]. Figure 1 shows a diagram of the basic version of the HS metaheuristics [2].

The HS algorithm determines a solution for one-criterion optimization problem with continuous decision variables that can be formulated, as follows [4]:

$$\min_{x \in X} f(x), \tag{1}$$

where:

- $f(x)$  – a value of an objective function  $f$  for solution  $x \in X, f : \mathbf{R}^{J_{max}} \rightarrow \mathbf{R}$ ;
- $x$  – a vector of decision variables,  $x = [x_1, \dots, x_j, \dots, x_{J_{max}}]^T$  for  $l_j \leq x_j \leq u_j$ ,  $j = \overline{1, J_{max}}$ ;
- $J_{max}$  – a number of decision variables;
- $X$  – a set of decision variables.

The lower limit vector is  $l = [l_1, \dots, l_j, \dots, l_{J_{max}}]^T$  and the upper limit vector is  $u = [u_1, \dots, u_j, \dots, u_{J_{max}}]^T$ , wherein  $l_j \in \mathbf{R}, u_j \in \mathbf{R}, l_j \leq u_j$  for  $j = \overline{1, J_{max}}$ . An initialization the harmonic memory HM (Fig. 1) occurs after setting the following parameters:

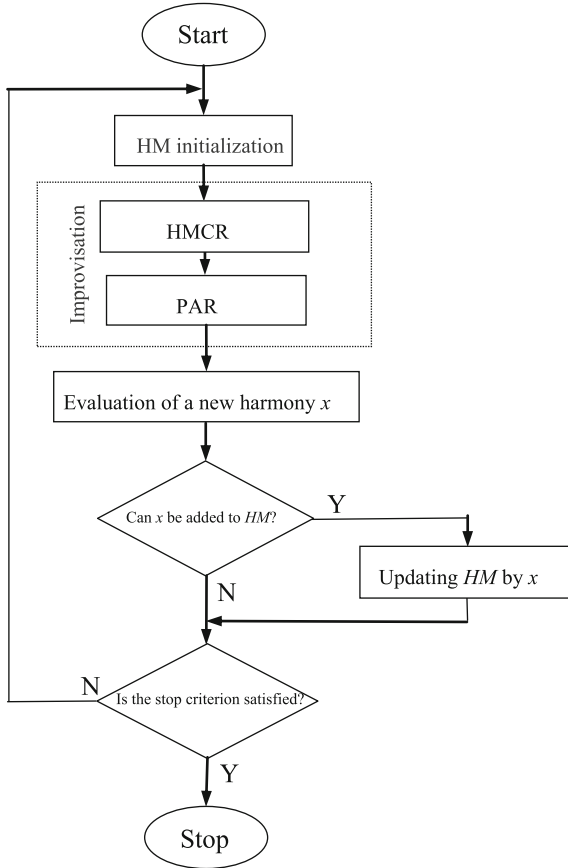


Fig. 1. A diagram of the harmony search algorithm [29]

- *HMS* - *Harmony Memory Size* [24];
- *HMCR* - *Harmony Memory Considering Rate* is the probability of a random event that the value of the decision variable during improvisation (constructing a solution) is drawn from the memory *HM*; an uniform distribution is assumed to draw;
- *PAR* - *Pitch Adjusting Rate* is the rate of the randomly selected decision variable;
- *NGmax* - *Number of Generations* (Improvisations);
- *BW* - *Bandwidth of Generations* that is the width of the interval to modify the value of the decision variable that is randomly selected from memory; the new value of the decision variable is modified by adding the value from the range  $[-BW, BW]$ .

In memory *HM*, there are stored *HMS* randomly generated solutions with  $J_{max}$  coordinates and the corresponding fitness function values  $fitness(x)$ . If restrictions are imposed on the solution, its efficiency is reduced by the appropriate punishment in case of violation of restrictions. The efficiency of each solution can be increased by an amount such that the accepted value of non-negative. The basic version of the harmony search algorithm has been repeatedly modified to adjust to solve some optimization problems [22].

### 3 Intelligent Agent Architecture for Big Data

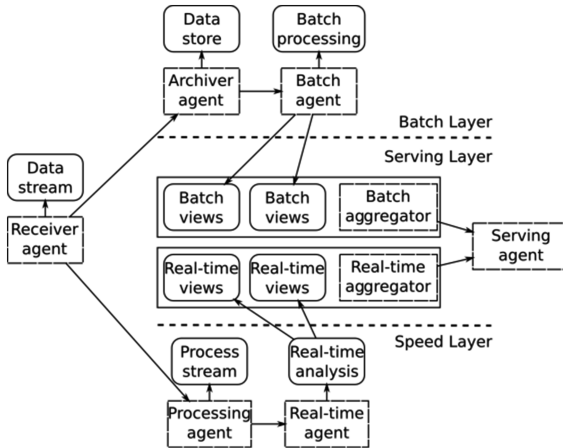
Big data (an acronym BD) is related to databases with petabyte capacities  $10^{15}$  B. 10 terabytes is a large capacity for a financial transaction system, but it is too small to test a web search engine. BD is uncooperative to work with using some relational database management systems like DB2, INGRES, Oracle, Sybase or SQL Server. Big data requires hundred thousand processors for data processing like supercomputers [36], grids [11] or clouds [8]. Especially, cloud architectures are preferred to BD processing because of commercial data centers with expensive information.

Tasks developed SQL-like queries to BD are massive parallel because the short time of a query performing is required. For instance, a query for multi-terabyte datasets at *BigQuery* service in *Google Cloud* is performed during few seconds. *BigQuery* service is scalable cloud like *IaaS Infrastructure as a Service*. Furthermore, this *RESTful* web service enables interactive analysis cooperating with *Google Storage* [16]. The most important tasks are related to analytics, capture, search, sharing, storage, and visualizing. Moreover, some BD mining tasks can be used to find predictions as well as some descriptive statistics tasks can be developed for business intelligence [23].

BD can be characterized by the 4Vs model due to high *volume*, extraordinary *velocity*, great data *variety*, and *veracity*. Data can be captured via Internet of Things from different sensors like smartphones, tablets, microphones, cameras, computers, radars, satellites, radio-telescopes and the other sensors. Moreover, data can be captured from social networks. A storage capacity can achieve many petabytes for one volume that is *high volume* [26]. *MongoDB* is one of perspective solutions for BD because the *NoSQL* database supports data stored to different nodes. Mongo DB can cooperate with massively parallel cluster with lots of CPUs, GPUs, RAM units and disks [27]. A crucial problem with BD is related to reading from a storage system to obtain the rapid answer on a complex query that is divided on some parallel operations acting on diverse data. Big data can be spread over some partitions that run on some separate nodes with own table spaces, logs, and configurations. In that case, a query is performed on all partitions concurrently [35, 44].

In an experimental grid called *Comcute*, two kinds of intelligent tasks have been considered to implement a middleware layer [13]. This grid is dedicated to parallel computing with using volunteer computing. Agents for data management send data from source databases to distribution agents. Then, distribution agents cooperate with web computers to calculate results and return them to management agents. Both types of agents can autonomously move from one host to another to improve quality of grid resource using. Moreover, the other agents based on harmony search have been introduced to optimize big data processing regarding some fault-tolerant aspects. These harmony search schedulers can cooperate with distributors and managers to give them information about optimal workload in a grid [9, 10].

The lambda architecture is developed for real-time BD analysis [26]. The batch layer of this architecture supports offline data processing by *MapReduce* framework [39]. This layer produces batch views of data, which can be exposed to external applications (Fig. 2). The serving layer offers prepared views to clients. The speed layer is responsible for real-time processing of data streams. It analyses data that was not yet processed



**Fig. 2.** Multi-agent real-time processing utilizing lambda architecture [37]

by the batch layer. Speed layer produces real-time views that can be coupled with batch views to create complete representation of the extracted knowledge [19].

The lambda architecture can be defined in terms of a heterogeneous multiagent system [37]. An implementation requires integration of a few components: one for batch processing, another one for serving views, a different solution for real-time stream analysis and a component that merges real-time views with batch views [43].

The use of multiagent environment will provide a common way for information exchange between different component and a common execution model [40]. The differences between individual components of the lambda architecture lead to inherently heterogeneous realizations so the ability to handle diversity in agent systems in another motivation for this approach [22].

## 4 Map-Reduce Model for Fault-Tolerant Grid

Grid and volunteer computing systems are different from super-computing systems because inexpensive hardware commodities have been widely deployed, which is helpful to the scalability. But it also brings a large number of hardware failures. Moreover, many machines constantly restart to update systems, which cause huge software failures [14]. Similarly, the popular cloud computing model *MapReduce* also has to overcome the failures [5, 7].

When a job consists of thousands tasks, the possibility of a few failed tasks is very high. Several fault-tolerant applications can be executed in the platform, which can use the result despite of some failed tasks. To support such fault-tolerant computing, an open source implementation of *MapReduce* can be applied. *Hadoop* has already provided the interface, by which the job can tolerate a given percentage of failed tasks. It was observed that optimizing the availability of individual task is not an effective approach for ensuring the high availability of these multi-task jobs [30].

However, *Hadoop* implicitly assume the nodes are homogeneous, but it doesn't hold in practice. These motivate to propose an optimal multi-task allocation scheme towards heterogeneous environments, which can tolerant a given percentage of failures to total tasks [28]. In this case the *reduce* function's responsibility is to sum the each *key's values* [34].

*MapReduce* is applied to solve several problems like large-scale machine learning for the *Google News*. Moreover, an extraction of data is used to produce reports of popular queries and extraction of geographical locations from a large corpus of web pages for localized search. In 2004, *Google* changed an indexing system that produces data used for web search service to system that used *MapReduce*. The new indexing system takes input documents that have been retrieved from a crawling system store as a set of files, and then they are processed by from five to ten *MapReduce* operations. It is easier to operate because of automatic resolving problems like machine failures, slow machines and networking hiccups [17, 25].

### 5 Harmony Search Agents for Local Grid Self-configuration

Intelligent agents can optimize a grid resource management for tasks related to big data queries. An agent based on harmony search metaheuristics AHS can reconfigure a local part of a grid. The whole grid is divided on zones and the AHS is assigned to its grid zone to support self-optimization of a system. The main part of AHS is a multi-objective scheduler for tasks from a middleware layer. This scheduler optimizes a probability that all tasks meet their deadlines, and the grid reliability [12]. We assume that each computer and each link between them are assumed to fail independently with exponential rates. It is preferred to allocate modules to computers on which failures are least likely to occur during the execution of task modules [3]. The rationale assumption is that repair and recovery times are largely implementation-dependent. Moreover, repair and recovery routines usually introduce too high time overheads to be used on-line for time-critical applications [6].

The overhead performing time of the task  $T_v$  by the computer  $\pi_j \in \Pi = \{\pi_1, \dots, \pi_j, \dots, \pi_J\}$  is represented by  $t_{vj}$ . Let the computer  $\pi_j$  be failed independently due to an exponential distribution with rate  $\lambda_j$ . Computers can be allocated to nodes and also tasks can be assigned to them in purpose to maximize the reliability function  $R$ , as below [21]:

$$R(x) = \prod_{v=1}^V \prod_{i=1}^I \prod_{j=1}^J \exp(-\lambda_j t_{vj} x_{vi}^m x_{ij}^\pi), \tag{2}$$

where

$$x_{ij}^\pi = \begin{cases} 1 & \text{if } \pi_j \text{ is assigned to the } w_i, \\ 0 & \text{in the other case.} \end{cases}$$

$$x_{vi}^m = \begin{cases} 1 & \text{if task } T_v \text{ is assigned to } w_i, \\ 0 & \text{in the other case,} \end{cases}$$

$$(x^m, x^\pi) = [x_{11}^m, \dots, x_{1I}^m, \dots, x_{vi}^m, \dots, x_{VI}^m, x_{11}^\pi, \dots, x_{1J}^\pi, \dots, x_{ij}^\pi, \dots, x_{I1}^\pi, \dots, x_{Ij}^\pi, \dots, x_{IJ}^\pi]^T.$$

Figure 3 shows the relation between the measure of system reliability  $R$  and time of using this system for the chosen two-computer system for  $\lambda_1 = 0.001$  [TU<sup>-1</sup>] and  $\lambda_2 = 0.002$  [TU<sup>-1</sup>].

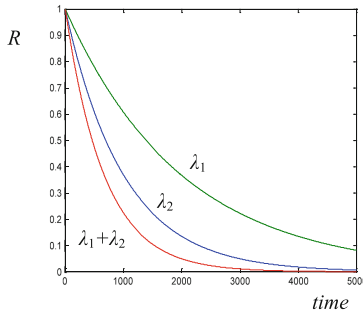


Fig. 3. The time-depended reliability of two-computer system

### 6 Task Scheduling Algorithm

Let the distributed application  $A_n$  starts running after  $\lambda_n$  and complete it before  $\delta_n$  [31]. Figure 4 shows an example of the task flow graph for two applications. Task  $m_2$  is performed with the probability  $q$  in a sub-graph denoted as  $OR$  (Fig. 4) and task  $m_3$  – with the probability  $(1-q)$ . Task may be performed at the most  $L_{max}$  times in a sub-graph denoted as  $Loop$ , and each repetition of this module is performed with the probability  $p$ . The task flow graph is split on some instances to schedule tasks if the sub-graph  $OR$  appears. There are  $2L_{max}$  instances for the task graph from Fig. 4. The instance, where task  $m_2$  appears and task  $m_5$  runs  $k$  times, occurs with the probability:

$$p_i = q(1 - p)p^{k-1} \tag{3}$$

An allocation of modules to computers  $(x^m, x^\pi)$  creates possibility to schedule tasks for each computer. Times of task completions  $(C_1, \dots, C_v, \dots, C_V)$  can be calculated for scheduled allocation modules to computers  $x$ . Let  $d_v$  represents the completion deadline for the  $v$ th task. If  $C_v \leq d_v$ , then the time constraint is satisfied what can be written as  $\xi(d_v - C_v) = 1$ . The state of deadline constraints regarding the  $i$ th instance of the flow graph with the set of tasks marked  $M_i$  is determined, as below [32]:

$$S_i = \prod_{m_v \in M_i} \xi(d_v - C_v(x)). \tag{4}$$

Probability that all tasks meet their deadlines for  $K$  instances of the flow graph is calculated, as follows [33]:

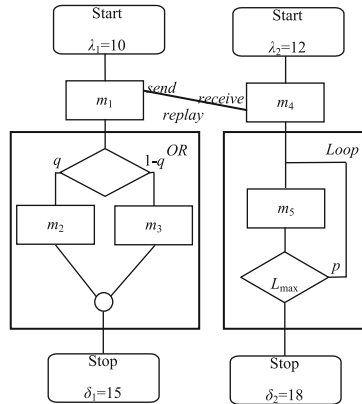


Fig. 4. A flow graph for two applications

$$P_D(x) = \sum_{i=1}^K p_i \prod_{m_v \in M_i} \xi(d_v - C_v(x)). \tag{5}$$

Figure 5 shows an example of a compromise configuration for its middleware zone in the *Comcute* grid that was found by the agent based on the harmony search for its area consisted on 14 modules divided among 2 computers.

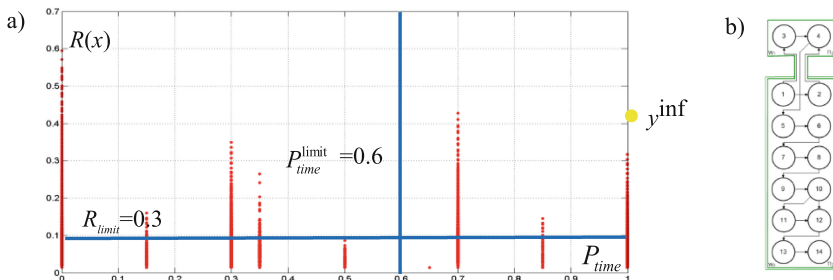


Fig. 5. A compromise configuration in the *Comcute* grid: (a) criterion space (b) a solution

## 7 Concluding Remarks and Future Work

Intelligent agents in the middleware of grid can significantly support efficiency of fault-tolerant self-configuration in grids. Agents based on harmony search can solve NP-hard multi-objective optimization problem of grid resource using to improve the level of fault-tolerance.

Our future works will focus on testing the other AI algorithms to find fault-tolerant configurations. Moreover, quantum-inspired algorithm can support big data, too [7].



## References

1. Afshari, S., Aminshahidy, B., Pishvaie, M.R.: Application of an improved harmony search algorithm in well placement optimization using streamline simulation. *J. Petrol. Sci. Eng.* **78**, 664–678 (2011)
2. Ahmed, A.M., Bakar, A.A., Hamdan, A.R.: Harmony search algorithm for optimal word size in symbolic time series representation. In: *Proceedings of Conference on Data Mining and Optimization, Malaysia*, pp. 57–62 (2011)
3. Ajith, A.P., Murthy, C.S.R.: Algorithms for reliability-oriented module allocation in distributed computing systems. *J. Syst. Softw.* **40**, 125–138 (1998)
4. Al-Betar, M.A., Khader, A.T., Zaman, M.: University course timetabling using a hybrid harmony search metaheuristic algorithm. *IEEE Trans. Syst. Man Cybern. Part C Appl. Rev.* **42**, 66–681 (2012)
5. Apache Hadoop. <http://hadoop.apache.org/>. Accessed 15 Mar 2016
6. Balicki, J.: Negative selection with ranking procedure in tabu-based multi-criterion evolutionary algorithm for task assignment. In: Alexandrov, V.N., Albada, G.D., Sloot, P.M., Dongarra, J. (eds.) *ICCS 2006*. LNCS, vol. 3993, pp. 863–870. Springer, Heidelberg (2006)
7. Balicki, J.: An adaptive quantum-based multiobjective evolutionary algorithm for efficient task assignment in distributed systems. In: Mastorakis N., et al. (eds.) *Recent Advances in Computer Engineering. Proceedings of the 13th WSEAS International Conference on Computers*, Rhodes, Greece, pp. 417–422 (2009)
8. Balicki, J., Korlub, W., Szymański, J., Zakidalski, M.: Big data paradigm developed in volunteer grid system with genetic programming scheduler. In: Rutkowski, L., Korytkowski, M., Scherer, R., Tadeusiewicz, R., Zadeh, L.A., Zurada, J.M. (eds.) *ICAISC 2014, Part I*. LNCS, vol. 8467, pp. 771–782. Springer, Heidelberg (2014)
9. Balicki, J., Kitowski, Z.: Multicriteria evolutionary algorithm with tabu search for task assignment. In: Zitzler, E., Deb, K., Thiele, L., Coello Coello, C.A., Corne, D.W. (eds.) *EMO 2001*. LNCS, vol. 1993, pp. 373–384. Springer, Heidelberg (2001)
10. Balicki, J., Korlub, W., Krawczyk, H., et al.: Genetic programming with negative selection for volunteer computing system optimization. In: Paja, W.A., Wilamowski, B.M. (eds.) *Proceedings the 6th International Conference on Human System Interactions*, Gdańsk, Poland, pp. 271–278 (2013)
11. BOINC. <http://boinc.berkeley.edu/>. Accessed 15 Mar 2016
12. Cao, L., Gorodetsky, V., Mitkas, P.A.: Agent mining: the synergy of agents and data mining. *IEEE Intell. Syst.* **24**, 64–72 (2009)
13. Comcute. <http://comcute.eti.pg.gda.pl/>. Accessed 25 Apr 2016
14. Dean, J., Ghemawat, S.: MapReduce: simplified data processing on large clusters. *Commun. ACM* **51**, 1–13 (2008)
15. Geem, Z.W., Kim, J.H., Loganathan, G.V.: A new heuristic optimization algorithm: harmony search. *Simulation* **76**, 60–68 (2001)
16. Gunarathne, T., et al.: Cloud computing paradigms for pleasingly parallel biomedical applications. In: *Proceedings of International Symposium on High Performance Distributed Computing*, Chicago, Illinois, pp. 460–469 (2010)
17. Guojun, L., Ming, Z., Fei, Y.: Large-scale social network analysis based on MapReduce. In: *Proceedings of International Conference on Computational Aspects of Social Networks*, pp. 487–490 (2010)

18. Huang, Z., Wang, C., Liu, L., Peng, Y.: Improve availability of fault-tolerant computing: optimal multi-task allocation in MapReduce. In: Proceedings of International Conference on Computer Science and Education, pp. 249–254 (2012)
19. Jennings, N.R., Wooldridge, M.: Applications of intelligent agents. In: Jennings, N.R., Wooldridge, M. (eds.) *Agent Technology*, pp. 3–28. Springer, New York (1998)
20. Kafil, M., Ahmad, I.: Optimal task assignment in heterogeneous distributed computing systems. *IEEE Concurr.* **6**, 42–51 (1998)
21. Kartik, S., Murthy, C.S.R.: Task allocation algorithms for maximizing reliability of distributed computing systems. *IEEE Trans. Comput.* **46**, 719–724 (1997)
22. Leyton-Brown, K., Shoham, Y.: *Multiagent Systems: Algorithmic, Game-Theoretic, and Logical Foundations*. Cambridge University Press, Cambridge (2008)
23. Li, H.X., Chosler, R.: Application of multilayered multi-agent data mining architecture to bank domain. In: Proceedings of International Conference on Wireless Communications, Networking and Mobile Computing, pp. 6721–6724 (2007)
24. Manjarres, D., et al.: A survey on applications of the harmony search algorithm. *Eng. Appl. Artif. Intell.* **26**, 1818–1831 (2013)
25. Mardani, S., Akbari, M.K., Sharifian, S.: Fraud detection in process aware information systems using MapReduce. In: Proceedings on Information and Knowledge Technology, pp. 88–91 (2014)
26. Marz, N., Warren, J.: *Big data - Principles and best practices of scalable realtime data systems* (2014)
27. O’Leary, D.E.: Artificial intelligence and big data. *IEEE Intell. Syst.* **28**, 96–99 (2013)
28. Ostrowski, D.A.: MapReduce design patterns for social networking analysis. In: Proceedings of International Conference on Semantic Computing, pp. 316–319 (2014)
29. Paluszak, J.: Optimizing the use of resources in distributed systems with grid architecture. Ph.D. dissertation, Gdańsk University of Technology, Gdańsk (2015). (in Polish)
30. Qiu, X., et al.: Using MapReduce technologies in bioinformatics and medical informatics. In: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, Portland (2009)
31. Sarvari, H., Zamanifar, K.: A self-adaptive harmony search algorithm for engineering and reliability problems. In: Second International Conference on Computer Intelligence, Modelling and Simulation, pp. 59–64 (2010)
32. Schneidewind, N.: Allocation and analysis of reliability: multiple levels: system, subsystem, and module. *Innov. Syst. Softw. Eng.* **2**, 121–136 (2006). Springer, London
33. Shatz, S.M., Wang, J.P.: Models and algorithms for reliability-oriented task-allocation in redundant distributed-computer systems. *IEEE Trans. Reliab.* **38**, 16–27 (1989)
34. Shvachko, K., et al.: The Hadoop distributed file system. In: MSST, pp. 1–10 (2010)
35. Snijders, C., Matzat, U., Reips, U.-D.: ‘Big Data’: big gaps of knowledge in the field of Internet. *Int. J. Internet Sci.* **7**, 1–5 (2012)
36. Shwe, T., Win, A.: A fault tolerant approach in cluster computing system. In: The 5th International Conference on Electrical Engineering/Electronics, Computer, Telecommunication and Information Technology, vol. 1, pp. 149–152 (2008)
37. Twardowski, B., Ryzko, D.: Multi-agent architecture for real-time big data processing. In: Proceedings of International Conference on Web Intelligence and Intelligent Agent Technologies, vol. 3, pp. 333–337 (2014)
38. Varvarigou, T., Trotter, J.: Module replication for fault-tolerant real-time distributed systems. *IEEE Trans. Reliab.* **47**(1), 8–18 (1998)
39. Vavilapalli, V.K.: Apache Hadoop Yarn: yet another resource negotiator. In: Proceedings of the 4th Annual Symposium on Cloud Computing, New York, USA, pp. 5:1–5:16 (2013)

40. Verbrugge, T., Dunin-Kępicz, B.: *Teamwork in Multi-Agent Systems. A formal Approach*. Wiley, London (2010)
41. Wang, L., Li, L.P.: A coevolutionary differential evolution with harmony search for reliability-redundancy optimization. *Expert Syst. Appl.* **39**, 5271–5278 (2012)
42. Węglarz, J., Błażewicz, J., Kovalyov, M.: Preemptable malleable task scheduling problem. *IEEE Trans. Comput.* **55**, 486–490 (2006)
43. Wooldridge, M.: *Introduction to MultiAgent Systems*. Wiley, London (2002)
44. Zhou, D., et al.: Multi-agent distributed data mining model based on algorithm analysis and task prediction. In: *Proceedings 2nd International Conference on Information Engineering and Computer Science*, pp. 1–4 (2010)
45. Zou, D., et al.: A novel global harmony search algorithm for reliability problems. *Comput. Ind. Eng.* **58**, 307–316 (2010)