© International Journal of Combinatorial Optimization Problems and Informatics, Vol. 7, No. 3, Sep-Dec 2016, pp. 7-19. ISSN: 2007-1558.

Branch and Bound Algorithm for the Heterogeneous Computing Scheduling Multi-Objective Problem

J. Carlos Soto-Monterrubio[1,2], Alejandro Santiago[1,2], H. J. Fraire-Huacuja1, Juan Frausto-Solís[1], J. David Terán-Villanueva[1]
[1]Tecnológico Nacional de México, Instituto Tecnológico de Ciudad Madero
[2]Universidad de Cádiz
E-mail: soto190@gmail.com, alejandro.santiagopi@mail.uca.es, automatas2002@yahoo.com.mx, juan.fruasto@gmail.com, david_teran01@yahoo.com.mx

A la memoria de Jesús Vicente Flores Morfín

Abstract. In this paper the Pareto optimization of the Heterogeneous Computing Scheduling Multi-Objective Problem (HCSMOP) is approached. The goal is to minimize two objectives which are in conflict: the overall completion time (makespan) and the energy consumed. In the revised literature, there are no reported exact algorithms which solve the HCSMOP. In this work, we propose a Branch and Bound algorithm to solve the problem and it is used to find the optimal Pareto front of a set of instances of the literature. This set is the first available benchmark to assess the performance of multiobjective algorithms with quality metrics that requires known the optimal front of the instances.

Keywords: Scheduling, optimal Pareto front, exact algorithm, branch and bound**.**

1 Introduction

The Heterogeneous Computing Systems are conformed by an interconnection of diverse resources, such as processors or machines, with the aim to provide a high computing power. These systems are used in a wide range of research areas to solve complex problems which require a high computing power. However, in these systems the economic cost and the heat generated increases while more energy is used to compute a set of tasks. For example, The data centers in United States of America in 2014 consumed an estimated of 70 billion kWh, representing about 1.8% of total United States of America electricity consumption [1].

In this paper, we approach the Heterogeneous Computing Scheduling Multi-Objective Problem (HCSMOP) with independent tasks. It consists in minimizing the total completion time (makespan) [2], and the total energy consumed [3]. Both objectives are in conflict because if the energy consumption is reduced a slower processors performance is used and higher computing times are required for the tasks, and vice versa; to reduce the makespan, it requires to increase the energy applied. The mono objective version of this problem and their variants have been widely approached with diverse metaheuristics [4 – 10], and also by intelligent systems [11]. To solve the HCSMOP we propose an exact algorithm, the Branch and Bound (B&B) because there have been no reported exact solution methods for the multi-objective version of this problem. Also, multi-objective problems are rarely tackled by exact methods [12]. A benchmark with unknown optimal Pareto front (PF) was chosen to evaluate the proposed algorithm [13, 14]. The main contributions in this work are: a multi-objective B&B algorithm based on Pareto dominance and the the first available benchmark for HCSMOP that include the optimal front of the instances.

Received Jan 29, 2016 / Accepted Aug 21, 2016
Editorial Académica Dragón Azteca (EDITADA.ORG)

The paper is organized as follows. Section 2 describes the HCSMOP formulation, the problem representation, and the solution presentation. Multi-objective optimization concepts and background are found in Section 3. Section 4 describes the main functions of the proposed B&B algorithm. The experiment, the benchmark and the results are shown in Section 5. Finally, in Section 6 the conclusions and future works are presented.

2 Problem description

In this section the makespan and energy model are described. The problem representation is presented in Section 1.1. In Section 1.2 the solution representation is detailed.

HCSMOP consists in searching the best allocation of a set of task in several machines, in such way that makespan and energy consumed are minimal [2]. The machines are Dynamic Voltage and Frequency Scaling (DVFS) enable [3], heterogeneous, and the tasks are independent.

To model HCSMOP, let a set of tasks $T = \{t_1, t_2, \dots, t_n\}$, a set of heterogeneous machines $M = \{m_1, m_2, \dots, m_k\}$, and the execution times of each tasks in each machine $P_{i,j} = \{p_{1,1}, p_{1,2}, \dots, p_{n,k}\}$. Since the machines uses DVFS technology, for each machine m_k there are different levels of configuration of voltage with an associated relative speed. If the maximum voltage is applied, then the speed associated to the machine is 1. When a lesser voltage is applied, then the associated speed decreases; e.g., for 80% of voltage consumption the speed of the computer might work at 0.8 (80% of the normal speed). The relative execution time $P'_{i,j}$ computed as:

$$P'_{i,j} = \frac{P_{i,j}}{speed}. \quad (1)$$

The energy model is derived from the energy consumption Complementary Metal-Oxide Semiconductor (CMOS) [15]. Therefore, the energy consumption is given by

$$E_c = \sum_{i=0}^{n} V_{l,i}^2 P'_{i,j} \ \forall j \in M . \quad (2)$$

where l is the index of the selected configuration in a machine with a voltage V_i.

The makespan is the required time for the completion of all the tasks and it is given by equation (3).

$$MAX_{j=1}^{k} \left\{ \sum P'_{i,j} \ \forall \ t_i \in \ m_j \right\}. \quad (3)$$

The objective functions to minimize are given by equations (2) and (3).

1.1 Problem representation

In this problem, $P_{i,j}$ are given as show in Table 1. Where the first column is the number of the task, the second, third, and fourth columns are the execution times of the task in each machine.

Table 1. Sample of processing times.

Task	M_0	M_1	M_2
0	15	14	12
1	12	10	7
2	10	15	11
3	9	18	12
4	13	9	15
5	7	12	9
6	14	11	15
7	10	18	15

The DVFS configurations are given as in Table 2. The first column is the level (l) of the configuration. The next columns contain the voltage (v) and the frequency (f) configuration for each machine. Notice that not all the machines have the same number of configurations.

Table 2. DVFS configurations for three machines [8].

l	M_0		M_1		M_2	
	v	f	v	f	v	f
0	1.75	1.00	1.50	1.00	2.20	1.00
1	1.40	0.80	1.40	0.90	1.90	0.85
2	1.20	0.60	1.30	0.80	1.60	0.65
3	0.90	0.40	1.20	0.70	1.30	0.50
4	-	-	1.10	0.60	1.00	0.35
5	-	-	1.00	0.50	-	-
6	-	-	0.90	0.40	-	-

Also, the Table 2 is represented by two matrixes, one contains the voltages and the other the frequencies. If the instance has more machines then a round-robin is applied, that means, the configurations of the first machine are copied to the fourth machine, the configurations of the second machine are copied to the fifth machine, and so on.

The Table 3 is a mapping of the levels configuration of the machines in Table 2. The first column is the mapping number, the second column is the corresponding machine, and the third column is the level configuration of the machine. For example, the mapping 1 is M_0 with $l = 1$, which implies that a voltage of 1.40 and a frequency of 0.80 is applied.

Table 3. Mapping configurations for three machines.

Mapping	M	l
0	0	0
1	0	1
2	0	2
3	0	3
4	1	0
5	1	1
6	1	2
7	1	3
8	1	4
9	1	5
10	1	6
11	2	0
12	2	1
13	2	2
14	2	3
15	2	4

With this mapping, we have a value which indicates the machine and the configuration to execute a task. Therefore, the range of the variables in the solution representation is the number of mappings.

1.2 Solution representation

The solution representation is an array of length *n*, where the first position corresponds to the first task, the second position to the second tasks, and so on. Each position contains the mapping indicating the machine and the configuration assigned to execute a task. Table 4 shows an example with eight tasks and the mappings from Table 3.

Table 4. Solution representation with eight tasks and 16 configurations.

T_0	T_1	T_2	T_3	T_4	T_5	T_6	T_7
0	4	7	0	3	13	2	11

From Table 4, T_0 uses the mapping 0 which means that it is executed in M_0 with $l = 0$ (level configuration 0 of Machine 0 is a voltage of 1.75 and a frequency of 1.00), T_1 uses the mapping 4 which indicates that it is executed in M_1 with $l = 0$, T_2 with the mapping 7 it is executed in M_1 with $l = 3$, and so on until all the tasks are allocated. This structure is used in the Branch and Bound algorithm.

3 Multi-objective optimization

The Pareto dominance and the basic concepts of multi-objective optimization are presented in this section [16, 17].

Definition 1. *Multi-objective Optimization Problem (MOP)*: Given a function vector $\vec{f}(\vec{x}) = [f_1(\vec{x}),\ f_2(\vec{x}), \ldots, f_k(\vec{x})]$ and a feasible space solutions Ω, a MOP consists on finding a vector $\vec{x} \in \Omega$ such that the vector function $\vec{f}(\vec{x})$ is optimized.

Definition 2. *Pareto dominance:* A vector $\vec{x}$ dominates $\vec{x}'$ (denoted by $\vec{x} \prec \vec{x}'$) if $f_i(\vec{x}) \leq f_i(\vec{x}')$ for all i in $\vec{f}$ and there exist at least one i such that $f_i(\vec{x}) < f_i(\vec{x}')$.

Definition 3. *Pareto optimum:* A vector $\vec{x}^*$ is a Pareto optimum if does not exist any $\vec{x} \in \Omega$ such that $\vec{x} \prec \vec{x}^*$.

Definition 4. *Optimum Pareto set:* Given a MOP, the optimum Pareto set is defined as $P^* = \{\vec{x}^* \in \Omega\}$.

Definition 5. *Pareto front.* Given a MOP and an optimum Pareto set P^*, the Pareto front is defined as $PF = \{f(\vec{x}) | \vec{x} \in P^*\}$.

4 Branch and bound algorithm

In this section the Multi-Objective B&B algorithm is described. In Section 4.1 the search tree and nodes are described. The lower bound and upper bound are presented in Section 4.2. The branching and pruning functions are explained in Section 4.3. The main functions of the B&B algorithm are detailed in Section 4.4. We adapt the algorithm from [18] to multi-objective optimization to solve the HCSMOP.

The B&B algorithm is widely used to solve combinatorial optimization problems because it allows reducing considerably the computation time needed to explore all the solution space. The B&B represents the solution space as a tree. The construction and exploration of the tree is done by the branching, bounding, selection, and pruning operators. The branching operator is responsible for the enumeration of all the solutions which will be evaluated. The selection operator chose the branch to explore. The pruning operator eliminates the branches which are not candidates to improve the bounding.

The adaptation of B&B algorithm to solve MOPs uses two functions: the function which verifies the improvement of the upper bound, and the function which updates the upper bound (UB). In the case of mono-objective optimization problems the upper bound is just one value, while in multi-objective problems the upper bound is a set of values which compose the PF. The B&B proposed is an extension of [19], where each objective was evaluated by an independent B&B. We propose a B&B which evaluates both objectives at the same time applying the dominance operator.

4.1 The search tree

The proposed B&B models the solution space by a tree where each branch represents a solution. The number of levels of the tree is equals to number of tasks (n). Each level corresponds to a position in the solution and each node of the tree is a mapping of the configuration. The number of descendants in each node is r, where r is the number of mappings. The number of nodes is $\sum_{q=1}^{n} r^q$ for a problem with n tasks and r mappings. The Figure 1 presents an example of the search tree for three machines and four mapping configurations.

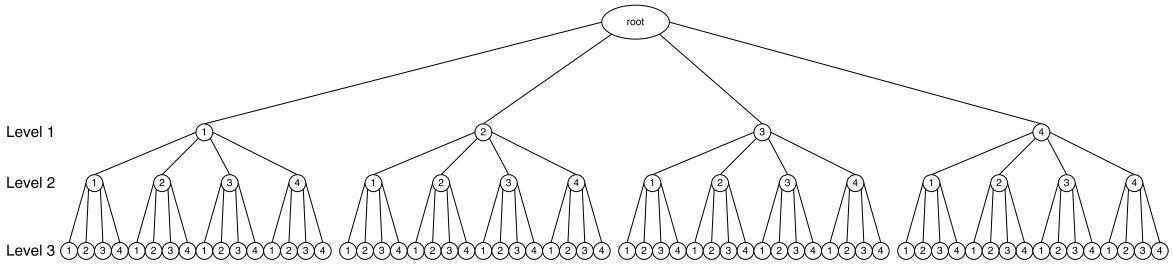

Fig. 1. Search tree generated by MOB&B for a problem with three machines and four configurations.

The search tree shown in Figure 1 has $n = 3$ levels, each node has $r = 4$ descendant nodes and $\sum_{q=1}^{3} 4^q = 84$ nodes in total. A complete solution is formed by going through the top (root node) to the last level (bottom of the tree). For example, the solution $\varphi = (1, 2, 3)$ is obtained by vising the node 1 from level 1, node 2 at level 2 (descendant of 1), and node 3 at the last level (descendant of 2).

The B&B performs a depth-first-search in the tree to visit all the solutions of an instance. The B&B uses a last-in-first-out stack (*TreeStack*) to store the nodes to visit, and a stack (*TreeLevel*) to store the corresponding level of the node to visit. All the structures used in the B&B are [18]:

- *TreeStack* is the active list of nodes, it is a vector structure which contains the pending tree nodes to be visited.
- *TreeLevel* is a vector structure which grows as *TreeStack* does. It stores the level in the search tree of the corresponding node in *TreeStack*. Thus, the *i*-th element of *TreeLevel* indicates the level of the *i*-th node in *TreeStack*.
- *Solution* is a vector of size *n*. It stores the visited nodes in the specified order. Thus, the node visited at level *p* is placed at *p*-th position in Solution.
- *UpperBound* is a list of the non-dominated solutions found. At the end of the exploration it will contain the Pareto front.

4.2 Lower bound and upper bound

The lower bound (LB) has associated the objective values of a node of the search tree. To decide whether to continue exploring from the active node or not the B&B computes the lower bound of the active node. The objective values of a solution *S* are obtained by evaluating *S* from the first tree level to *p*-th level. The objective values of the non-dominated solutions compose the upper bound (UB), if the lower bound is non-dominated by the upper bound, then the exploration continues.

At the beginning of the algorithm UB has one solution where all the tasks are fixed with the mapping 0. When a new solution is added to the UB all the dominated solutions are removed.

4.3 Branching and pruning functions

The branching functions allows the algorithm to found new solutions that could improve the UB. The prune functions avoid to guide the exploration to worse solutions. The branching function is performed when LB is non-dominated with respect to all the solutions in UB, or if LB dominates at least one solution in the UB, that means that the solution *S* could produce an improvement in UB when S is completed. The branching function push all the descendants of the visited node into *TreeStack* in reverse order. Also for each node pushed into *TreeStack*, its corresponding level is pushed into *TreeLevel*.

The pruning function is performed when one solution in the UB dominates the LB. The pruning function implies not to perform the branching function avoiding to push the descendants of the visited node into *TreeStack*.

4.4 Multi-objective B&B algorithm

In this section, we describe the main functions of the B&B algorithm. We start with the initialize function, the Algorithm 1 shows how the *StrackTree* and *StackLevel* are initialized. They are initialized with the mapping values in reverse order with the first level of the tree. With that the *StackTree* has all the pending nodes to visit.

Algorithm 1. Function that initialize the B&B.

```
1: program initialize ()
2:   S ← empty solution;
3:   UB ← empty front;
3:   level ← 0;                          ▷ The first level of the tree.
4:   for node = NumberOfMappings to 0 do ▷ To push in reverse order.
5:       push (StackTree, node);
6:       push (StackLevel, level);
7: end.
```

The algorithm 2 shows how the branching function is performed. The function push into *StackTree* all the descendants of the visited node with their corresponding level in *StackLevel*. In the counterpart, as was mentioned before, the prune function does nothing.

Algorithm 2. Branching function.

```
1: program branching (Solution S, integer currentLevel)
2:   nextTreeLevel ← currentLevel + 1;
3:   for i = NumberOfMappings to 0 do    ▷ To push in reverse order.
4:       push (StackTree, i);
5:       push (StackLevel, nextTreeLevel);
6: end.
```

The Algorithm 3 is the explore function which allows the B&B to move to the next tree level. This function pops from the *StackTree* the next node to visit and sets it in the corresponding position inside *S* indicated by level.

Algorithm 3. Function that explore a new node.

```
1: program explore-next-branch (Solution S)
2:   node  ← pop (StackTree);
3:   level ← pop (StackLevel);     ▷ Updates the level pointer.
4:   S_level ← node;
5: end.
```

The Algorithm 4 uses the equations (2) and (3) to evaluate a solution from the first position to the position indicated by level producing a partial evaluation of *S*.

Algorithm 4. Function that evaluate a partial solution.

```
1: program evaluate-partial-solution (Solution S, integer level)
2:   evaluates the solution from T_0 to T_level with equations (2) and (3).
3: end.
```

The function to decide if a solution improves the UB is shown in Algorithm 5. This algorithm compares *S* against all the solutions in UB using the dominance comparator.

Algorithm 5. Function that verifies if a solution S improves the upper bound.

```
1: program improves-upper-bound (Solution S, Front UB)
```

```
 2:   Dom_s ← 0;      ▷ Number of solutions which are dominated by S.
 3:   Dom_non ← 0;    ▷ Number of solutions which are non-dominated by S.
 4:   Dom_p ← 0;      ▷ Number of solutions which dominate S.
 5:   for each P ∈ UB do
 6:       if S ≺ P then
 7:         Dom_s ← Dom_s + 1;
 8:       if P ≺ S then
 9:         Dom_p ← Dom_p + 1;
10:       if S ⊀ P and P ⊀ S then
11:         Dom_non ← Dom_non + 1;
12:   if Dom_s > 0 or Dom_non = size (UB) then
13:       return True;
14:   return False;
15: end.
```

Algorithm 5 shows the function which verifies if a solution (S) improves the UB. From line 2 to 3, three counters are initialized. The first counter is the number of solutions which are dominated by S from UB. The second counts the total of solutions which S has a relation of non-dominance. The third counter is the number of solutions from UB which dominate S. It is said that S improves UB if at least one solution in LB is dominated by S or if it has a non-domination relationship with all the solutions in LB. From line 5 to 11, the algorithm compares S against all the solutions in the UB to verify the dominance relationship. Line 6, verifies if S dominate P. Line 8, verifies if P dominate S. Line 10, verifies if P and S have a relationship of non-domination. Line 13 to 15 verifies if S improves the UB.

Algorithm 6. Multi-objective Branch and Bound.

```
 1: program Multi-Objective-Branch-and-Bound ()
 2:   initialize ();
 3:   while tree-has-more-branches () do    ▷ While TreeStack has nodes.
 4:    explore-next-branch(S);
 5:    evaluate-partial-solution (S, level);
 6:    if a-leaf-has-not-been-reached () then
 7:      if improves-upper-bound (S, UB) then
 8:        branching(S);
 9:      else
10:        prune(S);
11:    else                                 ▷ a leaf has been reached.
12:      if improves-the-UB (S, UB) then
13:        add-to-UB (S, UB);
14:   return UB;                            ▷ UB is the Pareto front.
15: end.
```

Algorithm 6 shows the pseudocode from the B&B using all the main functions. Line 2 call the initialize function. Line 3 is the loop which keeps the exploration while the tree still has branches, it has branches if there are pending nodes to visit in *TreeStack*. Line 4 does an exploration of the tree by adding to S the next node corresponding to the level of the tree. In line 5 a partial evaluation of S is performed. Line 6 verify if a leaf has been reached, if not the algorithm continues in line 7 which validates if the partial solution can produce an improvement in the UB, and if it is so, then it performs a branching at line 8. If a leaf has been reached (line 11) line 12 verifies if S improves the UB, if it is true then S is added to UB and all the dominated solutions by S are removed. Finally, in line 14 the algorithm returns the optimum PF which is represented by UB.

5 Computational experiments

The experiment was done in a node of the Cluster Éhecatl from the Instituto Tecnológico de Ciudad Madero. Each node has the following specifications: 2x Intel Xeon with 12 cores E5-2670 v3, 64 GB of RAM in (8X8GB) DDR4-2133, and 120 GB in SSD. The experiment was done sequentially using one core from the node. The instances solved are a subset of benchmark datasets

reported in [13, 14] combined with instances from the literature, given a total of 40 instances (See Table 5), for those instances we ignored the precedencies section. The instances have between 2 and 5 machines. and between 6 and 15 tasks. This set we called Smallset. The name of each instance follows the next pattern name_#machines_#tasks, where name is the name of the author, #machines is the number of machines, and #tasks is the number of tasks. The configurations DVFS used in the machines are from Table 2. The experimentation had a time limit of 360 hours.

5.1 Results

The purpose of the experiments was to find the optimal PF for the instances of the Smallset. Table 1 shows the obtained results, column 1 contains the instance name, column 2 is the reference (Ref), column 3 indicates the number of solutions in the PF, and the last column contains the total time (T), in hours, required to find the optimal PF.

Table 5. Instances with their reference with the Pareto front size, and the total computation time given in hours.

Name	Ref	PF	T	Name	Ref	PF	T
Ahmad_3_9_28	[20]	110	0.0182	*Linshan_4_9_38*	[21]	157	0.1339
Bittencourt_3_9_184	[9]	212	0.8285	*Liu_2_8_364*	[22]	227	0.0087
Cao_3_10_536	[23]	327	2.5397	*Mohammad_2_11_64*	[24]	230	37.8343
Ching_3_10_84	[25]	148	3.4505	*Munir_3_10_76*	[26]	142	1.0393
Demiroz_3_7_47	[27]	108	0.0012	*Rahmani_3_7*	-	96	0.0003
Eswari_2_11_61	[28]	230	24.4703	*SahA_3_11_131*	[29]	376	310.8222
Gulzar_3_10_124	[30]	171	12.4246	*SahB_3_6_76*	[29]	107	0.0005
Hamid_3_10	[31]	267	0.6680	*Sakellariou_3_11_130*	[32]	-	360*
Hernandez_3_10_70	[33]	197	21.2947	*Samantha_5_11_31*	[34]	-	360*
heteropar_4_12_124	[35]	214	131.4986	*sample_3_10*	[13]	307	2.9908
heteropar_4_12_60	[36]	156	7.4750	*sample_3 _13*	-	-	360*
Ijaz_3_10_133	[37]	163	7.2786	*sample_3_8_100*	[13]	210	0.0126
IlavarasanIJCSIT_3_10	[38]	142	0.7264	*sample_4_11_25*	-	120	123.7853
Ilavarasan_3_10_77	[39]	142	0.7224	*Tao_3_10*	-	534	23.9643
Ilavarasan_3_11_27	[40]	220	57.1850	*Topcuoglu_3_10_80*	[5]	142	0.7170
Ilavarasan_3_15_114	[41]	-	360*	*Topcuoglu_3_8_51*	[42]	111	0.0037
Kang_3_10_76	[43]	142	0.7189	*Xu_3_8_66*	[44]	120	0.0104
Kang_3_10_84	[45]	142	0.7188	*YCLee_3_8_80*	[10]	108	0.0489
Kuan_3_10_28	[46]	130	1.3325	*Yu_4_10*	-	160	1.8916
Liang_3_10_80	[47]	142	0.7118	*Zhao_3_10_143*	[48]	273	10.0964

The B&B is capable to solve 36 instances in the given time limit. We classified the instances with respect to the time required for the B&B to achieve the PF. The instances which can be solved in one hour or less are considered as easy or lower hardness, between 1 and 24 hours are considered of medium hardness, and more than 24 hours are considered of high difficulty. Following this classification for the B&B, we have 18 instances of lower difficulty (white), 12 instances of medium difficulty (light grey), 6 instances of high difficulty (dark grey), and 4 instances that could not be solved (marked with asterisk).

Six instances required more than 24 hours to be solved, and four could not be solved in the given time. The graphs of the Pareto fronts found for the solved instances are shown in Figure 2 and Figure 3. Also the objective values for all the solutions in the Pareto fronts are shown in tables 6 to 41. In the tables an id (#), the makespan (m) and the energy (e) are given for each solution. These tables can be located in https://www.dropbox.com/s/ysm1yxk6j7ri16r/SMIO2016_paper_4_Tables6-41.pdf?dl=0.

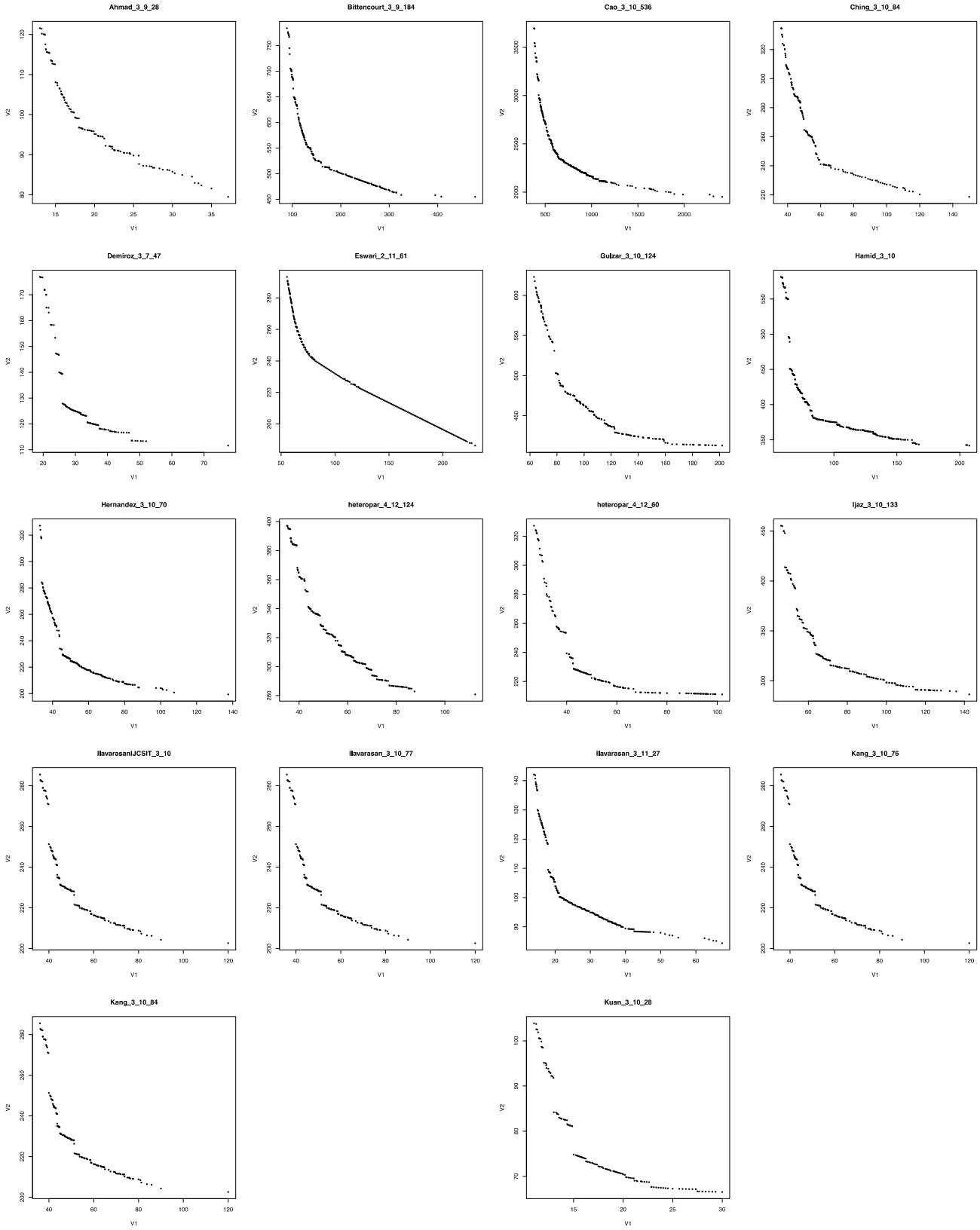

Fig. 2. Pareto fronts found for the first 18 instaces solved.

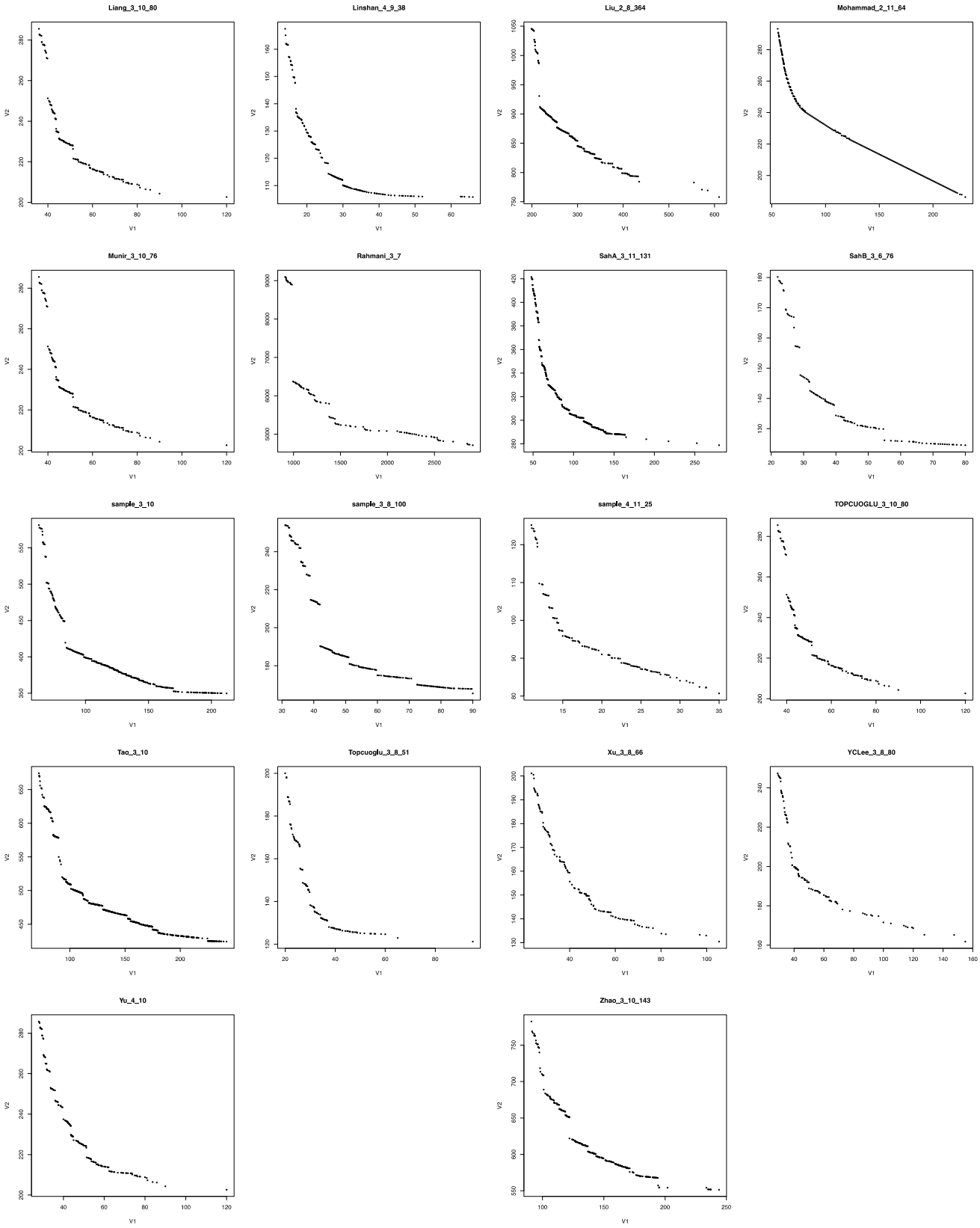

Fig. 3. Pareto fronts found for the last 18 instances solved.

6 Conclusions and future work

In this paper, we approached the problem of finding the optimal PF of HCSMOP instances. The contributions of this research are: a multi-objective B&B algorithm based on Pareto dominance and the optimal PF for 36 instances of the Smallset. The solved instances are the first available benchmark for the problem. The Pareto Fronts were completly documented to allow to another researchers use them to assess the performance of multiobjective algorithms with quality metrics that requires known the optimal front of the instances.

More research is needed to improve computational time of the proposed B&B, the improve-upper-bound function (Algorithm 5) because in each call it goes through all the solutions in the UB, and while more solutions are added to the UB it becomes more time consuming. Despite this, the B&B is capable of solve the benchmark.

As a future work we are planning to parallelize the algorithm using openMP and MPI with the purpose of reduce the required time to solve instances of higher difficulty. Also, the development of metaheuristics to initialize the upper bound with a set of good quality solutions. It is pending the improvement of the main functions and the research of techniques to accelerate the exploration of the tree. Another interesting research is to consider more objective such as max workload and total workload.

Acknowledgements

The authors would like to acknowledge with appreciation and gratitude to CONACYT, TECNM and PRODEP. Also, acknowledge to Laboratorio Nacional de Tecnologías de la Información del Instituto Tecnológico de Ciudad Madero for the access to the cluster. This work has been partial supported by CONACYT Project.254498. A. Santiago, J. Carlos Soto Monterrubio J. and David Terán-Villanueva would like to thank CONACyT for the supports 360199, 414092 and 177007.

References

1. A. Shehabi, S. J. Smith, D. A. Sartor, R. E. Brown, M. Herrlin, J. G. Koomey, E. R. Masanet, N. Horner, I. L. Azevedo, and W. Lintner, "United States Data Center Energy Usage Report," no. September, 2016.
2. S. Nesmachnow, H. Cancela, and E. Alba, "Heterogeneous computing scheduling with evolutionary algorithms," *Soft Comput.*, vol. 15, no. 4, pp. 685–701, Mar. 2010.
3. Y. C. Lee and A. Y. Zomaya, "Minimizing Energy Consumption for Precedence-Constrained Applications Using Dynamic Voltage Scaling," in *2009 9th IEEE/ACM International Symposium on Cluster Computing and the Grid*, 2009, pp. 92–99.
4. T. D. Braun, H. J. Siegel, N. Beck, L. L. Bölöni, 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, no. 6, pp. 810–837, Jun. 2001.
5. H. Topcuoglu, S. Hariri, and Min-You Wu, "Performance-effective and low-complexity task scheduling for heterogeneous computing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 13, no. 3, pp. 260–274, Mar. 2002.
6. S. Nesmachnow, H. Cancela, and E. Alba, "A parallel micro evolutionary algorithm for heterogeneous computing and grid scheduling," *Appl. Soft Comput.*, vol. 12, no. 2, pp. 626–639, Feb. 2012.
7. J. I. Hernández Hernández, "Reactive Scheduling of DAG Applications on Heterogeneous and Dynamic Distributed Computing Systems," *Comput. y Sist.*, vol. 13, pp. 221–237, 2009.
8. J. Kolodziej, S. U. Khan, and F. Xhafa, "Genetic Algorithms for Energy-Aware Scheduling in Computational Grids," in *2011 International Conference on P2P, Parallel, Grid, Cloud and Internet Computing*, 2011, pp. 17–24.
9. L. F. Bittencourt, R. Sakellariou, and E. R. M. Madeira, "DAG Scheduling Using a Lookahead Variant of the Heterogeneous Earliest Finish Time Algorithm," in *2010 18th Euromicro Conference on Parallel, Distributed and Network-based Processing*, 2010, pp. 27–34.
10. Young-Choon Lee and A. Zomaya, "A Novel State Transition Method for Metaheuristic-Based Scheduling in Heterogeneous Computing Systems," *IEEE Trans. Parallel Distrib. Syst.*, vol. 19, no. 9, pp. 1215–1223, Sep. 2008.
11. J. A. Ruiz-Vanoye, O. Díaz-Parra, and J. C. Zavala-Díaz, "Complexity Indicators applied to the Job Shop Scheduling Problem to discriminate the best Algorithm," *Int. J. Comb. Optim. Probl. Informatics*, vol. 2, no. 3,

pp. 25–31, 2011.
12. E.-G. Talbi, S. Mostaghim, T. Okabe, H. Ishibuchi, G. Rudolph, and C. A. Coello Coello, "Parallel Approaches for Multiobjective Optimization," in *Multiobjective Optimization*, Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 349–372.
13. H. J. Fraire Huacuja, J. J. Gonzalez Barbosa, P. Bouvry, A. A. S. Pineda, and J. E. Pecero, "An iterative local search algorithm for scheduling precedence-constrained applications on heterogeneous machines," *6th Multidiscip. Int. Conf. Sched. Theory Appl. (MISTA 2013)*, pp. 47–485, 2010.
14. J. E. Pecero, P. Bouvry, H. J. F. Huacuja, J. D. T. Villanueva, M. A. R. Zuniga, and C. G. G. Santillan, "Task Scheduling in Heterogeneous Computing Systems Using a MicroGA," in *2013 Eighth International Conference on P2P, Parallel, Grid, Cloud and Internet Computing*, 2013, pp. 618–623.
15. Y. C. Lee and A. Y. Zomaya, "Energy Conscious Scheduling for Distributed Computing Systems under Different Operating Conditions," *IEEE Trans. Parallel Distrib. Syst.*, vol. 22, no. 8, pp. 1374–1381, Aug. 2011.
16. E. Zitzler and L. Thiele, "Multiobjective evolutionary algorithms: a comparative case study and the strength Pareto approach," *IEEE Trans. Evol. Comput.*, vol. 3, no. 4, pp. 257–271, 1999.
17. C.-L. Hwang and A. S. M. Masud, *Multiple Objective Decision Making — Methods and Applications*, vol. 164. Berlin, Heidelberg: Springer Berlin Heidelberg, 1979.
18. H. F. Huacuja, N. Castillo-García, R. A. P. Rangel, J. A. M. Flores, J. J. G. Barbosa, and J. M. C. Valadez, "Two New Exact Methods for the Vertex Separation Problem," *Int. J. Comb. Optim. Probl. Informatics*, vol. 6, no. 1, p. 31, 2015.
19. J. C. Soto-Monterrubio, H. J. Fraire-Huacuja, J. Frausto-Solís, L. Cruz-Reyes, R. Pazos R., and J. Javier González-Barbosa, "TwoPILP: An Integer Programming Method for HCSP in Parallel Computing Centers," in *Advances in Artificial Intelligence and Its Applications*, Springer, 2015, pp. 463–474.
20. I. Ahmad, M. K. Dhodhi, and R. Ul–Mustafa, "DPS: dynamic priority scheduling heuristic for heterogeneous computing systems," *IEE Proc. - Comput. Digit. Tech.*, vol. 145, no. 6, p. 411, 1998.
21. L. Shen and T.-Y. Choe, "Posterior Task Scheduling Algorithms for Heterogeneous Computing Systems," in *High Performance Computing for Computational Science - VECPAR 2006*, vol. 4395, M. Daydé, J. M. L. M. Palma, A. L. G. A. Coutinho, E. Pacitti, and J. C. Lopes, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 172–183.
22. G. Q. Liu, K. L. Poh, and M. Xie, "Iterative list scheduling for heterogeneous computing," *J. Parallel Distrib. Comput.*, vol. 65, no. 5, pp. 654–665, May 2005.
23. H. Cao, H. Jin, X. Wu, S. Wu, and X. Shi, "DAGMap: efficient and dependable scheduling of DAG workflow job in Grid," *J. Supercomput.*, vol. 51, no. 2, pp. 201–223, Feb. 2010.
24. M. I. Daoud and N. Kharma, "A high performance algorithm for static task scheduling in heterogeneous distributed computing systems," *J. Parallel Distrib. Comput.*, vol. 68, no. 4, pp. 399–409, Apr. 2008.
25. C.-H. Hsu, C.-W. Hsieh, and C.-T. Yang, "A Generalized Critical Task Anticipation Technique for DAG Scheduling," in *Algorithms and Architectures for Parallel Processing*, Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 493–505.
26. A. Masood, E. U. Munir, M. M. Rafique, and S. U. Khan, "HETS: Heterogeneous Edge and Task Scheduling Algorithm for Heterogeneous Computing Systems," in *2015 IEEE 17th International Conference on High Performance Computing and Communications, 2015 IEEE 7th International Symposium on Cyberspace Safety and Security, and 2015 IEEE 12th International Conference on Embedded Software and Systems*, 2015, pp. 1865–1870.
27. B. Demiroz and H. R. Topcuoglu, "Static Task Scheduling with a Unified Objective on Time and Resource Domains," *Comput. J.*, vol. 49, no. 6, pp. 731–743, Nov. 2006.
28. R. Eswari and S. Nickolas, "Path-Based Heuristic Task Scheduling Algorithm for Heterogeneous Distributed Computing Systems," in *2010 International Conference on Advances in Recent Technologies in Communication and Computing*, 2010, pp. 30–34.
29. S. K. Sah and R. S. Singh, "Critical Path Based Scheduling of Multiple Applications in Heterogeneous Distributed Computing," in *2009 IEEE International Advance Computing Conference*, 2009, pp. 99–104.
30. S. G. Ahmad, E. U. Munir, and W. Nisar, "A Segmented Approach for DAG Scheduling in Heterogeneous Environment," in *2011 12th International Conference on Parallel and Distributed Computing, Applications and Technologies*, 2011, pp. 362–367.
31. H. Arabnejad and J. Barbosa, "Fairness Resource Sharing for Dynamic Workflow Scheduling on Heterogeneous Systems," in *2012 IEEE 10th International Symposium on Parallel and Distributed Processing with Applications*, 2012, no. September 2014, pp. 633–639.
32. R. Sakellariou, H. Zhao, and E. Deelman, "Mapping Workflows on Grid Resources: Experiments with the Montage Workflow," in *Grids, P2P and Services Computing*, Boston, MA: Springer US, 2010, pp. 119–132.
33. I. Hernandez and M. Cole, "Reactive grid scheduling of DAG applications," in *Parallel and Distributed Computing and Networks*, 2007, pp. 90–95.
34. S. Ranaweera and D. P. Agrawal, "A task duplication based scheduling algorithm for heterogeneous systems," in *Proceedings 14th International Parallel and Distributed Processing Symposium. IPDPS 2000*, 2000, no. February 2000, pp. 445–450.

35. H. Arabnejad and J. G. Barbosa, "Performance Evaluation of List Based Scheduling on Heterogeneous Systems," in *Proceedings of the 2011 international conference on Parallel Processing*, Berlin, Heidelberg: Springer-Verlag, 2012, pp. 440–449.
36. H. Arabnejad and J. G. Barbosa, "List Scheduling Algorithm for Heterogeneous Systems by an Optimistic Cost Table," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 3, pp. 682–694, Mar. 2014.
37. S. Ijaz, E. U. Munir, W. Anwar, and W. Nasir, "Efficient scheduling strategy for task graphs in heterogeneous computing environment," *Int. Arab J. Inf. Technol*, vol. 10, no. 5, pp. 486–492, 2013.
38. E. Ilavarasan and R. Manoharan, "High Performance and Energy Efficient Task Scheduling Algorithm for Heterogeneous Mobile Computing System," *Intl J Comput. Sci. Inf. Technol.*, vol. 2, no. 2, pp. 10–27, 2010.
39. E. Ilavarasan and P. Thambidurai, "Performance Effective Task Scheduling Algorithm for Heterogeneous Computing System," in *The 4th International Symposium on Parallel and Distributed Computing (ISPDC'05)*, 2007, vol. 3, no. 2, pp. 28–38.
40. E. Ilavarasan and P. Thambidurai, "Low Complexity Performance Effective Task Scheduling Algorithm for Heterogeneous Computing Environments," *J. Comput. Sci.*, vol. 3, no. 2, pp. 94–103, 2007.
41. E. Illvarasan and P. Thambidurai, "Levelized Scheduling of Directed A-Cyclic Precedence Constrained Task Graphs onto Heterogeneous Computing System," in *First International Conference on Distributed Frameworks for Multimedia Applications*, 2005, pp. 262–269.
42. H. Topcuoglu and C. Sevilmis, "Task Scheduling with Conflicting Objectives," in *Advis*, vol. 2457, 2002, pp. 346–355.
43. Y. Kang and Y. Lin, "A recursive algorithm for scheduling of tasks in a heterogeneous distributed environment," in *2011 4th International Conference on Biomedical Engineering and Informatics (BMEI)*, 2011, vol. 4, pp. 2099–2103.
44. Y. Xu, K. Li, L. He, and T. K. Truong, "A DAG scheduling scheme on heterogeneous computing systems using double molecular structure-based chemical reaction optimization," *J. Parallel Distrib. Comput.*, vol. 73, no. 9, pp. 1306–1322, Sep. 2013.
45. Y. Kang, Z. Zhang, and P. Chen, "An activity-based genetic algorithm approach to multiprocessor scheduling," in *2011 Seventh International Conference on Natural Computation*, 2011, vol. 2, pp. 1048–1052.
46. K.-C. Lai and C.-T. Yang, "A dominant predecessor duplication scheduling algorithm for heterogeneous systems," *J. Supercomput.*, vol. 44, no. 2, pp. 126–145, May 2008.
47. L.-T. Lee, C.-W. Chen, H.-Y. Chang, C.-C. Tang, and K.-C. Pan, "A Non-critical Path Earliest-Finish Algorithm for Inter-dependent Tasks in Heterogeneous Computing Environments," in *2009 11th IEEE International Conference on High Performance Computing and Communications*, 2009, pp. 603–608.
48. H. Zhao and R. Sakellariou, "An Experimental Investigation into the Rank Function of the Heterogeneous Earliest Finish Time Scheduling Algorithm," in *Euro-Par 2003 Parallel Processing*, vol. 2790, H. Kosch, L. Boszormenyi, and H. Hellwagner, Eds. Springer Berlin / Heidelberg, 2003, pp. 189–194.