Polyrhythmic Harmony Search for Workflow Scheduling

Mikhail Melnik, Tamara Trofimenko
ITMO University, Saint Petersburg, Russia
mihail.melnik.ifmo@gmail.com, tamaratrofimenko@gmail.com

Abstract
In this paper we propose the new conception of Harmony Search (HS) algorithm for scientific workflow scheduling called Polyrhythmic Harmony Search (PHS). The main idea of Harmony search is based on the artificial phenomenon found in jazz musical performance, namely the process of searching for better harmony. As it well known, jazz is characterized by the presence of several complex musical structures, such as polyrhythm, where a few different rhythmic patterns are played in the same time. According to technology of virtualization, which allows to divide physical computing resources into virtual machines, the main idea of the proposed algorithm is in the use of two heterogeneous harmonies – task scheduling on the computing environment and optimization of the computing environment for the tasks – as two different rhythms, which can be played in jazz. This algorithm is compared to several metaheuristic algorithms. Experimental results of PHS evaluation showed that proposed algorithm allows to find an optimal solution on a par with Genetic algorithm and with a greater convergence speed in terms of scheduling execution time.

Keywords: Workflow scheduling, Harmony search, Virtualization, Cloud computing

1 Introduction

Scientific community is under constant development and the complexity of tasks grows with more and more sophisticated research. To solve nowadays scientific problems it is frequently necessary to implement complex composite applications with division of initial task in subtasks, which can be processed on different computing resources. These composite applications can be represented as a workflow – directed acyclic graph (DAG), where each node denotes subtask and edges stand for dependencies between subtasks. There are several different optimization criteria for workflow scheduling. The most popular optimization criterion is makespan – total execution time of the workflow, what is essential for the urgent computing [17, 18]. Makespan can be significantly reduced by optimal scheduling of composite application processing in various heterogeneous computing

468 Selection and peer-review under responsibility of the Scientific Programme Committee of YSC 2015 © The Authors. Published by Elsevier B.V.
doi:10.1016/j.procs.2015.11.053
environments - clouds, grids and clusters [16]. There is variety of algorithms for solving such problems. On the one hand, there are heuristic algorithms such as Heterogeneous Earliest-Finish-Time (HEFT) [1, 2, 3], Critical-Path-on-a-Processor (CPOP) [1] or Lookahead [4]. Their basic idea is to create a sequence of tasks by assigning a certain rank or priority to each task, and then sort the list by this rank (priority). After the task queue is formed, the tasks can be selected one by one and assigned to a particular node by some set of rules. With the use of these algorithms the satisfactory solution can be obtained in a short time. Another class of algorithms are metaheuristic algorithms. These algorithms are used to find the most optimal solution but require more time for computation. The most popular among these algorithms is a Genetic Algorithm (GA) [5], but Particle Swarm Optimization (PSO) [6], Gravitational Search Algorithm (GSA) [7], Simulated Annealing (SA) [8] and Harmony Search (HS) [9] are also popular among scientists. The main idea of these algorithms is to generate the initial population and then to breed new and more optimal generations in accordance with the concept of the algorithm. In such computing environments as cloud computing, existing virtualization technology allows to divide or merge physical resources into virtual machines. Thus, computing environment can adapt to arising tasks.

HS is gaining popularity among scientists in the field of scheduling problems solution. Scientists are constantly trying to enhance the original concept of the algorithm for scheduling needs. In [10] authors proposed Hybrid HS to solve the flexible job shop scheduling problem. In this algorithm harmonies are presented as continuous vectors converted in two-vector code. Besides, a local optima search procedure is implemented to enhance algorithm efficiency. A local-best HS with dynamic sub-harmony memories [11] also uses the conversion of continuous vectors to discrete sequences and has specific scheme for initialization of Harmony Memory (HM) based on heuristics algorithm.

The main idea of proposed algorithm is inspired by real life musical phenomenon found in jazz musical performance - the process of searching for better harmony. Jazz is characterized by the use of multiple complex music structures like a polyrhythm - multiple simultaneous sounding of several incongruous rhythms [12]. The simple example of 3 over 4 polyrhythm is presented on the Figure 1. The Polyrhythmic Harmony Search algorithm proposed in this paper is based on HS. The main difference from original statement is the use of two heterogeneous harmonies, each of which optimizes different tasks. The first shares tasks between computing resources with a fixed set of computing resources, another is used to change the configuration of computational environment by changing a number or capacity of virtual resources. Proposed algorithm is compared with basic HS algorithm, GA, and Coevolution GA (CGA) [13, 15], which also can adapt computing environment to tasks.

2 Workflow scheduling

The simplest and the most popular way to describe a complex composite application is to design a high-level presentation via workflow. Workflow scheduling is a basic form of tasks scheduling where tasks are presented as a directed acyclic graph (DAG) $G(T, E)$, where $T$ - is a set of $m = |T|$ tasks $\{t_1, t_2, \ldots , t_m\}$ and $E$ - a set of directed links $\{e_{ij} | (t_i, t_j)\}$, where links represent internal data.
dependencies between tasks. Each task \( t \) represents an application with a certain computation complexity (execution cost), which is defined by million instructions (MIPS) or floating-point operations (FLOP). Each edge of the graph \( e \) is a limitation stating that the task with the smaller index should be processed earlier than a task with the higher index. Computing environment is presented as a set of \( n \) computing resources \( R = \{ r_1, r_2, ..., r_n \} \). Due to the virtualization technology, this set of computing resources \( R \) can be changed. The main goal of scheduling is to share \( m \) tasks between \( n \) computing resources with the view to optimize a result schedule \( S = \{ t_i, r_j, ST_{ij}, FT_{ij} \} \), where \( ST_{ij} \) is a start time of execution \( t_i \) on the \( r_j \) and \( FT_{ij} \) is a finish time of execution \( t_i \) on \( r_j \). In this work, the main optimization criterion is \( makespan = \max \{ FT_{ij} \} \).

### 3 Polyrhythmic Harmony Search

#### 3.1 Harmony Search approach

In HS algorithm firstly Harmony Memory(HM) is initialized as a set of solutions, and then new harmonies from HM are improvised with the aim to find the better solution [9].

The algorithm of HS can be presented as

1. The initialization of HM
2. While optimization criteria is not reached:
   a. Generation of new probe melody (randomly, from memory or by crossing the existing melodies with randomly initialized ones).
   b. This probe melody is modified.
   c. If a resulting probe melody is better than melodies from memory, then the probe melody replaces the worst melody in the HM.
3. Return the best melody in the HM as a result.

#### 3.2 Polyrhythmic Harmony Search (PHS) algorithm

As it was mentioned above, the main idea of the algorithm is based on the musician playing jazz. Jazz is known for its nonstandard musical structure with the use of such methods as polyrhythm, syncopation, swung notes. Polyrhythm means a simultaneous sounding of several different rhythmic patterns. In our algorithm this phenomenon can be presented as a search method for solutions for two optimization problems. The flowchart of proposed algorithm is presented on Figure 2.

In this algorithm two different Harmony Memories are used. First one contains information about the selection of optimal task schedule on fixed computing resources. The second one is combining a number and capacity of virtual resources. Thus, the complete solution is obtained by merging of two harmonies.
Figure 2: Polyrhythmic Harmony Search scheme

Primarily, two harmony memories are initialized randomly with one harmony obtained using HEFT algorithm. On each iteration for each HM a new harmonic is generated randomly, copied from HM or obtained by crossbreeding a random harmonic with a harmonic in HM. Thus, a full polyrhythmic harmony is constructed from these generated parts from each HM. Further, this probe harmony is improvised by using one of the improvisation operations either for schedule harmony or for resource harmony. This improvisation is repeated several times, however, only one type of improvisation (only for schedule harmony or only for resources harmony) can be applied on the one iteration. After improvisation, probe harmony replaces the worst harmony in HM, whose part of full solution was improvised during the iteration. To implement this algorithm it is necessary to construct several operations for new harmonies generation: random generation, crossbreeding two melodies (one generated randomly, one from HM). These operators should be constructed for each harmony type.

Two ways of schedule harmony improvisation are presented on Figure 3. The left one (a) shows the replacement of one task cluster node for another, another figure (b) indicates the process of exchange positions of two harmony elements.

Figure 3: Scheduling improvisation

Schedule harmonies crossbreeding is shown on Figure 4. Two points crossbreeding is presented - two randomly generated cut-points divide a harmony into three parts. The first and third parts are copied from first parent and the middle is taken from the remaining tasks of the another parent.

Figure 4: Scheduling crossbreeding
The next type of algorithm’s operators are operators for changing computing resources. Pool of computing resources is represented by amount of resources and their capacity. This resources pool has two restrictions for summary capacity of all resources and maximum capacity of each resource. Three ways for resource harmony improvisation are presented on Figure 5. The first one (a) introduces the case of resource deletion, (b) represents the addition of a resource from pool of available capacity, the last one (c) indicates the capacity reduction of one of resources.

![Figure 5: Resources improvisation](image)

Resource harmonies crossbreeding is shown on Figure 6. The nodes are copied from each parent in turn. The capacity of the last node is reduced with the aim to save the summary capacity of resource pool.

![Figure 6: Resources crossbreeding](image)

4 Experiments

Experiments to determine the effectiveness of the proposed algorithm PHS were conducted using our own simulator, which allows to simulate the process of the workflow execution on a set of computing resources. Descriptions of the most popular workflows in workflow scheduling field were taken from Pegasus [14] with different number of tasks (Montage with 25, 50, 75, 100 tasks;
CyberShake with 30, 50, 75 tasks; Inspiral with 30 tasks and Epigenomics with 24 tasks). Approximate time of tasks’ execution, volume of data transfer and dependencies between tasks are presented in these descriptions. Computing resources are estimated by their performance presented in arbitrary units. Bandwidth between all resources is constant, greater than zero. A set of computing resources has two restrictions: total capacity (TC), which means the maximum computing power of all resources, and max resource capacity (MRC) for each computing resource. Thus, the execution time of a task $t_i$ performed on a computing node $r_j$ can be evaluated as $\text{execTime}_{ij} = \text{execCost}_i \cdot \frac{\text{nodeCap}_j}{\text{idealCap}_i}$, where $\text{execCost}_i$ - approximate execution cost of task $t_i$ from workflow description, $\text{nodeCap}_j$ - resource $r_j$ capacity and $\text{idealCap}_i$ - constant, which determines the average resources’ capacity. Data transfer time between two computing tasks $t_i$ and $t_j$ can be estimated as $\text{transferTime}_{ij} = \frac{\text{requiredData}_{ij}}{\text{bandwidth}}$, where $\text{requiredData}_{ij}$ is a volume of data, which should be transferred from $t_i$ to $t_j$, and $\text{bandwidth}$ is a constant data transfer rate between any two nodes. Only one task can be executed on one node in the one moment of time. Total makespan of a schedule is represented as a finish time of the last executed task. In these experiments our optimization criterion and the aim of fitness function is the makespan. According to the technology of virtualization, a set of computing resources may be changed in PHS and CGA algorithms, and is a constant in HS and GA.

All experiments were executed 100 times. Harmony Memory size for HS and PHS is 10 harmonies (for each HM in PHS). Population size for GA and CGA is 100 and 50 chromosomes (for each population in CGA), respectively. Interactions number between populations for CGA is 200. Mutation and crossover operators in GA and CGA are similar to HS and PHS. Mutation and crossover occurrence probabilities are 0.9 and 0.6 accordingly and were selected during the sensitive analysis. Due to different computing weight of algorithms’ iterations, iteration number for HS and PHS is 10000 and 300 for GA and CGA. These iteration numbers were chosen this way with the view to obtain approximately equal computation time of all algorithms, due to the difference between the amount of operations (crossover, mutation for GA and crossbreeding, improvisation for HS) of each algorithm during one iteration. The solution provided by HEFT algorithm is embedded in initial HM or population for all algorithms, except CGA, where each population has only a part of full HEFT solution. Thus, the full HEFT solution may be found at the interactions between populations not immediately. TC for computing resources is 80 and MRC is 30. Initial set of resources has 4 nodes with capacities [10, 15, 25, 30]. The results of the experiments are presented in the Table 1.

| Workflow        | HEFT | GA  | HS  | CGA | PHS  |
|-----------------|------|-----|-----|-----|------|
| Montage_25      | 280  | 204 | 198 | 152 | 152  |
| Montage_50      | 344  | 304 | 289 | 274 | 286  |
| Montage_75      | 559  | 510 | 493 | 381 | 370  |
| Montage_100     | 578  | 453 | 446 | 479 | 459  |
| CyberShake_30   | 352  | 340 | 329 | 305 | 299  |
| CyberShake_50   | 484  | 461 | 457 | 461 | 455  |
| CyberShake_75   | 651  | 629 | 629 | 627 | 627  |
| Inspiral_30     | 2026 | 2025| 2015| 1934| 1918 |
| Epigenomics_24  | 6052 | 5967| 5963| 5360| 5355 |

Table 1: Results of the experiments
As can be seen from the table, proposed PHS algorithm outperforms other in the most cases, with the efficiency from 5% to 84% in compare to initial HEFT algorithm. It should be noted, that for Montage_100, the best performance is provided by HS, which means that an additional increase of the solution space by the computing environment adaptation may lead to less performance. In addition, in the Inspiral_30 and Epigenomics_24 cases, it can be seen that algorithms with static computing environment (HS and GA) virtually didn’t move from the initial HEFT solution, while algorithms with changing in the computing environment (PHS and CGA) may provide solutions with efficiency from 5 to 13% in these 2 cases.

Results of several experiments are presented on the next Figure 7 with the algorithms’ makespan on each iteration. Since the iterations number for HS and PHS were chosen 10000, each iteration for GA and CGA on these graphs presents 33 iterations for HS and PHS.
From the graphs it can be seen, that PHS algorithm has better convergence speed in average than CGA, as well as HS has better convergence speed relative to GA. Thus, it can be said that the proposed PHS can not only find a better solution, but also do it in less period of time.

5 Conclusion

The new conception of Harmony Search, called Polyrhythmic Harmony Search algorithm was proposed in this paper. This algorithm was applied for the workflow scheduling and compared to basic HS, popular GA and Coevolution GA algorithms. From the results it can be said, that proposed algorithm outperforms others in most cases with better convergence speed and performance improvement up to 84% in comparison to heuristic algorithm and up to 38% in comparison to other metaheuristic algorithms. Thus, ideas inspired by natural or artificial phenomena allow to achieve better efficiency in the encountered problems.

6 Acknowledgment

This paper is supported by Russian Foundation for Basic Research - 15-29-07034 "Technology of distributed cloud computing for the simulation of a large city»

References

1. Topcuoglu, Haluk, Salim Hariri, and Min-you Wu. "Performance-effective and low-complexity task scheduling for heterogeneous computing." Parallel and Distributed Systems, IEEE Transactions on 13.3 (2002): 260-274.
2. Durillo, J. J., & Prodan, R. (2014). Multi-objective workflow scheduling in Amazon EC2. Cluster Computing, 17(2), 169-189.
3. Durillo, J. J., Nae, V., & Prodan, R. (2014). Multi-objective energy-efficient workflow scheduling using list-based heuristics. Future Generation Computer Systems, 36, 221-236.
4. Arabnejad, H., & Barbosa, J. G. (2014). List scheduling algorithm for heterogeneous systems by an optimistic cost table. Parallel and Distributed Systems, IEEE Transactions on, 25(3), 682-694.

5. Leena, V. A., & Rajasree, M. S. (2016). Genetic Algorithm Based Bi-Objective Task Scheduling in Hybrid Cloud Platform. International Journal of Computer Theory and Engineering, 8(1), 7.

6. Areekijseree, K., & Achalakul, T. (2014, May). Volunteered mobile sourcing with multi-objective ant colony optimization. In Computer Science and Software Engineering (JCSSE), 2014 11th International Joint Conference on (pp. 248-253). IEEE.

7. Zarrabi, A., & Samsudin, K. (2014). Task scheduling on computational grids using gravitational search algorithm. Cluster Computing, 17(3), 1001-1011.

8. Yuan, Y., Xu, H., & Yang, J. (2013). A hybrid harmony search algorithm for the flexible job shop scheduling problem. Applied Soft Computing, 13(7), 3259-3272.

9. Pan, Q. K., Suganthan, P. N., Liang, J. J., & Tasgetiren, M. F. (2011). A local-best harmony search algorithm with dynamic sub-harmony memories for lot-streaming flow shop scheduling problem. Expert Systems with Applications, 38(4), 3252-3259.

10. Schuller, G. (1986). Early jazz: Its roots and musical development (Vol. 1). Oxford University Press.

11. Butakov, N., & Nasonov, D. (2014, October). Co-evolutional genetic algorithm for workflow scheduling in heterogeneous distributed environment. In Application of Information and Communication Technologies (AICT), 2014 IEEE 8th International Conference on (pp. 1-5). IEEE.

12. Deelman, E., Singh, G., Su, M. H., Blythe, J., Gil, Y., Kesselman, C., ... & Katz, D. S. (2005). Pegasus: A framework for mapping complex scientific workflows onto distributed systems. Scientific Programming, 13(3), 219-237.

13. Butakov N.A., Nasonov D., Boukhanovsky A.V. Evolutionary Inheritance in Workflow Scheduling Algorithms within Dynamically Changing Heterogeneous Environments. ECTA 2014 - Proceedings of the International Conference on Evolutionary Computation Theory and Applications. 2014. pp. 160-168.

14. Korkhov, V. V., Moscicki, J. T., & Krzhizhanovskaya, V. V. (2009). The user-level scheduling of divisible load parallel applications with resource selection and adaptive workload balancing on the grid. Systems Journal, IEEE, 3(1), 121-130.

15. Knyazkov, K. V., Nasonov, D. A., Tchurov, T. N., & Boukhanovsky, A. V. (2013). Interactive workflow-based infrastructure for urgent computing. Procedia Computer Science, 18, 2223-2232.

16. Nasonov, D., & Butakov, N. (2014). Hybrid Scheduling Algorithm in Early Warning Systems. Procedia Computer Science, 29, 1677-1687.