Received: 31 December 2021 | Revised: 5 May 2022 | Accepted: 13 July 2022

DOI: 10.1002/cpe.7285

SPECIAL ISSUE PAPER

WILEY

Efficient exact algorithms for continuous bi-objective performance-energy optimization of applications with linear energy and monotonically increasing performance profiles on heterogeneous high performance computing platforms

Hamidreza Khaleghzadeh[1] | Ravi Reddy Manumachu[2] | Alexey Lastovetsky[2]

[1]School of Computing, University of Portsmouth, Portsmouth, UK

[2]School of Computer Science, University College Dublin, Belfield, Dublin, Ireland

Correspondence
Ravi Reddy Manumachu, School of Computer Science, University College Dublin, Belfield, Dublin 4, Ireland.
Email: manumachu.reddy@gmail.com

Funding information
Science Foundation Ireland, Grant/Award Number: 14/IA/2474

Abstract

Performance and energy are the two most important objectives for optimization on heterogeneous high performance computing platforms. This work studies a mathematical problem motivated by the bi-objective optimization of data-parallel applications on such platforms for performance and energy. First, we formulate the problem and present an exact algorithm of polynomial complexity solving the problem where all the application profiles of objective type one are continuous and strictly increasing, and all the application profiles of objective type two are linear increasing. We then apply the algorithm to develop solutions for two related optimization problems of parallel applications on heterogeneous hybrid platforms, one for performance and dynamic energy and the other for performance and total energy. Our proposed solution methods are then employed to solve the two bi-objective optimization problems for two data-parallel applications, matrix multiplication and gene sequencing, on a hybrid platform employing five heterogeneous processors, namely, two different Intel multicore CPUs, an Nvidia K40c GPU, an Nvidia P100 PCIe GPU, and an Intel Xeon Phi.

KEYWORDS

bi-objective optimization, energy optimization, high performance computing, min-max optimization, min-sum optimization, performance optimization

1 | INTRODUCTION

Performance and energy are the two most important objectives for optimization on modern parallel platforms such as supercomputers, heterogeneous high performance computing (HPC) clusters, and cloud infrastructures.[1-4] State-of-the-art solutions for the bi-objective optimization problem for performance and energy on such platforms can be broadly classified into *system-level* and *application-level* categories.

System-level solution methods aim to optimize the performance and energy of the environment where the applications are executed. The methods employ application-agnostic models and hardware parameters as decision variables. The dominant decision variable in this category is Dynamic Voltage and Frequency Scaling (DVFS).[2,3,5-8]

Abbreviations: GPU, graphics processing unit; HPC, high performance computing.

These authors contributed equally to this study.

Research works[9-12] propose application-level solution methods that employ decision variables, which include the number of processes, number of threads, loop blocking factor, and workload distribution. The solution methods proposed in References 10,11 solve the bi-objective optimization problem of an application for performance and energy on homogeneous clusters of modern multicore CPUs. The solution method[9] considers the effect of heterogeneous workload distribution on bi-objective optimization of data analytics applications by simulating heterogeneity on homogeneous clusters.

Khaleghzadeh et al.[12] study bi-objective optimization of data-parallel applications for performance and energy on heterogeneous processors. The main contribution of this work is a bi-objective optimization algorithm for the case of *discrete* performance and energy functions with any arbitrary shape. The algorithm returns the Pareto front of load imbalanced solutions and best load balanced solutions. The authors also briefly study the continuous bi-objective optimization problem but *only for the simple case of two heterogeneous processors* with linear execution time and linear dynamic energy functions. They propose an algorithm to find the Pareto front and show that it is linear, containing an infinite number of solutions. While one solution is load balanced, the rest are load imbalanced. However, they do not present an algorithm to determine the workload distribution (solution in the decision vector space) corresponding to a point in the Pareto front (solution in the objective space).

In Reference 13, we study a more general continuous bi-objective optimization problem for a generic case of k heterogeneous processors. The problem is motivated by the bi-objective optimization for the performance and energy of data-parallel applications on heterogeneous HPC platforms. We now present an use case that highlights the problem.

Consider, for example, the bi-objective optimization of a highly optimized matrix multiplication application on a heterogeneous computing platform for performance and energy. The application computes the matrix product, $C = \alpha \times A \times B + \beta \times C$, where A, B, and C are matrices of size $M \times N$, $N \times N$, and $M \times N$, and α and β are constant floating-point numbers. The application invokes CUBLAS library functions for Nvidia GPUs and Intel MKL DGEMM library functions for CPUs and Intel Xeon Phi. The Intel MKL and CUDA versions used are 2017.0.2 and 9.2.148. Workload sizes range from $64 \times 10{,}112$ to $19{,}904 \times 10{,}112$ with a step size of 64 for the first dimension M.

The platform consists of five heterogeneous processors: Intel Haswell E5-2670V3 multi-core CPU (CPU_1), Intel Xeon Gold 6152 multi-core CPU (CPU_2), NVIDIA K40c GPU (GPU_1), NVIDIA P100 PCIe GPU (GPU_2), and Intel Xeon Phi 3120P (XeonPhi_1). Figure 1 presents the details of the computing platform.

Figure 2 shows the execution time functions $\{f_0(x), \ldots, f_4(x)\}$ and the dynamic energy functions $\{g_0(x), \ldots, g_4(x)\}$ of the processors against the workload size (x). Briefly, there are two types of energy consumption in computing platforms, *static* and *dynamic*. The static energy consumption is equal to the execution time of the application multiplied by the static power consumption of the platform. The application's dynamic energy consumption is equal to the platform's total energy consumption (E_T) minus the static energy consumption. The static energy consumption is the idle power of the platform (P_S) multiplied by the application's execution time (t). The static and dynamic energy consumptions during an application execution is obtained using power meters, which is considered the most accurate method of energy measurement.[14]

The execution time function shapes are continuous and strictly increasing. The energy function shapes can be approximated accurately by linear increasing functions. While the execution time profiles of the two CPUs are close to each other, the energy profile of CPU_1 is significantly higher

Intel Haswell E5-2670V3	
No. of cores per socket	12
Socket(s)	2
CPU MHz	1200.402
L1d cache, L1i cache	32 KB, 32 KB
L2 cache, L3 cache	256 KB, 30720 KB
Total main memory	64 GB DDR4
Memory bandwidth	68 GB/sec
NVIDIA K40c	
No. of processor cores	2880
Total board memory	12 GB GDDR5
L2 cache size	1536 KB
Memory bandwidth	288 GB/sec
Intel Xeon Phi 3120P	
No. of processor cores	57
Total main memory	6 GB GDDR5
Memory bandwidth	240 GB/sec

Intel Xeon Gold 6152	
Socket(s)	1
Cores per socket	22
L1d cache, L1i cache	32 KB, 32 KB
L2 cache, L3 cache	256 KB, 30976 KB
Main memory	96 GB
NVIDIA P100 PCIe	
No. of processor cores	3584
Total board memory	12 GB CoWoS HBM2
Memory bandwidth	549 GB/sec

FIGURE 1 Specifications of the five heterogeneous processors, Intel Haswell multicore CPU, Nvidia K40c, Intel Xeon Phi 3120P, Intel Skylake multicore CPU and Nvidia P100 PCIe

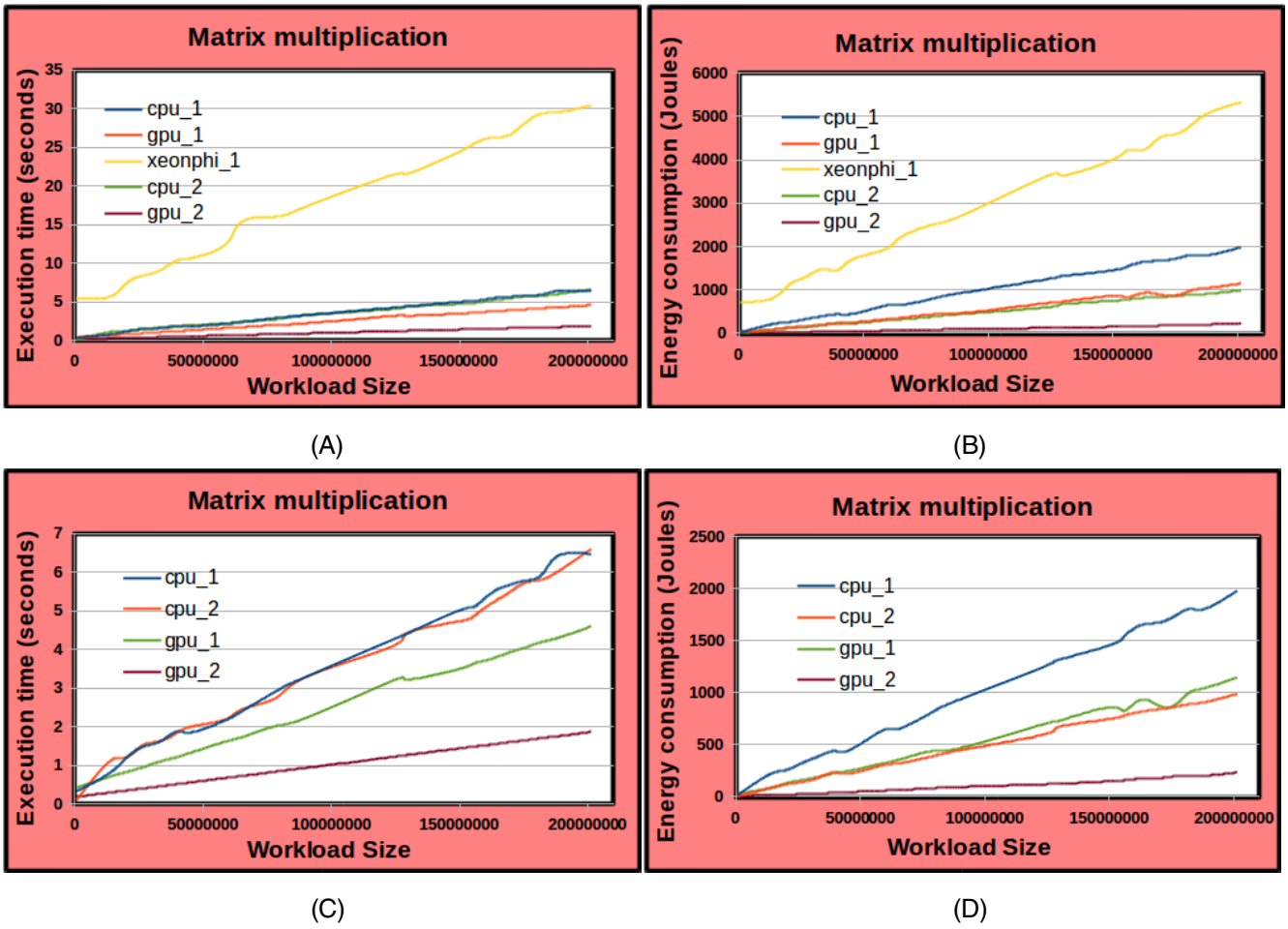


FIGURE 2 (A) and (B) contain the execution time and energy profiles of the five heterogeneous processors (Figure 1) employed in the matrix multiplication application. (C) and (D) are the same except that they do not contain the profiles for Xeon Phi whose energy profile dominates the other energy profiles. While the execution time profiles of the two CPUs are close to each other, the energy profile of CPU_1 is significantly higher than that of CPU_2

than that of CPU_2. The optimization goal is to find workload distributions of the workload size n ($\{x_0, \ldots, x_4\}, \sum_{i=0}^{4} x_i = n$) minimizing the execution time ($\max_{i=0}^{4} f_i(x_i)$) and the dynamic energy consumption ($\sum_{i=0}^{4} g_i(x_i)$) during the parallel execution of the application.

In Reference 13, we solve the continuous optimization problem for such shapes of performance and dynamic energy functions. We first formulate the mathematical problem, which for a given positive real number n aims to find a vector $X = \{x_0, \cdots, x_{k-1}\} \in \mathbb{R}^k_{\geq 0}$ such that $\sum_{i=0}^{k-1} x_i = n$, minimizing the max of k-dimensional vector of functions of objective type one and the sum of k-dimensional vector of functions of objective type two. We then propose an exact algorithm of polynomial complexity solving the case where all the functions of objective type one are continuous and strictly increasing, and all the functions of objective type two are linear increasing. The algorithm exhibits polynomial complexity.

In this work, we apply our proposed algorithm to solve two related optimization problems of parallel applications on heterogeneous hybrid platforms, one for performance and dynamic energy and the other for performance and total energy. We believe both optimization problems are pertinent to optimizing applications on modern heterogeneous hybrid HPC platforms due to following reasons. First, the thermal design power (TDP) of the multicore CPUs and accelerators has either remained the same or increased with each new generation. For example, the TDPs of the K40 and P100 GPUs used in our experiments are 235 and 250 W. The TDP of the latest generation A100 GPU is 250 W. The TDPs of Intel Xeon Gold 6152 and Intel Xeon E5-2670 v3 used in our experiments are 140 and 120 W. The TDP of the latest generation Icelake Xeon Gold 6354 is 205 W. Second, although the static energy consumption of devices is decreasing with every new generation, the total static energy consumption of a heterogeneous hybrid HPC platform with two or more such devices is still significant. Finally, the improvements in dynamic power consumption are not similar in magnitude to those for static power consumption.

We first formulate and solve a bi-objective optimization problem of parallel applications on heterogeneous hybrid platforms for performance and dynamic energy. The solution to the problem is a straightforward application of our algorithm proposed in Reference 13.

We then formulate a bi-objective optimization problem of parallel applications on heterogeneous hybrid platforms for performance and total energy. We prove a theorem that lays the foundation for our algorithm solving the theorem. The theorem states that a solution vector $X = \{x_0, \cdots, x_{k-1}\} \in \mathbb{R}^k_{\geq 0}$, $\sum_{i=0}^{k-1} x_i = n$ is Pareto-optimal for execution time and total energy if and only if it is Pareto-optimal for execution time and dynamic energy and there is no solution vector X_1 such that X_1 is Pareto-optimal for execution time and dynamic energy and $E_T(X_1) < E_T(X)$. Then, we propose an algorithm for solving the problem. The correctness of the algorithm follows from the theorem. Finally, we prove the algorithm has polynomial complexity.

The proposed algorithms (in Reference 13 and this work) are then employed to solve the two bi-objective optimization problems for two data-parallel applications, matrix multiplication and gene sequencing, employing five heterogeneous processors, two Intel multicore CPUs, an Nvidia K40c GPU, an Nvidia P100 PCIe GPU, and an Intel Xeon Phi (Figure 1). For the workloads and the platform employed in our experiments, the algorithms provide continuous piecewise linear Pareto fronts for performance and dynamic energy and performance and total energy where the performance-optimal point is the load balanced configuration of the application.

Based on our experiments, the maximum dynamic energy savings can be up to 17% while tolerating a performance degradation of 5% (an energy savings of 106 J for an execution time increase of 0.05 seconds) for the matrix multiplication application. The maximum total energy savings is 8%. The dynamic energy and total energy savings for the gene sequencing application accepting a 1% performance hit are 23% and 16%.

The main original contributions of this work are:

- Mathematical formulations of the bi-objective optimization problem of parallel applications on heterogeneous hybrid platforms for performance and dynamic energy and for performance and total energy;
- A theorem that lays the foundation for our algorithm solving the bi-objective optimization problem for performance and total energy. The theorem states that a solution vector X is Pareto-optimal for execution time and total energy if and only if it is Pareto-optimal for execution time and dynamic energy and there is no solution vector X_1 such that X_1 is Pareto-optimal for execution time and dynamic energy and $E_T(X_1) < E_T(X)$;
- An exact algorithm of polynomial complexity solving the bi-objective optimization problem for performance and total energy and whose correctness follows from the theorem;
- Experimental study of the practical efficacy of our proposed algorithms for optimization of two data-parallel applications, matrix multiplication and gene sequencing, on a platform comprising five heterogeneous processors that include two multicore CPUs, two GPUs, and one Intel Xeon Phi. We demonstrate that the algorithms provide continuous piecewise linear Pareto fronts for performance and dynamic energy and performance and total energy for the workloads and the platform employed in our experiments.

The rest of the paper is organized as follows. We discuss the related work in Section 2. The formulation of the bi-objective optimization problem is presented in Section 3. In Section 4, we propose an efficient and exact algorithm solving the bi-objective optimization problem. Section 5 presents application of our proposed algorithm to optimization of heterogeneous parallel applications for performance and energy. Section 6 contains the experimental results. Finally, we conclude the paper in Section 7.

2 | RELATED WORK

2.1 | Bi-objective optimization: background

A bi-objective optimization problem can be mathematically formulated as:[15,16]

$$\text{minimize} \quad \{T(X), E(X)\}, \quad \text{Subject to} \quad X \in \mathcal{S},$$

where there are two objective functions, $T : \mathbb{R}^k \to \mathbb{R}$ and $E : \mathbb{R}^k \to \mathbb{R}$. We denote the vector of objective functions by $\mathcal{F}(X) = (T(X), E(X))^T$. The decision vectors $X = (x_0, ..., x_{k-1})^T$ belong to the (nonempty) feasible region (set) $\mathcal{S}$, which is a subset of the decision variable space $\mathbb{R}^k$. We denote the image of the feasible region by $\mathcal{Z}$ $(= \mathcal{F}(\mathcal{S}))$, and call it a feasible objective region. It is a subset of the objective space $\mathbb{R}^2$. The elements of $\mathcal{Z}$ are called objective (function) vectors or criterion vectors and denoted by $\mathcal{F}(X)$ or $z = (z_1, z_2)^T$, where $z_1 = T(X)$ and $z_2 = E(X)$ are objective (function) values or criterion values.

The objective is to minimize both the objective functions simultaneously. The objective functions are at least partly conflicting or incommensurable, due to which it is impossible to find a single solution that would be optimal for all the objectives simultaneously.

Definition 1. A decision vector $X^* \in \mathcal{S}$ is *Pareto optimal* if there does not exist another decision vector $X \in \mathcal{S}$ such that $T(X) \leq T(X^*), E(X) \leq E(X^*)$ and either $T(X) < T(X^*)$ or $E(X) < E(X^*)$ or both.[15]

An objective vector $z^* \in \mathcal{Z}$ is Pareto optimal if there is not another objective vector $z \in \mathcal{Z}$ such that $z_1 \leq z_1^*, z_2 \leq z_2^*$ and $z_j < z_j^*$ for at least one index j.

There are several classifications for methods solving bi-objective optimization problems.[15,16] Since the set of Pareto optimal solutions is partially ordered, one classification is based on the involvement of the decision-maker in the solution method to select specific solutions. There are four categories in this classification, *No preference*, *A priori*, *A posteriori*, *Interactive*. The algorithms solving bi-objective optimization problems can be divided into two major categories, *exact methods* and *metaheuristics*. While branch-and-bound is the dominant technique in the first category, genetic algorithm (GA) is popular in the second category.

2.2 | Bi-objective optimization for performance and energy on HPC platforms

There are two principal categories of methods for optimizing applications on HPC platforms for performance and energy.

2.2.1 | System-level methods

The first category of system-level solution methods aims to optimize the performance and energy of the executing environment of the applications. The dominant decision variable in this category is DVFS. DVFS reduces the dynamic power consumed by a processor by throttling its clock frequency.

Rong et al.[17] present a runtime system (CPU MISER) based on DVFS that provides energy savings with minimal performance degradation by using a performance model. Huang et al.[18] propose an eco-friendly daemon that employs workload characterization as a guide to DVFS to reduce power and energy consumption with little impact on application performance. Mezmaz et al.[19] propose a parallel bi-objective GA to maximize the performance and minimize the energy consumption in cloud computing infrastructures. Fard et al.[1] present a four-objective case study comprising performance, economic cost, energy consumption, and reliability for optimization of scientific workflows in heterogeneous computing environments. Beloglazov et al.[20] propose heuristics that consider twin objectives of energy efficiency and Quality of Service for provisioning data center resources.

Durillo et al.[3] propose a multi-objective workflow scheduling algorithm for optimization of applications executing in heterogeneous high-performance parallel and distributed computing systems. Performance and energy consumption are among the objectives. Das et al.[21] propose task mapping to optimize for energy and reliability on multiprocessor systems-on-chip with performance as a constraint. Kolodziej et al.[5] propose multi-objective GAs that aim to maximize performance and energy consumption of applications executing in green grid clusters and clouds. The performance is modeled using computation speed of a processor. The decision variable is the DVFS level. Vaibhav et al.[22] present a runtime system that performs both processor and DRAM frequency scaling and demonstrate total energy savings with minimal performance loss. Abdi et al.[23] propose multicriteria optimization where they minimize the execution time under three constraints, the reliability, the power consumption, and the peak temperature. DVFS is a key decision variable in all of these research works.

The methods proposed in References 6-8 optimize for performance under a energy budget or optimize for energy under an execution time constraint. The methods proposed in References 2,3,5 solve bi-objective optimization for performance and energy with no time constraint or energy budget.

2.2.2 | Application-level methods

The second category of application-level solution methods[9-12,24,25] use application-level decision variables and models. The most popular decision variables include the loop tile size, workload distribution, number of processors, and number of threads.

Tarplee et al.[26] employ task-mapping as a decision variable for bi-objective optimization of applications for performance and energy in a HPC platform. Aba et al.[27] present an approximation algorithm for bi-objective optimization of parallel applications running on a heterogeneous resources system for performance and total energy. The decision variable is task scheduling. Their algorithm ignores all solutions where energy consumption exceeds a given constraint and returns the solution with minimum execution time.

Reddy et al.[11,25] study bi-objective optimization of data-parallel applications for performance and energy on homogeneous clusters multicore CPUs employing only one decision variable, the workload distribution. They propose an efficient solution method. The method

accepts as input the number of available processors, the discrete function of the processor's energy consumption against the workload size, the discrete function of the processor's performance against the workload size. It outputs a Pareto-optimal set of workload distributions. Chakraborti et al.[9] consider the effect of heterogeneous workload distribution on bi-objective optimization of data analytics applications by simulating heterogeneity on homogeneous clusters. The performance is represented by a linear function of problem size and the total energy is predicted using historical data tables. Khaleghzadeh et al.[12] propose a solution method solving the bi-objective optimization problem of data-parallel applications for performance and energy on heterogeneous processors and comprising of two principal components. The first component is a data partitioning algorithm that takes as an input discrete performance and dynamic energy functions with no shape assumptions. The second component is a novel methodology employed to build the discrete dynamic energy profiles of individual computing devices, which are input to the algorithm.

Khokhriakhov et al.[28] propose a novel solution method for bi-objective optimization of multithreaded data-parallel applications for performance and dynamic energy on a single multicore processor. The method uses two decision variables, the number of identical multithreaded kernels (threadgroups) executing the application and the number of threads per threadgroup, with a given workload partitioned equally between the threadgroups.

3 | FORMULATION OF THE BI-OBJECTIVE OPTIMIZATION PROBLEM

Given a positive real number $n \in \mathbb{R}_{>0}$ and two sets of k functions each, $F = \{f_0, f_1, \cdots, f_{k-1}\}$ and $G = \{g_0, g_1, \cdots, g_{k-1}\}$, where $f_i, g_i : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}, i \in \{0, \cdots, k-1\}$, the problem is to find a vector $X = \{x_0, \cdots, x_{k-1}\} \in \mathbb{R}^k_{\geq 0}$ such that $\sum_{i=0}^{k-1} x_i = n$, minimizing the objective functions $T(X) = \max_{i=0}^{k-1} f_i(x_i)$ and $E(X) = \sum_{i=0}^{k-1} g_i(x_i)$. We use $T \times E$ to denote the objective space of this problem, $\mathbb{R}_{\geq 0} \times \mathbb{R}_{\geq 0}$.

Thus, the problem can be formulated as follows:

BOPGV(n, k, F, G):

$$
\begin{aligned}
& T(X) = \max_{i=0}^{k-1} f_i(x_i), E(X) = \sum_{i=0}^{k-1} g_i(x_i) \\
\underset{X}{\text{minimize}} \quad & \{T(X), E(X)\} \\
\text{s.t.} \quad & x_0 + x_1 + \cdots + x_{k-1} = n.
\end{aligned} \tag{1}
$$

We aim to solve BOPGV by finding both the Pareto front containing the optimal objective vectors in the objective space $T \times E$ and the decision vector for a point in the Pareto front. Thus, our solution finds a set of triplets $\Psi = \{(T(X), E(X), X)\}$ such that X is a Pareto-optimal decision vector, and the projection of Ψ onto the objective space $T \times E$ is the Pareto front symbolized by $\Psi \downarrow_{T \times E}$.

4 | BI-OBJECTIVE OPTIMIZATION PROBLEM FOR MAX OF CONTINUOUS FUNCTIONS AND SUM OF LINEAR FUNCTIONS

In this section, we solve BOPGV for the case where all functions in the set F are continuous and strictly increasing, and all functions in the set G are linear increasing, that is, $G = \{g_0, \cdots, g_{k-1}\}, g_i(x) = b_i \times x, b_i \in \mathbb{R}_{>0}, i = 0, \ldots, k-1$. Without loss of generality, we assume that the functions in G are sorted in the decreasing order of coefficients, $b_0 \geq b_1 \geq \ldots \geq b_{k-1}$.

Our solution consists of two algorithms, Algorithms 1 and 2. The first one, which we call LBOPA, constructs the Pareto front of the optimal solutions in the objective space $\Psi \downarrow_{T \times E}$. The second algorithm finds the decision vector for a given point in the Pareto front.

The inputs to LBOPA (see Algorithm 1 for pseudo-code) are two sets of k functions each, F and G, and an input value, $n \in \mathbb{R}_{>0}$. LBOPA constructs a continuous Pareto front, consisting of $k-1$ segments $\{s_0, s_1, \cdots, s_{k-2}\}$. Each segment s_i has two endpoints, (t_i, e_i) and (t_{i+1}, e_{i+1}), which are connected by curve $P_f(t) = b_i \times n - \sum_{j=i+1}^{k-1} (b_i - b_j) \times f_j^{-1}(t)$ $(0 \leq i \leq k-2)$. Figure 3 illustrates the functions in the sets, F and G, when all functions in F are linear, $f_i(x) = a_i \times x$. In this particular case, the Pareto front returned by LBOPA will be piecewise linear, $P_f(t) = b_i \times n - t \times \sum_{j=i+1}^{k-1} \frac{b_i - b_j}{a_j}$ $(0 \leq i \leq k-2)$, as shown in Figure 3.

The main loop of the Algorithm 1 computes k end-points of the segments of the Pareto front (Lines 3-7). In an iteration i, the minimum value of objective T, t_i, is obtained using the algorithm, solving the single-objective min-max optimization problem, $\min_X \{\max_{j=i}^{k-1} f_j(x_j)\}$. We do not present the details of this algorithm. Depending on the shapes of functions, $\{f_0, \ldots, f_{k-1}\}$, one of the existing polynomial algorithms solving this problem can be employed.[29,30]

The end point $(t_{\min}, e_{\max}) = (t_0, e_0)$ represents decision vectors with the minimum value of objective T and the maximum value of objective E, while the end point $(t_{\max}, e_{\min}) = (t_{k-1}, e_{k-1})$ represents decision vectors with the maximum value of objective T and the minimum value of objective E (as illustrated for the case of all linear increasing functions in Figure 3).

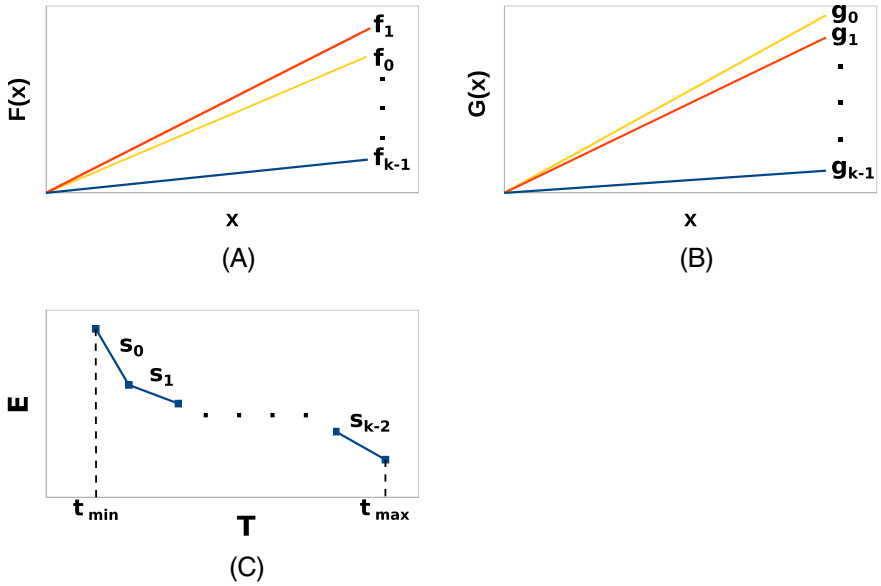


FIGURE 3 Sets F and G of k linear increasing functions each. Functions in G are arranged in the decreasing order of slopes. LBOPA returns a piecewise linear Pareto front shown in Figure (C) comprising a chain of $k-1$ linear segments.

Algorithm 1. Algorithm LBOPA constructing the Pareto front of the optimal solutions, minimizing the max of continuous and strictly increasing functions and the sum of linear increasing functions, in the objective space $T \times E$

1: **function** LBOPA(n, k, F, G)
2: $S \leftarrow \varnothing$
3: **for** $i \leftarrow 0, k-1$ **do**
4: $t_i \leftarrow \min_X \{ \max_{j=i}^{k-1} \sim f_j(x_j) \}$
5: $e_i \leftarrow b_i \times n - \sum_{j=i+1}^{k-1}(b_i - b_j) \times f_j^{-1}(t_i)$
6: $S \leftarrow S \cup (t_i, e_i)$
7: **end for**
8: **for** $i \leftarrow 0, k-2$ **do**
9: Connect (t_i, e_i) and (t_{i+1}, e_{i+1}) by curve $b_i \times n - \sum_{j=i+1}^{k-1}(b_i - b_j) \times f_j^{-1}(t)$
10: **end for**
11: **return** S
12: **end function**

Given an input $t \in [t_0, t_{k-1}]$, PARTITION (Algorithm 2) finds a decision vector $X = \{ x_0, x_1, \cdots, x_{k-1}\}$ such that $\sum_{i=0}^{k-1} x_i = n$, $\max_{i=0}^{k-1} f_i(x_i) = t$, and $\sum_{i=0}^{k-1} g_i(x_i)$ is minimal. The algorithm first initializes X with $\{x_0, x_1, \cdots, x_{k-1} \mid x_i = f_i^{-1}(t)\}$ (Line 2) so that $f_i(x_i) = t$ for all $i \in \{0, \cdots, k-1\}$. For this initial X the condition $\max_{i=0}^{k-1} f_i(x_i) = t$ is already satisfied but $\sum_{i=0}^{k-1} x_i$ may be either equal to n or greater than n. If $\sum_{i=0}^{k-1} x_i = n$, then this initial X will be the only decision vector such that $\sum_{i=0}^{k-1} x_i = n$ and $\max_{i=0}^{k-1} f_i(x_i) = t$ and hence the unique (Pareto-optimal) solution. Otherwise, $\sum_{i=0}^{k-1} x_i = n + n_{plus}$ where $n_{\text{plus}} > 0$. In that case, this initial vector X will maximize both $\sum_{i=0}^{k-1} x_i$ and $\sum_{i=0}^{k-1} g_i(x_i)$ in the set $\mathcal{X}_t$ of all vectors in the decision space satisfying the condition $\max_{i=0}^{k-1} f_i(x_i) = t$.

The algorithm then iteratively reduces elements of vector X until their sum becomes equal to n. Obviously, each such reduction will also reduce $\sum_{i=0}^{k-1} g_i(x_i)$. To achieve the maximum reduction of $\sum_{i=0}^{k-1} g_i(x_i)$, the algorithm starts from vector element x_i, the reduction of which by an arbitrary amount Δx will result in the maximum reduction of $\sum_{i=0}^{k-1} g_i(x_i)$. In our case, it will be x_0 as the functions in G are sorted in the decreasing order of coefficients b_i. Thus, at the first reduction step, the algorithm will try to reduce x_0 by n_{plus}. If $x_0 \geq n_{\text{plus}}$, it will succeed and find a Pareto-optimal decision vector $X = \{x_0 - n_{\text{plus}}, x_1, \cdots, x_{k-1}\}$. If $x_0 < n_{\text{plus}}$, it will reduce n_{plus} by x_0, set $x_0 = 0$ and move to the second step. At the second step, it will try to reduce x_1 by the reduced n_{plus}, and so on. This way the algorithm minimizes $\sum_{i=0}^{k-1} g_i(x_i)$, preserving $\max_{i=0}^{k-1} f_i(x_i) = t$ and achieving $\sum_{i=0}^{k-1} x_i = n$.

Algorithm 2. Algorithm finding a Pareto-optimal decision vector $X = \{x_0, x_1, \cdots, x_{k-1}\}$ for the problem $BOPGV(n, k, F, G)$, where functions in F are continuous and strictly increasing and functions in G are linear increasing, for a given point (t, e) from the Pareto front of this problem, $(t, e) \in \Psi \downarrow_{T\times E}$. Only the first coordinate of the input point, t, is required for this algorithm

```
1:  function PARTITION(n, k, F, t)
2:      X = {x_0, ..., x_{k-1} | x_i ← f_i^{-1}(t)}
3:      n_plus ← Σ_{i=0}^{k-1} x_i − n
4:      if n_plus<0 then return (0, 0, ∅) end if
5:      i ← 0
6:      while (n_plus>0) ∧ (i<k − 1) do
7:          if x_i ≥ n_plus then
8:              x_i ← x_i − n_plus
9:              n_plus ← 0
10:         else
11:             n_plus ← n_plus − x_i
12:             x_i ← 0
13:             i ← i + 1
14:         end if
15:     end while
16:     if n_plus>0 then return (0, 0, ∅) end if
17:     return X
18: end function
```

The correctness of LBOPA and PARTITION is proved in Theorem 1.

Theorem 1. *Consider bi-objective optimization problem $BOPGV(n, k, F, G)$ where all functions in F are continuous and strictly increasing and $G = \{g_i(x) \mid g_i(x) = b_i \times x, b_i \in \mathbb{R}_{>0}, i \in \{0, \cdots, k-1\}\}$. Then, the piecewise function S, returned by LBOPA n, k, F, G (Algorithm 1) and consisting of $k-1$ segments, is the Pareto front of this problem, $\Psi \downarrow_{T\times E}$, and for any $(t, e) \in \Psi \downarrow_{T\times E}$, Algorithm 2 returns a Pareto-optimal decision vector X such that $T(X) = t$ and $E(X) = e$.*

Proof. First, consider Algorithm 2 and arbitrary input parameters $n > 0$ and $t > 0$. If after initialization of X (Line 2) we will have $\sum_{i=0}^{k-1} x_i < n$, it means that t is too small for the given n, and for any vector $Y = \{y_0, y_1, \cdots, y_{k-1}\}$ such that $\sum_{i=0}^{k-1} y_i = n$, $\max_{i=0}^{k-1} f_i(y_i) > t$. In this case, there is no solution to the optimization problem, and the algorithm terminates abnormally.

Otherwise, the algorithm enters the *while* loop (Line 6). If $i < k-1$ upon exit from this loop, then the elements of vector X will be calculated as

$$x_j = \begin{cases} 0 & j < i \\ n - \sum_{m=j+1}^{k-1} f_m^{-1}(t) & j = i \\ f_j^{-1}(t) & j > i \end{cases}, \tag{2}$$

and therefore satisfy the conditions $\sum_{j=0}^{k-1} x_j = n$ and $\max_{j=0}^{k-1} f_j(x_j) = t$. Moreover, the total amount of n will be distributed in X between vector elements with higher indices, which have lower G cost, $g_i(x)$, because $b_i \geq b_{i+1}, \forall i \in \{0, \cdots, k-2\}$. Therefore, for any other vector $Y = \{y_0, y_1, \cdots, y_{k-1}\}$ satisfying these two conditions, we will have $\sum_{i=0}^{k-1} g_i(y_i) \geq \sum_{i=0}^{k-1} g_i(x_i)$. Indeed, such a vector Y can be obtained from X by relocating certain amounts from vector elements with higher indices to vector elements with lower indices, which will increase the G cost of the relocated amounts. Thus, when the algorithm exits from the *while* loop with $i < k-1$, it returns a Pareto-optimal vector X.

If the algorithm exits from the *while* loop with $i = k-1$, it will mean that t is too big for the given n. We would still have $n_{\text{plus}} > 0$ to take off the last vector element, x_{k-1}, but if we did it, we would make $\max_{j=0}^{k-1} f_j(x_j) < t$. This way we would construct for the given n a decision vector, which minimizes $\sum_{i=0}^{k-1} g_i(x_i)$ but whose $\max_{j=0}^{k-1} f_j(x_j)$ will be less than t, which means that no decision vector X such that $\max_{j=0}^{k-1} f_j(x_j) = t$ can be Pareto optimal. Therefore, in this case the algorithm also terminates abnormally.

Thus, for any $t \in T$, Algorithm 2 either finds a Pareto-optimal decision vector X such that $T(X) = t$ and $E(X) = \sum_{i=0}^{k-1} b_i \times x_i = e$, or returns abnormally if such a vector does not exist. Let Algorithm 2 return normally, and the loop variable i be equal to s upon exit from the loop. Then, according to Formula (2), $e = \sum_{i=0}^{k-1} b_i \times x_i = b_s \times (n - \sum_{i=s+1}^{k-1} f_i^{-1}(t)) + \sum_{i=s+1}^{k-1} (b_i \times f_i^{-1}(t)) = b_s \times n - \sum_{i=s+1}^{k-1} (b_s - b_i) \times f_i^{-1}(t)$, where s, n, b_i, b_s, a_i are all known constants. Therefore, the Pareto front $e = P_f(t)$ can be expressed as follows:

$$e = P_f(t) = b_s \times n - \sum_{i=s+1}^{k-1} (b_s - b_i) \times f_i^{-1}(t)$$

$$t_{\min} = \min_X \{ \max_{j=i}^{k-1} f_j(x_j) \}, t_{\max} = f_{k-1}(n)$$

$$t \in [t_{\min}, t_{\max}], \quad s \in \mathbb{Z}_{[0,k-2]},$$

which is the analytical expression of the piecewise function constructed by Algorithm 1 (LBOPA). ∎

Theorem 2. *The time complexity of LBOPA (Algorithm 1) is* $\mathcal{O}(k^3 \times \log_2 n)$. *The time complexity of PARTITION (Algorithm 2) is* $\mathcal{O}(k)$.

Proof. The *for* loop in LBOPA (Algorithm 1, Lines 3–7) has k iterations. At each iteration i, the computation of t_i has a time complexity of $\mathcal{O}(k^2 \times \log_2 n)$,[29] the computation of e_i has a time complexity of $\mathcal{O}(k)$, and the insertion of the point in the set $\mathcal{S}$ has complexity $\mathcal{O}(1)$. Therefore, the time complexity of the loop is $\mathcal{O}(k^3 \times \log_2 n)$. The time complexity of the loop (Lines 8–10) is $\mathcal{O}(k)$. Therefore, the time complexity of LBOPA is $\mathcal{O}(k^3 \times \log_2 n)$.

Let us consider the PARTITION algorithm. The initialization of X (Line 2) and computation of n_{plus} has time complexity $\mathcal{O}(k)$ each. The while loop (Lines 6–15) iterates as long as $n_{\text{plus}} > 0$ and $i < k - 1$, of which $i < k - 1$ is the worst case scenario. The time complexity of the loop is, therefore, $\mathcal{O}(k)$. Therefore, the time complexity of PARTITION is bounded by $\mathcal{O}(k)$. ∎

5 | APPLICATION OF THE BI-OBJECTIVE ALGORITHMS TO OPTIMIZATION OF HETEROGENEOUS PARALLEL APPLICATIONS FOR PERFORMANCE AND ENERGY

In this section, we apply the LBOPA and PARTITION algorithms to optimization of heterogeneous parallel applications for performance and energy. We look at two bi-objective optimization problems, performance and dynamic energy and performance and total energy.

5.1 | Bi-objective optimization for performance and dynamic energy

The bi-objective optimization problem for performance and dynamic energy is a direct application of BOPGV where the functions in F and G are the execution time and dynamic energy functions, respectively. $f_i(x)$ is the execution time of the problem size x on processor P_i and $g_i(x)$ represents the amount of dynamic energy consumed by P_i to execute the problem size x.

The solution to the problem is given by LBOPA and PARTITION algorithms for the case where all functions in the set F are continuous and strictly increasing, and all functions in the set G are linear increasing. The algorithms determine a set of tuples, $\Psi_{DE} = \{(T(X), E(X), X)\}$ where X is a Pareto-optimal decision vector and $T(X)$ and $E(X)$ are the execution time and the dynamic energy consumption corresponding to X. The projection of Ψ_{DE} onto the objective space $T \times E$ is the Pareto front symbolized by $\Psi_{DE} \downarrow_{T \times E}$. The projection of Ψ_{DE} onto the decision vector space X is the set of workload distributions signified by $\Psi_{DE} \downarrow_X$.

If the application requires integer solutions, $X_I = \{x_0, \cdots, x_{k-1}\} \in \mathbb{Z}^k_{\geq 0}$ such that $\sum_{i=0}^{k-1} x_i = n$, we will find the closest approximation to the real-valued solution vector, X, output by PARTITION, in the Euclidean space.

5.2 | Bi-objective optimization for performance and total energy

We start with the formulation for bi-objective optimization problem for performance and total energy. It is an extension of BOPGV.

5.2.1 | Problem formulation

Consider a workload size n executed using p heterogeneous processors, whose execution time and dynamic energy functions are given by the two sets, F and G. Let $P_S \in \mathbb{R}^+$ be the static power consumption of the platform, which is a constant. The problem is then to find a vector $X = \{x_0, \cdots, x_{k-1}\} \in \mathbb{R}^k_{\geq 0}$ such that $\sum_{i=0}^{k-1} x_i = n$, minimizing the objective functions $T(X)$ and $E_T(X) = E(X) + P_S \times T(X)$. We use $T \times E_T$ to denote the objective space of this problem, $\mathbb{R}_{\geq 0} \times \mathbb{R}_{\geq 0}$.

Thus, the problem can be formulated as follows:

$$\textbf{BOPPTE}(n, k, F, G, P_S):$$

$$T(X) = \max_{i=0}^{k-1} f_i(x_i), E(X) = \sum_{i=0}^{k-1} g_i(x_i)$$

$$\underset{X}{\text{minimize}} \quad \{T(X), E(X) + P_S \times T(X)\}$$

$$\text{s.t.} \quad x_0 + x_1 + \cdots + x_{k-1} = n. \tag{3}$$

Our solution for BOPPTE finds a set of tuples, $\Psi_{TE} = \{(T(X), E_T(X), X)\}$, where X is a Pareto-optimal decision vector, $T(X)$ and $E_T(X)$ are the execution time and the total energy consumption corresponding to X. The projection of Ψ_{TE} onto the objective space $T \times E_T$ is the Pareto front symbolized by $\Psi_{TE} \downarrow_{T \times E_T}$. The projection of Ψ_{TE} onto the decision vector space X is the set of solutions (workload distributions) represented by $\Psi_{TE} \downarrow_X$.

5.2.2 | Solution using the bi-objective algorithms LBOPA and PARTITION

In this section, we propose a solution that employs the LBOPA and PARTITION algorithms to solve BOPPTE for the case where all functions in the set F are continuous and strictly increasing, and all functions in the set G are linear increasing.

The solution is based on the following observations. Let $\mathcal{X}$ be the space of all feasible solutions to the BOPGV problem, that is, a set consisting of all vectors $X = \{x_0, \cdots, x_{k-1}\} \in \mathbb{R}^k_{\geq 0}$ such that $\sum_{i=0}^{k-1} x_i = n$. Let $\mathcal{O}$ be the image of $\mathcal{X}$ in the objective space $T \times E$. Then, $\mathcal{X}$ will also be the space of all feasible solutions to the BOPPTE problem. However, its image in the BOPPTE objective space, $\mathcal{O}_{T \times E_T}$. will be different, obtained by moving each point $(t, e) \in \mathcal{O}$ vertically by $P_S \times t$. This transformation guarantees that no solution $X \in \mathcal{X}$, which is not Pareto optimal for BOPGV, will become Pareto optimal for BOPPTE. Indeed, as we consider the case when all functions in F are continuous and strictly increasing, and all functions in G are linear increasing, the Pareto front constructed by LBOPA for the BOPGV problem will be a continuous decreasing function. Therefore, there exists a BOPGV Pareto-optimal X^*, which dominates X so that $T(X^*) = T(X)$ and $E(X^*) < E(X)$. The images of X^* and X in the BOPPTE objective space will be $(T(X), E(X^*) + P_S \times T(X))$ and $(T(X), E(X) + P_S \times T(X))$, respectively. As $(T(X), E(X^*) + P_S \times T(X)) < (T(X), E(X) + P_S \times T(X))$, X^* will be dominating X in the BOPPTE space as well.

Thus, solutions, which are not BOPGV Pareto optimal, cannot be BOPPTE Pareto optimal. It means that if a solution is BOPPTE Pareto optimal, then it must be BOPGV Pareto optimal; that is, the BOPPTE set of Pareto-optimal solutions is a subset of the BOPGV set of Pareto-optimal solutions. By construction, the image of the set of BOPGV Pareto-optimal solutions in the BOPPTE objective space will be a continuous function of objective T but not necessarily decreasing. Therefore, different points belonging to this function will have different T coordinates but may have the same E_T coordinate. Obviously, if we have two points with the same E_T coordinate, the one with greater T coordinate will be the image of the inferior solution, which should be removed from the BOPPTE set of Pareto-optimal solutions.

These observations can be summarised as the following theorem.

Theorem 3. *A solution vector $X \in \mathcal{X}$ is Pareto-optimal for execution time and total energy if and only if it is Pareto-optimal for execution time and dynamic energy and there is no solution vector X_1 such that X_1 is Pareto-optimal for execution time and dynamic energy and $E_T(X_1) < E_T(X)$.*

We now propose an algorithm LBOPA-TE that along with PARTITION solves BOPPTE for the case where all functions in the set F are continuous and strictly increasing, and all functions in the set G are linear increasing. It returns the Pareto front for execution time and total energy. The correctness of LBOPA-TE follows from the Theorem 3. It invokes LBOPA to obtain the piecewise linear Pareto front for execution time and dynamic energy.

LBOPA-TE (Algorithm 3) takes input the workload size n; the two sets of k functions, $\{F, G\}$; the base power of the computing platform, P_S, and the machine precision, ε. It returns a piecewise linear Pareto front for execution time and total energy. The Pareto front is a set of segments, S_{TE}, where a segment s_i is represented by a 4-tuple, $(t_i, e_i, t_{i+1}, e_{i+1})$, with the left endpoint, (t_i, e_i), and the right endpoint, (t_{i+1}, e_{i+1}). The coordinates of the points in each segment have indices $\{0, 1, 2, 3\}$. Therefore, $S_{TE}[i][j]$ gives the j th coordinate in the i th segment.

Given the first coordinate of the point, $(t, e_t) \in \Psi_{TE} \downarrow_{T \times E_T}$, the workload size n, the set, F, the PARTITION algorithm returns the workload distribution associated with the point.

We now explain the main steps of LBOPA-TE (Algorithm 3). Line 2 contains the call to LBOPA to determine the piecewise linear Pareto front for execution time and dynamic energy. In the *for* loop (Lines 4–7), the structure, S_{tmp}, containing tuples (line segments) for execution time and total energy is initialized. The slopes of the line segments are determined and stored in $\mathcal{M}_{TE}$ (Line 6). In Line 9, the output Pareto front, S_{TE}, is initialized with the performance-optimal point, $(t_{\text{min}}, e_{\text{max}}, \phi, \phi)$, corresponding to the load-balanced workload distribution. The second *for* loop (Lines 10–20) iterates through the line segments, $i \in S_{\text{tmp}}, \forall i = \{0, \cdots, k-2\}$. The first if condition checks whether the slope of the line segment i, $\mathcal{M}_{TE}[i]$, is greater than or equal to 0, or the energy of its right endpoint is greater than e_{min}. If so, the line segment i is not added to S_{TE} since all the solutions lying on it are dominated by the solution with the energy, e_{min}.

Algorithm 3. Algorithm LBOPA-TE determining the Pareto front for execution time and total energy

1: **function** LBOPA-TE($n, k, F, G, P_S, \epsilon$)
2: $\{(t_i, e_i, t_{i+1}, e_{i+1}) \mid i = 0, \cdots, k-2\} \leftarrow$ LBOPA(n, k, F, G)
3: $S_{tmp} \leftarrow \Phi$; $\mathcal{M}_{tmp} \leftarrow \Phi$
4: **for** $i \leftarrow 0, k-2$ **do**
5: $S_{tmp}[i] \leftarrow (t_i, e_i + P_S \times t_i, t_{i+1}, e_{i+1} + P_S \times t_{i+1})$
6: $\mathcal{M}_{tmp}[i] \leftarrow \frac{e_{i+1} - e_i + P_S \times (t_{i+1} - t_i)}{t_{i+1} - t_i}$
7: **end for**
8: $t_{min} \leftarrow S_{tmp}[0][0]$; $e_{max} \leftarrow S_{tmp}[0][1]$; $e_{min} \leftarrow S_{tmp}[0][3]$
9: $S_{TE} \leftarrow \{(t_{min}, e_{max}, \phi, \phi)\}$
10: **for** $i \leftarrow 0, k-2$ **do**
11: **if** ($\mathcal{M}_{tmp}[i] \geq 0$) **Or** ($S_{tmp}[i][3] > e_{min}$) **then**
12: **continue**
13: **else if** ($S_{tmp}[i][1] > e_{min}$) **And** ($S_{tmp}[i][3] < e_{min}$) **then**
14: $e_{poi} \leftarrow e_{min} - \epsilon$; $t_{poi} \leftarrow S_{tmp}[i][0] + \frac{e_{poi} - S_{tmp}[i][1]}{\mathcal{M}_{tmp}[i]}$
15: $S_{TE}[i] \leftarrow (t_{poi}, e_{poi}, S_{tmp}[i][2], S_{tmp}[i][3])$
16: $e_{min} \leftarrow S_{tmp}[i][3]$
17: **else**
18: $S_{TE}[i] \leftarrow S_{tmp}[i]$; $e_{min} \leftarrow S_{tmp}[i][3]$
19: **end if**
20: **end for**
21: **return** S_{TE}
22: **end function**

Lines 13–17 represent the case when e_{min} lies between the energies of the endpoints of the line segment i signifying that a fragment of the line segment satisfies Pareto-optimality. The point of intersection, $(t_{\text{poi}}, e_{\text{min}} - \varepsilon)$, of the lines $y = e_{\text{min}} - \varepsilon$ and the line segment with the slope $\mathcal{M}_{\text{tmp}}[i]$ is determined. The line segment represented by the tuple, $(t_{\text{poi}}, e_{\text{poi}}, S_{\text{tmp}}[i][2], S_{\text{tmp}}[i][3])$, is Pareto-optimal and is added to S_{TE}. Line 18 represents the case of the line segment whose points satisfy Pareto-optimality. Therefore, it is stored in S_{TE} at index i.

If no solutions are added to S_{TE} in the *for* loop, then S_{TE} will contain only the performance-optimal point corresponding to the load-balanced workload distribution.

We illustrate LBOPA-TE using an example shown in the Figure 4. The number of processors employed in the example is four. The static power consumption, P_S, is assumed to be 5 W. The Pareto front for execution time and dynamic energy ($\Psi_{DE} \downarrow_{T \times E}$) is given by the blue line in Figure 4A. It contains four segments. The static energy consumption as a function of execution time ($5 \times t$) is shown as an orange line. Figure 4B shows the execution time versus total energy curve (highlighted in green) obtained by adding the static energy consumptions to the energies in the execution time and dynamic energy Pareto front. In Figure 4C, the solutions highlighted in red are the non-Pareto-optimal solutions removed by LBOPA-TE in Lines 13–17. The output Pareto front for execution time and total energy, $\Psi_{TE} \downarrow_{T \times E_T}$, is shown Figure 4D.

Theorem 4. *The time complexity of LBOPA-TE is* $\mathcal{O}(k^3 \times \log_2 n)$.

Proof. LBOPA-TE invokes LBOPA in Line 2. The time complexity of LBOPA is $\mathcal{O}(k^3 \times \log_2 n)$. In the loop (Lines 4–7), the insertion of a line segment represented by a 4-tuple in Line 5 and the insertion of the segment slope in Line 6 each have a time complexity of $\mathcal{O}(1)$. In the loop (Lines 9–20), the time complexity of adding a line segment to S_{TE} is also $\mathcal{O}(1)$. Therefore, the *for* loops (Lines 4–7, Lines 9–20) each have a time complexity of $\mathcal{O}(k)$. Hence, the time complexity of LBOPA-TE is $\mathcal{O}(k^3 \times \log_2 n)$. ∎

6 | EXPERIMENTAL RESULTS AND DISCUSSION

We analyze the proposed algorithms for two data-parallel applications, matrix multiplication and gene sequencing, executed on a platform comprising the five heterogeneous processors illustrated in Figure 1.

We first describe the methodology to construct the discrete execution time and the dynamic energy profiles based on system-level physical power measurements using power meters for the processors involved in the execution of our applications. We then present the applications and the experimental results.

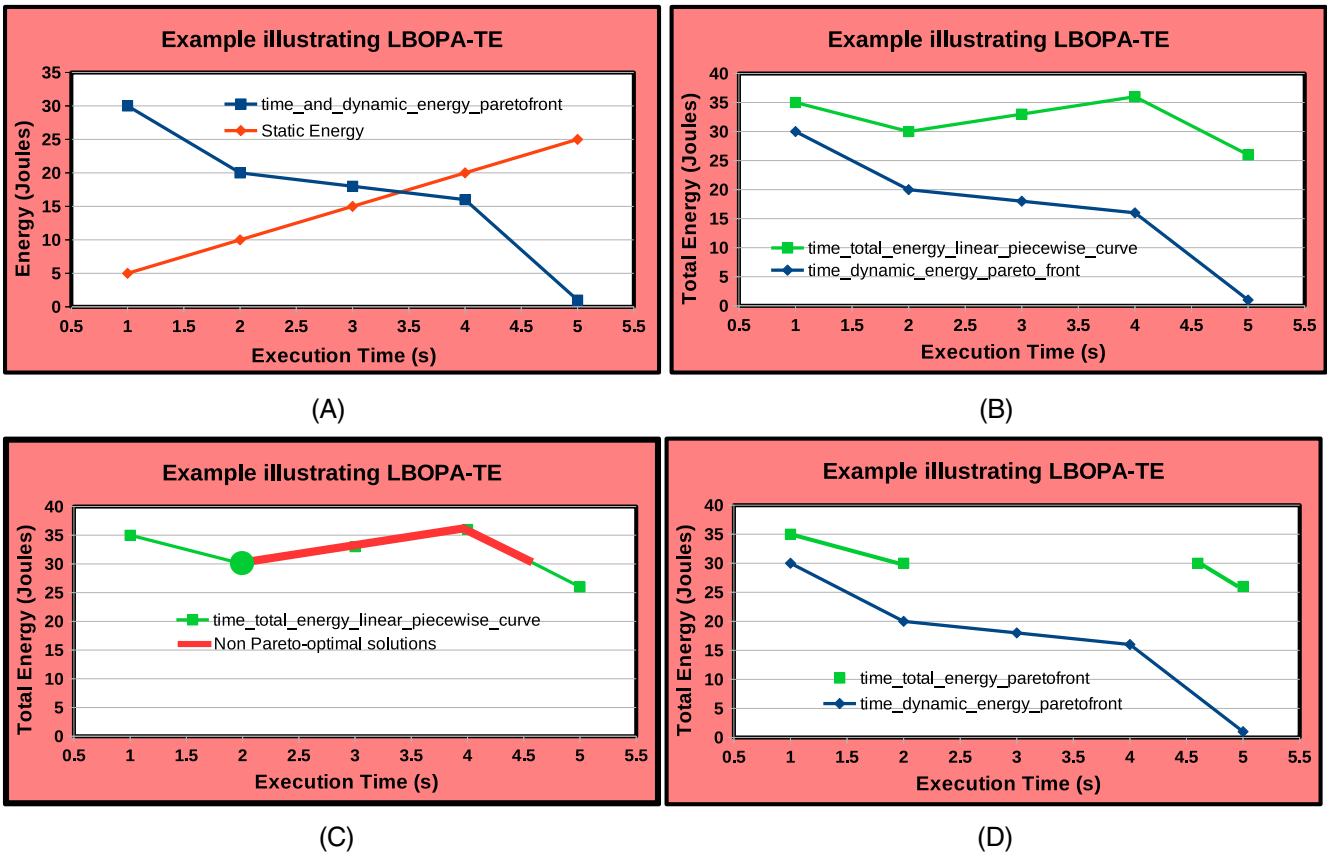


FIGURE 4 Example illustrating LBOPA-TE using four processors. The static power assumed is 5 W. (A) shows the piecewise linear Pareto front for the execution time and dynamic energy. (B) displays the piecewise linear curve for execution time and total energy obtained by adding the static energy consumption to the Pareto front for the execution time and dynamic energy in the first *for* loop of the algorithm. (C) shows the non-Pareto-optimal solutions highlighted in red that are removed. The non-Pareto-optimal solutions are dominated by the solution displayed as a solid green circle. (D) shows the output Pareto front for execution time and total energy.

Our platform is equipped with WattsUp Pro power meters between the wall A/C outlets and the input power sockets. The power meters capture the total power consumption of the node. They have data cables connected to USB ports of the node. A Perl script collects the data from the power meter using the serial USB interface. The execution of these scripts is nonintrusive and consumes insignificant power. The power meters are periodically calibrated using an ANSI C12.20 revenue-grade power meter, Yokogawa WT210. The maximum sampling speed of the power meters is one sample every second. The accuracy specified in the data sheets is $\pm 3\%$. The minimum measurable power is 0.5 watts. The accuracy at 0.5 W is ± 0.3 W.

To ensure the reliability of our results, we follow a statistical methodology where a sample mean for a response variable (energy, time, PMC, utilization variables) is obtained from multiple experimental runs. The sample mean is calculated by executing the application repeatedly until it lies in the 95% confidence interval and a precision of 0.025 (2.5%) is achieved. For this purpose, Student's *t*-test is used assuming that the individual observations are independent and their population follows the normal distribution. We verify the validity of these assumptions using Pearson's chi-squared test.

6.1 | Methodology to construct execution time and dynamic energy profiles

We employ an experimental methodology,[14 31] that accurately models the energy consumption by a hybrid data-parallel application executing on a heterogeneous HPC platform containing different computing devices using system-level power measurements provided by power meters. The automated software tool, HCLWATTSUP,[32] provides the dynamic and total energy consumptions based on system-level physical power measurements using power meters. The tool has no overhead and, therefore, does not influence the energy consumption of the application. HCLWATTSUP gives the static power consumption of our platform when it does not execute any application. Based on our measurements, it is 410 W.

A data-parallel application executing on this heterogeneous hybrid platform consists of several kernels (generally speaking, multithreaded) running in parallel on different computing devices of the platform. The proposed algorithms for solving the bi-objective optimisation problem for performance and energy requires individual performance and dynamic energy profiles of all the kernels.

Due to tight integration and severe resource contention in heterogeneous hybrid platforms, the load of one computational kernel may significantly impact others' performance to the extent of preventing the ability to model the performance and energy consumption of each kernel in hybrid applications individually.[33] Therefore, we only consider configurations where one CPU kernel or accelerator kernel runs on the corresponding device. Each group of cores executing an individual kernel of the application is modeled as an abstract processor so that the executing platform is represented as a set of heterogeneous abstract processors. We ensure that the sharing of system resources is maximized within groups of computational cores representing the abstract processors and minimized between the groups. This way, the contention and mutual dependence between abstract processors are minimized.

We thus model our platform by five abstract processors, CPU_1, GPU_1, xeonphi_1, CPU_2, and GPU_2. CPU_1 contains 22 (out of the total 24 physical) CPU cores. GPU_1 involves the Nvidia K40c GPU and a host CPU core connected to this GPU via a dedicated PCI-E link. CPU_2 comprises 10 (out of the total 12 physical) CPU cores. XeonPhi_1 is made up of one Intel Xeon Phi 3120P and its host CPU core connected via a dedicated PCI-E link. GPU_2 involves the Nvidia P100 PCI-E GPU and a host CPU core connected to this GPU via a dedicated PCI-E link. Since there should be a one-to-one mapping between the abstract processors and computational kernels, any hybrid application executing on the node should consist of three kernels, one kernel per computational device, running in parallel. Because the abstract processors contain CPU cores that share some resources such as main memory and QPI, they cannot be considered entirely independent. Therefore, the performance of these loosely coupled abstract processors must be measured simultaneously, thereby taking into account the influence of resource contention.

The execution time profiles of the abstract processors are experimentally built separately using an automated build procedure using OpenMP threads where one thread is mapped to one abstract processor. To account for the influence of resource contention, all the abstract processors execute the same workload simultaneously and their execution times are measured. The execution time for accelerators includes the time taken to transfer data between the host and devices.

The dynamic energy profiles of the abstract processors are constructed using the additive approach.[12] In the additive approach, the dynamic energy profiles of the five processors are constructed serially. The combined profile where the individual dynamic energy consumptions are totaled for each data point is then obtained. Then, the dynamic energy profile employing all the processors in parallel is built. The difference between the parallel and combined dynamic energy profiles is observed. We find that the average difference between parallel and combined dynamic energy profiles is around 2.5% for the applications and within the statistical accuracy threshold set in our experiments. Both the parallel and combined profiles also follow the same pattern. Therefore, we conclude that the processors in our experiments satisfy the additive hypothesis: the abstract processors are loosely coupled and do not interfere during the application. *Thus, we conclude that the dynamic energy profiles of the five processors can be constructed serially or in parallel for our experimental platform and applications.*

6.1.1 | Precautions to rule out interference of other components in dynamic energy consumption

Several precautions are taken in computing energy measurements to eliminate any potential interference of the computing elements that are not part of the given abstract processor running the given application kernel. First, we group abstract processors so that a given abstract processor constitutes solely the computing elements involved to run a given application kernel. Hence, the dynamic energy consumption will solely reflect the work done by the computing elements of the given abstract processor executing the application kernel.

Consider the DGEMM application kernel executing on the abstract processor CPU_1, which comprises CPU and DRAM. The *HCLWattsUp* API function gives the total energy consumption of the server during the execution of an application. The energy consumption includes the contribution from all components such as NIC, SSDs, and fans. We ensure that the application exercises only the CPUs and DRAM and not the other components so that the dynamic energy consumption reflects the contribution of only these two components. The following steps are employed to achieve this goal:

- The disk consumption is monitored before and during the application run and ensure no I/O is performed by the application using tools such as *sar*, and *iotop*;
- The problem size used in executing an application does not exceed the main memory and that swapping (paging) does not occur;
- The application does not use the network by monitoring using tools such as *sar*, and *atop*;
- The application kernel's CPU affinity mask is set using SCHED API's system call, SCHED_SETAFFINITY(). To bind the DGEMM application kernel, we set its CPU affinity mask to 11 physical CPU cores of Socket 1 and 11 physical CPU cores of Socket 2.

Fans are also a great contributor to energy consumption. On our platform, fans are controlled in two zones: (a) zone 0: CPU or System fans, (b) zone 1: Peripheral zone fans. There are four levels to control the speed of fans:

- Standard: BMC control of both fan zones, with the CPU zone based on CPU temp (target speed 50%) and Peripheral zone based on PCH temp (target speed 50%);
- Optimal: BMC control of the CPU zone (target speed 30%), with Peripheral zone fixed at low speed (fixed 30%);
- Heavy IO: BMC control of CPU zone (target speed 50%), Peripheral zone fixed at 75%;
- Full: all fans are running at 100%.

We set the fans at full speed before launching the experiments to rule out fans' contribution to dynamic energy consumption. When set at full speed, the fans run consistently at $\sim$ 13,400 rpm. In this way, fans consume the same amount of power that is included in the static power of the platform. Furthermore, we monitor the server's temperatures and the fans' speeds with the help of Intelligent Platform Management Interface (IPMI) sensors, both with and without the application run. We find that there are no significant differences in temperature and the speeds of fans are the same in both scenarios.

Thus, we ensure that the dynamic energy consumption measured reflects the contribution solely by the abstract processor executing the given application kernel.

6.2 | Applications used in the experiments

The matrix multiplication application computes $C = \alpha \times A \times B + \beta \times C$, where A, B, and C are matrices of size $m \times n$, $n \times n$, and $m \times n$, and α and β are floating-point constants. The application uses Intel MKL DGEMM for CPUs, ZZGEMMOOC out-of-card package[34] for Nvidia GPUs, and XeonPhiOOC out-of-card package[34] for Intel Xeon Phis. ZZGEMMOOC and XeonPhiOOC packages reuse CUBLAS and MKL BLAS for in-card DGEMM calls. The out-of-card packages allow the GPUs and Xeon Phis to execute computations of arbitrary size. The Intel MKL and CUDA versions used on HCLServer01 are 2017.0.2 and 7.5, and on HCLServer02 are 2017.0.2 and 9.2.148. Workload sizes range from $64 \times 10{,}112$ to $28{,}800 \times 10{,}112$ with a step size of 64 for the first dimension m. The speed of execution of a given problem size $m \times n$ is calculated as $(2 \times m \times n^2)/t$ where t is the execution time.

The gene sequencing application deals with the alignment of DNA or protein sequences. It employs the Smith-Waterman algorithm (SW),[35,36] which uses a dynamic programming (DP) approach to determine the optimal local alignment score of two sequences, a query sequence of length m and a database sequence of length n. The time and space complexities of the SW DP algorithm are $O(m \times n)$ and $O(m)$, where $m < n$, assuming the use of refined linear-space methods. The speed of execution of the application for a given workload size, $(m + n)$, is calculated as $(m \times n)/t$ where t is the execution time. The speed is usually measured in GCUPS, which stands for Billions of Cell Updated per Second.

The application employs the five heterogeneous processors, CPU_1, GPU_1, xeonphi_1, CPU_2, and GPU_2. The application invokes optimized SW routines provided by SWIPE for Multicore CPUs,[37] CUDASW++3.0 for Nvidia GPU accelerators,[38] and SWAPHI for Intel Xeon Phi accelerators.[39] All the computations are in-card.

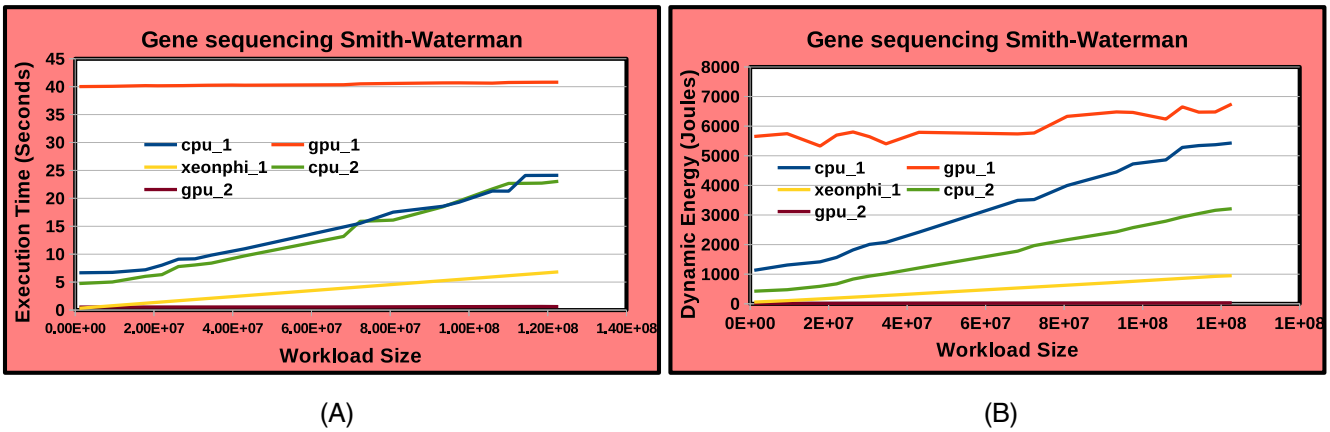
The performance and dynamic energy profiles for the matrix multiplication and gene sequencing applications are shown in the Figures 2 and 5. The input performance and dynamic energy functions, (F, G), to LBOPA and PARTITION are linear approximations of the profiles.

To demonstrate the practical efficacy and the most interesting aspects of our algorithms, we select two workloads, $12{,}352 \times 10{,}112$ and $15{,}552 \times 10{,}112$, for the matrix multiplication and the workload $29{,}312 \times 163{,}841$ for the gene sequencing application. First, the workloads provide shapes of Pareto fronts with steep slopes and a wide range of performance-energy tradeoffs for both performance and dynamic energy and performance and total energy. Second, the workloads allow us to demonstrate scenarios where the set of Pareto-optimal solutions for performance and total energy is equal to the set of Pareto-optimal solutions for performance and dynamic energy ($\Psi_{TE}\downarrow_X = \Psi_{DE}\downarrow_X$) and where it is only a proper subset ($\Psi_{TE}\downarrow_X \subset \Psi_{DE}\downarrow_X$).

6.3 | Performance-dynamic energy Pareto fronts

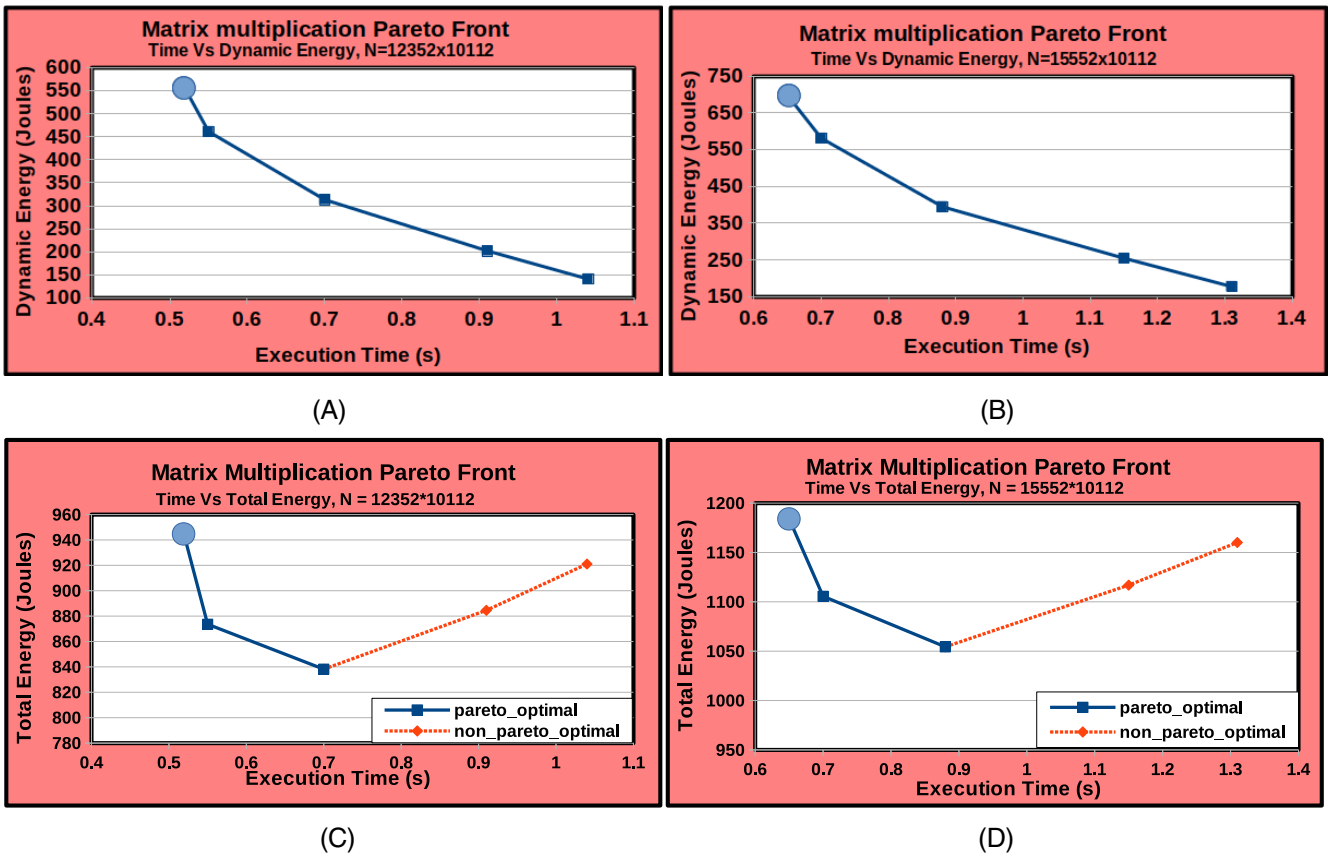
Figure 6 shows the Pareto fronts for the matrix multiplication application for two workloads, $12{,}352 \times 10{,}112$ and $15{,}552 \times 10{,}112$. Each Pareto front contains four linear segments. Each segment is connected by two endpoints. All the points lying on a segment are the optimal solutions in the objective space. The solution (shown as a circle) with the minimal execution time is the load-balancing solution.

For the workload $12{,}352 \times 10{,}112$, 17% dynamic energy savings is gained while allowing 5% performance degradation. Similarly, for the workload $15{,}552 \times 10{,}112$, 13% energy savings is achieved while tolerating 5% performance degradation.



(A) (B)

FIGURE 5 The execution time and dynamic energy profiles of the five heterogeneous processors (Figure 1) employed in the gene sequencing application



(A) (B)

(C) (D)

FIGURE 6 Pareto fronts for the matrix multiplication application using five heterogeneous processors for two workloads. (A) and (B) contain the Pareto fronts for execution time and dynamic energy. Each Pareto front contains four linear segments. (C) and (D) contain the Pareto fronts for execution time and total energy. The non-Pareto-optimal solutions due to high static energy consumption are highlighted in red. The solution (represented as a circle) with the minimal execution time is the load-balancing solution.

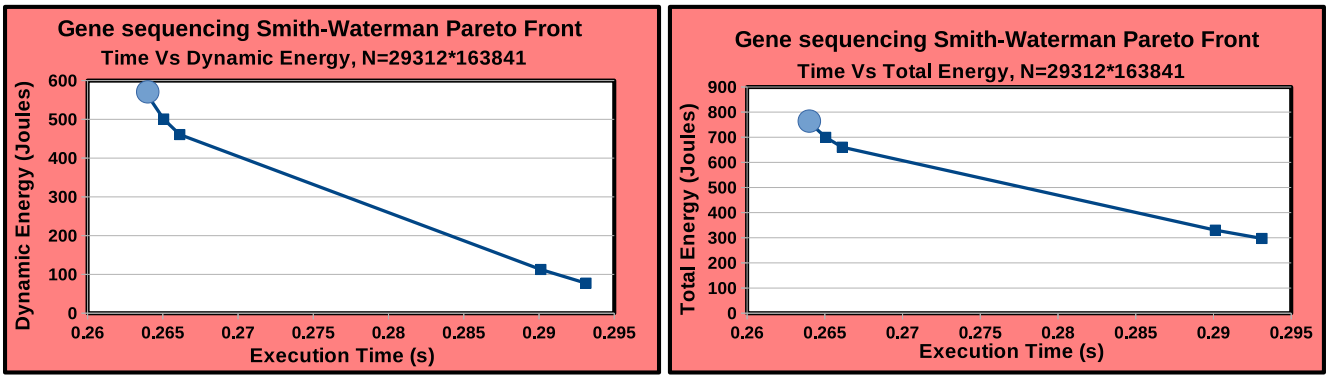


FIGURE 7 Pareto fronts for the gene sequencing application using the five heterogeneous processors for the workload 29,312 × 163,841. (A) contains the Pareto front for execution time and dynamic energy. It has four linear segments. (B) displays the Pareto front for execution time and total energy. The solution (represented as a circle) with the minimal execution time is the load-balancing solution.

The first linear segment has a steep slope signifying a significant dynamic energy savings for a slight increase in execution time. The energy savings are 93 and 106 J for execution time increases of 0.03 and 0.05 s for the two workloads. The energy-performance tradeoff (i.e., the gain in energy savings for a corresponding increase in execution time) decreases with each next linear segment.

Figure 7 shows the Pareto front for the gene sequencing application solving a problem size 29,312 × 163,841. The dynamic energy savings accepting a 1% performance hit is 23%.

6.4 | Performance-total energy Pareto fronts

Figure 6 presents the Pareto fronts for execution time and total energy for the matrix multiplication application for two workloads, 12,352 × 10,112 and 15,552 × 10,112. The solution (represented as a circle) with the minimal execution time is the load-balancing solution. The dotted red lines show the non-Pareto-optimal solutions removed by LBOPA-TE due to high static energy consumption.

Figure 7 shows the Pareto fronts for execution time and total energy for the gene sequencing application for the workload 29,312 × 163,841. The total energy savings accepting a 1% performance hit is 16%.

6.5 | Discussion

Following are our salient observations:

- The set of Pareto-optimal solutions (workload distributions) for execution time and dynamic energy is optimal for total energy only for the first two linear segments starting from the performance-optimal endpoint for the matrix multiplication application. The third and fourth linear segments with the positive slopes contain non-Pareto-optimal solutions due to high static energy consumptions;
- The shapes of the two Pareto fronts for execution time and dynamic energy and execution time and total energy are similar, suggesting that the qualitative conclusions apply for all workloads;
- For the gene-sequencing application, the set of Pareto-optimal solutions (workload distributions) for execution time and dynamic energy is also optimal for total energy for the workload employed in our experiments;
- Based on an input user-specified energy-performance tradeoff, one can selectively focus on a specific segment in the Pareto fronts to return the Pareto-optimal solutions (workload distributions). A steep slope in the line segment with the load-balanced solution as the performance-optimal endpoint will provide significant energy savings while tolerating little performance degradation. It signifies that introducing a small load imbalance can provide good energy savings.
- The execution times of our proposed algorithms range from milliseconds to 1 s to find Pareto-optimal solutions for the workload sizes used in the experiments. These execution times are insignificant compared to the execution times of the applications where our proposed algorithms are employed to find the workload distribution.

7 | CONCLUSION

Performance and energy are the two most important objectives for optimization on heterogeneous HPC platforms. Khaleghzadeh et al.[12] studied bi-objective optimization of data-parallel applications for performance and energy on heterogeneous processors. They proposed an algorithm for the case of *discrete* performance and energy functions with any arbitrary shape. They also briefly studied the continuous bi-objective optimization problem but *only for the simple case of two heterogeneous processors* with linear execution time and linear dynamic energy functions. They proposed an algorithm to find the Pareto front and showed that it is linear, containing an infinite number of solutions. While one solution is load balanced, the rest are load imbalanced.

In Reference 13, we studied a more general continuous bi-objective optimization problem for a generic case of k heterogeneous processors. The problem is motivated by the bi-objective optimization for the performance and dynamic energy of data-parallel applications on heterogeneous HPC platforms. We first formulated the problem, which for a given positive real number n aims to find a vector $X = \{x_0, \cdots, x_{k-1}\} \in \mathbb{R}^k_{\geq 0}$ such that $\sum_{i=0}^{k-1} x_i = n$, minimizing the max of k-dimensional vector of functions of objective type one and the sum of k-dimensional vector of functions of objective type two. We then proposed an exact algorithm of polynomial complexity solving the problem where all the functions of objective type one are continuous and strictly increasing, and all the functions of objective type two are linear increasing.

In this work, we applied the problem and the algorithm proposed in Reference 13 to solve two related optimization problems of parallel applications on heterogeneous hybrid platforms, one for performance and dynamic energy and the other for performance and total energy. First, we formulated and solved the bi-objective optimization problem for performance and dynamic energy. The problem and the solution are a direct application of the problem and algorithm proposed in Reference 13.

We then formulated the bi-objective optimization problem of parallel applications on heterogeneous hybrid platforms for performance and total energy. We proved a theorem that states that a solution vector X is Pareto-optimal for execution time and total energy if and only if it is Pareto-optimal for execution time and dynamic energy and there is no solution vector X_1 such that X_1 is Pareto-optimal for execution time and dynamic energy and $E_T(X_1) < E_T(X)$. Finally, we proposed an algorithm of polynomial complexity to solve the problem and whose correctness follows from the theorem.

Using the algorithms (proposed in Reference 13 and this work), we solved the two bi-objective optimization problems for two applications, matrix multiplication and gene sequencing, employing five heterogeneous processors, two Intel multicore CPUs, an Nvidia K40c GPU, an Nvidia P100 PCIe GPU, and an Intel Xeon Phi. For the workloads and the platform employed in our experiments, the algorithms provide continuous piecewise linear Pareto fronts for performance and dynamic energy and performance and total energy where the performance-optimal point is the load balanced configuration of the application.

Finally, 17% dynamic energy savings was achieved while tolerating a performance degradation of 5% (a saving of 106 J for an execution time increase of 0.05 s) for the matrix multiplication application. The dynamic energy and total energy savings for the gene sequencing application accepting a 1% performance hit were 23% and 16%.

In our future work, we will study bi-objective performance-energy optimization of applications with both continuous performance and energy profiles on heterogeneous hybrid HPC platforms.

ACKNOWLEDGMENTS

This publication has emanated from research conducted with the financial support of Science Foundation Ireland (SFI) under Grant Number 14/IA/2474. Open access funding provided by IReL.

CONFLICT OF INTEREST

The authors declare no potential conflict of interest.

DATA AVAILABILITY STATEMENT

Data are available on request from the authors.

ORCID

Hamidreza Khaleghzadeh https://orcid.org/0000-0003-4070-7468
Ravi Reddy Manumachu https://orcid.org/0000-0001-9181-3290

REFERENCES

1. Fard HM, Prodan R, Barrionuevo JJD, Fahringer T. A multi-objective approach for workflow scheduling in heterogeneous environments. Paper presented at: Proceedings of the 2012 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (Ccgrid 2012) CCGRID'12; IEEE Computer Society; 2012; Ottawa, ON: 300–309.
2. Kessaci Y, Melab N, Talbi EG. A pareto-based metaheuristic for scheduling HPC applications on a geographically distributed cloud federation. *Cluster Comput*. 2013;16(3):451-468.

3. Durillo JJ, Nae V, Prodan R. Multi-objective energy-efficient workflow scheduling using list-based heuristics. *Future Gener Comput Syst*. 2014;36:221-236.
4. Rossi FD, Xavier MG, De Rose CA, Calheiros RN, Buyya R. E-eco: performance-aware energy-efficient cloud data center orchestration. *J Netw Comput Appl*. 2017;78:83-96.
5. Kołodziej J, Khan SU, Wang L, Zomaya AY. Energy efficient genetic-based schedulers in computational grids. *Concurr Pract Exp*. 2015;27(4):809-829.
6. Yu L, Zhou Z, Wallace S, Papka ME, Lan Z. Quantitative modeling of power performance tradeoffs on extreme scale systems. *J Parallel Distrib Comput*. 2015;84:1-14.
7. Gholkar N, Mueller F, Rountree B. Power tuning HPC jobs on power-constrained systems. Paper presented at: Proceedings of the 2016 International Conference on Parallel Architectures and Compilation ACM; 2016; Haifa, Israel: 179–191.
8. Rountree B, Lowenthal DK, Funk S, Freeh VW, de Supinski BR, Schulz M. Bounding energy consumption in large-scale MPI programs. Paper presented at: SC '07: Proceedings of the 2007 ACM/IEEE Conference on Supercomputing; 2007; Reno, NV: 1–9.
9. Chakrabarti A, Parthasarathy S, Stewart C. A pareto framework for data analytics on heterogeneous systems: implications for green energy usage and performance. Paper presented at: Parallel Processing (ICPP), 2017 46th International Conference on IEEE; 2017; Bristol, UK: 533–542.
10. Lastovetsky A, Reddy R. New model-based methods and algorithms for performance and energy optimization of data parallel applications on homogeneous multicore clusters. *IEEE Trans Parallel Distrib Syst*. 2017;28(4):1119-1133.
11. Manumachu RR, Lastovetsky A. Bi-objective optimization of data-parallel applications on homogeneous multicore clusters for performance and energy. *IEEE Trans Comput*. 2018;67(2):160-177.
12. Khaleghzadeh H, Fahad M, Shahid A, Manumachu RR, Lastovetsky A. Bi-objective optimization of data-parallel applications on heterogeneous HPC platforms for performance and energy through workload distribution. *IEEE Trans Parallel Distrib Syst*. 2021;32(3):543-560.
13. Khaleghzadeh H, Manumachu RR, Lastovetsky A. A novel algorithm for bi-objective performance-energy optimization of applications with continuous performance and linear energy profiles on heterogeneous HPC platforms. Paper presented at: Euro-Par 2021 Workshops, Lecture Notes in Computer Science. Springer; 2022; Gottingen, Germany.
14. Fahad M, Shahid A, Manumachu RR, Lastovetsky A. A comparative study of methods for measurement of energy of computing. *Energies*. 2019;12(11):2204.
15. Miettinen K. *Nonlinear Multiobjective Optimization*. Kluwer; 1999.
16. Talbi EG. *Metaheuristics: From Design to implementation*. Vol 74. John Wiley & Sons; 2009.
17. Ge R, Feng X, Feng WC, Cameron KW. CPU MISER: a performance-directed, run-time system for power-aware clusters. Paper presented at: International Conference on Parallel Processing. IEEE Computer Society; 2007; Xi'an, China.
18. Huang S, Feng W. Energy-efficient cluster computing via accurate workload characterization. Paper presented at: 9th IEEE/ACM International Symposium on Cluster Computing and the Grid, IEEE Computer Society; 2009; Shanghai, China.
19. Mezmaz M, Melab N, Kessaci Y, et al. A parallel bi-objective hybrid metaheuristic for energy-aware scheduling for cloud computing systems. *J Parallel Distrib Comput*. 2011;71(11):1497-1508.
20. Beloglazov A, Abawajy J, Buyya R. Energy-aware resource allocation heuristics for efficient management of data centers for Cloud computing. *Future Gener Comput Syst*. 2012;28(5):755-768. Special Section: Energy efficiency in large-scale distributed systems.
21. Das A, Kumar A, Veeravalli B, Bolchini C, Miele A. Combined DVFS and mapping exploration for lifetime and soft-error susceptibility improvement in MPSoCs. Paper presented at: 2014 Design, Automation Test in Europe Conference Exhibition (DATE); 2014; Dresden, Germany: 1–6.
22. Sundriyal V, Sosonkina M. Joint frequency scaling of processor and DRAM. *J Supercomput*. 2016;72(4):1549-1569.
23. Abdi A, Girault A, Zarandi HR. ERPOT: a quad-criteria scheduling heuristic to optimize execution time, reliability, power consumption and temperature in multicores. *IEEE Trans Parallel Distrib Syst*. 2019;30:2193-2210.
24. Lang J, Rünger G. An execution time and energy model for an energy-aware execution of a conjugate gradient method with CPU/GPU collaboration. *J Parallel Distrib Comput*. 2014;74(9):2884-2897.
25. Reddy Manumachu R, Lastovetsky AL. Design of self-adaptable data parallel applications on multicore clusters automatically optimized for performance and energy through load distribution. *Concurr Comput Practice Exp*. 2019;31(4):e4958.
26. Tarplee KM, Friese R, Maciejewski AA, Siegel HJ, Chong EK. Energy and makespan tradeoffs in heterogeneous computing systems using efficient linear programming techniques. *IEEE Trans Parallel Distrib Syst*. 2016;27(6):1633-1646.
27. Aba MA, Zaourar L, Munier A. Approximation algorithm for scheduling a chain of tasks on heterogeneous systems. Paper presented at: European Conference on Parallel Processing. Springer; 2017; Santiago de Compostela, Spain: 353–365.
28. Khokhriakov S, Manumachu RR, Lastovetsky A. Multicore processor computing is not energy proportional: An opportunity for bi-objective optimization for energy and performance. *Appl Energy*. 2020;268:114957.
29. Lastovetsky A, Reddy R. Data partitioning with a realistic performance model of networks of heterogeneous computers. Paper presented at: 18th International Parallel and Distributed Processing Symposium, 2004; 2004; Santa Fe, NM: 104.
30. Lastovetsky A, Reddy R. Data partitioning with a functional performance model of heterogeneous processors. *Int J High Perform Comput Appl*. 2007;21:76-90.
31. Fahad M, Shahid A, Manumachu RR, Lastovetsky A. Accurate energy modelling of hybrid parallel applications on modern heterogeneous computing platforms using system-level measurements. *IEEE Access*. 2020;8:93793-93829.
32. Fahad M, Manumachu RR. *HCLWattsUp: Energy API Using System-Level Physical Power Measurements Provided by Power Meters*. Heterogeneous Computing Laboratory, University College Dublin; 2021. https://csgitlab.ucd.ie/manumachu/hclwattsup
33. Zhong Z, Rychkov V, Lastovetsky A. Data partitioning on multicore and multi-GPU platforms using functional performance models. *IEEE Trans Comput*. 2015;64(9):2506-2518.
34. Khaleghzadeh H, Zhong Z, Reddy R, Lastovetsky A. Out-of-core implementation for accelerator kernels on heterogeneous clouds. *J Supercomput*. 2018;74(2):551-568.
35. Smith TF, Waterman MS. Identification of common molecular subsequences. *J Mol Biol*. 1981;147(1):195-197.
36. Gotoh O. An improved algorithm for matching biological sequences. *J Mol Biol*. 1982;162(3):705-708.
37. Rognes T. Faster Smith-Waterman database searches with inter-sequence SIMD parallelisation. *BMC Bioinform*. 2011;12(1):1.

38. Liu Y, Wirawan A, Schmidt B. CUDASW++ 3.0: accelerating Smith-Waterman protein database search by coupling CPU and GPU SIMD instructions. *BMC Bioinform*. 2013;14(1):1.
39. Liu Y, Schmidt B. SWAPHI: Smith-Waterman protein database search on Xeon Phi coprocessors. Paper presented at: 2014 IEEE 25th International Conference on Application-Specific Systems, Architectures and Processors. IEEE; 2014; Zurich, Switzerland: 184–185.

How to cite this article: Khaleghzadeh H, Reddy Manumachu R, Lastovetsky A. Efficient exact algorithms for continuous bi-objective performance-energy optimization of applications with linear energy and monotonically increasing performance profiles on heterogeneous high performance computing platforms. *Concurrency Computat Pract Exper*. 2022;e7285. doi: 10.1002/cpe.7285