Scheduling independent tasks on heterogeneous computing systems by optimizing various objectives

Christos Gogos · Christos Valouxis · Panayiotis Alefragis · Iordanis Xanthopoulos · Efthymios Housos

Abstract Scheduling tasks on a set of heterogeneous machines is a problem with both theoretical and practical interest. In this paper, the problem of scheduling independent tasks while trying to optimize four different objectives is studied. These objectives are makespan, flowtime, resource utilization and matching proximity. Four mathematical models are presented that optimize each individual objective. Then, certain objectives are combined forming a weighted sum single objective in order to reach good compromise solutions. A number of experiments were undertaken that demonstrated the applicability of using a mathematical integer programming solver in solving several instances of the problem. An interesting result is that when makespan is the single objective, as a side-effect, high resource utilization is obtained also.

Keywords scheduling · mathematical programming · independent tasks

C. Gogos
Technological Educational Institute of Epirus, Department of Computer and Informatics Engineering, Laboratory of Knowledge and Intelligent Computing, Arta, Greece
Tel.: +30-2681-050349
E-mail: cgogos@teiep.gr

C. Valouxis
Ministry of Education, Achaia Branch Office, Patras, Greece

P. Alefragis
Technological Educational Institute of Western Greece, Department of Computer and Informatics Engineering, Patras, Greece

I. Xanthopoulos
University of Patras, Department of Electrical and Computer Engineering, Patras, Greece

E. Housos
University of Patras, Department of Electrical and Computer Engineering, Patras, Greece

1 Introduction

Heterogeneous systems consist of several computers with different hardware characteristics and some kind of interconnect among them. A major purpose of assembling a heterogeneous system is to execute, fast and reliably, sets of tasks. A relevant term referring to such systems is Grid Computing [6]. The premise of executing multiple processing tasks of varying complexity, in a parallel and cost effective manner is appealing. Nevertheless, efficient task scheduling over a computing infrastructure is a significant problem that relates to the overall performance and the usage cost of grid computing systems [3]. Due to the importance of the problem, several approaches have been proposed in order to address it. Such approaches are heuristics [12], meta-heuristics [11] and techniques originating from Operations Research [7].

The problem examined in this paper is the Heterogeneous Computing Scheduling Problem (HCSP) which consists of assigning a set of independent tasks to available resources while trying to optimize certain objectives. The objectives that are used in this paper are the following four: makespan, which refers to the latest task's finish time, flowtime, which refers to the finishing times of every task, resource utilization, which measures the degree of how much the resources are utilized, and matching proximity, which is a metric of the proximity of a schedule to the one that assigns each task to the processor that executes it faster. Each one of the aforementioned objectives can be optimized independently or the combination of two or more of them can formulate a multi-objective optimization problem. The most common objective used as performance metric for evaluating scheduling algorithms is makespan.
This paper is organized as follows. In the next session the problem is introduced, and the four objectives are described in detail. Then, the mathematical formulation for each one of the problem's objectives is given. The next session examines the possibility of combining two or three objectives into a weighted sum. Next, experiments of applying a mathematical solver to the models in order to solve certain problem instances are presented. Finally, conclusions are discussed and future work is presented.

2 Problem description

Mapping is the term referring to the assignment of each task to a specific processor. The set of all tasks forms the meta-task. The objectives of the problem refer to this meta-task. In this paper, tasks are considered to be independent from each other. This means that each task can be executed without prior knowledge of other tasks' statuses. Moreover, the mapping of the meta-task takes place before the beginning of the execution of the tasks, so this problem can be described as static scheduling.
The model used for simulating the different Heterogeneous Computing environments is the Expected Time to Compute (ETC) model that was introduced by Ali et al. in [1]. In this model the ETC matrix is a matrix that consists of

as many rows as the number of tasks and as many columns as the number of processors. Each cell $ETC[t][p]$ indicates the expected time to compute task t in processor p. The ETC matrix is considered to be known.
The ETC Model defines three metrics: task heterogeneity, processor heterogeneity and consistency type. Task heterogeneity refers to the relation among execution times of the tasks. If the execution times are similar, the task heterogeneity is considered low else it is considered to be high. Processor heterogeneity takes into account the execution times of a single task at different processors. For example, if a processor executes a task significantly faster than another processor, the processor heterogeneity is considered high. The final metric is the type of consistency, which can be consistent, inconsistent or partially consistent. An ETC matrix is characterized as consistent if a processor executes a specific task faster than any other processor, then it should execute all tasks faster than the other processors. On the other hand, if the previous statement is not valid, the ETC matrix is considered to be inconsistent. Finally when a ETC matrix is inconsistent but there are subsets of it that are consistent, then the ETC matrix is characterized as partially consistent. Braun et al. in [2] proposed twelve instances of the ETC model with 512 tasks and 16 processors that we use as a testbed for our experiments.

2.1 Objectives of task scheduling

As mentioned before the objectives of task scheduling that are used in this paper are makespan, flowtime, resource utilization and matching proximity. Makespan denotes the time that the final task finishes. In other words, it is the completion time of the meta-task. Letting T be the set of tasks and P the set of processors, C_t is defined as the completion time of each task $t \in T$ in the resulting schedule. Then, makespan can be defined as:

$$makespan = max(C_t) \quad \forall t \in T \tag{1}$$

Flowtime on the other hand is the sum of the completion times of each task. It is a metric that can be used to estimate the Quality of Service of the system. It can be defined as:

$$flowtime = \sum_{t \in T} C_t \tag{2}$$

Since flowtime is expected to be in some cases a very large number, average flowtime can be used instead which is the quotient of the division of flowtime by the numbers of tasks T.

$$average\_flowtime = \frac{flowtime}{|T|} \tag{3}$$

Resource utilization is the degree of utilization of the resources. It is in the interest of the operators of the system to maximally utilize its resources.

Considering CP_p the completion time of each processor $p \in P$ in the resulting schedule, resource utilization can be defined as:

$$resource\ utilization = \frac{\sum_{p \in P} CP_p}{makespan * |P|} \tag{4}$$

Finally, matching proximity, measures the proximity of the schedule under examination to the schedule that has been derived from assigning each task to the processor that executes it faster. Formally, this is defined as:

$$matching\ proximity = \frac{\sum_{t \in T} ETC[t][p_t]}{\sum_{t \in T} ETC[t][p_{MET_t}]} \tag{5}$$

where p_t is the processor that task t is scheduled to and p_{MET_t} is the processor that the Minimum Execution Time (MET) heuristic, which assigns each task to the processor that has the smallest execution time for that task, would have scheduled task t to be. Since the numerator of equation 5 is always greater than or equal to the denominator the best value that can be achieved is 1, while bigger values indicate failure to achieve good values for this objective.

An example of a problem instance with four tasks and three processors is given in order to better describe the problem and the objectives. The ETC matrix alongside with two possible solutions is presented in figure 1. The first solution (schedule A) has makespan=22, flowtime=65, average flowtime=16.25, resource utilization=87.88% and matching proximity=1.0943. On the other hand the second solution (schedule B) has makespan=28, flowtime=70, average flowtime=17.5, resource utilization=63.10% and matching proximity=1. As a matter of fact, schedule A is the optimal solution according to makespan, flowtime and resource utilization, while schedule B is the optimal solution according to matching proximity.

In the following section a mathematical model formulating each one of the above objectives is provided.

3 Mathematical formulation

In order to formulate the mathematical models, a set of parameters and decision variables have to be defined first. So, parameters w_{tp} store the execution time of task $t \in T$ when executed at processor $p \in P$ and binary variables y_{tp} are defined over each $t \in T$ and $p \in P$. Each variable y_{tp} assumes value 1 if task t is scheduled to processor p or 0 otherwise.

3.1 Objective: Makespan

As already mentioned makespan is the most common objective used. The mathematical formulation of the problem with makespan as the objective is presented below.

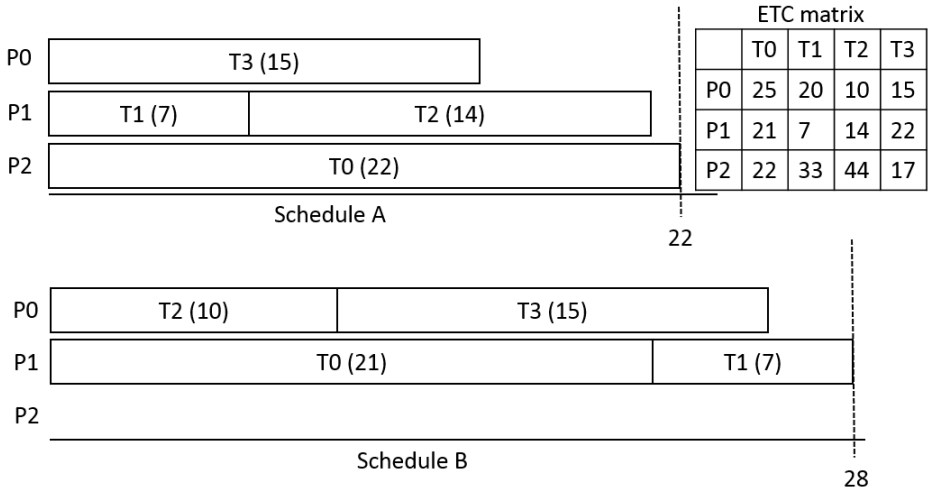

Fig. 1 Example of a small Heterogeneous Computing Scheduling Problem with two solutions

$$minimize\ m \tag{6}$$

$$\sum_{p \in P} y_{tp} = 1 \qquad \forall t \in T \tag{7}$$

$$m \geq \sum_{t \in T} w_{tp} y_{tp} \qquad \forall p \in P \tag{8}$$

Variable m assumes the maximum value among execution times over all processors which is by definition the makespan. Each task should be assigned to exactly one processor. This is enforced by equation 7. The right side of inequality 8 assumes the value of the total execution time of all tasks on processor p. So, variable m assumes the maximum value among execution times across all processors.

3.2 Objective: Flowtime

The finishing times of each task are needed in order to compute the flowtime. These values are not provided by the mathematical model that uses makespan as objective. It can be observed that if the processor that each task will be executed is known, the optimal sequence of execution among tasks on the same processor is defined by scheduling its tasks in increasing order of execution time. This fact is exploited by the mathematical formulation of the problem

that follows.

$$minimize \sum_{t\in T}\sum_{t\in P} w_{tp}y_{tp} + \left[\sum_{t\in T}\sum_{t'\in T}\sum_{p\in P} k_{tt'p}w_{t'p} \qquad \forall t,t' \in T, p \in P : t < t', w_{t'p} < w_{tp}\right] \tag{9}$$

$$\sum_{p\in P} y_{tp} = 1 \qquad \forall t \in T \tag{10}$$

$$k_{tt'p} - y_{tp} - y_{t'p} \geq -1 \qquad \forall t,t' \in T, p \in P : t \neq t', w_{t'p} < w_{tp} \tag{11}$$

A set of binary variables $k_{tt'p}$ are introduced that assume value 1 when tasks t and t' are scheduled at the same processor p. Otherwise, they assume value 0. So, for each processor p tasks are ordered by their execution times and for each task t for all tasks t' that are executed faster than t the expression $k_{tt'p}w_{t'p}$ is added to the objective function resulting in expression 9.

Each task should be scheduled in one processor as stated in equation 10. Moreover, for each pair of tasks t and t' and for each processor p such that $w_{t'p} < w_{tp}$ inequality 11 should hold.

3.3 Objective: Resource Utilization

Resource utilization measures the degree that the resources are utilized with respect to the schedule. Since the completion time of each processor is needed in order to compute this value, a new set of variables c_p are introduced to the model of section 3.1. These variables are defined over $p \in P$ and denote the finishing time of each processor. The model of the problem follows.

$$maximize \sum_{p\in P} c_p - |P|m \tag{12}$$

$$\sum_{p\in P} y_{tp} = 1 \qquad \forall t \in T \tag{13}$$

$$m \geq \sum_{t\in T} w_{tp}y_{tp} \qquad \forall p \in P \tag{14}$$

$$c_p \leq \sum_{t\in T} w_{tp}y_{tp} \qquad \forall p \in P \tag{15}$$

Since c_p participates in the objective function with positive sign it will assume the largest possible value not exceeding $\sum_{t\in T} w_{tp}y_{tp}$ at the optimal solution. Likewise since m has a negative sign its value will be the smallest possible.

3.4 Objective: Matching Proximity

Matching proximity measures the degree of proximity of a schedule to the schedule that results from assigning each task to the processor that can execute it faster. It can be easily programmed and the resulting heuristic is called Minimum Execution Time. The mathematical formulation of the model for this case follows:

$$maximize \sum_{t \in T} y_{tp_{best_t}} \tag{16}$$

$$\sum_{p \in P} y_{tp} = 1 \qquad \forall t \in T \tag{17}$$

where p_{best_t} is the processor with the fastest execution time for task t.

4 Combining objectives

Since HCSP is by its nature a multi-objective problem a solution approach that simultaneously tries to optimize more than one objectives seems to be a good idea. The objectives might be combined in pairs or even form a triplet of objectives. The objective pairs that were selected was the makespan and resource utilization pair and the matching proximity and resource utilization pair. Furthermore, the objectives selected for the triplet was makespan, resource utilization and matching proximity. Flowtime is not included in the above combinations since it increases the complexity of the resulting model beyond the size that can be directly managed by the current mathematical solvers capabilities.

The solution approach used here is to simple combine the objectives in a weighted sum and to merge the constraints present in each model into one. We acknowledge that more sophisticated methods exist that produce non-dominated sets of solutions (i.e. solutions that cannot be improved in one objective without sacrificing some other objective). Typically, these solutions consist an approximation of the Pareto front which is a set of high quality solutions. Then, one or more solutions from the set are selected [5]. Nevertheless, in our approach the goal was to test the applicability of combining the mathematical models in order to reach good solutions that might serve as initials solutions perhaps for other multi-objective methods.

5 Experiments

Experiments were performed against the 12 Braun et al. [2] datasets. Each dataset describes a problem with 512 tasks and 16 processors. Table 1 presents representative results from the bibliography regarding the makespan objective

Table 1 Braun et al. datasets [2] and results from the bibliography regarding makespan

Instances	MinMin	TPB	pμ-CHC	CPR
u_c_hihi.0	8460675.0	7575121.1	7381570.0	7372307.9
u_c_hilo.0	161805.4	155627.1	153105.4	152865.4
u_c_lohi.0	275837.4	248411.5	239260.0	238859.8
u_c_lolo.0	5441.4	5213.8	5147.9	5137.8
u_i_hihi.0	3513919.3	3046010.7	2938380.8	2930720.1
u_i_hilo.0	80755.7	74045.9	73387.0	73244.4
u_i_lohi.0	120517.7	105503.4	102050.6	101791.8
u_i_lolo.0	2785.6	2556.0	2541.4	2538.2
u_s_hihi.0	5160342.8	4233055.3	4103500.3	4094481.3
u_s_hilo.0	104375.2	98128.9	95787.4	95682.5
u_s_lohi.0	140284.5	130740.0	122083.3	121519.6
u_s_lolo.0	3806.8	3544.4	3433.5	3423.5

using the heuristics MinMin [9] and TPB [7] and the more time and resource connsuming approaches pμ-CHC [10] and CPR [7].

The computer used for running the experiments had an Intel Core i7 860 2.8GHz processor with 16GB RAM memory and run Windows 7 64 bit. The IP solver used was Gurobi 6.5.0 64 bit [8] and it was allowed to run with a time limit of 60 seconds for each problem instance.

5.1 Experiments over single objectives

For each one of the objectives we tried to apply the IP solver directly on the models described in section 2.1. Unfortunately, for the flowtime objective the size of the generated model was prohibitively large to be solved directly by the IP solver since it had over 2 million rows and 2 million columns. Nevertheless, the model had successfully produced results for smaller problem instances and solved problems up to 100 tasks and 16 processors.

In the following tables we present the values of all four objectives. In the last column (Matching Proximity) two values are presented. The first one corresponds to the value of equation 5 while the second one is the percentage of tasks that was scheduled at the processors that would have executed them faster.

Table 2 shows the results when the objective is makespan. It can be observed that as a side effect the resource utilization values approximate the best possible values.

Table 3 shows the results when the resource utilization is used as the objective. The betterment of the resource utilization values is negligible when compared with the values of Table 2. Moreover, values over the other 3 objectives are significantly worse than those in Table 2. This suggests that resource utilization might not be a good choice as an objective when interest on other features of the solution exist.

Table 2 Objective: Makespan

Instances	Makespan	Flowtime (avg)	Resource Utilization	Matching Proximity
u_c_hihi.0	7358942.8	2101022.2	0.9986	2.4768 (44.53%)
u_c_hilo.0	152850.4	54078.9	0.9996	2.0628 (17.19%)
u_c_lohi.0	238876.1	70479.4	0.9986	2.6267 (45.12%)
u_c_lolo.0	5136.6	17950.8	0.9996	2.0756 (16.21%)
u_i_hihi.0	2929297.6	700811.4	0.9987	1.0323 (87.89%)
u_i_hilo.0	73202.6	24445.9	0.9988	1.0083 (88.48%)
u_i_lohi.0	101688.0	24485.8	0.9974	1.0179 (88.28%)
u_i_lolo.0	2533.9	8551.5	0.9988	1.0056 (88.48%)
u_s_hihi.0	4082317.5	1028939.8	0.9973	1.4594 (65.43%)
u_s_hilo.0	95621.3	32023.9	0.9993	1.3101 (51.56%)
u_s_lohi.0	121018.5	29699.3	0.9973	1.4677 (64.65%)
u_s_lolo.0	3421.5	11755.9	0.9988	1.3050 (52.54%)
Averages			0.9986	1.5707 (59.20%)

Table 3 Objective: Resource Utilization

Instances	Makespan	Flowtime (avg)	Resource Utilization	Matching Proximity
u_c_hihi.0	13751868.3	4022725.5	0.9990	4.6303 (49.80%)
u_c_hilo.0	207950.3	76629.1	0.9994	2.8060 (30.08%)
u_c_lohi.0	426054.5	127359.2	0.9994	4.6886 (50.20%)
u_c_lolo.0	6992.6	23888.6	0.9991	2.8240 (19.34%)
u_i_hihi.0	28488233.0	9462383.8	0.9994	10.0459 (04.49%)
u_i_hilo.0	307459.7	101162.8	0.9993	4.2372 (05.66%)
u_i_lohi.0	932162.1	277334.1	0.9995	9.3506 (05.27%)
u_i_lolo.0	10202.9	34934.5	0.9992	4.0507 (05.08%)
u_s_hihi.0	20666644.6	6170283.8	0.9996	7.4045 (29.69%)
u_s_hilo.0	251178.4	86497.8	0.9994	3.4419 (17.77%)
u_s_lohi.0	626067.7	174016.4	0.9995	7.6097 (13.48%)
u_s_lolo.0	9427.7	32502.6	0.9996	3.5986 (16.02%)
Averages			0.9994	5.3907 (20.57%)

The results when matching proximity is used as the objective are presented in Table 4. It should be noted that the time needed by the IP solver in order to produce them was negligible but still much slower than the MET heuristic that gives the same results. Furthermore, the values of the other objectives are much worse than the ones produced using the two previous objectives.

5.2 Experiments over multiple objectives

Makespan and resource utilization are combined with equal weights in a weighted sum. The results are presented in Table 5. Since, it has been already showed in Table 4 that by optimizing makespan, resource utilization gets also optimized it is no surprise that the values of Tables 5 and 4 are quite similar. Indeed the values of resource utilization become marginally better but at the expense of slightly worse values for makespan and the other objectives.

Table 4 Objective: Matching Proximity

Instances	Makespan	Flowtime (avg)	Resource Utilization	Matching Proximity
u_c_hihi.0	47472299.4	9461581.5	0.0625	1.0000 (100.00%)
u_c_hilo.0	1185093.0	382392.4	0.0625	1.0000 (100.00%)
u_c_lohi.0	1453098.0	305853.9	0.0625	1.0000 (100.00%)
u_c_lolo.0	39582.3	124991.2	0.0625	1.0000 (100.00%)
u_i_hihi.0	4508506.8	722303.5	0.6286	1.0000 (100.00%)
u_i_hilo.0	96610.5	24958.6	0.7506	1.0000 (100.00%)
u_i_lohi.0	185694.6	24778.6	0.5366	1.0000 (100.00%)
u_i_lolo.0	3399.3	8658.3	0.7404	1.0000 (100.00%)
u_s_hihi.0	25162058.1	3045160.8	0.1109	1.0000 (100.00%)
u_s_hilo.0	605363.8	111590.3	0.1205	1.0000 (100.00%)
u_s_lohi.0	674689.5	84090.9	0.1219	1.0000 (100.00%)
u_s_lolo.0	21042.4	38543.0	0.1244	1.0000 (100.00%)
Averages			0.2820	1.0000 (100.00%)

Table 5 Objectives: Makespan + Resource Utilization

Instances	Makespan	Flowtime (avg)	Resource Utilization	Matching Proximity
u_c_hihi.0	7369427.4	2108228.9	0.9992	2.4819 (43.95%)
u_c_hilo.0	152832.9	54110.4	0.9997	2.0627 (17.38%)
u_c_lohi.0	239180.4	70583.2	0.9993	2.6318 (45.51%)
u_c_lolo.0	5142.2	17981.9	0.9995	2.0776 (15.82%)
u_i_hihi.0	2933390.3	705857.7	0.9996	1.0346 (87.11%)
u_i_hilo.0	73395.7	24704.3	0.9996	1.0118 (87.11%)
u_i_lohi.0	101678.8	24548.9	0.9993	1.0198 (88.28%)
u_i_lolo.0	2533.9	8567.5	0.9995	1.0063 (87.70%)
u_s_hihi.0	4098582.5	1033170.8	0.9996	1.4684 (64.84%)
u_s_hilo.0	95707.1	32125.1	0.9996	1.3117 (50.98%)
u_s_lohi.0	121346.2	30131.2	0.9991	1.4744 (62.50%)
u_s_lolo.0	3426.1	11780.1	0.9995	1.3076 (50.59%)
Averages			0.9995	1.5741 (58.48%)

Table 6 shows the results obtained when the resource utilization and the matching proximity objectives are combined into a weighted sum. The weight factor for resource utilization is 1. The weight factor for the matching proximity objective is $\frac{max-min}{|T|}$ where max and min are the maximum and minimum execution times of all tasks over all processors. These values were selected based on experimentation. These two objectives are in contrast one with the other. Results show that better matching proximity values are obtained when the datasets are of the inconsistent type. This is expected since there is less competition for the same resources. In general, the solutions obtained seem to be a good compromise between the two objectives especially when compared with values of Tables 3 and 4.

Finally, Table 7 shows the results when the objective is a weighted sum of makespan, resource utilization and matching proximity. The weight factor of makespan is $|P| + 1$, while the weight factors for the two other objectives are the same as in the previous paragraph. Makespan values obtained are much

Table 6 Objectives: Resource Utilization + Matching Proximity

Instances	Makespan	Flowtime (avg)	Resource Utilization	Matching Proximity
u_c_hihi.0	12733132.0	4078525.8	0.9992	4.2883 (66.41%)
u_c_hilo.0	240234.1	93804.9	0.9994	3.2415 (39.84%)
u_c_lohi.0	410558.7	129560.0	0.9993	4.5175 (64.84%)
u_c_lolo.0	8045.3	31517.6	0.9992	3.2493 (42.97%)
u_i_hihi.0	4938638.0	888468.6	0.9982	1.7393 (89.06%)
u_i_hilo.0	106572.7	29568.5	0.9985	1.4674 (83.79%)
u_i_lohi.0	168806.8	29548.3	0.9981	1.6910 (91.60%)
u_i_lolo.0	3176.1	9470.4	0.9979	1.2593 (90.04%)
u_s_hihi.0	9852759.2	2036427.6	0.9994	3.5295 (77.15%)
u_s_hilo.0	188574.3	55095.6	0.9991	2.5832 (69.53%)
u_s_lohi.0	288848.5	60014.7	0.9989	3.5089 (83.79%)
u_s_lolo.0	6466.8	18774.3	0.9992	2.4675 (70.90%)
Averages			0.9989	2.7952 (72.49%)

better than the ones in the two previous tables, while values for the other two objectives are also good.

Table 7 Objectives: Makespan + Resource Utilization + Matching Proximity

Instances	Makespan	Flowtime (avg)	Resource Utilization	Matching Proximity
u_c_hihi.0	7564660.2	2372454.0	0.9991	2.5473 (56.05%)
u_c_hilo.0	155760.3	56537.7	0.9996	2.1020 (27.34%)
u_c_lohi.0	245483.1	79167.0	0.9991	2.7006 (56.25%)
u_c_lolo.0	5150.4	18030.5	0.9998	2.0815 (18.36%)
u_i_hihi.0	2993769.5	708694.5	0.9979	1.0541 (92.77%)
u_i_hilo.0	73772.8	24688.3	0.9993	1.0167 (91.21%)
u_i_lohi.0	102758.5	24603.0	0.9988	1.0301 (92.77%)
u_i_lolo.0	2539.7	8608.9	0.9994	1.0085 (87.70%)
u_s_hihi.0	4205302.2	1105178.5	0.9984	1.5049 (75.20%)
u_s_hilo.0	96888.9	32882.3	0.9990	1.3272 (62.50%)
u_s_lohi.0	127125.7	33838.9	0.9990	1.5445 (75.98%)
u_s_lolo.0	3427.8	11766.4	0.9996	1.3085 (53.71%)
Averages			0.9991	1.6022 (65.82%)

6 Conclusions

In this paper we formulated four mathematical models that optimize different objectives for the Heterogeneous Computing Scheduling Problem. These objectives were makespan, flowtime, resource utilization and matching proximity. In the flowtime model, the finish time of each task had to be modeled. Since the finish time of each task depends on other tasks scheduled at the same processor, $O(T^2P)$ constraints were included into the model resulting into a model that was directly solvable only for small problem instances.

In general, focusing on a single objective should probably cause other objectives to get inferior values. Nevertheless, experiments showed that optimizing makespan resulted in highly optimized values for the resource utilization objective also but not vice versa. The objective that seemed to be more in odds with the other objectives was the matching proximity.

Three scenarios of combining objectives using weighted sums were tested. The first one combined makespan and resource utilization and expectedly the results were similar to the results obtained by optimizing makespan alone. The second pair of objectives examined was matching proximity and resource utilization and the results were good compromises regarding both objectives while better values for makespan and flowtime were obtained than the results of optimizing matching proximity and resource utilization in isolation. The last scenario involved makespan, resource utilization and matching proximity and gave good results when compared with the bi-objectives since it managed to keep makespan to low values while achieving overall good values for the other objectives.

We plan to incorporate our findings into a multi-objective approach for the problem exploiting the concepts of non dominated solutions and Pareto front. Solutions generated from our models might then be the initial solutions used in a multi-objective approach like NSGA-II [4] and SPEA2 [13]. We also plan to address much bigger problems than the ones in this paper using decomposition techniques like the ones described in [7].

References

1. Ali, S., Siegel, H.J., Maheswaran, M., Hensgen, D.: Task execution time modeling for heterogeneous computing systems. In: Heterogeneous Computing Workshop, 2000.(HCW 2000) Proceedings. 9th Heterogeneous Computing Workshop, pp. 185–199. IEEE (2000)
2. Braun, T.D., Siegel, H.J., Beck, N., Bölöni, L.L., Maheswaran, M., Reuther, A.I., Robertson, J.P., Theys, M.D., Yao, B., Hensgen, D., et al.: A comparison of eleven static heuristics for mapping a class of independent tasks onto heterogeneous distributed computing systems. Journal of Parallel and Distributed computing **61**(6), 810–837 (2001)
3. Buyya, R., Abramson, D., Giddy, J., Stockinger, H.: Economic models for resource management and scheduling in grid computing. Concurrency and computation: practice and experience **14**(13-15), 1507–1542 (2002)
4. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A fast and elitist multiobjective genetic algorithm: Nsga-ii. Evolutionary Computation, IEEE Transactions on **6**(2), 182–197 (2002)
5. Ferreira, J.C., Fonseca, C.M., Gaspar-Cunha, A.: Methodology to select solutions from the pareto-optimal set: a comparative study. In: Proceedings of the 9th annual conference on Genetic and evolutionary computation, pp. 789–796. ACM (2007)
6. Foster, I., Kesselman, C.: The Grid: Blueprint for a New Computing Infrastructure, vol. 1. Morgan Kaufmann (2004)
7. Gogos, C., Valouxis, C., Alefragis, P., Goulas, G., Voros, N., Housos, E.: Scheduling independent tasks on heterogeneous processors using heuristics and column pricing. Future Generation Computer Systems **60**, 48–66 (2016)
8. Gurobi Optimization, I.: Gurobi optimizer reference manual (2015). URL http://www.gurobi.com
9. Ibarra, O.H., Kim, C.E.: Heuristic algorithms for scheduling independent tasks on nonidentical processors. Journal of the ACM (JACM) **24**(2), 280–289 (1977)

10. Nesmachnow, S., Cancela, H., Alba, E.: A parallel micro evolutionary algorithm for heterogeneous computing and grid scheduling. Applied Soft Computing **12**(2), 626–639 (2012)
11. Xhafa, F., Abraham, A.: Meta-heuristics for grid scheduling problems. Springer (2008)
12. Xhafa, F., Barolli, L., Durresi, A.: Batch mode scheduling in grid systems. International Journal of Web and Grid Services **3**(1), 19–37 (2007)
13. Zitzler, E., Laumanns, M., Thiele, L., Zitzler, E., Zitzler, E., Thiele, L., Thiele, L.: SPEA2: Improving the strength pareto evolutionary algorithm (2001)