An Evolutionary Technique for Performance-Energy-Temperature Optimized Scheduling of Parallel Tasks on Multi-Core Processors

Hafiz Fahad Sheikh, Ishfaq Ahmad, *Fellow, IEEE*, and Dongrui Fan, *Member, IEEE*

Abstract—This paper proposes a multi-objective evolutionary algorithm (MOEA)-based task scheduling approach for determining Pareto optimal solutions with simultaneous optimization of performance (P), energy (E), and temperature (T). Our algorithm includes problem-specific solution encoding, determining the initial population of the solution space, and the genetic operators that collectively work on generating efficient solutions in fast turnaround time. Multiple schedules offer a diverse range of values for makespan, energy consumed, and peak temperature and thus present an efficient way of identifying trade-offs among the desired objectives, for a given application and machine pair. We also present a methodology for selecting one solution from the Pareto front given the user's preference. The proposed algorithm for scheduling tasks to cores achieves three-way optimization with fast turnaround time. The proposed algorithm is advantageous because it reduces both energy and temperature together rather than in isolation. We evaluate the proposed algorithm using implementation and simulation, and compare it with integer linear programming as well as with other scheduling algorithms that are energy- or thermal-aware. The time complexity of the proposed scheme is considerably better than the compared algorithms.

Index Terms—Energy-efficient computing, thermal-efficient computing, task allocation, evolutionary algorithms, task graphs, static scheduling

1 Introduction

Sustainability in computing has gained immense importance for managing the energy needs of the future cyber infrastructures. The building blocks of such infrastructures are often multi-core processors that continue to grow with enhanced complexity and ever increasing number of cores on the same chip. With an emphasis on energy and thermal issues, the raw computational speed of these processors can only be harnessed with effective scheduling and mapping tools. Higher power dissipation levels resulting into thermal problems and higher cooling costs is one of the major factors in limiting the scalability of these systems. High power consumption can also cause unacceptably high temperatures that in turn can lead to loss in performance, reliability and lifespan, and even total failures ([6], [7]). Most of the multi-core chips are equipped with basic power control mechanisms such as Dynamic Voltage-Frequency Scaling (DVFS), which can potentially be exploited in resource scheduling for assigning tasks to cores. A resource management scheme for allocating tasks to cores for optimizing P, E, and T has to consider a number of important factors:

1) *Achieving triple objectives (P, E, and T)*: Current task-to-machine mapping algorithms are primarily energy-aware [11], [22], or thermal-aware [8], [9], [10], but not both. The problem is often formulated as a dual objective optimization problem. In energy-aware algorithms, one factor (e.g., energy or performance) is given as a constraint while the second factor is to be optimized [20]. Likewise, in temperature-aware optimization, either temperature or performance is given as constraint and the other factor is to be optimized. Since energy and performance are correlated, and temperature and performance are correlated as well, therefore performance (P), energy (E) and temperature (T) (*PET quantities*) need to be optimized simultaneously.
2) *Harnessing DVFS*: Despite the challenges encountered in DVFS, such as the inevitable static power (largely due to leakage currents) and associated overheads [31], [34]; it remains one of the predominant options for controlling the chip power. DVFS is also aggressively researched and is widely incorporated in emerging architectures [41], allowing both the hardware and software controls-the later leaving a large room for software designers to harness.
3) *Addressing both E and T*: Both E and T must be controlled together and not in isolation. As confirmed by the experimental results shown in the paper, scheduling algorithms focusing on temperature alone can incur high values of energy [8]. Only a

- *H. F. Sheikh is with the Department of Computer Science and Engineering, University of Texas at Arlington, 500 UTA Blvd, Arlington 76019, TX. E-mail: hafizfahad.sheikh@mavs.uta.edu.*
- *I. Ahmad is with the Department of Computer Science and Engineering, University of Texas at Arlington, 500 UTA Blvd, Arlington 76019, TX. E-mail: iahmad@uta.edu.*
- *D. Fan is with the SKL Computer Architecture, Institute of Computing Technology, CAS, No. 6 Kexueyuan South Road Zhongguancun, Haidian District Beijing 100190, China. E-mail: fandr@ict.ac.cn.*

Manuscript received 22 June 2014; revised 22 Mar. 2015; accepted 25 Mar. 2015. Date of publication 8 Apr. 2015; date of current version 12 Feb. 2016. Recommended for acceptance by H. Shen.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TPDS.2015.2421352

few research efforts have reported thermal management techniques resulting in improved energy [11]. Even then these improvements are only a side effect of these schemes and not a direct result of the joint optimization. Moreover, T cannot be ignored for energy saving because there are bounds and limitations on the temperature which may vary from system to system.

4) *Incurring low complexity*: The complexity of the optimization process should remain at an acceptable level in order to allow scheduling of large workloads within a reasonable amount of time. The multi-objective prospect compounds the complexity of an already NP-hard task scheduling problem [17].

The contributions of this paper are as follows: First, this paper addresses and formulates the problem of simultaneous three-way performance, energy, and temperature optimized scheduling (PETOS) of task graphs on a given multi-core system with DVFS capability. For this problem, there can be various application scenarios. Typically, one or two of the *PET factors* may be given as a constraint while the third factor may need to be optimized. For example, a user may like to know the performance level for certain energy and temperature constraints (and vice versa), for a given application and machine pair. Second, the paper proposes a multi-objective evolutionary algorithm (MOEA)-based scheduling methodology for solving the PETOS problem. We design problem-specific genetic operators as well as a technique for obtaining an initial population for the MOEA-based approach. The result of these techniques is the proposed algorithm called E-FORCE (Evolutionary- Frequency ORchestration and Core allocation Equability) that aims to obtain multiple solution points (Pareto front) with trade-offs among the *PET quantities* while allocating tasks on multi-core systems. Here, each point in the Pareto fronts represents a schedule for task allocations and frequency selections. Multiple schedules result in diverse range of values for makespan, total energy consumed and peak temperature and thus present an efficient way of identifying trade-offs among the desired objectives. Third, we present a methodology to choose a solution out of the Pareto front. E-FORCE achieves *PET values* comparable to ECSIdle [20] (performance-aware energy optimization approach) and PostTM [8] (a thermal-aware scheduling approach), at the same time produces multiple schedules (trade-off points) instead of a single solution. The execution time of the proposed approach is comparable with the energy- and thermal-aware scheduling (heuristic) schemes and scales better with increasing number of tasks.

This paper significantly extends the preliminary results presented in [19] and is organized as follows: Section 2 provides the related work on the evolutionary task scheduling as well as on the energy- and thermal-aware scheduling. Section 3 formulates the PETOS problem. Section 4 provides the details of proposed solution to the problem and Section 5 highlights the experimental setup. Section 6 explains the results while Section 7 provides the concluding remarks.

2 Related Work

Previous work on dual-objective optimization for performance and temperature can be found in [8], [9], [10], [38], [39]. For example, pre-built look-up tables representing the thermal effects due to power activation at different locations of the chip are used in [8] to allocate tasks to different cores. The algorithm reported in [38] addresses the problem using a hardware-software co-design. An operating system level heuristic of scheduling hot-job before a cool-job to minimize the number of DTM (dynamic thermal management) events is presented in [9]. Similarly, a convex optimization problem for allocating frequencies to cores under different workloads to meet the thermal threshold and hotspot limits has been formulated in [10]. In these works, the joint optimization problem for improving performance, energy and thermal profile of multi-core system remains unaddressed.

Some research efforts report the improvement in either performance and energy or performance and temperature. For instance, [11] proposes an agent-based power distribution approach for DTM which achieves 44.2 and 44.4 percent improvement in performance and energy respectively when compared with other DTM schemes. However, these improvements are only a side effect of an efficient DTM scheme and not due to the joint optimization. A software-based optimization method that takes into account all three *PET quantities* is proposed in [40]. However, this profiling scheme provides insight into the software's structure specific to an application; it does not provide solution to the scheduling problem. Similarly, methods for identifying energy-performance Pareto fronts in dual-objective space [36] cannot be extended to solve the *PET optimization scheduling* problem because even their two-objective optimization has prohibitively high execution time.

A few schemes applying evolutionary/genetic approaches to solve scheduling and allocation problems for different domains have been proposed. The algorithm proposed in [2] addresses multidimensional QoS issues, the algorithm in [3] aims to maximize throughput, and the algorithm in [5] addresses the task allocation problem for performance. An energy-performance optimization scheme using voltage selection (but not task scheduling) is introduced in [4]. It optimizes energy consumption of a given task set by selecting voltages for tasks while satisfying their precedence and deadline constraints. However it does not address task allocation or re-allocation, nor does it consider the temperature. In contrast, E-FORCE is the first approach in solving the three-way performance, energy and temperature optimization at the scheduling level for parallel task graphs.

3 Problem Formulation

Parallel programs are commonly represented as DAGs [17]. Each node in the DAG represents a task where the weight associated with each node is the estimated execution time of the task (while executing at the highest frequency level). An edge in the DAG represents the dependency relationship between a pair of tasks and the corresponding weight is the estimated time required to complete the communication. The *critical path* is the path with the longest length in a DAG [17]. The nodes constituting a critical path are called *critical path nodes* (CPNs) and the cores on which these tasks are

TABLE 1
Scalars, Parameters, and Additional Variables

	Symbol	Description
Scalars and Parameters	N	Total number of tasks.
	M	Total number of cores.
	K	Total number of voltage levels.
	C_{eff}	Effective capacitance.
	R_{TH}	Thermal Resistance.
	λ_{th}	Threshold Temperature.
	γ	Constant factor which relates voltage level and clock cycle time.
	CCR	Communication to Computation ratio.
Additional Variables	et_i^o	Execution time of ith task at the highest available frequency level.
	v_i	Voltage level selected for the execution of ith task.
	$\boldsymbol{V}$	$\{v_i, \forall\, 1 \leq i \leq \text{N}\}$(Set of voltage levels selected for all the tasks)
	f_i	Frequency at which the ith task will be executed.
	$\boldsymbol{L}$	Set of all available voltage levels.
	et_i	Time required for the execution of ith task.
	$\boldsymbol{ET}$	$\{et_i, \forall\, 1 \leq i \leq \text{N}\}$(Set of execution times of all tasks)
	p_i	Power consumed while executing ith task.
	$\boldsymbol{P}$	$\{p_i, \forall\, 1 \leq i \leq \text{N}\}$(Set of power levels of all tasks)
	st_i	Start time of ith task.
	$\boldsymbol{ST}$	$\{st_i, \forall\, 1 \leq i \leq \text{N}\}$(Set of starting times of all tasks)
	ft_i	Finish time of ith task.
	$\boldsymbol{FT}$	$\{ft_i, \forall\, 1 \leq i \leq \text{N}\}$(Set of finish times of all tasks)
	$\beta_{(a,b)}$	Constant factor which relates the power of a core (a) to the temperature of its neighboring core (b).
	$\mathbf{B}$	$\mathbf{B} = [\beta_{(a,b)}]$ where $\beta_{(a,b)} = 0 \forall a = b$, and $1 \leq a,b \leq M$. (A symmetric matrix that contains all $\beta_{(a,b)}$ values)
	α_i^j	Temperature of the jth core consequential to the allocation of ith task excluding "neighbor-effect".
	T_i^j	Temperature of the jth core consequential to the allocation of ith task including "neighbor-effect".

scheduled are called the critical cores. Nodes having successors on a critical path are known as *In-bound nodes* (IBNs). All other tasks are called *out-bound nodes* (OBNs) [17]. We initially obtain the scheduling order of tasks from a performance-aware scheduler called Dynamic Critical Path scheduler (DCP) [18]. Table 1 lists the scalars and parameters as well as additional variables and data structures used in the problem setup. For a DAG with N tasks that is to be allocated to M cores, we define our task allocation and voltage selection decision variables as follows:

$$\forall 1 \leq i \leq N, 1 \leq j \leq M, 1 \leq k \leq K$$

$$x_{i,j} = \begin{cases} 1 & if\, ith\, task\, executes\, on\, jth\, core \\ 0 & otherwise, \end{cases} \tag{1}$$

$$y_{i,k} = \begin{cases} 1 & if\, ith\, task\, uses\, kth\, voltage\, level \\ 0 & otherwise, \end{cases} \tag{2}$$

subject to

$$\sum_{j=1}^{M} x_{i,j} = 1 \wedge \sum_{k=1}^{K} y_{i,k} = 1.$$

From (2), we can obtain the value of voltage level selected for each task as:

$$\forall 1 \leq i \leq N$$

$$v_i = \sum_{k=1}^{K} y_{i,k}\, l_k, \tag{3}$$

where l_k is the kth element of the set $\boldsymbol{L}$. The corresponding frequency level (f_i) can be obtained from the selected voltage level using $f_i = \gamma(v_i)^{\gamma_o}$. Here, γ and γ_o are technology dependent constants which govern the scaling relationship between voltage level and frequency of a core (typical values of γ_o can vary from 1-2). Assume that the execution time of each task at the maximum frequency level is known. We can then use the scaling relationship [35] for CPU-bounded tasks to calculate the execution time of each task at the selected frequency as:

$$\forall 1 \leq i \leq N$$

$$et_i = (f_o/f_i) et_i^o. \tag{4}$$

In (4), f_o is the highest available frequency level while et_i^o is the execution time of the ith task at f_o (i.e., the weight on a DAG node). The power consumption of a core while executing the ith task is given by:

$$p_i = p_{static} + C_{eff}(v_i)^{\gamma'}. \tag{5}$$

In the above equation, γ' is another technology dependent constant with typical value between 2-3 while p_{static} is the power dissipated by a core at idle due to the leakage current. Now, If PD_i represents the set of all predecessors of ith task then the start time of the ith task can be given as:

$$\forall\, pred_j \in PD_i$$

$$st_i = \max_{pred_j}[ft_{pred_j} + d_{(pred_j,i)}], \tag{6}$$

where $d_{(pred_j,i)}$ is the time required to complete the data transfer between $pred_j$ and the ith task. Based on the earliest possible start time of the ith task, we can determine the earliest completion time of the ith task as:

$$ft_i = st_i + et_i. \tag{7}$$

In order to calculate the temperature of the cores, we define a variable for every task pair to determine their overlap in time while executing on their corresponding cores,

$$\forall 1 \leq i, j \leq N \wedge i \neq j$$

$$z_{i,j} = \begin{cases} 1 & st_j \leq st_i \leq ft_j \\ 0 & otherwise. \end{cases} \tag{8}$$

The variable $z_{i,j}$ is used for implementing the neighbor-effect [13]. Neighbor-effect is the impact of the power

dissipation of the neighboring cores on the temperature of a particular core. First, we find the steady state temperature of all the cores without considering the neighbor-effect as:

$$\forall\, 1 \leq i \leq N, 1 \leq j \leq M \qquad \alpha_i^j = R_{th}\, p_i\, x_{i,j} + T_A, \tag{9a}$$

where T_A is the ambient temperature and R_{th} is the thermal resistance of the system [37]. Next, the neighbor-effect can be incorporated to update the steady state temperature of each core as:

$$T_i^j = \alpha_i^j + \sum_{r=1}^{N} z_{i,r}\big(\boldsymbol{x_i}\mathbf{B}\boldsymbol{x_r}^T\big)p_r, \tag{9b}$$

where $\boldsymbol{x_i}$ and $\boldsymbol{x_r}$ are vectors representing the task-core allocation decisions of ith and rth task. Here, $\boldsymbol{x_i}$ can be defined as $\boldsymbol{x_i} = \{x_{i,j}, \forall 1 \leq j \leq M\}$. For experimental evaluations, we obtained power and thermal values by profiling an actual multi-core system (see Section 5.1). Our algorithm allows any thermal and energy model and does not focus on modeling.

Using (1)-(9), the objective functions for performance, energy and temperature are as follows:

$$Minimize \quad \max_{1\leq i\leq N} ft_i, \tag{10}$$

$$Minimize \quad \sum_{i=1}^{N} p_i\, et_i, \tag{11}$$

$$Minimize \quad \max_{1\leq i\leq N} \max_{1\leq j\leq M} T_i^j. \tag{12}$$

Equations (10), (11) and (12) refer to the minimization of the completion time (makespan), minimization of the total energy consumption, and minimization of the maximum temperature, respectively. We term this problem as *PET optimized scheduling* (PETOS) problem which is formulated without any constraints on P, E, or T. However, after obtaining these fronts, a user may like to impose constraints during the solution selection phase (Section 4.6) to filter out only those solutions which satisfy the given requirements.

4 Proposed Solution

For the PETOS problem with conflicting objectives, the concept of optimality is transformed into that of Pareto optimality. Pareto optimality can be defined on the basis of the dominance relationship between two solutions to a multi-objective optimization problem [12]. Let us consider x_1 and x_2 to be the two solutions to an m-objective minimization problem, the domination of x_1 over x_2 ($x_1 \succ x_2$) is defined as:

$$\begin{aligned} &x_1 \succ x_2 \quad if \\ &\exists k | f_k(x_1) < f_k(x_2), \quad k \in \{1, 2, \ldots, m\} \\ &and\ \forall 1 \leq j \leq m \\ &f_j(x_1) \leq f_j(x_2), \end{aligned} \tag{13}$$

where $f_j(x)$ represents the value of the jth objective for solution x. Here, we assume $F(x) = \{f_1, f_2, \ldots, f_m\}$ represents the set of all objective functions and that each objective needs to be minimized. Then, x_1 is said to dominate x_2 if it achieves lower value at least along one objective while achieving equal or lesser values along all other objectives as compared to x_2. Solutions that are not dominated by other solution members are known as *Pareto optimal/non-dominated* solutions. A collection of such non-dominated solutions form a *Pareto front* (see [12] for more details). Fig. 4 (Section 4.3) illustrates such Pareto relationship. The figure shows that in the feasible region defined by deadline (D) and peak temperature constraint ($T_{\max}$), no single solution among points 1-7 is better than the other six solutions along both performance and temperature, rather each solution presents a different trade-off between performance and temperature. In order to achieve a better value of makespan we have to compromise temperature and vice versa.

The proposed E-FORCE algorithm aims to generate such Pareto front between all three *PET quantities* at the scheduling level. The core of the proposed method is based on the principles of a MOEA technique called Strength Pareto Evolutionary Algorithm [1]. The generic evolutionary techniques, such as SPEA-II [1], have been utilized in various scientific and engineering problems. The basic SPEA-II technique iteratively updates a set of solutions population until a stopping criterion is achieved. In every iteration/generation, an elite population which is a set of best known solutions found so far is combined with a newly generated population. The new population is obtained by forming mating pairs among the elite population members and then applying genetic operations. The genetic operations for creating the offsprings have to be designed appropriately for the given problem.

The E-FORCE algorithm is designed on the principles of SPEA-II. The design of the algorithm for solving the PETOS problem includes several steps including techniques for determining the initial population of the solution space as well as genetic operators, specific to this problem. E-FORCE also includes a solution selection scheme, which is based on user's preference, to select a solution from the Pareto front generated by the evolutionary process. The pseudocode in Fig. 1 describes the salient steps in E-FORCE algorithm: *systemmodel* includes the given set of cores, available set of frequencies, as well as models required to calculate power and temperature of the system under different settings. The input argument *params* is the set of algorithmic parameters including those used during the genetic operations. Parameter p represents the preference vector provided by the user for selecting a solution from the generated Pareto front. The details of each step in E-FORCE are presented next.

4.1 Solution Encoding

In E-FORCE, each population member for a DAG with N tasks is a string of size $2N$. The indices 1 to N correspond to the task allocation decisions and indices $N+1$ to $2N$ encode voltage/frequency selection decisions. Thus, each population member is a possible schedule. Fig. 2a presents a DAG for FFT and Fig. 2b shows the corresponding scheduling decisions including task-core mapping, frequency selection, and task ordering for each task. Fig. 2c shows the corresponding Gantt chart.

Algorithm 1 E-FORCE

```
1: procedure E-FORCE(DAG, systemmodel, params, p)
2:     initialpop, curr_pop, child_pop, parent_pop ← Φ
3:     schedule ← INITIALIZESCHEDULE(DAG, systemmodel)
4:     initialpop.ADDDCPBASEDSET (DAG, schedule, systemmodel)
5:     initialpop.ADDRANDOMFIXEDSET(DAG, systemmodel)
6:     initialpop.ADDRANDOMRANDOMSET(DAG, systemmodel)
7:     curr_pop.ADDMEMBERS(initialpop)
8:     while ( !done ) do
9:         curr_pop.PETvals ←
10:        GETPETVALS(curr_pop.members, systemmodel, params)
11:        curr_pop.fitnessvals ←
12:        GETFITNESS(curr_pop.members, curr_pop.PETvals)
13:        parent_pop ← SELECTTOPMEMBERS(params, curr_pop)
14:        if ( termination criterion met ) then
15:            done ← true
16:        else
17:            matingpool ← POPULATEMATINGPOOL(parent_pop)
18:            child_pop ← APPLYGENETICOPS(matingpool, params)
19:            curr_pop ← parent_pop
20:            curr_pop.ADDMEMBERS(child_pop)
21:        endif
22:    end while
23:    paretoFront ← parent_pop
24:    paretoFront.ranks ← GETRANKS(paretoFront.members, p)
25:    selectedSolution ← GETMINRANKSOLUTION(paretoFront)
26:    return selectedSolution
27: end procedure
```

Fig. 1. Algorithmic outline of E-FORCE.

4.2 Initial Population

The initial population for the PETOS problem consists of three subsets each with size ($\eta/3$), where η is a user defined parameter representing the required size of the Pareto front. Our methodology for the generation of these three subsets namely DCP-based, Random-Fixed, and Random-Random along with their significance is explained as follows:

The population members in the DCP-based subset are generated using the Dynamic Critical Path scheduler [18]. DCP is efficiently performance-aware and keeps track of the dynamic changes in the critical path of the DAG while scheduling tasks on a given system. All members of the DCP-based subset have the same task-core mappings but their frequency selection decisions are different. Since the size of each subset is $\eta/3$ and assuming $(\eta/3) > K$ (the total number of frequency levels), we assign kth frequency level to all the tasks in the kth member of the DCP-based subset. Therefore, the first solution in the DCP-based subset has task-core mappings from DCP while all tasks run at the first (lowest) frequency level. The second solution has the identical task-core mappings as of the first but each task is executed at the second lowest frequency level. Thus, there are K such solutions having the same task-core mappings from DCP and all tasks to be run at one of the K frequency levels. The remaining $(\eta/3 - K)$ solutions have their task-core mappings from DCP, but their frequency levels are selected randomly using a uniform distribution defined over the interval [1, K]. This subset facilitates keeping performance as a major objective.

For the solutions in Random-Fixed population, each task-core mapping is drawn from a uniform probability distribution defined over the interval [1, M] while satisfying the precedence constraints. The frequency of execution of each task is fixed at $k = 1$ (i.e., lowest frequency level). This population adds bias towards lowering peak temperatures.

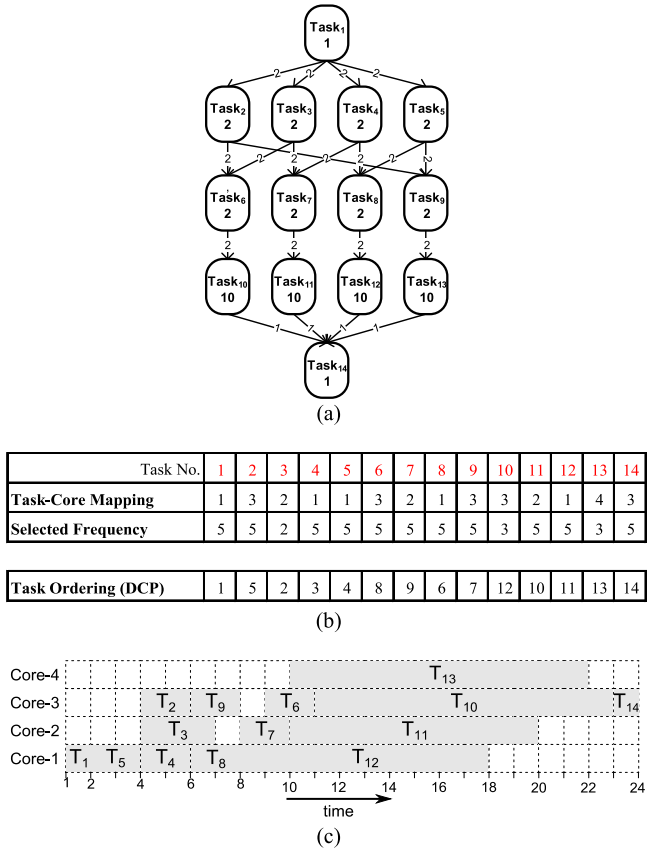

Fig. 2. (a) A 14-node FFT task graph [15]. (b) A possible schedule with DVFS settings. (c) The Gantt chart of FFT task graph.

For the solutions in Random-Random, both task allocation and frequency selection decisions are random, adding general randomness into the population to enable search space not covered by the other two subsets. Table 2 summarizes the construction of each of these subsets. Fig. 3 shows a sample initial population for a task graph with $N = 14$ and $\eta = 30$ while assuming a system with 16 cores and 5 frequency levels ($M = 16$ and $K = 5$).

TABLE 2
Initial Population Subsets

Subset	Task-Core mapping	Frequency Selection
DCP-based	As generated by the DCP scheduler.	• First K members: All tasks run at the same level selected from {1,2, . . ., K}. • Remaining $(\eta/3 - K)$ members: Randomly selected frequency level for each task (using uniform probability distribution defined over [1, K]).
Random-Fixed	Drawn from uniform probability distribution defined over [1, M].	• All tasks run at the lowest frequency level.
Random-Random	Drawn from uniform probability distribution defined over [1, M].	• Randomly selected frequency level for each task (using uniform probability distribution defined over [1, K]).

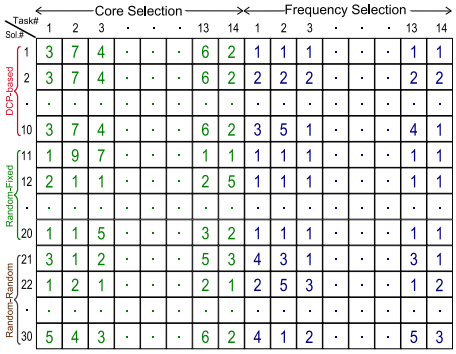

Fig. 3. Initial population used by E-FORCE with $N = 14$ and $\eta = 30$.

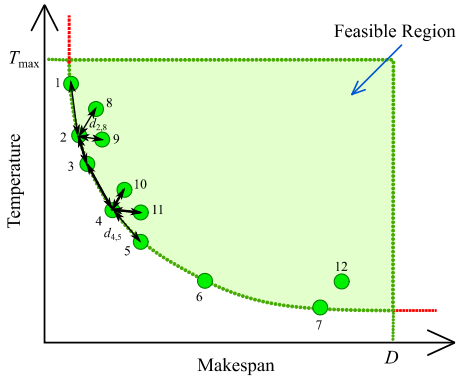

Fig. 4. An example of density value estimation for Fitness Assignment.

4.3 Fitness Assignment

The algorithm works with two populations: the current population (***G***) and the elite/best population (***G′***). E-FORCE starts by setting the initial population to ***G*** while ***G′*** is empty. The algorithm iteratively updates the population to be ***G′*** U ***G***. The members of ***G′*** U ***G*** are first evaluated to obtain the corresponding *PET values* for each solution using DAG's information and system models. The fitness value of each solution is then obtained by using these *PET values*. There are many ways to assign fitness [12]. The fitness value used in E-FORCE [1] has been shown to be effective for improving solution spaces for higher dimensional problems [42],

$$\forall i \in G \cup G' \qquad fitness(i) = frailty(i) + Density(i). \tag{14}$$

The *frailty* of a solution *i* is the sum of the number of solutions dominated by the members that dominate *i*, where domination relationship is given by (13). Now, if *Domination Strength* (*DS*) is defined as the number of solutions dominated by a solution then *frailty* in (14) is given as:

$$\forall i, j, k \in G \cup G' \qquad DS(j) = \sum_{k} u_{j,k}, \quad where \quad u_{j,k} = 1 \quad iff \quad j \succ k, \tag{15}$$

$$\forall j \succ i \quad frailty(i) = \sum_{j} DS(j). \tag{16}$$

The second component of fitness (*Density*) in (14) is based on the distance between a population member and its *k*th nearest neighbor (*kNN*) calculated in the objective space. Solutions with larger *kNN* distances are assigned a smaller value of density to prefer solutions in less dense regions of Pareto front. This density assignment also allows breaking ties between the solutions with equal *frailty* based on the density of solutions around them. The density value to a solution $i \in$ ***G′*** U ***G***, is [1]:

$$Density(i) = \frac{1}{d_{i,kNN} + 2}, \tag{17}$$

where, $d_{i,kNN} = ||i - kNN||$ and *kNN* represents the *k*th nearest neighbor of the population member *i* in the objective space. Typically, $k = \sqrt{}$(population size) can be used [1], [33]. Consider a feasible region defined by deadline (*D*) and peak temperature constraint ($T_{\max}$) as shown in Fig. 4. The closest neighbors of solution 2 are 3, 9, 8, and 1 while the closest neighbors of solution 4 are 10, 11, 5, and 3. If we select $k = 3$ for (17), then the third nearest neighbor of solution 2 is solution 8. Therefore, the distance between solution 2 and solution 8 ($d_{2,8}$) will be used to calculate the *Density* for solution 2. Similarly, the distance between solution 4 and 5 will be used to calculate the density value of solution 4 ($d_{4,5}$) when $k = 3$.

The solutions with lower fitness values are considered better solutions. Lines 9-12 of the pseudocode in Fig. 1 highlight the fitness assignment.

4.4 Population Selection

Once all solutions in ***G′*** U ***G*** has been assigned a fitness value, the algorithm selects the top η solutions. For an example population in Fig. 5, the shaded rows show the selected solutions. These solutions are assigned to set ***G′***. From this updated ***G′***, a mating pool is generated through a binary tournament, where each solution is allowed to take part in two matches. In each match a solution is compared with another solution based on their fitness values, the solution with lower fitness values wins and is copied to the mating pool. Every *i*th member of ***G′*** competes with solution (*i*-1) mod η and (*i* mod η)+1 where $1 < i \leq \eta$ [30]. After η matches are completed we get a mating pool of size η. This tournament selection favors better solutions to become part of the mating pool as in this scheme a "good" solution can potentially beat two "poor" solutions and thus can be selected twice for the mating pool. Fig. 5 demonstrates the population selection mechanism by starting from the set ***G′*** U ***G*** and then illustrating the fitness assignment, update of ***G′*** and tournament selection for mating pool generation.

4.5 Genetic Operations

The members of the mating pool generated above form pairs randomly and then each pair undergoes three genetic operations namely, uniform crossover, simulated binary crossover and mutation, each with a specified probability. Uniform crossover is applied to the population with a probability p_{c_uc}. During uniform crossover, a starting index is randomly picked from the range [1, ..., 2*N*] for each pair. The members of that pair swap the values of their decision variables from that starting index up to the last index thus creating two child solutions. Fig. 6 illustrates the uniform crossover applied to a mating pair (set of parents) from the mating pool.

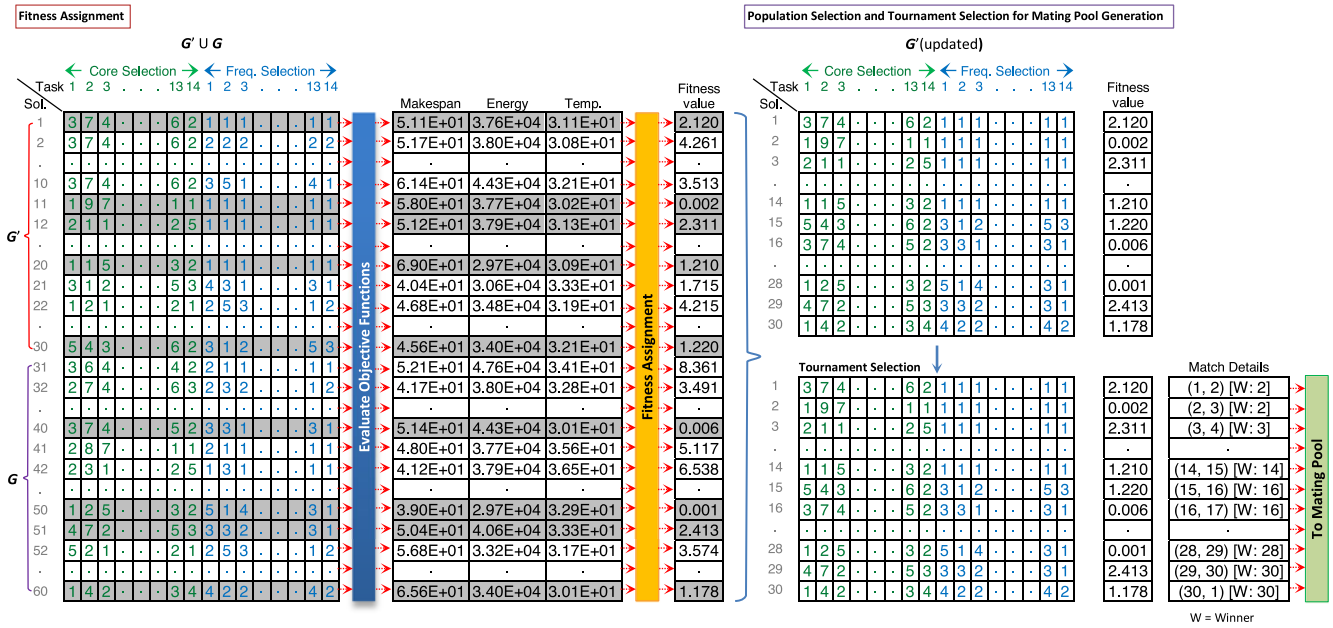

Fig. 5. Fitness assignment, selection, and binary tournament used in E-FORCE.

Next, we apply simulated binary crossover [32] using binomial distributions to the child solutions generated after the uniform crossover. The simulated binary crossover is better suited to the problems with non-binary decision variables [32]. For the PETOS problem, we construct two binomial distributions $B(parentval, 2\Delta + 1)$, where $parentval$ represents the value of the decision variable in the parent member and Δ is the difference between the parent values for the decision variable being updated (Fig. 6). Here, $B(\alpha, \beta)$ represents a binomial distribution with mean equal to α and size β. These distributions allow an offspring to take value close to the parent values when the difference among the parents is small and at the same time provide the opportunity for an offspring to differ significantly from the parent members when the difference among them is large [12]. Since simulated binary crossover is applied to each decision variable individually, there can be two probabilities associated with it [30]. Let us represent the probability that a particular pair will undergo simulated binary crossover as p_{c_mem} and the probability that a decision variable (from the range $[1, \ldots, 2N]$) is updated during simulated binary crossover as p_{c_dec}. For an example, we consider a task graph with 14 nodes, mating pool with 30 solutions/schedules, $p_{c\_mem} = 0.8$ and $p_{c\_dec} = 0.5$, then on average, there will be about 24 members/12 pairs undergoing simulated binary crossover (sbx) in each iteration. Each such sbx operation will result in the update of 14 out of 28 decision variables in each member on average (as $p_{c\_dec} = 0.5$). Fig. 6 shows the simulated binary crossover applied to a decision variable having values 3 and 1 in its parents. Two binomial distributions are generated one for each parent. For first parent, the binomial distribution has its mean at 3, while for second parent the mean is set at 1. Both distributions have a size of 5 as $\Delta = 2$. The value of the corresponding decision variable in child 1 is drawn from the distribution $B(3, 5)$ whereas child 2 gets its value from $B(1, 5)$. In case, the value drawn from distribution is outside the lower and upper limits of a decision variable, it is rounded to the nearest lower/upper limit value.

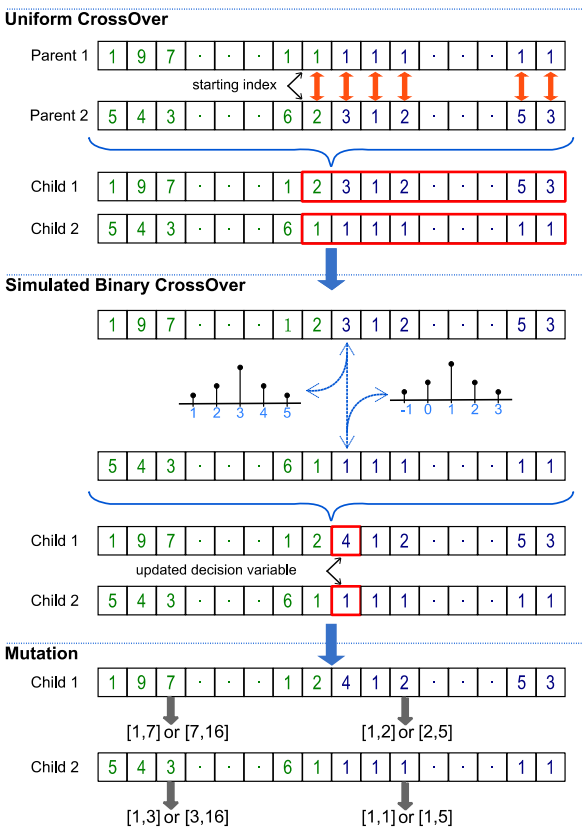

Fig. 6. Genetic operations used in E-FORCE.

Following sbx, each child member undergoes mutation individually. The probability that a particular member will undergo mutation is p_{m_mem} while the probability with which each decision variable in a solution will undergo mutation is given by p_{m_dec}. During mutation, for each decision variable, we generate two uniform distributions with the range [*low*, *value*] and [*value*, *high*]. The low and high represent the lower and upper bounds for that decision variable. The lower and upper bounds (*lower*, *upper*) for

task-allocation and frequency selection decision variables are (1, *M*) and (1, *K*) respectively. Fig. 6 shows the uniform distributions generated for different decision variables during mutation. We can then pick one distribution randomly and use it to draw the value of the decision variable to apply the mutation operation.

Once all mating pairs complete the genetic operations; the iteration is complete and provides a new set of solutions/schedules, called offspring/current population (***G***). The offspring population (***G***) is combined with ***G′*** and the whole process repeats for the given number of iterations. Upon termination, the set ***G′*** (defined as *parent_pop* in Fig. 1) will contain the required Pareto front.

4.6 Solution Selection

The evolutionary process yields multiple solutions (Pareto front) for scheduling a given DAG on a target multi-core system. The next consideration is to pick a schedule from the Pareto front. Here, we use an aggregation scheme based on a *preference vector* (similar to those presented in [43]). Let us define a *preference vector* $\boldsymbol{p}$ as:

$$\boldsymbol{p} = \left\{ w_i | 0 \leq w_i \leq 1 \wedge \sum_i w_i = 1, \quad \forall 1 \leq i \leq 3 \right\}. \tag{18}$$

Now, before we use the vector $\boldsymbol{p}$ to select a solution, we need to apply the given system-based constraints on the obtained Pareto front. For example, performance constraints in the form of task deadlines, energy constraints in the form of total energy budget and thermal constraints including the thermal limit of the system can be used to define the feasible Pareto space ($\boldsymbol{P}_{feasible}$). Thus, $\boldsymbol{P}_{feasible}$ contains those solutions from ***G′*** for which the objective function values satisfy the imposed constraints. Now, if $f_m(x)$ represents the value of the mth objective function (for PETOS, $1 \leq m \leq 3$) generated by the solution x, then we can rank each solution in $\boldsymbol{P}_{feasible}$ as:

$$\forall\ x \in \boldsymbol{P}_{feasible} \wedge w_m \in \boldsymbol{p}$$

$$rank_x = \sum_{m=1}^{3} w_m \frac{f_m(x)}{f_m^*}, \tag{19}$$

where $f_m^* = \min f_m(x), \forall x \in G'$

Hence, we rank each solution based on the ratios of their objective function values to the minimum values along each objective and then scale each term with the corresponding weight from the preference vector. Now, the solution with minimum rank can be selected for execution:

$$\forall x \in \boldsymbol{P}_{feasible}$$

$$x^{selected} = \arg\min_{\forall x}(rank_x). \tag{20}$$

The selection scheme ensures that best trade-off solution is selected from the Pareto front and thus avoids excessive degradation in one quantity while trying to optimize the other two. Lines 23-26 in Fig. 1 summarize the solution selection process of E-FORCE.

5 Experimental Setup

5.1 Power and Thermal Models

To obtain power and thermal models for our evaluations, we collected sample values for power and temperature under different settings on a 16-core system (AMD Opteron-6272 [25]). We varied the number of active cores as well as their execution frequency across different settings. The power values for CPU were obtained by measuring current through the 12 V supply lines to the board at regular intervals using the data acquisition card *NI-USB6008* [23]. The temperature readings were collected from the on-chip temperature sensors through *lm-sensors* [24]. Each setting was sampled for a period of 30 seconds to properly record the average power draw as well as to correctly capture the temperature transients. For each setting, we used the same initial value of temperature i.e., 23°C, this allows to minimize the impact of temperature on power readings across different settings. In addition, we collected temperature data for longer durations under various settings to estimate the thermal RC-constant for the system. These data were then used to derive power and thermal models for the system using the regression tool in Matlab [27]. We used polynomials of the form:

TABLE 3
Characteristics of Task Graphs

Task Graph	Nodes (*N*)	Edges	CCR
FFT	14	20	0.36
Laplace	16	24	0.67
Gauss	20	29	1.19
Fpppp	334	1,145	1.61
100	96	134	0.1, 1.0, 10.0
500	458	645	0.1, 1.0, 10.0
1,000	958	1,393	0.1, 1.0, 10.0

$$P, T(f) = \alpha_n f^n + \alpha_{n-1} f^{n-1} + \cdots + \alpha_1 f^1 + \alpha_0, \tag{21}$$

where f represents the sum of frequencies of all the cores at any given time. We varied n to find its smallest value that can achieve an $R^2 > 0.95$ to obtain the following equations for power and temperature:

$$P(f) = 2 \times 10^{-08} f^2 + 1.8 \times 10^{-03} f + 1.83\, P_{Static} \tag{22}$$

$$T_s(f) = 6 \times 10^{-04} f + T_A. \tag{23}$$

P_{Static} refers to the power consumed by CPU while idling at the lowest frequency level with only one active core. T_A is the ambient temperature, in our case it was set to 23°C. It should be noted here that the temperature model, based on (23), was only used to find the expected steady state temperature for a given setting with a specific initial temperature. The transient temperature values were determined by using the RC model based temperature equation [37]:

$$T(t) = T_s(f) - (T_s(f) - T_i) e^{-[(t-t_i)/\tau]}, \tag{24}$$

where t_i is the time just before the system changed its setting to the current one and T_i is the temperature of the system at that instant. For $t_i = 0$, the value of T_i was set to 23°C. The steady state temperature values for a system under consideration can also be obtained using (9) instead of (23). However, it will require knowing the values of system parameters like *Rth* and **B** as well as power values of each core under different settings. Since such data are usually not readily available, we chose to obtain the thermal model from the sampled data.

TABLE 4
Algorithmic Parameters

Evolutionary Parameters	Selected Value
Member's Sbx Probability (p_{c_mem})	1
Member's Mutation Probability (p_{m_mem})	0.2
Variable's Sbx Probability (p_{c_dec})	1
Variable's Mutation Probability (p_{m_dec})	0.2
Uniform Crossover Probability (p_{c_uc})	1

TABLE 5
DVFS Parameters For ILP-Based Comparison

f(MHz)	Power(W)
1600	23.61
2000	48.9
2200	72.48
2400	93.12
2600	105

5.2 Workload

The workload used for the evaluation of our proposed work comprises both synthetic and application task graphs. For synthetic task graphs, we used *tgff-3.5* [21] to generate task graphs with varying number of nodes and communication to computation ratio (*CCR*). For application task graphs, we selected diverse types of applications that include Fast Fourier Transform (FFT) [15], Gauss Elimination [15], Laplace Equation [16], as well as an application task graph from Standard Task Graph set (STG) [14] namely Fpppp. Table 3 presents the characteristics of the workload used for evaluations.

5.3 Algorithm/System Parameters

The parameters used for the genetic operations can impact the performance of evolution-based algorithms. Therefore, we thoroughly evaluated the performance of E-FORCE under various parameter settings for different task graphs. Specifically, we varied the mutation and crossover probabilities from a low value of 0.2 to as high as 1.0; we also used different initial population biasing levels as well as different task ordering methods to find the best parameter setting for E-FORCE. We obtained task ordering based on DCP generated schedule, bottom-level values of each task (b-levels [17]), and the critical path based classification of tasks (CPN, IBN, OBN) to generate priority lists. The priority list obtained using critical path based classification is usually termed as CPDS [17]. We found that E-FORCE performs best when DCP-based list is used for task ordering. (Please see [17] for more details on b-levels and CPDS.)

The size of the population (η) was set to 30 whereas 50 generations/iterations of evolution were allowed for generating the Pareto front of each application. A 16-core system with five frequency levels (1400, 1500, 1700, 1900, and 2100 MHz) was used for task graphs with $N < 300$. For $N \geq 300$, a system with 64 cores was assumed. The evaluations do not assume time-sharing and preemption of tasks.

Table 4 presents the list of the parameters along with the value selected for each of the parameter. Selected value in Table 4 corresponds to the value of the parameter that resulted in maximum number of unique solutions along each axis and with Pareto front closer to the origin.

6 Results

The experimental evaluation and comparison consists of three parts. First, we simulated and compared E-FORCE with the solutions obtained by the mixed integer linear programs (ILPs) for small task graphs. Second, we evaluated the proposed algorithm against energy- and thermal-aware scheduling algorithms on a real multi-core system. Third, we used simulation for very large task graphs which were not implemented due to practical limitations of the machine.

6.1 Comparison with ILPs

We formulated mixed integer linear programs to obtain minimum makespan and minimum energy consumption for task graphs with $N \leq 20$. These ILPs similar to those presented in [28] were solved using LPSolve3.5 [26]. The setup and execution times of such ILPs are extremely large (of the order of hours for small task graphs [28]) even with P and E only. Thus T was not calculated through ILPs because of the prohibitively high complexity. For the same reason, we used simpler power-performance models for both simulations and the ILPs in this set of experiments. Table 5 shows the power-frequency settings used for this set of experiments.

Fig. 7 shows the Pareto fronts obtained by E-FORCE for FFT, Gauss, and Laplace Equation along with the ILP solutions for minimum makespan (ILP (Min-Makespan)) and

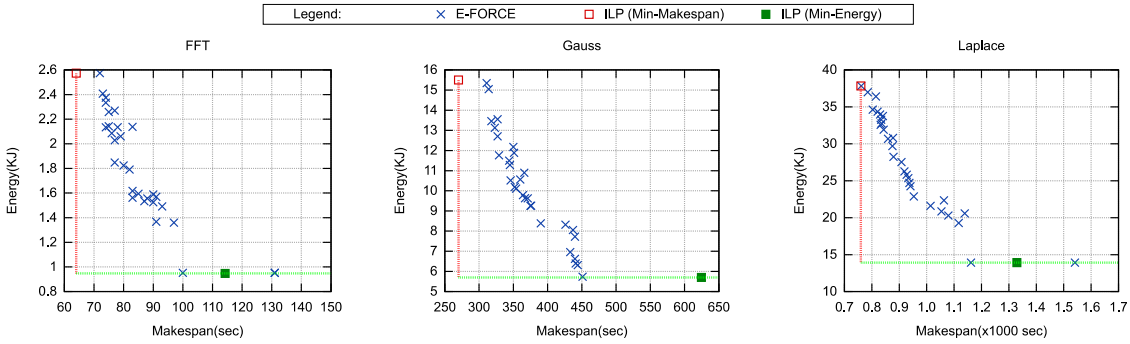

Fig. 7. Comparison of E-FORCE with solutions of ILPs for minimum makespan and minimum energy for different task graphs.

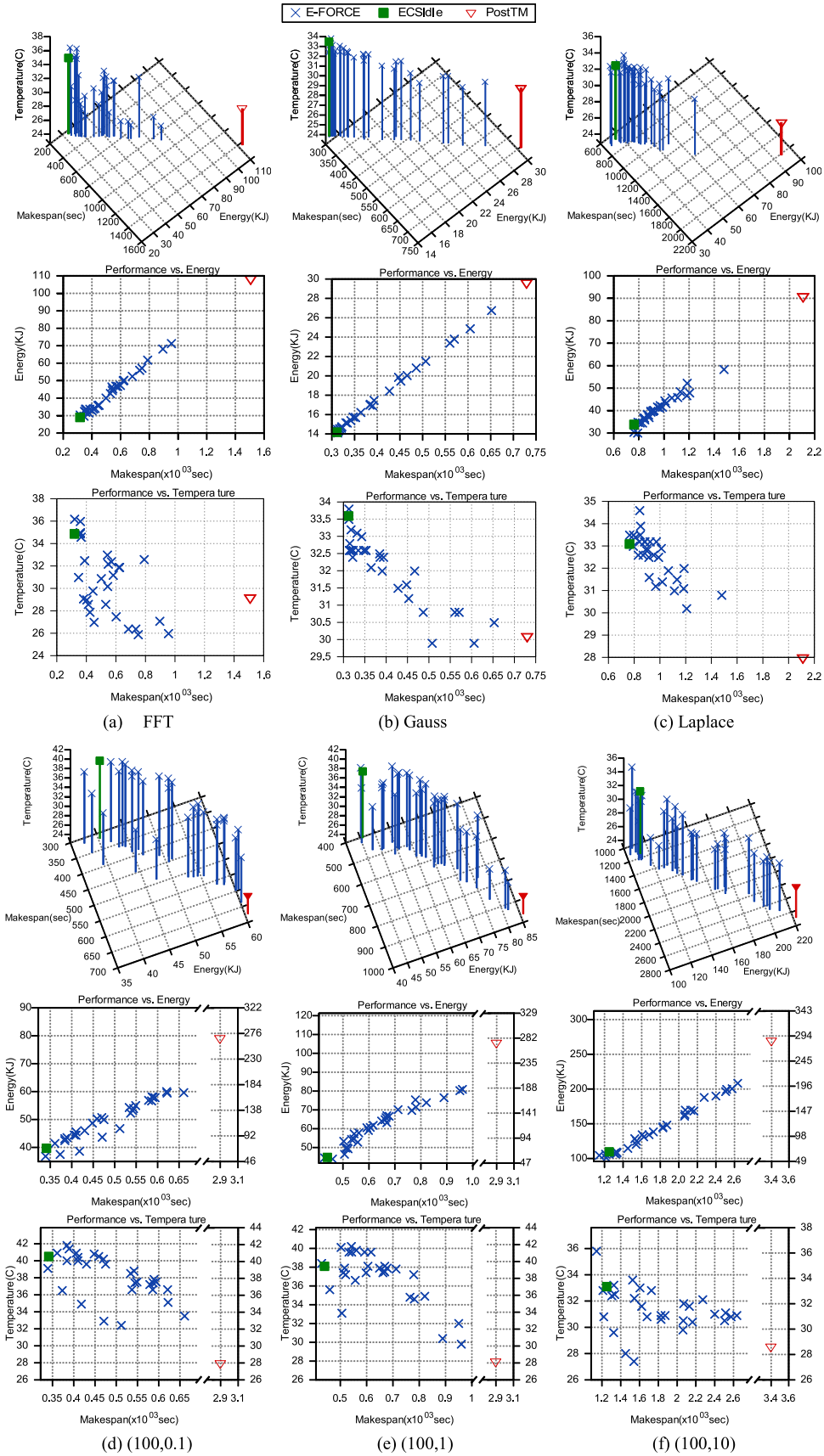

Fig. 8. Comparison between the Pareto fronts generated by E-FORCE vs. ECSIdle and PostTM.

minimum energy (ILP (Min-Energy)). Each point in Fig. 7 represents a pair (makespan, energy) corresponding to the schedules generated by the algorithms. We observe that E-FORCE produces schedules that achieve the same minimum energy as that of ILP. The Pareto front generated by E-FORCE attained about 8.5 percent higher makespan

TABLE 6
% Difference and Number of Unique Solutions for Pareto Fronts Generated by E-FORCE

Task Graph	% Difference			Unique Solutions		
	P	*E*	*T*	*P*	*E*	*T*
FFT	71.35%	47.81%	6.76%	20	30	11
Laplace	63.93%	64.10%	13.58%	29	30	20
Gauss	70.54%	61.88%	12.24%	28	30	17
(100,0.1)	64.40%	47.87%	25.34%	28	30	23
(100,1)	76.46%	59.26%	29.71%	29	30	21
(100,10)	78.98%	69.43%	26.58%	29	30	22
Mean	70.94%	58.39%	19.04%	27.17	30.00	19.00

value on average as compared to the ILP-based solution while this deviation was as low as 0.14 percent and as high as 18 percent. These deviations are acceptable considering the solution time of ILPs as compared to the execution time of E-FORCE and also because E-FORCE tries to optimize all three objectives as compared to single objective in the case of ILP. Another important observation is that most of the points comprising the Pareto front present a unique trade-off between performance and energy and tend to balance the two extremities (i.e., min. makespan and min. energy).

6.2 Comparison with Other Algorithms

We compared E-FORCE with ECSIdle [20] (an energy-aware scheduling algorithm), and PostTM(maxTemp) [8] (a thermal-aware scheduling algorithm) by executing schedules on a 16-core system. Each core was allocated its corresponding busy and idle slots based on the schedules generated by these algorithms. During busy slots, each core executed the CPU cycles burning program [29] at the selected frequency, resulting in 100 percent core utilization for the specified period on that core. For idle slots, cores executed *sleep* command. The idle slots correspond to the time slots for which a core has to wait due to the dependency constraints among the tasks. We modified PostTM(maxTemp) to incorporate frequency selection and the precedence constraints among tasks. Fig. 8, where each point represents a possible schedule, shows the Pareto fronts obtained by E-FORCE along with the *PET values* of ECSIdle and PostTM for both the application and synthetic task graphs. For each task graph, the top row compares the algorithms in three dimensional objective-space while the next two rows delineate performance vs. energy and performance vs. temperature, respectively. The synthetic applications in Figs. 8d, 8e, and 8f are represented by attributes pair (number of tasks, *CCR*). For most of the task graphs, PostTM(maxTemp) results in excessively large makespan making it difficult to compare it with ECSIdle and E-FORCE along the three objectives. For such task graphs, we mapped PostTM point just outside the x- and y-axis with a filled *nabla* symbol while comparing them in 3D objective-space. The figures for performance vs. energy and performance vs. temperature depict the actual values achieved by PostTM against those by E-FORCE and ECSIdle.

Fig. 8 indicates that for most of the task graphs, the Pareto fronts generated by E-FORCE contained solutions with comparable *PET values* to those by ECSIdle. While for some of the task graphs, E-FORCE achieved better values of peak temperature with identical or better values of performance and energy as compared to ECSIdle. On the other hand, PostTM generates schedules with the lowest peak temperatures; it does so at the cost of excessively larger makespan; as much as seven times greater than the minimum makespan generated by E-FORCE (see Fig. 8d). In some cases, where PostTM results in comparable values of makespan (Fig. 8b), E-FORCE achieves a temperature comparable to that of PostTM but with slightly better makespan. Although the proposed algorithm performs comparably with the other two algorithms on the two objectives alone, the point is that the other algorithms ignore the third objective and hence are limited in their scope and usefulness. In contrast, E-FORCE, allows simultaneous optimization of all three objectives without letting any one objective get out of control.

TABLE 7
PET Values and Rank of Solutions in (100,0.1)'s Pareto Front

No.	*P* (ms)	*E* (J)	*T* (°C)	*rank* p = {0.4, 0.4, 0.2}	*rank* p = {0.1, 0.1, 0.8}
1	**339**	**3.68E + 04**	**39.1**	**1.04**	1.15
2	361	4.14E + 04	40.9	1.12	1.21
3	373	3.75E + 04	36.5	1.07	1.10
4	384	4.25E + 04	40	1.16	1.20
5	384	4.31E + 04	41.8	1.18	1.25
...	...	...	...	...	...
10	418	3.86E + 04	34.9	1.13	1.08
11	**471**	**4.37E + 04**	**32.9**	1.23	**1.06**
12	448	4.87E + 04	40.8	1.31	1.26
...	...	...	...	...	...
30	661	5.96E + 04	33.5	1.63	1.17

Table 6 indicates that E-FORCE generates multiple schedules with a broad range of values in the objective domain. For example, the percentage difference between the maximum and minimum values for E-FORCE is as high as 78.98, 69.43, and 29.71 percent in performance, energy, and temperature, respectively. In addition, the number of unique solutions along all objectives is mostly greater than 20 and usually close to 30 (which is the maximum allowed size of the solution set (η) used in our evaluations). Table 7 shows the *PET values* and the solution selection quality in terms of *rank* (19) obtained by our algorithm for the given preference vectors. The selected solution for each case is shown in bold.

We measured the percentage decrease in the minimum *PET values* achieved by E-FORCE against the *PET values* of ECSIdle and PostTM along all objectives. The results in Fig. 9 indicate that for almost all cases, E-FORCE achieved the minimum peak temperature significantly lower/better than that of ECSIdle (shown as % Improvement in Fig. 9a). At the same time, it performs comparably along performance and energy objectives. Compared to PostTM, E-FORCE achieved peak temperatures higher than PostTM but at the same time resulted in mostly 40 percent or higher percentage decrease/improvement in performance and energy (Fig. 9b). Again, the output of E-FORCE is not a single schedule; rather it generates the whole Pareto front comprising multiple schedules that not only achieve better or comparable values than ECSIdle and PostTM but also facilitate choosing a trade-off point between the *PET quantities*.

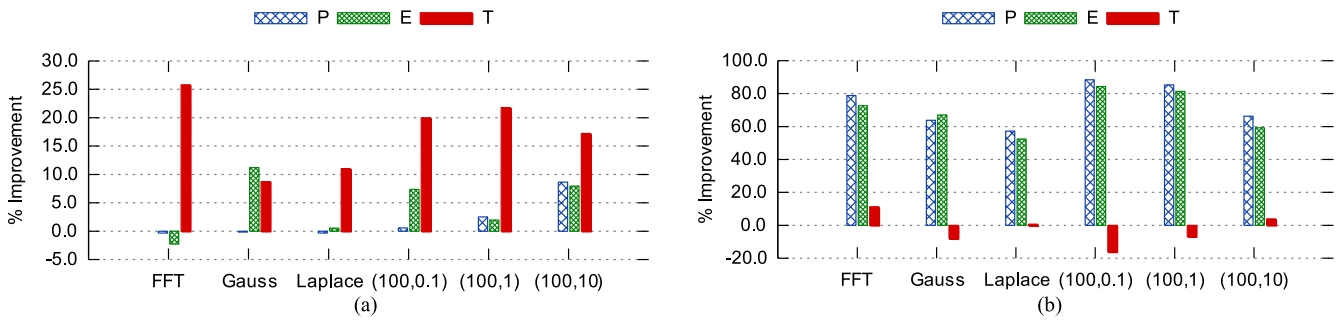

Fig. 9. Percentage improvement in the minimum *PET values* achieved by E-FORCE over (a) ECSIdle and (b) PostTM on a 16-core system.

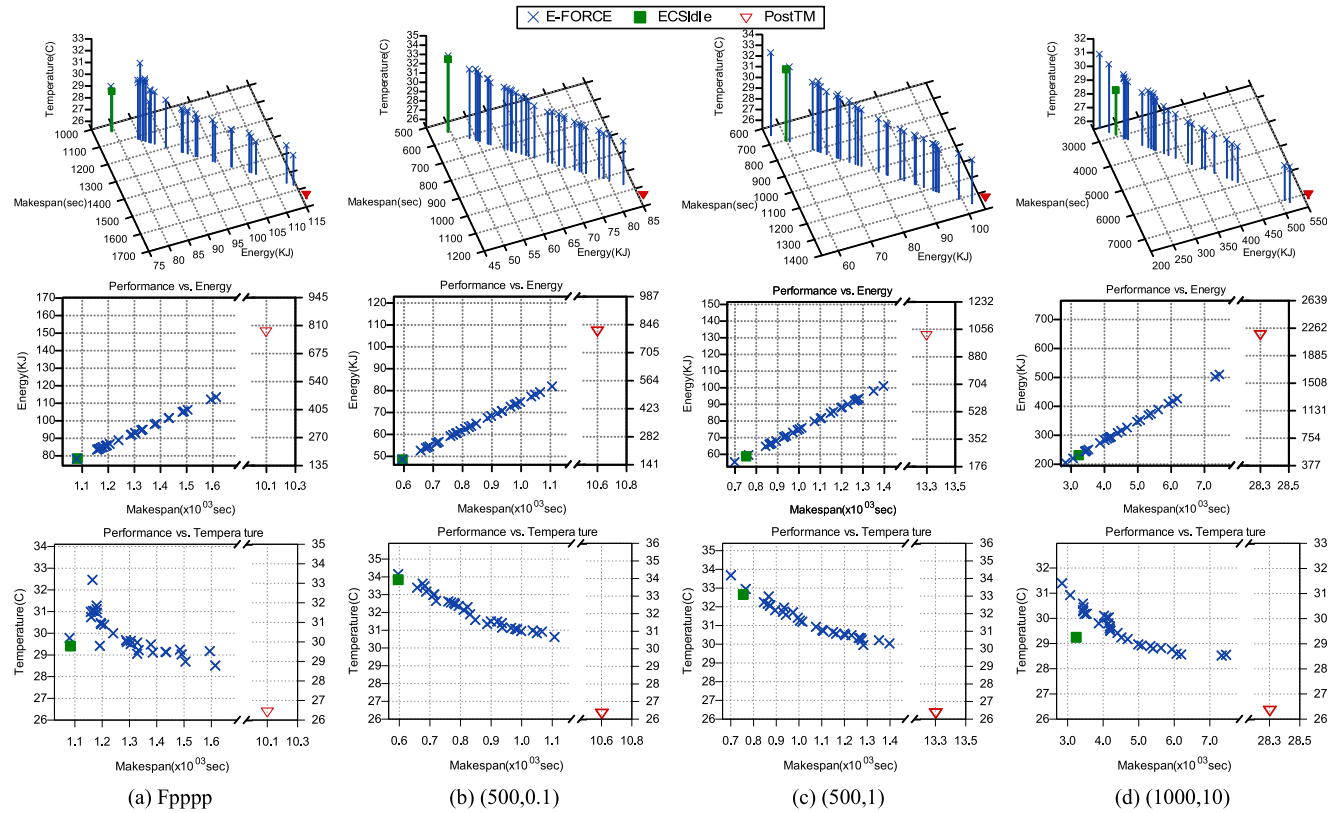

Fig. 10. Comparison between the Pareto fronts generated by E-FORCE vs. ECSIdle and PostTM for large task graphs.

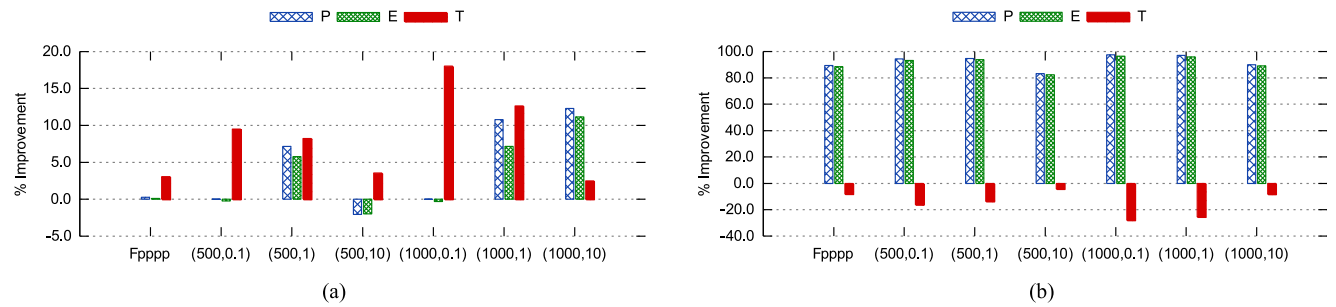

Fig. 11. Percentage improvement achieved by E-FORCE in the minimum values of the *PET quantities* as compared to (a) ECSIdle and (b) PostTM.

6.3 Simulation for Large Task Graphs

These larger tasks graphs required either very long execution traces or needed a larger system than the available Opteron-6200 platform, and hence were simulated. Fig. 10 shows that for these tests, E-FORCE produced schedules that perform comparably to ECSIdle along performance- and energy-axis while improving the peak temperature. Similarly, the relative comparison between E-FORCE and PostTM is also the same as noted in 6.2. Fig. 11 compares the minimum values of *PET quantities* achieved by E-FORCE with ECSIdle and PostTM.

Table 8 presents the total execution times of each algorithm when executed on a single core running at

TABLE 8
Total Execution Times in Seconds

Algorithm	Task Graph												
	FFT(14)	Gauss(16)	Laplace(20)	(100, 0.1)	(100, 1)	(100, 10)	Fpppp(334)	(500, 0.1)	(500, 1)	(500, 10)	(1000, 0.1)	(1000, 1)	(1000, 10)
E-FORCE	33	40	48	358	**338**	**264**	**783**	**1,580**	**1,471**	**1,260**	**5,099**	**5,146**	**3,617**
ECSIdle	15	16	22	2,450	2,091	2,667	13,608	38,741	38,301	45,521	167,227	221,304	270,439
PostTM	**3**	**14**	**10**	**312**	371	658	6,146	2,564	3,024	6,492	10,776	13,706	39,619

2100 MHz. E-FORCE exhibits rich scalability and is at least an order of magnitude faster than ECSIdle for large task graphs. For 1,000 node task graph, the maximum time taken by the proposed algorithm is about 1 hour 25 minutes, while ECSIdle and PostTM take 75 hours and 11 hours respectively. The reason for the slowness of the other algorithms is that they are essentially exhaustive while the proposed algorithm uses a systematic and efficient method of searching the solution space. Another important aspect of the evaluation is the relationship between performance and energy. Figs. 8 and 10 indicate that the schedules with the minimum makespan also achieve the minimum energy. This is due to two reasons: (a) static power in current/emerging multi-core systems constitutes a large part of the total power draw [34]; (b) the modified schedules may potentially create additional or longer idle slots as compared to the minimum-makespan schedule thus consuming more energy in pursuit of reducing it as also observed in [20]. However, note that we only measured the energy consumption for the duration of the execution of a schedule and did not include idle energy consumption of the machine if it has to sit idle after finishing earlier. We observe that by varying the relationship between static and dynamic power we obtain a different relationship between performance and energy. For example, assume that P_d (dynamic power) is the total amount of power that can be controlled by either disabling cores or by changing their frequencies and P_s (static power) represents the minimum power drawn by the processor while idling with only one active core. Fig. 12 shows the impact of decreasing the ratio P_s/P_d on the relationship between performance and energy. We note that the relationship reverses at $P_s/P_d = 0.25$ and the solutions generated by E-FORCE now allow trading off performance to improve energy consumption. Regardless of the relationship between performance and energy, E-FORCE effectively explores the scheduling decision space for generating the multiple schedules to provide trade-off opportunities available for the given system.

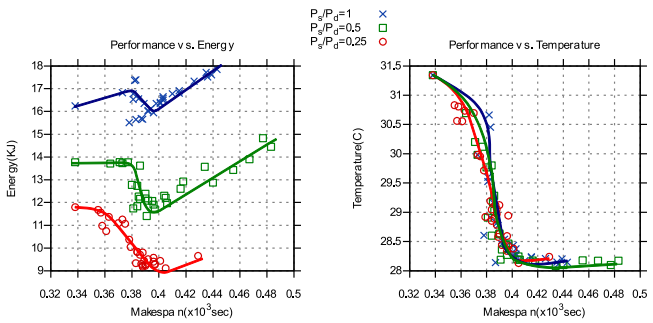

Fig. 12. Impact of varying relationship between static and dynamic power on Performance-Energy and Performance-Temperature trade-offs.

7 Conclusions

In this paper, we proposed an efficient algorithm utilizing concepts from evolutionary computing to solve the PETOS problem. Compared to other energy- and thermal-aware scheduling schemes, E-FORCE solves the PETOS problem by generating a set of schedules (Pareto front) with diversely spread *PET values* rather than a clustered schedule for the given requirements. This set of solutions can then be used in the solution selection phase of E-FORCE to pick a solution from the Pareto front based on user's preference. E-FORCE achieved values as close as 0.14 percent to the global minimum values along performance, and energy. At the same time, E-FORCE attained comparable or better values of performance, energy, and temperature in comparison to energy-aware (ECSIdle) and thermal-aware (PostTM) scheduling schemes. In addition, the execution time of E-FORCE scales better with increasing number of tasks as compared to ECSIdle and PostTM.

Acknowledgments

This work is in part supported by the National Natural Science Foundation of China under Grant No. 61173007, No. 61332009, and the National Grand Fundamental Research 973 Program of China under Grant No. 2011CB302501.

References

[1] E. Zitzler, M. Laumanns, and L. Thiele, "SPEA2: Improving the strength pareto evolutionary algorithm," Swiss Federal Inst. Technol., Dept. of Elect. Eng., Zürich, Switzerland, Tech. Rep. TIK-Report 103, 2001.

[2] H. SongFa and Z. Ying, "NSGA-II based grid task scheduling with multi-QoS constraint," in *Proc. 3rd Int. Conf. Genetic Evol. Comput.*, 2009, pp. 306–308.

[3] S. Zheng, W. Shu, and D. Shangping, "Task scheduling model design using hybrid genetic algorithm," in *Proc. 1st Int. Conf. Innovative Comput., Inform. Control*, 2006, vol. 3, pp.316–319.

[4] B. Gorji-Ara, C. Pai, N. Bagherzadeh, M. Reshadi, and D. Jensen, "Fast and efficient voltage scheduling by evolutionary slack distribution," in *Proc. Asia South Pacific Des. Autom. Conf.*, Jan. 2004, pp. 659–662.

[5] A. J. Page, T. M. Keane, and T. J. Naughton, "Multi-heuristic dynamic task allocation using genetic algorithms in a heterogeneous distributed system," *J. Parallel Distrib. Comput.*, vol. 70, no. 7, pp. 758–766, Jul. 2010.

[6] R. Viswananth, V. Wakharkar, A. Watwe, and V. Lebonheur, "Thermal performance challenges from silicon to systems," *Intel Technol. J., Q3*, vol. 23, p. 16, 2000.

[7] A. H. Ajami, K. Banerjee, and M. Pedram, "Modeling and analysis of nonuniform substrate temperature effects on global ULSI interconnects," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 24, no. 6, pp. 849–861, Jun. 2005.

[8] J. Cui and D. L. Maskell, "Dynamic thermal-aware scheduling on chip multiprocessor for soft real-time systems," in *Proc. 19th ACM Great Lakes Symp.*, May 2009, pp. 393–396.

[9] J. Yang, X. Zhou, M. Chrobak, Y. Zhang, and L. Jin, "Dynamic thermal management through task scheduling," in *Proc. IEEE Int. Symp. Perform. Anal. Syst. Softw.*, Apr. 2008, pp. 191–201.
[10] S. Murali, A. Mutapcic, D. Atienza, R. Gupta, S. Boyd, L. Benini, and G. De Micheli, "Temperature control of high-performance multi-core platforms using convex optimization," in *Proc. Conf. Des., Autom. Test Eur.*, 2008, pp. 110–115.
[11] T. Ebi, M. Faruque, and J. Henkel, "TAPE: Thermal-aware agent-based power economy multi/many-core architectures," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Des. - Digest Techn. Papers*, Nov. 2009, pp. 302–309.
[12] K. Deb, *Multi-Objective Optimization Using Evolutionary Algorithms, (Wiley-Interscience Series in Systems and Optimization)*. Chichester, U.K.: Wiley, 2001.
[13] Z. Wang and S. Ranka, "A simple thermal model for MPSoC and its application to slack allocation," in *Proc. IEEE Int. Parallel Distrib. Process. Symp.*, 2010, pp. 1–11.
[14] Standard Task Graph Set, STG, [Online]. Available: http://www.kasahara.elec.waseda.ac.jp/schedule/, 2007.
[15] I. Ahmad, Y.-K. Kwok, M.-Y. WU, and W. Shu, "CASCH: A tool for computer-aided scheduling," *IEEE Concurrency*, vol. 8, no. 4, pp. 21–33, Oct. 2000.
[16] M.-Y. Wu and D. Gajski, "Hypertool: A programming aid for message-passing systems," *IEEE Trans. Parallel Distrib. Syst.*, vol. 1, no. 3, pp. 330–343, Jul. 1990.
[17] Y.-K. Kwok and I. Ahmad, "Static scheduling algorithms for allocating directed task graphs to multiprocessors," *ACM Comput. Surveys*, vol. 31, no. 4, pp. 406–471, 1999.
[18] Y.-K. Kwok and I. Ahmad, "Dynamic critical-path scheduling: An effective technique for allocating task graphs to multiprocessors," *IEEE Trans. Parallel Distrib. Syst.*, vol. 7, no. 5, pp. 506–521, May 1996.
[19] H. F. Sheikh and I. Ahmad, "Simultaneous Optimization of performance, energy and temperature for DAG scheduling in multi-core processors," in *Proc. Int. Green Comput. Conf.*, Jun. 2012, pp. 1–6.
[20] Y. C. Lee and A. Y. Zomaya, "Energy conscious scheduling for distributed computing systems under different operating conditions," *IEEE Trans. Parallel Distrib. Syst.*, vol. 22, no. 8, pp. 1374–1381, Aug. 2011.
[21] TGFF, [Online]. Available: http://ziyang.eecs.umich.edu/~dickrp/tgff/, 2011.
[22] H. Liu, Z. Shao, M. Wang, and P. Chen, "Overhead-aware system-level joint energy and performance optimization for streaming applications on multiprocessor systems-on-chip," in *Proc. Euromicro Conf. Real-Time Syst.*, Jul. 2008, pp. 92–101.
[23] National Instruments, NI USB-6008, [Online]. Available: http://sine.ni.com/nips/cds/view/p/lang/en/nid/201986, 2015.
[24] Lm-Sensors, lm-sensors, [Online]. Available: http://www.lm-sensors.org/, 2014.
[25] AMD Opteron 6200 Series Processors, AMD, [Online]. Available: http://www.amd.com/en-us/products/server/opteron/6000/6200#, May 2015.
[26] LPSolve, [Online]. Available: http://lpsolve.sourceforge.net/5.5/, 2010.
[27] NonLinear Regression, MathWorks, [Online]. Available: http://www.mathworks.com/discovery/nonlinear-regression.html, May 2015.
[28] A. K. Coskun, T. S. Rosing, K. A. Whisnant, and K. C. Gross, "Static and dynamic temperature-aware scheduling for multiprocessor SoCs," *IEEE Trans. Very Large Scale Integr. Syst.*, vol. 16, no. 9, pp. 1127–1140, Sep. 2008.
[29] BurnP6, [Online]. Available: http://manpages.ubuntu.com/manpages/precise/man1/cpuburn.1.html, 2011.
[30] Genetic Operations in NSGA-II, [online], Available: http://goo.gl/Qg3BQC, 2009.
[31] E. Le Sueur and G. Heiser, "Dynamic voltage and frequency scaling: The laws of diminishing returns," in *Proc. Int. Conf. Power Aware Comput. Syst.*, 2010, pp. 1–8.
[32] K. Deb and R. B. Agrawal, "Simulated binary crossover for continuous search space," Indian Institute of Technology, Dept. of Mech. Eng., Kanpur, UP, India, Tech. Rep. IITK/ME/SMD-94027, 1994.
[33] B.W. Silverman, *Density Estimation for Statistics and Data Analysis*. London, U.K.: Chapman and Hall, 1986.
[34] N. S. Kim, T. Austin, D. Blaaw, T. Mudge, K. Flautner, J. S. Hu, M. J. Irwin, M. Kandemir, and V. Narayanan, "Leakage current: Moore's law meets static power," *IEEE Comput.*, vol. 36, no. 12, pp. 68–75, Dec. 2003.
[35] C-H. Hsu and U. Kremer, "The design, implementation, and evaluation of a compiler algorithm for CPU energy reduction," in *Proc. ACM SIGPLAN 2003 Conf. Programm. Lang. Des. Implementation*, 2003, pp. 38–48.
[36] G. Palermo, C. Silvano, and V. Zaccaria, "Multi-objective design space exploration of embedded systems," *J. Embedded Comput.*, vol. 1, no. 3, pp. 305–316, Aug. 2005.
[37] S. Zhang and K. S. Chatha, "Approximation algorithm for the temperature-aware scheduling problem," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Des.*, 2007, pp. 281–288.
[38] O. Khan and S. Kundu, "Hardware/software co-design architecture for thermal management of chip multiprocessors," in *Proc. Conf. Exhib. Des., Autom. Test Eur.*, Apr. 2009, pp. 952–957.
[39] J. S. Lee, K. Skadron, and S. W. Chung, "Predictive temperature-aware DVFS," *IEEE Trans. Comput.*, vol. 59, no. 1, pp. 127–133, Jan. 2010.
[40] M. A. Khan, C. Hankendi, A. K. Coskun, and M. C. Herbordt, "Software optimization for performance, energy, and thermal distribution: initial case studies," in *Proc. Int. Green Comput. Conf. Workshops*, Jul. 25–28, 2011, pp. 1–6.
[41] E. Rotem, A. Naveh, D. Rajwan, A. Ananthakrishnan, and E. Weisman, "Power management architectures of the intel microarchitecture code-named sandy bridge," *IEEE Micro*, vol. 32, no. 2, pp. 20–27, Mar. 2012.
[42] K. Deb, A. Pratab, S. Agrawal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Trans. Evol. Comput.*, vol. 6, no. 2, pp. 182–197, Apr. 2002.
[43] J. Koski and R. Silvennoinen, "Norm methods and partial weighting in multicriterion optimization of structures," *Int. J. Numerical Methods Eng.*, vol. 24, no. 6, pp. 1101–1121, 1987.

Hafiz Fahad Sheikh received the BSc degree in electrical engineering from the University of Engineering and Technology, Pakistan in 2004 and the MSc degree in computer engineering from the same University in 2008. He is currently working towards his PhD at the University of Texas at Arlington since 2010. His research interests include energy- and thermal-aware task allocation and scheduling techniques. He was a graduate technical intern at Intel Corporation during summer semester in 2013.

Ishfaq Ahmad received the BSc degree in electrical engineering from the University of Engineering and Technology, Pakistan, in 1985, the MS degree in computer engineering, and the PhD degree in computer science from Syracuse University, New York, in 1987 and 1992, respectively. He is a professor of computer science and engineering at the University of Texas at Arlington (UTA). His research focus is on the broader areas of parallel and distributed computing systems and their applications, optimization algorithms, multimedia systems, video compression, and energy-aware green computing. He has received numerous research awards, including three best paper awards at leading conferences and 2007 best paper award for *IEEE Transactions on Circuits and Systems for Video Technology*, IEEE Service Appreciation Award, and 2008 outstanding area editor Award from the *IEEE Transactions on Circuits and Systems for Video Technology*. He is the founding editor-in-chief of the new Journal, *Sustainable Computing: Informatics and Systems*. He is a fellow of the IEEE.

Dongrui Fan received the PhD degree of computer architecture in 2005 from the Institute of Computing Technology, Chinese Academy of Sciences, and now he is an associate professor of the institute. His research interests are focused on computer architecture, on-chip multi-core/many-core architecture, and low-power embedded micro-architecture design. He is a technical committee member of Computer Architecture and System Software of China Computer Federation (CCF). He served as the program vice-chair of Multi-Core and Parallel Systems in the International Conference on Parallel Processing (ICPP) in 2011. He is a member of the IEEE.