MOL2NET, International Conference Series on Multidisciplinary Sciences

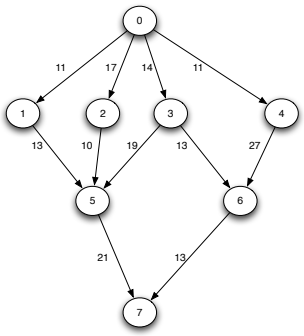
Optimal Scheduling for Precedence-Constrained Applications on Heterogeneous Machines

Carlos Soto[a,c], Alejandro Santiago[b], Héctor Fraire[a], Bernabé Dorronsoro[c].

[a] *Madero City Institute of Technology*
[b] *Polytechnic University of Altamira*
[c] *University of Cadiz*

Graphical Abstract

Task	m_1	m_2	m_3	$\overline{p_i}$
0	11	13	9	11.0
1	10	15	11	12.0
2	09	12	14	11.6
3	12	16	10	12.3
4	15	11	19	15.0
5	13	09	05	8.60
6	11	15	13	12.3
7	11	15	10	12.0



Abstract.
High-Performance Computing (HPC) is a growing necessity of our technological society, HPC demands high loads of parallel computing jobs, an optimal scheduling of the parallel applications tasks is a priority to meet the demands of its users on time. Branch-and-bound (BB) Algorithms and Mathematical Programming (MP) solve complex optimization problems in an optimal manner, some MP or BB even have parallel computing capabilities, making them suitable solutions to solve real-world problems. In this paper, we propose two exact algorithms, a BB and an MP Model for scheduling precedence-constrained applications, on heterogeneous computing systems, as far as we known the first ones on his kind presented in the state of the art. One major contribution of the work is the proposed formulations of the objective function in both methods. Experimental results obtained more than twenty optimal values for synthetic applications from the literature.

Introduction

Normally a heterogeneous computing system (HCS) provides high computing machines on parallel and/or distributed systems which works cooperatively to solve problems that require an intensive computing power and diverse computational requirements. The heterogeneous computing systems have been used to solve a wide variety of problems that require a high computing power [1].

The principal issue in an HCS consists in finding the best schedule of tasks on machines such that satisfies some requirements related to efficiency, workload, economic benefits, costs, and others. This is a scheduling problem where the scheduling considers diverse operations per job and dependencies between jobs. Typically, dependency is modeled by a directed graph [2], [3]. Either scheduling tasks

problems with and without precedence-constraints as in the parallel application scheduling, are known to be NP-Hard [4]. That means no deterministic algorithm is available to solve it in polynomial time, hence its relevance in solving it in an efficient way.

Parallel program representation

Generally, a parallel application is represented by a Directed Acyclic Graph (DAG) with the following description. Given a DAG, $G = (T, E)$, consists of a set T of n corresponding tasks t_i of the parallel program and set E of edges. In general, the nodes present segments from an application that can be computed independently; each edge $(t_i, t_j) \in$E represents a precedence constraint such that tasks t_j cannot start until t_i finish their execution. The edge $(t_i, t_j) \in E$ between tasks t_i, t_j also represents inter-task communication. The HCS is represented by a set of machines $M = \{m_1, m_2, \ldots, m_n\}$ with different processing times for every task $t_i \in T$. The problem formulation is represented by the next three equations.

$$\boldsymbol{ts_i = max\{\max\{tf_j + C_{j,k}\}\forall(t_j, t_i) \in E, Pw_{i,k} \in m_k\}} \quad , \qquad (1)$$

$$\boldsymbol{tf_i = ts_i + P_{i,k}} \quad , \qquad (2)$$

$$\boldsymbol{Makespan = \max_{i\in\{t_1, \ldots, t_n\}}\{tf_i\}} \quad , \qquad (3)$$

where $P_{i,k}$ is the computation time of the task t_i in the machine m_k, $Pw_{i,k}$ is the start time of an available window of size $P_{i,k}$ in the machine m_k, after the execution of their precedence tasks, and $C_{j,k}$ is the communication of the task t_j to the machine m_k. $C_{j,k}$ is equal to the edge weight (t_j, t_i) when the tasks t_i and t_j are executed in a different machine, otherwise $C_{j,k} = 0$. ts_i is the starting time of task i and tf_i is the ending time of task i. Although the above formulation, the correct computed *makespan* can vary; if the tasks are not executed in the best priority way. This is because the system can execute the tasks in any feasible order that do not violate the precedence-constraints. Needing to compute different priorities execution for tasks, this best priority verification is computationally expensive, for example when two tasks are assigned to the same machine and ready to go at the same time, Which one should be executed first? the obvious answer is the one that minimizes the overall *makespan*, but this needs to be computed first to be known.

Task	m_1	m_2	m_3	$\overline{p_i}$
0	11	13	9	11.0
1	10	15	11	12.0
2	09	12	14	11.6
3	12	16	10	12.3
4	15	11	19	15.0
5	13	09	05	8.60
6	11	15	13	12.3
7	11	15	10	12.0

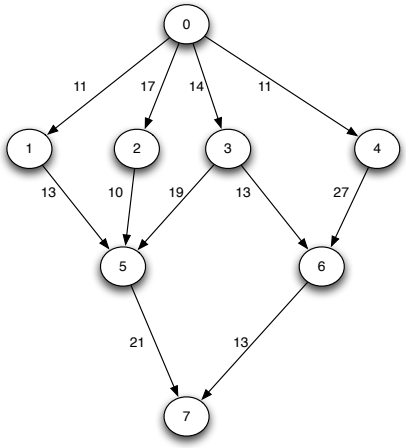


Figure 1. Instance Sample_3_8_100 at the right is the sample DAG with task labels. The inside nodes are the task identifiers and the values of communication cost are next to the corresponding edges. At the left are the computation costs.

In order to avoid these inconsistencies prioritizes scheduling heuristics had been developed, a popular one is list scheduling. These heuristics makes use of two attributes: 1) the b-level computation, which is

the length of the longest path from the exit task of the DAG job to the task; 2) the t-level computation, which is the length of the longest path from the entry task to the task [5]. The **¡Error! No se encuentra el origen de la referencia.** shows a recursive computation of b-level attribute. The heuristic used in this work is the Heterogeneous Earliest-Finish-Time (HEFT) [6]. The heuristic computes the priority of all the tasks and schedules each task on its best processor, which minimizes the task's computation time.

Algorithm 1. Recursive computation of b-level.

```
1: procedure RECURSIVE B-LEVEL(Task t_current)
2:     temp ← p̄i[t_current]
3:     max ← temp
4:     for (t_current, t_u) ∈ E do
5:         temp ←B-LEVEL(t_u, (temp + (t_current, t_u) + p̄i[t_u]))
6:         if max < temp then
7:             max ← temp
8:         end if
9:     end for
10:    return max
11: end procedure
```

It is easier to simplify the problem formulation from Equation 1, when the tasks are executed in a topological order (feasible), allowing a lower computational cost of the objective function.

Objective function with topological order

Table 1 shows the values of b-level for each task from the example instance.

Table 1. b-level value from tasks from instance Sample_3_8_100.

Task	*0*	*1*	*2*	*3*	*4*	*5*	*6*	*7*
b-level	*101.33*	*66.66*	*6.33*	*73.00*	*79.33*	*41.66*	*37.33*	*12.00*

The Algorithm 2 shows the computation of the objective value with a topological order.

Algorithm 2. Objective function with a topological order.

```
1: procedure OBJECTIVE-FUNCTION-TOPOLOGICAL-ORDER
2:     for x in 0 to n do
3:         t_current ← o_x
4:         if ((u, current) ∈ E) is null then
5:             ts_current ← CTM_j
6:             tf_current ← ts_current + P_current,j
7:         else
8:             u* ← argmax_{u∈T|(u,current)∈E} {tf_u + C'_{u,current}}
9:             ts_current ← MAX(tf_u* + C'_{u*,actual}, CTM_j)
10:            tf_current ← ts_current + P_current,j
11:        end if
12:    end for
13:    makespan ← MAX(tf_x) ∀x ∈ T
14: end procedure
```

Experimental results

The experimentation consists of solving a set of 21 instances from the literature. The reference for each instance can be found in Table 2. The time limit to solve each instance was set to 37297 seconds.

Table 2 shows the results provided by a branch-and-bound and a MILP model. The first column is the instance name. The second column is the optimal value corresponding to the instance. The reference to the instance is in the third column. The fourth and fifth columns are the objective value retrieved and the required time of the branch-and-bound to find it, respectively. The sixth and seventh columns are the objective value found and the required time of the MILP.

Table 2. The results for synthetic application in the literature.

Instance	Optimal	Reference	BB	Time	MILP	Time
Ahmad_3_9_28	22	[7]	22	0.13	22	0.46
Bittencourt_3_9_184	184	[8]	184	0.41	184	0.73
Hsu_3_10_84	80	[9]	80	14.36	80	0.60
Demiroz_3_7_47	47	[10]	47	0.00	47	0.13
Eswari_2_11	100	[11]	100	5.61	100	0.71
Eswari_2_11_61	56	[11]	56	3.69	56	1.52
Hamid_3_10	100	[12]	100	5.59	100	0.70
Ilavarasan_3_10_77	73	[13]	73	6.84	73	0.64
Ilavarasan_3_15_114	121	[14]	121	37297.23	121	28.05
Kang_3_10_76	73	[15]	73	7.88	73	2.34
Kang_3_10_84	79	[16]	79	17.80	79	0.76
Kuan_3_10_28	25	[17]	26	1.89	26	0.75
Liang_3_10_80	73	[18]	73	6.83	73	0.65
Linshan_4_9_38	40	[19]	40	11.31	40	1.99
Mohammad_2_11_64	56	[20]	56	3.83	56	1.51
SahA_3_11_131	118	[21]	118	6.88	118	3.43
SahB_3_6_76	66	[21]	66	0.00	66	0.13
Samantha_5_11_31	32	[22]	32	3696.81	32	6.21
Sample_3_8_100	81	[23]	81	0.08	81	0.75
Liang_3_10_80	73	[18]	73	6.83	73	0.67
YCLee_3_8_80	66	[24]	66	0.11	66	0.28
Average time (sec.)				1956.86		2.52

Conclusions

In this work, a novel model for Heterogenous Computing Scheduling Problem is presented. The mixed integer linear programming model is compared against a branch-and-bound based on HEFT. Both strategies achieve the optimal values, but the MILP gets a better computational time.

References

[1] T. D. Braun *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*, vol. 61, no. 6, pp. 810–837, Jun. 2001.

[2] A. Jones, L. C. Rabelo, and A. T. Sharawi, "Survey of Job Shop Scheduling Techniques," in *Wiley Encyclopedia of Electrical and Electronics Engineering*, Hoboken, NJ, USA: John Wiley & Sons, Inc., 1999.

[3] R. W. Conway, W. L. Maxwell, and L. W. Miller, *Theory of scheduling*. Courier Corporation, 2012.

[4] O. Sinnen, *Task Scheduling for Parallel Systems*. Hoboken, NJ, USA: John Wiley & Sons, Inc., 2007.

[5] 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," *Journal of Parallel and Distributed Computing*, vol. 73, no. 9, pp. 1306–1322, Sep. 2013.

[6] H. Topcuoglu, S. Hariri, and Min-You Wu, "Performance-effective and low-complexity task scheduling for heterogeneous computing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 13, no. 3, pp. 260–274, Mar. 2002.

[7] I. Ahmad, M. K. Dhodhi, and R. Ul–Mustafa, "DPS: dynamic priority scheduling heuristic for heterogeneous computing systems," *IEE Proceedings - Computers and Digital Techniques*, vol. 145, no. 6, p. 411, 1998.

[8] 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*, Pisa, Italy, 2010, pp. 27–34.

[9] 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.

[10] B. Demiroz and H. R. Topcuoglu, "Static Task Scheduling with a Unified Objective on Time and Resource Domains," *The Computer Journal*, vol. 49, no. 6, pp. 731–743, Nov. 2006.

[11] 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*, Washington, DC, USA, 2010, pp. 30–34.

[12] 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.

[13] P. T. E. Ilavarasan, "Performance Effective Task Scheduling Algorithm for Heterogeneous Computing System," in *The 4th International Symposium on Parallel and Distributed Computing (ISPDC'05)*, Washington, DC, USA, 2007, vol. 3, pp. 28–38.

[14] 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*, Besançon, France, 2005, pp. 262–269.

[15] Y. Kang, Z. Zhang, and P. Chen, "An activity-based genetic algorithm approach to multiprocessor scheduling," in *2011 Seventh International Conference on Natural Computation*, Shanghai, 2011, vol. 2, pp. 1048–1052.

[16] 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)*, Shanghai, 2011, vol. 4, pp. 2099–2103.

[17] K.-C. Lai and C.-T. Yang, "A dominant predecessor duplication scheduling algorithm for heterogeneous systems," *The Journal of Supercomputing*, vol. 44, no. 2, pp. 126–145, May 2008.

[18] 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*, Seoul, 2009, pp. 603–608.

[19] 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.

[20] M. I. Daoud and N. Kharma, "A hybrid heuristic–genetic algorithm for task scheduling in heterogeneous processor networks," *Journal of Parallel and Distributed Computing*, vol. 71, no. 11, pp. 1518–1531, Nov. 2011.

[21] 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.

[22] 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*, Washington, DC, USA, 2000, pp. 445–450.

[23] 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 Multidisciplinary International Conference on Scheduling : Theory and Applications (MISTA 2013)*, pp. 47–485, 2010.

[24] Young-Choon Lee and A. Zomaya, "A Novel State Transition Method for Metaheuristic-Based Scheduling in Heterogeneous Computing Systems," *IEEE Transactions on Parallel and Distributed Systems*, vol. 19, no. 9, pp. 1215–1223, Sep. 2008.