- Formulation of *BOPPE* and an efficient and exact global optimization algorithm, *ALEPH*, for solving *BOPPE*. These are based on a new decision variable, the problem size. Unlike existing solvers, *ALEPH* specifically targets inter-node optimization of data-parallel applications for performance and energy. Also unlike existing solvers, which assume linear relationship between performance and problem size and energy and problem size, *ALEPH* takes as inputs, functions of performance and dynamic energy consumption against problem size. We show that these functions realistically represent the performance and energy profiles of data-parallel applications on modern multicore CPUs.
- We show that *ALEPH* gives a diverse set of globally Pareto-optimal solutions whereas existing solvers give only one solution when the problem size and number of processors are fixed.
- We demonstrate how the Pareto-optimal front (determined by *ALEPH*) and paths of optimization algorithms for performance and energy (*POPTA* and *EOPTA*) can be used together to analyze the interplay between performance and energy.

The rest of the paper is organized as follows. Section 2 presents the challenges posed to solving *BOPPE* by the inherent complexities in modern multicore CPUs. Section 3 presents related work on bi-objective optimization problems for parallel platforms. Section 4 contains theory and notation of multi-objective optimization and the concept of optimality. We then formulate the bi-objective optimization of data-parallel applications on homogeneous clusters of multicore CPU nodes for performance and energy and analyse classical methods to solve the problem. We discuss why the state-of-the-art solvers are inadequate for direct solution of the bi-objective optimization problem. Section 5 presents an efficient exact algorithm solving *BOPPE*. Section 6 contains experimental analysis of the algorithm and analysis of interplay between performance and energy using *ALEPH* and its building blocks, *POPTA* and *EOPTA*. Section 7 concludes the paper.

2 Performance and Energy of Data-parallel Applications on Homogeneous Clusters of Multicore CPUs

In this section, we present the dramatic changes observed in performance and energy profiles of real-life scientific data-parallel applications executing on parallel platforms with multicore CPUs compared to parallel platforms composed of uniprocessors. Using this presentation, we specifically highlight the challenges posed to solving *BOPPE* by the new complexities in modern multicore CPUs. At the same time, we demonstrate why problem size has now become an important decision variable that can not be ignored.

In parallel platforms with uniprocessors, the performance and energy profiles of real-life scientific data-parallel applications were smooth and exhibited certain properties, which essentially rendered the bi-objective optimization problem of performance and energy (*BOPPE*) mono-objective. This meant that users had to use hardware-level intra-node parameters such as DVFS to tackle *BOPPE* where they reported noteworthy improvements in energy consumption while causing minimal degradation of performance.

TABLE 1: Specification of the Intel Haswell workstation used to build the uniprocessor speed and energy models.

Technical Specifications	Intel Haswell i5-4590
Processor	Intel(R) Core(TM) i5-4590 3.3 GHz
Microarchitecture	Haswell
Memory	8 GB
Socket(s)	1
Core(s) per socket	4
L1d cache	32 KB
L11 cache	32 KB
L2 cache	256 KB
L3 cache	6144 KB
TDP	84 W
Base Power	22.3 W
Max Turbo Frequency	3.7 GHz

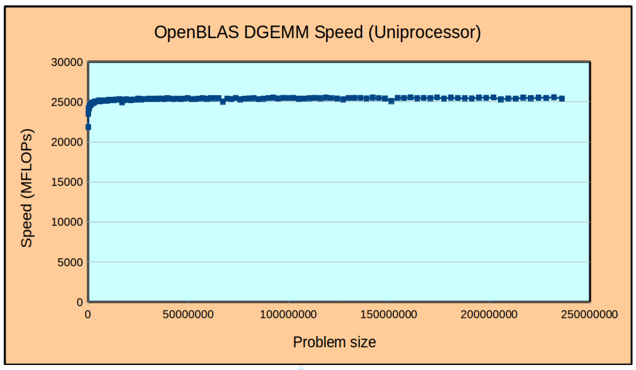
TABLE 2: Specification of the Intel Haswell server used to build the FPM and energy model for multithreaded OpenBLAS DGEMM and FFTW applications.

Technical Specifications	Intel Haswell Server
Processor	Intel E5-2670 v3 @ 2.30GHz
OS	CentOS 7
Microarchitecture	Haswell
Memory	64 GB
Socket(s)	2
Core(s) per socket	12
NUMA node(s)	2
L1d cache	32 KB
L11 cache	32 KB
L2 cache	256 KB
L3 cache	30720 KB
TDP	240 W
Base Power	58 W

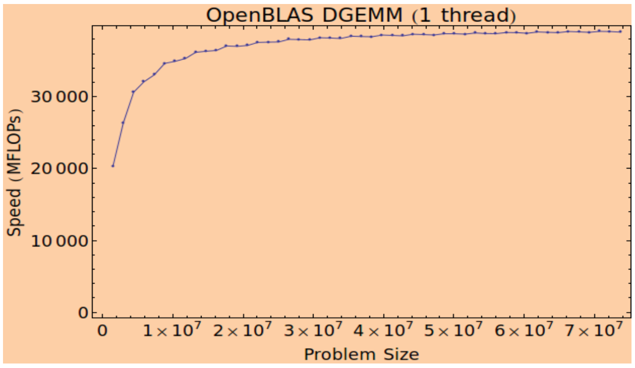
However, due to new formidable challenges imposed by the inherent complexities in modern multicore CPUs, *BOPPE* has become very difficult to solve. This is because the cores in a modern multicore CPU are very tightly integrated and arranged in a highly hierarchical fashion. Due to this tight integration, they contend for various shared on-chip resources such as Last Level Cache (LLC) and interconnect (For example: Intel's Quick Path Interconnect, AMD's Hyper Transport), thereby causing severe resource contention and non-uniform memory access (NUMA). These inherent complexities have posed the new challenges to solving *BOPPE*.

To elucidate these challenges, we compare the typical shapes of real-life scientific data-parallel applications on platforms consisting of uniprocessors and modern multicore CPUs. For this purpose, we select two widely known and highly optimized scientific routines, OpenBLAS DGEMM [12] and FFTW [13].

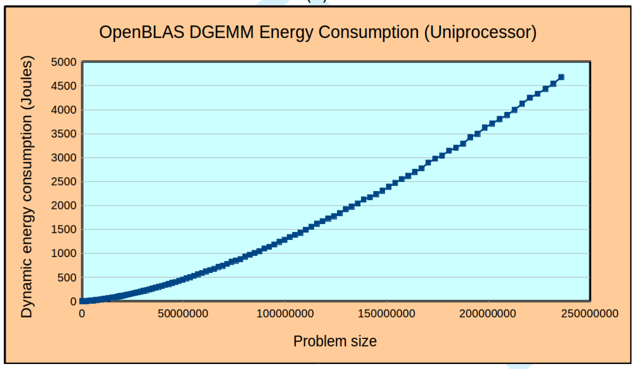
Consider the shapes of the speed and dynamic energy consumption functions of the OpenBLAS DGEMM application built experimentally by executing it on a single core of an Intel Haswell workstation (specification shown in Table 1). The application multiples two square matrices of size $n \times n$ (problem size is equal to n^2). In these experiments, the *numactl* tool is used to bind the application to one core. The dynamic energy consumptions are obtained using Watts Up Pro power meter. We must mention that there are two types



(a)

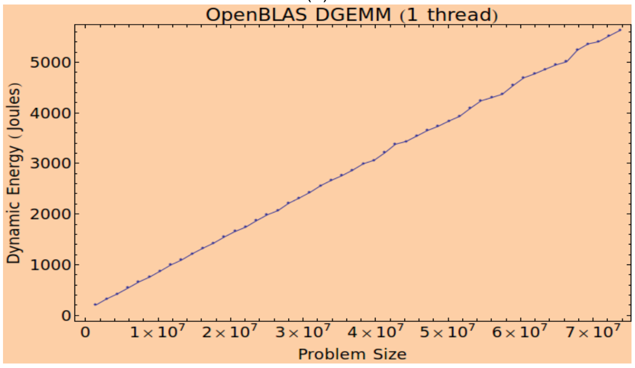


(a)



(b)

Fig. 1: a). Speed function of OpenBLAS DGEMM application executed on a single core on the Intel Haswell workstation. b). Dynamic energy consumption of OpenBLAS DGEMM application executed on a single core on the Intel Haswell workstation.



(b)

Fig. 2: a). Speed function of OpenBLAS DGEMM application executed on a single core on the Intel Haswell server. b). Dynamic energy consumption of OpenBLAS DGEMM application executed on a single core on the Intel Haswell server.

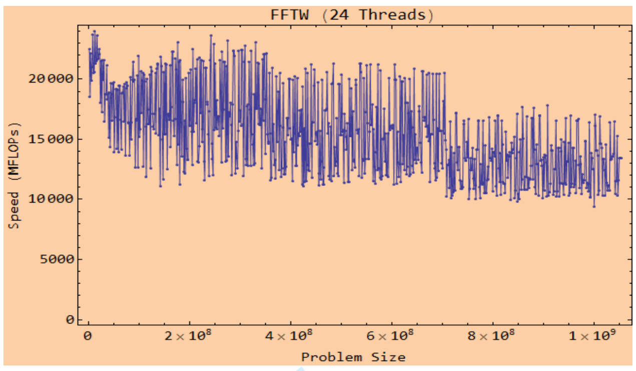
of energy consumptions, dynamic energy and static energy. We define the static energy consumption as the energy consumption of the platform without the application execution. The static energy consumption of the platform during the application execution is calculated by multiplying the idle power (static power) of the platform by the execution time of the application. Dynamic energy consumption is calculated by subtracting this static energy consumption from the total energy consumption of the platform during the application execution measured using the Watts Up Pro. In this work, we consider only the dynamic energy consumption due to several reasons, which are explained in Section 2 of the supplemental. However, we would like to mention that static power consumption can be easily incorporated in our problem formulations and algorithms.

Fig. 1a and 1b respectively show the shapes, whose properties can be summarized below:

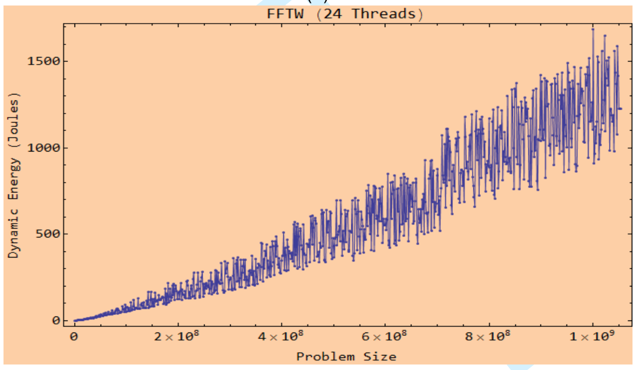
- The functions are smooth.
- The speed function satisfies the following properties:
  - Monotonically increasing.
  - Concave.
  - Any straight line coming through the origin of the coordinate system intersects the graph of the function in no more than one point.
- The dynamic energy consumption is a monotonically increasing convex function of problem size.

Lastovetsky et al. [14], [15], [11] prove that for such shapes, the solutions determined by the traditional and the state-of-the-art load-balancing algorithms [16], [17], [18], [19], [20], simultaneously minimize the execution time and dynamic energy consumption of computations in the parallel execution of the application. Fig. 2a and 2b respectively show the shapes of the speed and dynamic energy consumption functions of the same application built experimentally by executing it on a single core of an Intel Haswell server (specification shown in Table 2). It can be seen that while the shape of the speed function is the same as before, the shape of the dynamic energy consumption is linear. This implies that all workload distributions will result in same dynamic energy consumption and therefore parallelization has no effect on the dynamic energy consumption of computations in the parallel execution of the application.

Therefore, on platforms composed of uniprocessors, one can see that solutions determined by load-balancing algorithms that minimize the execution time either minimized the energy consumption or did not have any effect on it. Owing to this, methods solving BOPPE used energy-efficiency techniques such as DVFS, where the clock frequency of the processor is dynamically adjusted, to deter-

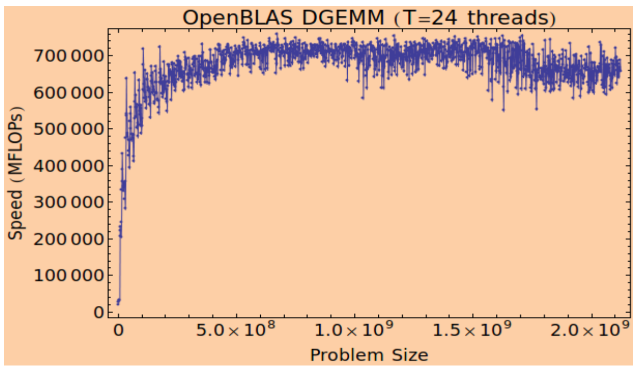


(a)

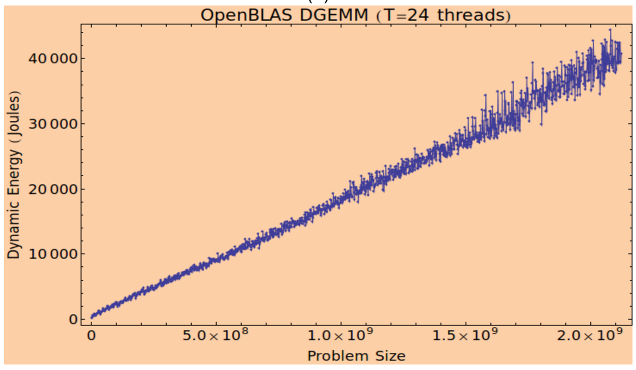


(b)

Fig. 3: a). Speed function of FFTW executing 24 threads on the Intel Haswell server. b). Function of dynamic energy consumption against problem size for FFTW executing 24 threads on the Intel Haswell server.



(a)



(b)

Fig. 4: a). Speed function of OpenBLAS DGEMM executing 24 threads on the Intel Haswell server. b). Function of dynamic energy consumption against problem size for OpenBLAS DGEMM executing 24 threads on the Intel Haswell server.

mine any trade-offs between execution time and energy. The main objective of these methods was to minimize the energy consumption but at the same time causing minimal performance degradation. We do not delve further to present the essence of these methods since we do not consider DVFS in our work.

Now, on modern homogeneous clusters composed of multicore CPUs, due to the newly introduced complexities such as resource contention and NUMA, the performance and energy profiles of real-life scientific applications executing on these platforms are not smooth and may deviate significantly from the shapes observed before. This is illustrated in Fig. 3 and 4, which respectively show the speed function and dynamic energy consumption graphs for multi-threaded OpenBLAS DGEMM and FFTW applications executed with 24 threads on the Intel Haswell server. The FFTW application performs a 2D FFT of size $n \times n$ (the problem size being n^2).

To make sure the experimental results are reliable, we follow a detailed methodology, which is described in Section 3 in the supplemental. It contains the following main steps: 1). We make sure the server is fully reserved and dedicated to our experiments and is exhibiting clean and normal behaviour by monitoring its load continuously for a week. 2). For each data point in the speed and energy functions of an application, the sample mean is used, which is calculated by executing the application repeatedly until the sample mean lies in the 95% confidence interval and a precision of 0.025 (2.5%) has been 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 by plotting the distributions of observations.

Therefore, the variation observed is not noise but is an inherent trait of applications executing on multicore servers with resource contention and NUMA. Other interesting properties about the variations are discussed in [11] and summarized below:

- They can be quite large. This is evident from the speed and energy functions shown in Fig. 3a and 3b respectively. From the speed function plot, one can observe performance drops of around 70% for many problem sizes.
- The variations presented in the paper cannot be explained by the constant and stochastic fluctuations due to OS activity or a workload executing in a node in common networks of computers. In such networks, a node is persistently performing minor routine computations and communications by being an integral part of the network. Examples of such routine applications include e-mail clients, browsers,

text editors, audio applications, etc. As a result, the node will experience constant and stochastic fluctuations in the workload. This changing transient load will cause a fluctuation in the speed of the node in the sense that the speed will vary for different runs of the same workload. One way to represent these inherent fluctuations in the speed is to use a speed band rather than a speed function. The width of the band characterizes the level of fluctuation in the speed due to changes in load over time [16], [17], [18]. For a node with uniprocessors, the width of the band has been shown to decrease as the problem size increases. For a node with a very high level of network integration, typical widths of the speed bands were observed to be around 40% for small problem sizes and narrowing down to 3% for large problem sizes. Therefore, as the problem size increases, the width of the speed band is observed to decrease. Therefore, for long running applications, one would observe the width to become quite narrow (3%). However, this is not the case for variations in the presented graphs. The dynamic energy consumption in Fig. 4b and 3b (for the number of threads equal to 24) show the widths of the variations increasing as problem size increases. These widths reach a maximum of 70% and 125% respectively for large problem sizes. The speed functions in Fig. 4a and 3a (for the number of threads equal to 24) demonstrate that the widths are bounded with the averages around 17% and 60% respectively. This suggests therefore that the variation is largely due to the newly introduced complexities and not due to the fluctuations arising from changing transient load.

Although we use two standard scientific kernels to illustrate the drastic variations in performance and energy profiles, these variations have been the central research focus in [14], [15] where the authors study them for a real-life scientific application, Multidimensional Positive Definite Advection Transport Algorithm (MPDATA). MPDATA is a core component of the EULAG (Eulerian/semi-Lagrangian fluid solver) geophysical model [21], which is an established computational model developed for simulating thermo-fluid flows across a wide range of scales and physical scenarios.

Therefore, these variations are not singular and will become natural because chip manufacturers are increasingly favouring and thereby rapidly progressing towards tighter integration of processor cores, memory, and interconnect in their products.

It is these variations that have now made the *BOPPE* difficult to solve. To demonstrate why this is the case, we zoom into the energy function of the OpenBLAS DGEMM application to analyze its properties. The speed functions also exhibit these properties. Figure 5 shows the energy function between two arbitrarily chosen points A and B. Assume, for the sake of simplicity, that we are allowed to use only this partial energy function in our algorithms.

- One can observe that the energy function is characterized by many local minima $(Q_1, Q_2, ...)$ and many local maxima $(P_1, P_2, ...)$. There is one global maximum P and one global minimum Q.
- The function (feasible region) is non-linear and non-convex.
- The function is not differentiable. The function lacks first-order and second-order derivatives. Hence, solvers that rely on the existence of derivatives for efficient search cannot be directly used. There are two approaches to deal with this problem. One is to estimate first-order derivatives by finite-differences and in the other, solvers must choose search directions purely based on functional values. One popular approach is to use Nelder-Mead algorithm [22], which is however designed to solve the classical unconstrained optimization problem without derivatives.

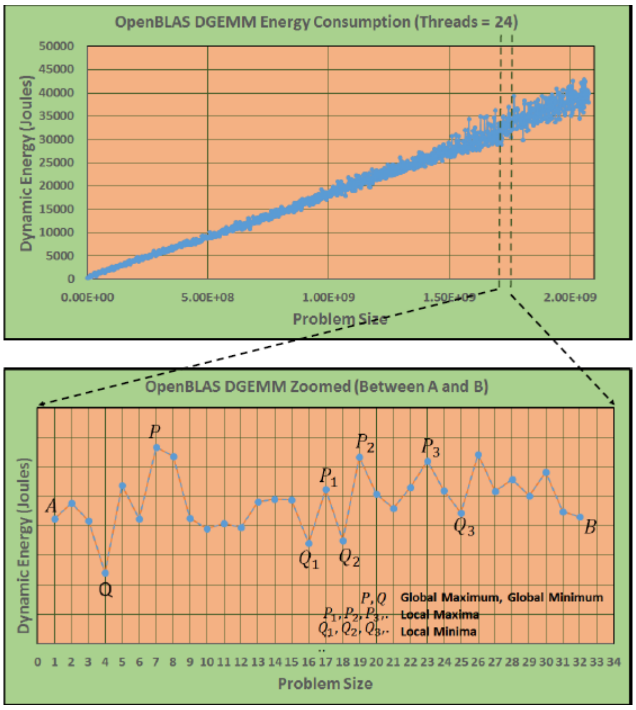


Fig. 5: Zoomed energy function of OpenBLAS DGEMM application between two arbitrarily chosen points A and B. The points are connected by dashed lines for clarity.

We describe in the supplemental why state-of-the-art optimization softwares are not directly amenable to tackle optimization problems where objective functions (in this case, execution time and energy) exhibit such complex properties.

To summarize, the new inherent complexities introduced in modern multicore CPUs have made the *BOPPE* difficult to solve. We also show that problem size has become an important decision variable.

3 Related Work

In this section, we present the state-of-the-art approaches studying *BOPPE* for parallel platforms. We focus exclusively on what kind of optimization is achieved by the approach, what parameters (which can be set at application-level) and decision variables are used in the problem formulation, and what dependence of performance and energy consumption on these parameters is considered.

Freeh et al. [1] is an intra-node optimization method that analyses the performance-energy trade-offs of serial and parallel applications on a cluster of DVFS-capable AMD nodes. They use three parameters in their study: β, which compares the application slowdown to the CPU slowdown, *memory pressure*, which is determined using hardware performance counters such as memory of operations retired and L2 cache misses, and *slack*, which predicts communication bottlenecks. They do not consider problem size as a parameter. In our work, we propose an inter-node optimization method and we consider only one parameter, the problem size.

Ishfaq et al. [2] formulate a bi-objective optimization problem of power-aware scheduling of tasks onto heterogeneous and homogeneous multicore processor architectures. The twin objectives in their problem formulation are minimization of energy consumption and the makespan of computationally intensive scientific problems. Their approach aims to achieve intra-node optimization by considering node-level parameters such as DVFS, computational cycles, and core architecture type in their optimization problem. They suggest a solution that combines a classical game-theoretic approach and Karush-Kuhn-Tucker (KKT) conditions to simultaneously optimize both the objectives. They do not consider problem size as a parameter.

Subramaniam et al. [3] use multi-variable regression to study the performance-energy trade-offs of the high-performance LINPACK (HPL) benchmark. Their model contains four parameters: N, the problem size, NB, the block size, P, Q, the rows and columns respectively of the process grid. They do not consider problem size as a parameter whereas, in our work, the problem size is the only parameter. If the problem size and number of processors are fixed, their approach gives a single solution whereas our algorithm gives a diverse set of globally Pareto-optimal solutions.

Song et al. [4] propose an iso-energy-efficiency model to study the performance-power trade-offs of parallel applications and systems. It is based on lines similar to performance iso-efficiency function. Their model contains twenty-eight parameters. One of the power-related parameters is the processor clock frequency. However, they do not consider problem size as a parameter.

Demmel et al. [5] present an intra-node and inter-node optimization approach that studies energy savings at the algorithmic level. They prove that a region of perfect strong scaling in energy exists for matrix multiplication (classical and Strassen) and the direct ($O(n^2)$) n-body problem. This means that for a for a given problem size n, the energy consumption remains constant as the number of processors p increases and the runtime decreases proportionally to p. In their analysis, they use performance and energy models containing six and twelve parameters respectively. The energy consumption is considered to be a summation of energy costs of computations, communications, and leakage (static power). They do not consider problem size as a parameter. In our work, we consider only one parameter, the problem size, and the dynamic energy consumption is represented by a function of problem size.

Choi et al. [6] present an energy roofline model based on the time-based roofline model [7]. Choi et al. [8] extend the roofline model by adding an extra parameter, *power caps*, to their execution time model. These two works [6], [8] present an intra-node optimization approach to study the performance-energy trade-offs. They do not consider problem size as a parameter.

Drozdowski et al. [9] propose a concept called an iso-energy map, which represent points of equal energy consumption in a multi-dimensional space of system and application parameters. They use iso-energy maps to study performance-energy trade-offs. They present iso-energy maps for three models of parallel computations, Amdahl's law, Gustafson's law, and divisible loads representing data-parallel computations in distributed systems. For the models for Amdahl's law and Gustafson's law, three intra-node parameters are used: f, the size of the parallel portion of the application, m, the number of processors in the system, and k, which represents the ratio of powers in active and idle states. Their iso-energy map analysis concludes that the laws give opposing indications from the energy point of view. For the model for divisible computations, they have eight parameters giving rise to twenty-eight iso energy maps (which are two-dimensional) and they present analyses for a subset. They do not consider problem size as a parameter. One of the key assumptions in their model is that the energy consumption is constant and independent of problem size. This they confirm to be true for the applications and the platform used in their experiments. In our work, the problem formulation and the algorithm are based on only one parameter, the problem size. The speed and the dynamic energy consumptions are represented by functions of problem size. We show using experiments on a modern multicore CPU that these functions have highly complex properties.

Balaprakash et al. [23] is an intra-node optimization approach that explores trade-offs among power, energy, and performance using various application-level tuning parameters such as number of threads and hardware parameters such as DVFS. They do not consider problem size as a parameter.

Marszakowski et al. [10] analyze the impact of memory hierarchies on time-energy trade-off in parallel computations, which are represented as divisible loads. They represent execution time and energy by two linear functions on problem size, one for in-core computations and the other for out-of-core computations. They use this formalization in their bi-objective optimization problem of minimizing time and energy in parallel processing of divisible loads. They have twelve parameters in their models and problem formulation. Due to large parameter space, they restrict their study of performance-energy trade-offs where they just have one parameter, the number of processors (all the other parameters have fixed values). In our work, we consider only one parameter, problem size. The other inputs, which are the speed and the dynamic energy consumptions are represented by functions of problem size. We show using experiments on a modern multicore CPU that these functions have highly complex properties and cannot be represented by piece-wise linear functions.