Energy-Aware Fast Scheduling Heuristics in Heterogeneous Computing Systems

Cesar O. Diaz, Mateusz Guzek, Johnatan E. Pecero, Gregoire Danoy and Pascal Bouvry
CSC Research Unit, University of Luxembourg, Luxembourg
{firstname.lastname}@uni.lu
Samee U. Khan
Department of Electrical and Computer Engineering
North Dakota State University, Fargo, ND 58108
{samee.khan@ndsu.edu}@uni.lu

ABSTRACT

In heterogeneous computing systems it is crucial to schedule tasks in a manner that exploits the heterogeneity of the resources and applications to optimize systems performance. Moreover, the energy efficiency in these systems is of a great interest due to different concerns such as operational costs and environmental issues associated to carbon emissions. In this paper, we present a series of original low complexity energy efficient algorithms for scheduling. The main idea is to map a task to the machine that executes it fastest while the energy consumption is minimum. On the practical side, the set of experimental results showed that the proposed heuristics perform as efficiently as related approaches, demonstrating their applicability for the considered problem and its good scalability.

KEYWORDS: Heterogeneous computing systems, energy efficiency, scheduling, optimization

1. INTRODUCTION

Modern-day computing platforms such as grid or cloud computing are composed of many new features that enable sharing, selection, and aggregation of highly heterogeneous resources for solving large scale and complex real problems. All these heterogeneous computing systems (HCS) are widely used as a cheap way of obtaining powerful parallel and distributed systems. However, the required electrical power to run these systems and to cool them is of a great interest due to different concerns. This results in extremely large electricity bills, reduced system reliability and environmental issues due to carbon emissions [1]. Therefore, energy efficiency in HCS is the great interest.

HCS comprises different hardware architectures, operating systems and computing power. In this paper, heterogeneity refers to the processing power of computing resources and to the different requirements of the applications. To take advantage of the different capabilities of a suite of heterogeneous resources, a scheduler commonly allocates the tasks to the resources and determines a date to start the execution of the tasks. In this paper, we assume that energy is the amount of power used over a specific time interval [2]. For each task the information on its processing time and the voltage rate of the processor to execute one unit of time is sufficient to measure the energy consumption for that task. In this context, we additionally promote the heterogeneity capability of the computing system to efficiently use the energy of the system [3–5]. The main idea is to match each task with the best resource to execute it, that is, the resource that optimizes the completion time of the task and executes it fastest with minimum energy.

The main objective of our work is to contribute to the efficient energy consumption in HCS. This target is achieved by providing a new set of scheduling algorithms. These algorithms take advantage of the resource capabilities and feature a very low overhead. The algorithms are based on list scheduling approaches and they are considered as batch mode dynamic scheduling heuristics [6]. In the batch mode, the applications are scheduled after predefined time intervals.

As a part of this work, we compare the proposed algorithms by analyzing the results of numerous simulations featuring high heterogeneity of resources, and/or high heterogeneity of applications. Simulations studies are performed to compare these algorithms with the min-min algorithm [7,8]. We used min-min as a basis of comparison because it is one of the most used algorithm in the literature in the context of HCS, and it has a good performance behavior [3, 8]. We have considered the minimization of the makespan (i.e., the maximum completion time) and energy as a basis of comparison. Most of related work consider only the makespan as a performance criterion. The goal has been to find a feasible schedule such that the total energy consumption over the entire time horizon is as small as possible. It gives insight into effective energy conservation, however, it ignores the important aspect that users typically expect good response times for their job [9]. In this context, we also compare these heuristics based on the *flowtime*. The flowtime of a task is the length of the time interval between the release time and completion time of the task. Flowtime is commonly used as a quality of service measure that allows guaranting good response times. The large set of experimental results shown that the investigated heuristics perform as efficiently as the related approach for most of the studied instances although their low running time, showing their applicability for the considered scheduling problem.

The remainder of this paper is organized as follows. The system, energy and scheduling models are introduced in Section 2. Section 3 briefly reviews some related approaches. We provide the resource allocation and scheduling heuristics in Section 4. Experimental results are given in Section 5. Section 6 concludes the paper.

2. MODELS

2.1. System and Application Models

We consider a HCS composed of a set of $M = \{m_1, ..., m_m\}$ machines. We assume that the machines are incorporated with an effective energy-saving mechanism for idle time slots [10]. The energy consumption of an idle resource at any given time is set using a minimum voltage based on the processor's architecture. In this paper, we consider two voltage levels: *maximum*, when the processor is performing work or it is in an active state and *idle* level, when processor is in an idle state. We consider a set of independent tasks $T = \{t_1, ..., t_n\}$ to be executed onto the system. The tasks are considered as an indivisible unit of workload. Each task has to be processed completely on a single machine. The computational model we consider in this work is the ETC model. In this model, it is assumed that we dispose of estimation or prediction of the computational load of each task, the computing capacity of each resource, and an estimation of the prior load of the resources. Moreover, we assume that the ETC matrix of size $t \times m$ is known. Each position $ETC[t_i][m_j]$ in the matrix indicates the expected time to compute task t_i on machine m_j. This model allows to represent the heterogeneity among tasks and machines. Machine heterogeneity evaluates the variation of execution times for a given task across the computing resources. *Low machine* heterogeneity represents computing systems composed by similar computing resources (almost-homogeneous). On the contrary, *high machine* heterogeneity represents computing systems integrated by resources of different type and capacity power. For the case of task heterogeneity, it represents the degree of variation among the execution time of tasks for a given machine. *Low task* heterogeneity models the case when tasks are quasi homogeneous (i.e., when the complexity, and the computational requirement of tasks are quite similar), they have similar execution times for a given machine. *High task* heterogeneity describes those scenarios in which different types of applications are submitted to execute in the heterogeneous computing system ranging from simple applications to complex programs which require large computational time to be performed. Additionally, the ETC model also tries to reflect the characteristics of different scenarios using different ETC matrix consistencies defined by the relation between a task and how it is executed in the machines according to heterogeneity of each one [8]. The scenarios are *consistent*, *semi-consistent* and *inconsistent*. The consistent scenario models the SPMD applications executing with local input data, that is if a given machine m_j executes any task t_i faster than machine m_k, then machine m_j executes all tasks faster than machine m_k. The inconsistent scenario represents the most generic scenario for a HCS system that receives different tasks, from easy applications to complex parallel programs. Finally, the semi-consistent scenario models those inconsistent systems that include a consistent subsystem.

2.2. Energy Model

The energy model used in this work is derived from the power consumption model in digital complementary metal-oxide semiconductor (CMOS) logic circuitry. The power consumption of a CMOS-based microprocessor is defined to be the summation of capacitive power, which is dissipated whenever active computations are carried out, short-circuit and leakage power (static power dissipation). The capacitive power (P_c) (dynamic power dissipation) is the most significant factor of the power consumption. It is directly related to frequency and supply voltage, and it is defined as [11]:

$$P_c = AC_{eff}V^2f, \tag{1}$$

where A is the number of switches per clock cycle, C_{eff} denotes the effective charged capacitance, V is the supply voltage, and f denotes the operational frequency. The energy consumption of any machines in this paper is defined as:

$$E_c = \sum_{i=1}^{n} AC_{ef}V_i^2fETC[i][M[i]], \tag{2}$$

where $M[i]$ represents a vector containing the machine m_j where task t_i is allocated, V_i is the supply voltage of the machine m_j. The energy consumption during idle time is defined as:

$$E_i = \sum_{j=1}^{m} \sum_{idle_{jk} \in IDLES_j} AC_{ef} Vmin_j^2 I_{jk}, \tag{3}$$

where $IDLES_j$ is the set of idling slots on machine m_j, V_{minj} is the lowest supply voltage on m_j, and I_{jk} is the amount of idling time for $idle_{jk}$. Then the total energy consumption is defined as:

$$E_t = E_c + E_i \tag{4}$$

2.2. Scheduling Model

The scheduling problem is formulated as follows. Formally, given the heterogeneous computing systems composed of the set of m machines, and the set of n tasks. Any task is scheduled without preemption from time $\sigma(t_i)$ on machine m_j, with an execution time $ETC[t_i][m_j]$. The task t_i completes at time C_i equals to $\sigma(t_i)$ + $ETC[t_i][m_j]$. The objective is to minimize the maximum completion time ($C_{max} = max(C_i)$) or makespan with minimum energy E_t used to execute the tasks. Additionally, in this paper we also aim to guarantee good response times. In this context, response time is modeled as flowtime. As we already mentioned, the flowtime of a task is the length of the time interval between the completion time and release time. We consider that the release time is 0 for all the tasks. Hence, the flowtime represents the sum of completion time of jobs, that is, $\sum_{i=1}^{n} C_i$, the aim is to minimize $\sum_{i=1}^{n} C_i$. Let us mention that the tasks considered in our study are not associated with deadlines, which is the case for many computational systems.

3. RELATED WORK

The job scheduling problem in heterogeneous computing systems without energy consideration has been shown to be NP-complete [7]. Therefore, a large number of heuristics have been developed. One of the most widely used batch mode dynamic heuristic for scheduling independent tasks in the heterogeneous computing system is the min-min algorithm. It begins by considering that all tasks are not mapped. It works in two phases. In the first phase, the algorithm establishes the minimum completion time for every unscheduled job. In the second phase, the task with the overall minimum expected completion time is selected and assigned to the corresponding machine. The task is then removed from the set and the process is repeated until all tasks are mapped. The run time of min-min is $O(t^2m)$.
Some strategies for energy optimization in HCS systems by exploiting heterogeneity have been proposed and investigated. In [12], the authors investigated the tradeoff between energy and performance. The authors proposed a method for finding the best match of the number of cluster nodes and their uniform frequency. However, the authors did not consider much about the effect of scheduling algorithms. The authors in [5] introduced an online dynamic power management strategy with multiple power-saving states. Then, they proposed an energy-aware scheduling algorithm to reduce energy consumption in heterogeneous computing systems. The proposed algorithm is based on min-min. In [3] the authors considered the problem of scheduling tasks with different priorities and deadline constrained in an ad hoc grid environment with limited battery capacity that used DVS for power management. In this context, the resource manager needed to exploit the heterogeneity of the tasks and resources while managing the energy. The authors introduced several online and batch mode dynamic heuristics and they showed by simulation that batch mode heuristics performed the best. These heuristics were based on min-min. However, they required significantly more time.

4. PROPOSED ALGORITHMS

In the scheduling problem on heterogeneous computing systems, near-optimal solutions would suffice rather than searching for optimality for most practical applications. Therefore, we investigate low-cost heuristics with good quality schedules and low energy consumption. These heuristics are based on the min-min algorithm. However, we took special care to decrease the computational complexity. The main idea is to avoid the loop on all the pairs of machines and tasks in the min-min algorithm corresponding to the first phase of the heuristic. One alternative is to consider one task at a time, the task that should be scheduled next. For that we propose to order the tasks by a predefined priority, so that they can be selected in constant time. Once the order of tasks is determined, in the second phase that we call the mapping event, we consider assigning the task to the machine that minimizes its expected completion time as well as its execution time. This modification lies essentially with the calculation of a mapping event of an application to a machine. We propose a weighted function, that we named the *score function* $SF(t_i, m_j)$ (see Eq. 5), that tries to balances both objectives. The rational is to minimize the workload of machines and intrinsically minimize the energy used to carry out the work. This is the main principle of the scheduling heuristics we are interested in this work. However, the main difference among them is the priority used to construct the list. To optimize the flowtime we apply the classical shortest processing time rule on each machine after the schedule is constructed.

4.1. Low-cost heuristics

Algorithm 1 depicts the general structure of the proposed heuristics. It is based on classical list scheduling algorithms for what well founded theoretical performance guarantees have been proven [7]. The heuristics start by computing the *Priority* of each task according to some objective (line 1). Hence, we compute the sorted list of tasks (line 2). The order of the list is not modified during the execution of the heuristics. Next, the heuristics proceed to allocate the tasks to the machines and determine the starting date for them (main loop line 3). One task at a time is scheduled. The heuristics always consider the task t_i at the top of the list (highest priority) and remove it from that (line 4). A score function $SF(t_i, m_j)$ for the selected task is evaluated on all the machines (lines 5 and 6). Then each heuristic selects the machine for which the value of the score function is optimized for task t_i and we schedule the task on that machine (line 8). It corresponds to the second phase of the min-min heuristic, with a different evaluation function. In the case of min-min, the evaluation function is only based on the completion time of the selected task on all the machines. Then, the algorithm selects the machine that gives the minimum completion time for that task and the task is assigned on that machine. The list is updated (line 9) and we restart the main loop. Once all task have been scheduled we apply the shortest processing time rule on all machines to optimize the flowtime (lines 11 and 12).
The score of each mapping event is calculated as in equation 5. For each machine m_j,

$$SF(t_i) = \lambda \cdot \frac{C_i}{\sum_{k=1}^{m} C_{ik}} + (1-\lambda) \cdot \frac{ETC[t_i][m_j]}{\sum_{k=1}^{m} ETC[t_i][m_k]}, \quad (5)$$

where $\sum_{k=1}^{m} C_{ik}$ is the sum of the completion time of the task t_i over all machines and $\sum_{k=1}^{m} ETC[t_i][m_k]$ is the sum of the expected time to complete of task t_i over all machines. The first term of equation 5 aims to minimize the completion time of the tasks t_i, while the second term aims to assign the task to the fastest machine or the machine on which the task takes the minimum expected time to complete. The heuristics differ on the objective used to compute the priorities. For that, *maximum* (Algorithm 2), *minimum* (Algorithm 3) and *average* (Algorithm 4) completion time of the task are used as if it was the only task to be scheduled on the computing system. Let's note that it corresponds to the execution time ($ETC[t_i][m_j]$) of task t_i on machine m_j. The name of the heuristics are MaxMax Min (Algorithm 2) (maximum completion time of tasks, sorted in decreasing order of its maximum completion time, and scheduled based on the minimum completion time). MinMax Min (Algorithm 3) (minimum completion time of tasks, sorted in decreasing order of its minimum completion time, and scheduled based on the minimum completion time). MinMean Min (Algorithm 4) (average completion time of tasks, sorted in decreasing order of its average completion time, and scheduled based on the minimum completion time).

Algorithm 1 Pseudo-code for the low-cost heuristics

```
1: Compute **Priority** of each task $t_i \in T$ according to some predefined objective;
2: Build the list L of the tasks sorted in decreasing order of Priority;
3: while L $\neq \emptyset$ do
4:     Remove the first task $t_i$ from $L$;
5:     for each machine $m_j$ do
6:         Evaluate Score Function SF($t_i$);
7:     end for
8:     Assign $t_i$ to the machine $m_j$ that optimize the Score Function;
9:     Update the list L;
10: end while
11: for all machine $m_j$ do
12:     Sort the tasks $t_k$ on $m_j$ in increasing ETC[$t_k$][$m_j$];
13: end for
```

Algorithm 2 Pseudo-code for heuristic (MaxMax min)

```
1: for all task $t_i$ do
2:     for each machine $m_j$ do
3:         Evaluate CompletionTime($t_i$, $m_j$);
4:     end for
5:     Select the maximum completion time for each task $t_i$;
6: end for
```

Algorithm 3 Pseudo-code for heuristic (MinMax min)

```
1: for all task $t_i$ do
2:     for each machine $m_j$ do
3:         Evaluate CompletionTime($t_i$, $m_j$);
4:     end for
5:     Select the minimum completion time for each task $t_i$;
6: end for
```

Algorithm 4 Pseudo-code for heuristic (MinMean min)

```
1: for all task $t_i$ do
2:     for each machine $m_j$ do
3:         Evaluate CompletionTime($t_i$, $m_j$);
4:     end for
5:     Compute the average completion time for each task $t_i$;
6: end for
```

4.2. Computational complexity of the heuristics

The computational complexity of the algorithms is as follows: computing the value of the priorities for the tasks and the construction of the sorted list have an overall cost of $O(tm\ log\ t)$. The execution of the main loop (line 3) in Algorithm 1 has an overall cost of $O(tm)$. Sorting the tasks for all the machines takes $O(km\ log\ k)$ (line 12), where $k \leq t$. Therefore, the asymptotic overall cost of the heuristics is $O(tm\ log\ t)$, which is less than one order of magnitude to the related approaches.

5. EXPERIMENTAL EVALUATION

In this section, we compare by simulations the proposed algorithms and min-min on a set of randomly built ETCs. Table 1 shows the twelve combinations of heterogeneity types (tasks and machines) and consistency classifications in the ETC model that we use in this paper. The consistency categories are named for the correspondent initial letter (*c* stands for consistent, *i* for inconsistent, *s* for semi-consistent, *lo* stands for low heterogeneity and *hi* for high heterogeneity). Hence, a matrix named *c_hihi* corresponds to a consistent scenario with hi task heterogeneity and hi machine heterogeneity.

Table 1. Consistency and heterogeneity combinations in the ETC model

Consistency		
Consistent	*Semi-consistent*	*Inconsistent*
c_lolo	s_lolo	i_lolo
c_lohi	s_lohi	i_lohi
c_hilo	s_hilo	i_hilo
c_hihi	s_hihi	i_hihi

5.1. Experiments

For the generation of these ETC matrices we have used the coefficient of variation based method (COV) introduced in [13]. To simulate different heterogeneous computing environments we have changed the parameters μ_{task}, V_{task} and $V_{machine}$, which represent the mean task execution time, the task heterogeneity, and the machine heterogeneity, respectively. We have used the following parameters: V_{task} and $V_{machine}$ equal to 0.1 for low case respectively and 0.6 for high case, and $\mu_{task} = 100$. The heterogeneous ranges were chosen to reflect the fact that in real situations there is more variability across the execution time for different tasks on a given machine than that across the execution time for a single task on different machines [14].
As we are considering batch mode algorithms, we assume in both cases that all tasks have arrived to the system before the scheduling event. Furthermore, we consider that all the machines are idle or available at time zero, this can be possible by considering advance reservation. We have generated 1200 instances, 100 for each twelve cases to evaluate the performance of the heuristics. We have generated instances with 512 tasks in size to be scheduled on 16 machines. Additionally, we have considered different voltages for the machines. We randomly assigned these voltages to machines by choosing among three different set. The first set considers 1.95 and 0.8 Volts for active state and idle state, respectively. The second set is 1.75 Volts at maximum state and 0.9 Volts at idle state. Finally, the last set considers 1.6 Volts for active level and 0.7 Volts at idle level.

5.2. Results

The results for the algorithms are depicted from Figure 1 to 3. We show normalized values of makespan, flowtime and energy for each heuristic against min-min for λ-values in the interval [0, 1]. The normalized data were generated by dividing the results for each heuristic by the maximum result computed by these heuristics. We only show the curves for the high task and high machine heterogeneity for the three different scenarios which are the most significant results. The legends *m-m n_mksp*, *m-m n_flow* and *m-m n_energy* in the figures stand for makespan, flowtime and energy of min-min.
We can observe from these figures that the proposed heuristics follow the same performance behavior according to the scenarios. Relative values range are biggest for the consistent instances than semi-consistent and inconsistent. The results clearly demonstrate that energy efficiency is the best for the consistent instances. It may be related to the fact, that the makespan has worse results. However, for value of λ = 0.8 the proposed heuristics can perform as well as min-min for all the three considered metrics. We can also observe that the proposed algorithms can improve makespan and flowtime results for lambda for semi-consistent and inconsistent instances. Interestingly, if the instance is more inconsistent, the new algorithms performs better. The benefit of exploiting the heterogeneity of the applications and resources to maximize the performance of the system and energy is more apparent. This is mainly because these instances are the ones presenting the highest inconsistency and heterogeneity. In terms of flowtime, all the heuristics are as efficient as min-min, however, the proposed heuristics have lower complexity.

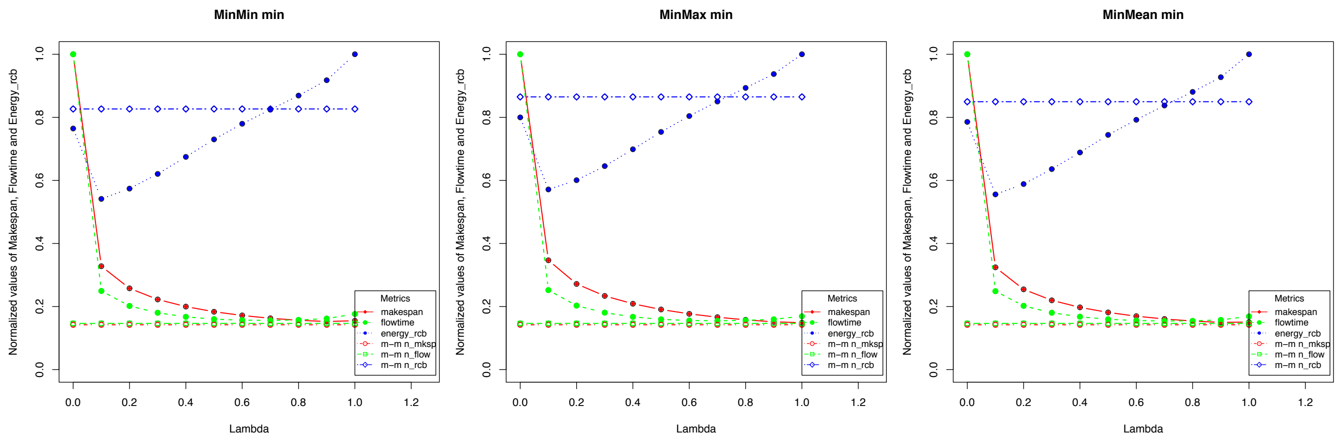

Figure 1. Relative performances of the schedules produced by different heuristics in the c_hihi instances.

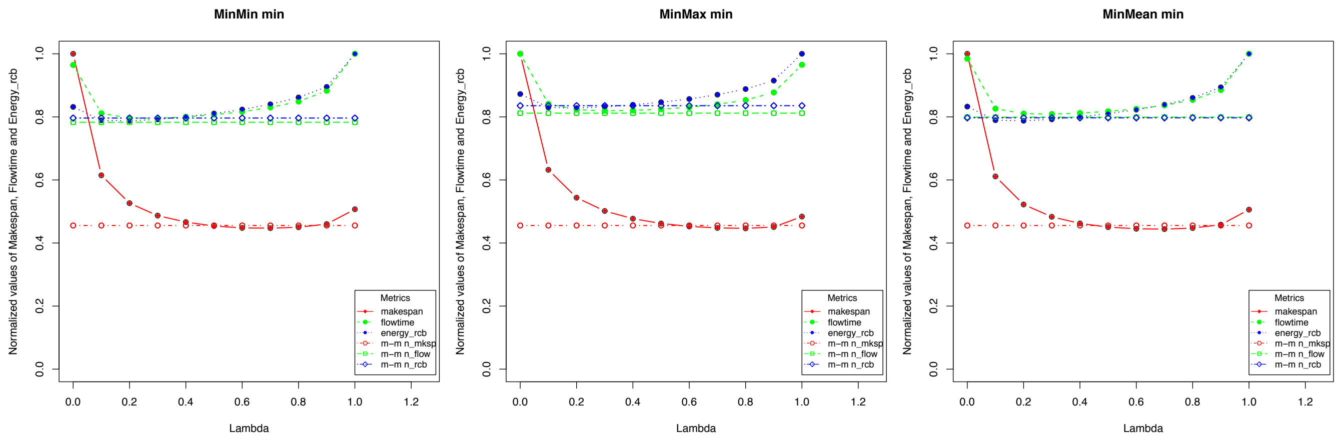

Figure 2. Relative performances of the schedules produced by different heuristics in the s_hihi instances.

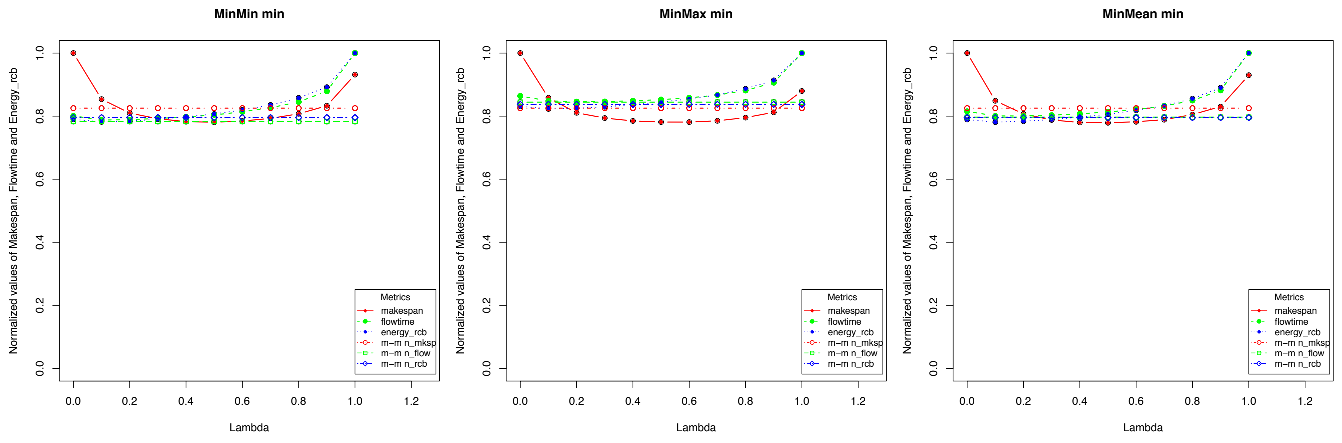

Figure 3. Relative performances of the schedules produced by different heuristics in the i_hihi instances.

6. CONCLUSIONS

This paper investgated three batch mode scheduling algorithms in the context of performance and energy efficiency in heterogeneous computing systems. These algorithms are based on the well-known list scheduling approaches. The set of experimental results showed that the investigated heuristics perform as efficiently as the related approaches although featuring lower complexity, lower running time, showing their applicability for the considered scheduling problem and their good efficiency.
As part of future work, we intend to implement the proposed heuristics in a packet-level simulator of energy-aware cloud computing data center, GreenCloud [15] and we plan to extend these heuristics to include thermal aspects. Additionally, we consider using dynamic frequency and voltage scaling techniques. We have considered that the estimated time to compute each task on every machine is known. However in most of modern computing systems performance perturbation should be considered. For that, we plan to investigate these heuristics in the context of robust scheduling.

AKNOWLEDGMENT

This work is supported by the National Research Fund (FNR) of Luxembourg through project Green-IT no. C09/IS/05 and the Luxembourg research grant scheme (AFR) through the grant no. PDR-08-010.

REFERENCES

[1] T.Heath, B.Diniz, E.V.Carrera, W.J.Meira, R.Bianchini, ”Energy conservation in heterogeneous server clusters, 10th ACM SIGPLAN symposium on Principles and practice of parallel programming, PPoPP 05. New York, NY, USA: ACM, 2005, pp. 186–195.

[2] P.Lindberg, J.Leingang, D.Lysaker, S.U.Khan, J.Li, ”Comparison and analysis of eight scheduling heuristics for the optimization of energy consumption and makespan in large-scale distributed systems,*Journal of Supercomputing*, vol. 53, April, 2010.

[3] J.-K. Kim, H. J. Siegel, A. A. Maciejewski, R. Eigenmann, ”Dynamic resource management in energy constrained heterogeneous computing systems using voltage scaling”, *IEEE Trans. on Parallel and Distrib. Syst.*, vol. 19(11), pp. 1445–1457, 2008.

[4] A. Al-Qawasmeh, T. Maciejewski, R. Roberts, H. J. Siegel, ”Characterizing task machine affinity in heterogeneous computing environments”, in IPDPS Workshops - Int Heterogeneity in Computing Workshop, Alaska, USA, 2011, pp. 1–11.

[5] Y. Li, Y. Liu, D. Qian, ”A heuristic energy-aware scheduling algorithm for heterogeneous clusters”, in ICPADS. IEEE, 2009, pp. 407413.

[6] H. J. Siegel, S. Ali, ”Techniques for mapping tasks to machines in heterogeneous computing systems”, J. Syst. Archit., vol. 46, pp. 627 639, 2000.

[7] O. H. Ibarra, C. E. Kim, ”Heuristic algorithms for scheduling independent tasks on nonidentical processors”, J. ACM, vol. 24, pp. 280 289, 1977.

[8] T. D. Braun, H. J. Siegel, N. Beck, L. L. Blni, M. Maheswaran, A. I. Reuther, J. P. Robertson, M. D. Theys, B. Yao, D. Hensgen, and R. F. Freund, ”A comparison of eleven static heuristics for mapping a class of independent tasks onto heterogeneous distributed computing systems”, J. Parallel Distrib. Comput., vol. 61, pp. 810–837, 2001.

[9] S. Albers, Energy-efficient algorithms, Commun. ACM, vol. 53, pp.86–96, May 2010.

[10] Y.C. Lee, A.Y. Zomaya, ”Energy efficient utilization of resources in cloud computing systems”, *The Journal of Supercomputing*, pp. 1–13, 2010.

[11] Y. C. Lee and A. Y. Zomaya, Minimizing energy consumption for precedence-constrained applications using dynamic voltage scaling, in 9th IEEE/ACM International Symposium on Cluster Computing and the Grid, 2009, pp. 92–99.

[12] F. Pan, V. W. Freeh, and D. M. Smith, Exploring the energy-time tradeoff in high-performance computing, in Proceedings of the 19th IEEE International Parallel and Distributed Processing Symposium (IPDPS05) - Workshop 11 - Volume 12, ser. IPDPS 05. Washington, DC, USA: IEEE Computer Society, 2005.

[13] S. Ali, H. J. Siegel, M. Maheswaran, D. Hensgen, and S. Ali, Representing task and machine heterogeneities for heterogeneous computing systems, Journal of Science and Engineering, vol. 3(3), pp. 195 207, 2000.

[14] P. Luo, K. Lu, Z. Shi, A revisit of fast greedy heuristics for mapping a class of independent tasks onto heterogeneous computing systems, J. Parallel Distrib. Comput., vol. 67, pp. 695714, 2007.

[15] D. Kliazovich, P. Bouvry, and S. U. Khan, Dens: Data center energy- efficient network-aware scheduling, in The 2010 IEEE/ACM International Conference on Green Computing and Communications (Greencom), Hangzhou, China, 2010.