Supplementary Material for Energy-Efficient Task Scheduling on Multiple Heterogeneous Computers: Algorithms, Analysis, and Performance Evaluation

Keqin Li, *Fellow, IEEE*

1 Paper Outline

The paper is organized as follows. (All section numbers refer to the main paper.) In Section 2, we review related research in energy-efficient heterogeneous computing. In Sections 3–5, we address energy-efficient scheduling of independent tasks on multiple heterogeneous computers. In Section 3.1, we define the energy-constrained scheduling problem and develop a method of optimal power allocation for a given schedule, such that the total task execution time is minimized. In Section 3.2, we define the time-constrained scheduling problem and develop a method of optimal power allocation for a given schedule, such that the energy consumption is minimized. The significance of Sections 3.1 and 3.2 is to reduce our scheduling problems to the optimal workload partition problems. In Section 4.1, we develop a method to find an optimal partition of a given workload, such that the total task execution time is minimized. In Section 4.2, we develop a method to find an optimal partition of a given workload, such that the total energy consumption is minimized. The significance of Sections 4.1 and 4.2 is two fold. First, the optimal workload partition can be used to guide us in finding an optimal schedule to solve a scheduling problem. Second, we get lower bounds for the optimal solutions, such that our solutions can be compared with the optimal solutions. In Section 5.1, we describe the MLS algorithm for scheduling independent tasks. In Section 5.2, we give examples of our numerical calculations. In Section 5.3, we present simulation data to demonstrate the performance of our heuristic algorithms.

In Section 6, we address energy-efficient scheduling of precedence constrained tasks on multiple heterogeneous computers. In Section 6.1, we describe the LL-MLS algorithm for scheduling precedence constrained tasks. In Section 6.2, we discuss optimal energy allocation to levels of a dag. In Section 6.3, we discuss optimal time allocation to levels of a dag. The significance of Sections 6.2 and 6.3 is to optimize the performance of the level-by-level scheduling method. In Section 6.4, we present simulation data to demonstrate the performance of our heuristic algorithms. In Section 7, we conclude the paper.

• K. Li is with the Department of Computer Science, State University of New York, New Paltz, New York 12561, USA.
E-mail: lik@newpaltz.edu

2 Heterogeneous Cloud Computing

Several authors have incorporated dynamic voltage and frequency scaling into study. In [6], the authors addressed optimal power allocation and load distribution for multiple heterogeneous multicore server processors across clouds and data centers as optimization problems, i.e., power constrained performance optimization and performance constrained power optimization. In [17], the author considered the problem of optimal power allocation among multiple heterogeneous servers in a data center, i.e., minimizing the average task response time of multiple heterogeneous computer systems with energy constraint. In [21], the author investigated the technique of using workload dependent dynamic power management (i.e., variable power and speed of processor cores according to the current workload) to improve system performance and to reduce energy consumption. In [31], the authors optimized the performance and power consumption tradeoff for multiple heterogeneous servers with continuous and discrete speed scaling.

3 Numerical Algorithms

In this section, we give the pseudo-codes of all the numerical algorithms developed in this paper. The classic bisection method, or the binary search algorithm ([5], §2.1, p. 21) is repeatedly used to solve complicated equations. It is a common sense that all meaningful large numbers in the everyday world are not that large. Consider astronomically large numbers. The estimated number of atoms in the observable universe is only $10^{80} < 2^{266}$. The age of the universe is only 4.355×10^{26} nanoseconds. Since all variables in our study denote real quantities, it is safe to claim that the bisection method can be finished in $O(10^2)$ time. Hence, the bisection method is very efficient. For instance, all the data in Tables 1–2 are produced instantly, while all the data in Tables 5–6 are produced in seconds.

In all our algorithms, the small constant ϵ used to terminate the bisection method is set as 10^{-9}.

Algorithm 4 employs the Gaussian elimination and backward substitution algorithm for solving a linear system of equations ([5], §6.2, p. 265).

Algorithm 7 refers to the following equation from Section 6.3 of the paper:

$$\phi = \sum_{k=1}^{m} \left(\frac{\phi_l}{\alpha_k}\right)^{\alpha_k/(\alpha_k-1)} - \frac{R_l}{T_l}\left(\sum_{k=1}^{m} \frac{1}{\alpha_k(\alpha_k-1)}\left(\frac{\alpha_k}{\phi_l}\right)^{(\alpha_k-2)/(\alpha_k-1)}\right)^{-1} \sum_{k=1}^{m}\left(\frac{1}{\alpha_k-1}\left(\frac{\phi_l}{\alpha_k}\right)^{1/(\alpha_k-1)}\right). \quad (1)$$

All algorithms are properly cited in the main paper.

Algorithm 1: Find_R_k's$(m, \alpha_1, \alpha_2, ..., \alpha_m, R, E, T)$.

Input: $m, \alpha_1, \alpha_2, ..., \alpha_m, R, E$, and T.
Output: $R_1, R_2, ..., R_m$.

//Set W_1 and W_2
$W_0 \leftarrow 1$;
$W_1 \leftarrow W_0$;
do
    $W_1 \leftarrow W_1/2$;
    for ($k \leftarrow 1$; $k \le m$; k++) **do**
        $R_k \leftarrow T\,(E/\alpha_k W_1)^{1/(\alpha_k-1)}$;
    end do;
while ($W_1 \ge R_1/\alpha_1 + R_2/\alpha_2 + \cdots + R_m/\alpha_m$);
$W_2 \leftarrow W_0$;
do
    $W_2 \leftarrow 2W_2$;
    for ($k \leftarrow 1$; $k \le m$; k++) **do**
        $R_k \leftarrow T\,(E/\alpha_k W_2)^{1/(\alpha_k-1)}$;
    end do;
while ($W_2 \le R_1/\alpha_1 + R_2/\alpha_2 + \cdots + R_m/\alpha_m$);
//Search W in $[W_1, W_2]$
while ($W_2 - W_1 > \epsilon$) //ϵ is a very small constant
    $W \leftarrow (W_1 + W_2)/2$;
    for ($k \leftarrow 1$; $k \le m$; k++) **do**
        $R_k \leftarrow T\,(E/\alpha_k W)^{1/(\alpha_k-1)}$;
    end do;
    if ($R_1/\alpha_1 + R_2/\alpha_2 + \cdots + R_m/\alpha_m > W$)
        $W_1 \leftarrow W$;
    else
        $W_2 \leftarrow W$;
    end if;
end while;
return $R_1, R_2, ..., R_m$.

Fig. 1. An algorithm to find $R_1, R_2, ..., R_m$.

REFERENCES

[1] http://developer.amd.com/resources/heterogeneous-computing/what-is-heterogeneous-system-architecture-hsa/

[2] http://www.spec.org/power_ssj2008/results/power_ssj2008.html

[3] S. Albers, "Energy-efficient algorithms," *Communications of the ACM*, vol. 53, no. 5, pp. 86-96, 2010.

[4] A. Beloglazov, R. Buyya, Y. C. Lee, and A. Zomaya, "A taxonomy and survey of energy-efficient data centers and cloud computing systems," *Advances in Computers*, vol. 82, pp. 47-111, 2011.

[5] R. L. Burden, J. D. Faires, and A. C. Reynolds, *Numerical Analysis*, 2nd edition, Prindle, Weber & Schmidt, Boston, MA., 1981.

[6] J. Cao, K. Li, and I. Stojmenovic, "Optimal power allocation and load distribution for multiple heterogeneous multicore server processors across clouds and data centers," *IEEE Transactions on Computers*, vol. 63, no. 1, pp. 45-58, 2014.

[7] A. P. Chandrakasan, S. Sheng, and R. W. Brodersen, "Low-power CMOS digital design," *IEEE Journal on Solid-State Circuits*, vol. 27, no. 4, pp. 473-484, 1992.

[8] S. Crago, K. Dunn, P. Eads, L. Hochstein, D.-I. Kang, M. Kang, D. Modium, K. Singh, J. Suh, J. P. Walters, "Heterogeneous cloud computing," *IEEE International Conference on Cluster Computing*, pp. 378-385, 2011.

[9] S. P. Crago and J. P. Walters, "Heterogeneous cloud computing: the way forward," *IEEE Computer*, vol. 48, no. 1, pp. 59-61, 2015.

[10] H. Esmaeilzadeh, T. Cao, X. Yang, S. M. Blackburn, and K. S. McKinley, "Looking back and looking forward: power, performance, and upheaval," *Communications of the ACM*, vol. 55, no. 7, pp. 105-114, 2012.

[11] S. Garg, S. Sundaram, and H. D. Patel, "Robust heterogeneous data center design: a principled approach," *ACM SIGMETRICS Performance Evaluation Review*, vol. 39, no. 3, pp. 28-30, December 2011.

[12] R. L. Graham, "Bounds on multiprocessing timing anomalies," *SIAM J. Appl. Math.*, vol. 2, pp. 416-429, 1969.

[13] R. Iyer, S. Srinivasan, O. Tickoo, Z. Fang, R. Illikkal, S. Zhang, V. Chadha, P. M. Stillwell Jr., S. E. Lee, "CogniServe: heterogeneous server architecture for large-scale recognition," *IEEE Micro*, vol. 31, no. 3, pp. 20-31, May/June 2011.

[14] K. Li, "Performance analysis of power-aware task scheduling algorithms on multiprocessor computers with dynamic voltage and speed," *IEEE Transactions on Parallel and Distributed Systems*, vol. 19, no. 11, pp. 1484-1497, 2008.

[15] K. Li, "Energy efficient scheduling of parallel tasks on multiprocessor computers," *Journal of Supercomputing*, vol. 60, no. 2, pp. 223-247, 2012.

[16] K. Li, "Scheduling precedence constrained tasks with reduced processor energy on multiprocessor computers," *IEEE Transactions on Computers*, vol. 61, no. 12, pp. 1668-1681, 2012.

[17] K. Li, "Optimal power allocation among multiple heterogeneous servers in a data center," *Sustainable Computing: Informatics and Systems*, vol. 2, no. 1 pp. 13-22, 2012.

[18] K. Li, "Energy-efficient and high-performance processing of large-scale parallel applications in data centers," *Data Centers*, S. U. Khan and A. Y. Zomaya, eds., Chapter 1, pp. 1-33, Springer, 2015.

[19] K. Li, "Power and performance management for parallel computations in clouds and data centers," *Journal of Computer and System Sciences*, vol. 82, pp. 174-190, 2016.

[20] K. Li, "Energy and time constrained task scheduling on multiprocessor computers with discrete speed levels," *Journal of*

Algorithm 2: Find_$T(m, \alpha_1, \alpha_2, ..., \alpha_m, R, E)$.

Input: $m, \alpha_1, \alpha_2, ..., \alpha_m, R$, and E.
Output: $R_1^*, R_2^*, ..., R_m^*$, and T.

//Set T_1 and T_2
$T_0 \leftarrow 1$;
$T_1 \leftarrow T_0$;
do
  $T_1 \leftarrow T_1/2$;
  Find_R_k's$(m, \alpha_1, \alpha_2, ..., \alpha_m, R, E, T_1)$;
while $(R_1 + R_2 + \cdots + R_m \geq R)$;
$T_2 \leftarrow T_0$;
do
  $T_2 \leftarrow 2T_2$;
  Find_R_k's$(m, \alpha_1, \alpha_2, ..., \alpha_m, R, E, T_2)$;
while $(R_1 + R_2 + \cdots + R_m \leq R)$;
//Search T in $[T_1, T_2]$
while $(T_2 - T_1 > \epsilon)$ //ϵ is a very small constant
  $T \leftarrow (T_1 + T_2)/2$;
  Find_R_k's$(m, \alpha_1, \alpha_2, ..., \alpha_m, R, E, T)$;
  //The return values are $R_1^*, R_2^*, ..., R_m^*$
  if $(R_1^* + R_2^* + \cdots + R_m^* < R)$
    $T_1 \leftarrow T$;
  else
    $T_2 \leftarrow T$;
  end if;
end while;
return $R_1^*, R_2^*, ..., R_m^*$, and T.

Fig. 2. An algorithm to find T.

Algorithm 3: Find_$E(m, \alpha_1, \alpha_2, ..., \alpha_m, R, T)$.

Input: $m, \alpha_1, \alpha_2, ..., \alpha_m, R$, and T.
Output: $R_1^*, R_2^*, ..., R_m^*$, ϕ, and E.

//Set ϕ_1 and ϕ_2
$\phi_0 \leftarrow 1$;
$\phi_1 \leftarrow \phi_0$;
do
  $\phi_1 \leftarrow \phi_1/2$;
while $(\sum_{k=1}^{m} (\phi_1/\alpha_k)^{1/(\alpha_k-1)} \geq R/T)$;
$\phi_2 \leftarrow \phi_0$;
do
  $\phi_2 \leftarrow 2\phi_2$;
while $(\sum_{k=1}^{m} (\phi_2/\alpha_k)^{1/(\alpha_k-1)} \leq R/T)$;
//Search ϕ in $[\phi_1, \phi_2]$
while $(\phi_2 - \phi_1 > \epsilon)$ //ϵ is a very small constant
  $\phi \leftarrow (\phi_1 + \phi_2)/2$;
  if $(\sum_{k=1}^{m} (\phi/\alpha_k)^{1/(\alpha_k-1)} < R/T)$
    $\phi_1 \leftarrow \phi$;
  else
    $\phi_2 \leftarrow \phi$;
  end if;
end while;
for $(k \leftarrow 1$; $k \leq m$; k++) **do**
  $R_k^* \leftarrow (\phi/\alpha_k)^{1/(\alpha_k-1)} T$;
end do;
$E \leftarrow T \sum_{k=1}^{m} (\phi/\alpha_k)^{\alpha_k/(\alpha_k-1)}$;
return $R_1^*, R_2^*, ..., R_m^*$, ϕ, and E.

Fig. 3. An algorithm to find E.

Algorithm 4: Find_$R'_{l,k}$'s$(m, \alpha_1, ..., \alpha_m, R_l, E_l, \phi)$.

Input: $m, \alpha_1, \alpha_2, ..., \alpha_m, R_l, E_l$, and ϕ.
Output: $R'_{l,1} + R'_{l,2} + \cdots + R'_{l,m}$.

Find_$T(m, \alpha_1, \alpha_2, ..., \alpha_m, R_l, E_l)$;
//The results are $R_{l,1}, R_{l,2}, ..., R_{l,m}$ and T_l
Solve the system of linear equations in Section 6.2;
//The results are $R'_{l,1}, R'_{l,2}, ..., R'_{l,m}$
return $R'_{l,1} + R'_{l,2} + \cdots + R'_{l,m}$.

Fig. 4. An algorithm to find $R'_{l,1} + R'_{l,2} + \cdots + R'_{l,m}$.

Parallel and Distributed Computing, vol. 95, pp. 15-28, 2016.

[21] K. Li, "Improving multicore server performance and reducing energy consumption by workload dependent dynamic power management," *IEEE Transactions on Cloud Computing*, vol. 4, no. 2, pp. 122-137, 2016.

[22] K. Li, X. Tang, and K. Li, "Energy-efficient stochastic task scheduling on heterogeneous computing systems," *IEEE Transactions on Parallel and Distributed Systems*, vol. 25, no. 11, pp. 2867-2876, 2014.

[23] S. U. R. Malik, K. Bilal, S. U. Khan, B. Veeravalli, K. Li, and A. Y. Zomaya, "Modeling and analysis of the thermal properties exhibited by cyber physical data centers," *IEEE Systems Journal*, in press, 2016.

[24] J. Mateo, J. Vilaplana, L. M. Plá, J. L. Lérida, and F. Solsona, "A green strategy for federated and heterogeneous clouds with communicating workloads," *The Scientific World Journal*, Published online 2014 Nov 11. http://www.ncbi.nlm.nih.gov/pmc/articles/PMC4244950/

[25] J. Mei, K. Li, and K. Li, "Energy-aware task scheduling in heterogeneous computing environments," *Cluster Computing*, vol. 17, no. 2, pp. 537-550, 2014.

[26] H. S. Narman, M. S. Hossain, and M. Atiquzzaman, "h-DDSS: heterogeneous dynamic dedicated servers scheduling in cloud computing," *IEEE International Conference on Communications*, pp. 3475-3480, 2014.

[27] V. Petrucci, E. V. Carrera, O. Loques, J. C. B. Leite, D. Mossé, "Optimized management of power and performance for virtualized heterogeneous server clusters," *11th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*, pp. 23-32, 2011.

[28] C. Reiss, A. Tumanov, G. R. Ganger, R. H. Katz, and M. A. Kozuch, "Heterogeneity and dynamicity of clouds at scale: Google trace analysis," *Proceedings of the 3rd ACM Symposium on Cloud Computing*, Article No. 7, 2012.

[29] D. Sun, G. Zhang, S. Yang, W. Zheng, S. U. Khan, and K. Li, "Re-Stream: Real-time and energy-efficient resource scheduling in big data stream computing environments," *Information Sciences*, vol. 319, pp. 92-112, 2015.

[30] Z. Tang, L. Qi, Z. Cheng, K. Li, S. U. Khan, and K. Li, "An energy-efficient task scheduling algorithm in DVFS-enabled cloud environment," *Journal of Grid Computing*, vol. 14, no. 1, pp. 55-74, 2016.

[31] Y. Tian, C. Lin, and K. Li, "Managing performance and power consumption tradeoff for multiple heterogeneous servers in cloud computing," *Cluster Computing*, vol. 17, no. 3, pp. 943-955, 2014.

[32] G. L. Valentini, W. Lassonde, S. U. Khan, N. Min-Allah, S. A.

Algorithm 5: Find_$E_l(m, \alpha_1, \alpha_2, ..., \alpha_m, R_l, \phi)$.

Input: $m, \alpha_1, \alpha_2, ..., \alpha_m, R_l$, and ϕ.
Output: E_l.

```
//Set E_1 and E_2
E_0 ← 1;
E_1 ← E_0;
while (Find_R'_{l,k}'s(m, α_1, ..., α_m, R_l, E_1, φ) < 0)
    E_1 ← E_1/2;
end while;
E_2 ← E_0;
while (Find_R'_{l,k}'s(m, α_1, ..., α_m, R_l, E_2, φ) > 0)
    E_2 ← 2E_2;
end while;
//Search E_l in [E_1, E_2]
while (E_2 − E_1 > ε) //ε is a very small constant
    E_l ← (E_1 + E_2)/2;
    if (Find_R'_{l,k}'s(m, α_1, ..., α_m, R_l, E_l, φ) > 0)
        E_1 ← E_l;
    else
        E_2 ← E_l;
    end if;
end while;
return E_l.
```

Fig. 5. An algorithm to find E_l.

Algorithm 6: Min_$T(m, \alpha_1, ..., \alpha_m, R_1, ..., R_v, E)$.

Input: $m, \alpha_1, \alpha_2, ..., \alpha_m, R_1, R_2, ..., R_v$, and E.
Output: $E_1, E_2, ..., E_v$, and T.

```
//Set φ_1 and φ_2
φ_0 ← −1;
φ_1 ← φ_0;
do
    φ_1 ← 2φ_1;
    for (l ← 1; l ≤ v; l++) do
        E_l ← Find_E_l(m, α_1, ..., α_m, R_l, φ_1);
    end do;
while (E_1 + E_2 + ··· + E_v > E);
φ_2 ← φ_0;
do
    φ_2 ← φ_2/2;
    for (l ← 1; l ≤ v; l++) do
        E_l ← Find_E_l(m, α_1, ..., α_m, R_l, φ_2);
    end do;
while (E_1 + E_2 + ··· + E_v < E);
//Search φ in [φ_1, φ_2]
while (φ_2 − φ_1 > ε) //ε is a very small constant
    φ ← (φ_1 + φ_2)/2;
    for (l ← 1; l ≤ v; l++) do
        E_l ← Find_E_l(m, α_1, ..., α_m, R_l, φ);
    end do;
    if (E_1 + E_2 + ··· + E_v < E)
        φ_1 ← φ;
    else
        φ_2 ← φ;
    end if;
end while;
//Calculate T
for (l ← 1; l ≤ v; l++) do
    Find_T(m, α_1, α_2, ..., α_m, R_l, E_l) to get T_l;
    T ← T + T_l;
end do;
return E_1, E_2, ..., E_v, and T.
```

Fig. 6. An algorithm to find $E_1, E_2, ..., E_v$, and T.

Madani, J. Li, L. Zhang, L. Wang, N. Ghani, J. Kolodziej, H. Li, A. Y. Zomaya, C.-Z. Xu, P. Balaji, A. Vishnu, F. Pinel, J. E. Pecero, D. Kliazovich, and P. Bouvry, "An overview of energy efficiency techniques in cluster computing systems," *Cluster Computing*, vol. 16, no. 1, pp. 3-15, 2013.

[33] L. Van Doorn, "Heterogeneous server architectures will dominate the future data center," Open Server Summit, Santa Clara, California, November 13, 2014. https://vimeo.com/119496605

[34] W. Wang, B. Liang, and B. Li, "Multi-Resource fair allocation in heterogeneous cloud computing systems," *IEEE Transactions on Parallel and Distributed Systems*, vol. 26, no. 10, pp. 2822-2835, 2015.

[35] Y. Wang, K. Li, H. Chen, L. He, and K. Li, "Energy-aware data allocation and task scheduling on heterogeneous multiprocessor systems with time constraints," *IEEE Transactions on Emerging Topics in Computing*, vol. 2, no. 2, pp. 134-148, 2014.

[36] B. Zhai, D. Blaauw, D. Sylvester, and K. Flautner, "Theoretical and practical limits of dynamic voltage scaling," *Proceedings of the 41st Design Automation Conference*, pp. 868-873, 2004.

[37] L. Zhang, K. Li, Y. Xu, J. Mei, F. Zhang, and K. Li, "Maximizing reliability with energy conservation for parallel task scheduling in a heterogeneous cluster," *Information Sciences*, vol. 319, pp. 113-131, 2015.

[38] L. M. Zhang, K. Li, D. C.-T. Lo, and Y. Zhang, "Energy-efficient task scheduling algorithms on heterogeneous computers with continuous and discrete speeds," *Sustainable Computing: Informatics and Systems*, vol. 3, no. 2, pp. 109-118, 2013.

[39] S. Zhuravlev, J. C. Saez, S. Blagodurov, A. Fedorova, and M. Prieto, "Survey of energy-cognizant scheduling techniques," *IEEE Transactions on Parallel and Distributed Systems*, vol. 24, no. 7, pp. 1447-1464, 2013.

Algorithm 7: Find_$T_l(m, \alpha_1, \alpha_2, ..., \alpha_m, R_l, \phi)$.

Input: $m, \alpha_1, \alpha_2, ..., \alpha_m, R_l$, and ϕ.
Output: T_l.

//Set T_1 and T_2
$T_0 \leftarrow 1$;
$T_1 \leftarrow T_0$;
do
$T_1 \leftarrow T_1/2$;
Find_$E(m, \alpha_1, \alpha_2, ..., \alpha_m, R_l, T_1)$ to get ϕ_l;
while (the right-hand side of Eq. (1) $> \phi$);
$T_2 \leftarrow T_0$;
do
$T_2 \leftarrow 2T_2$;
Find_$E(m, \alpha_1, \alpha_2, ..., \alpha_m, R_l, T_2)$ to get ϕ_l;
while (the right-hand side of Eq. (1) $< \phi$);
//Search T in $[T_1, T_2]$
while ($T_2 - T_1 > \epsilon$) //ϵ is a very small constant
$T_l \leftarrow (T_1 + T_2)/2$;
Find_$E(m, \alpha_1, \alpha_2, ..., \alpha_m, R_l, T_l)$;
if (the right-hand side of Eq. (1) $< \phi$);
$T_1 \leftarrow T_l$;
else
$T_2 \leftarrow T_l$;
end if;
end while;
return T_l.

Fig. 7. An algorithm to find T_l.

Algorithm 8: Min_$E(m, \alpha_1, ..., \alpha_m, R_1, ..., R_v, T)$.

Input: $m, \alpha_1, \alpha_2, ..., \alpha_m, R_1, R_2, ..., R_v$, and E.
Output: $T_1, T_2, ..., T_v$, and E.

//Set ϕ_1 and ϕ_2
$\phi_0 \leftarrow -1$;
$\phi_1 \leftarrow \phi_0$;
do
$\phi_1 \leftarrow 2\phi_1$;
for ($l \leftarrow 1$; $l \le v$; l++) **do**
$T_l \leftarrow$ Find_$T_l(m, \alpha_1, ..., \alpha_m, R_l, \phi_1)$;
end do;
while ($T_1 + T_2 + \cdots + T_v > T$);
$\phi_2 \leftarrow \phi_0$;
do
$\phi_2 \leftarrow \phi_2/2$;
for ($l \leftarrow 1$; $l \le v$; l++) **do**
$T_l \leftarrow$ Find_$T_l(m, \alpha_1, ..., \alpha_m, R_l, \phi_2)$;
end do;
while ($T_1 + T_2 + \cdots + T_v < T$);
//Search ϕ in $[\phi_1, \phi_2]$
while ($\phi_2 - \phi_1 > \epsilon$) //$\epsilon$ is a very small constant
$\phi \leftarrow (\phi_1 + \phi_2)/2$;
for ($l \leftarrow 1$; $l \le v$; l++) **do**
$T_l \leftarrow$ Find_$T_l(m, \alpha_1, ..., \alpha_m, R_l, \phi)$;
end do;
if ($T_1 + T_2 + \cdots + T_v < T$)
$\phi_1 \leftarrow \phi$;
else
$\phi_2 \leftarrow \phi$;
end if;
end while;
//Calculate E
for ($l \leftarrow 1$; $l \le v$; l++) **do**
Find_$E(m, \alpha_1, \alpha_2, ..., \alpha_m, R_l, T_l)$ to get E_l;
$E \leftarrow E + E_l$;
end do;
return $T_1, T_2, ..., T_v$, and E.

Fig. 8. An algorithm to find $T_1, T_2, ..., T_v$, and E.