

# Energy-Efficient Computing using Agent-Based Multi-Objective Dynamic Optimization

Alexandru-Adrian Tantar, Grégoire Danoy, Pascal Bouvry, Samee U. Khan

**Abstract** Nowadays distributed systems face a new challenge, almost nonexistent a decade ago: energy-efficient computing. Due to the rising environmental and economical concerns and with trends driving operational costs beyond the acquisition ones, *green computing* is of more actuality than never before. The aspects to deal with, e.g. dynamic systems, stochastic models or time-dependent factors, call nonetheless for paradigms combining the expertise of multiple research areas. An agent-based dynamic multi-objective evolutionary algorithm relying on simulation and anticipation mechanisms is presented in this chapter. A first aim consists in addressing several difficult energy-efficiency optimization issues, in a second phase, different open questions being outlined for future research.

## 1 Introduction

High Performance Computing (HPC) evolved over the past three decades into increasingly complex distributed models. After attaining Gigaflops and Teraflops performance with Cray2 in 1986, respectively with Intel ASCI Red in 1997, the Petaflops barrier was crossed in 2008 with the IBM Roadrunner system [31]. And trends indicate that we should reach Exaflops in the next ten to fifteen years [15]. The shift towards decentralized paradigms raised nonetheless scalability, resilience and, last but not least, energy-efficiency issues [8]. Disregarded or seen as an extraneous factor in the HPC's beginnings, the carbon emissions footprint of data centers escalated to levels comparable to those of highly-developed countries [21]. Estimates place the energy consumption of an Exaflops scale system in the range of

---

Alexandru-Adrian Tantar, Grégoire Danoy, Pascal Bouvry  
Faculty of Sciences, Technology and Communication, University of Luxembourg e-mail:  
{alexandru.tantar,gregoire.danoy,pascal.bouvry}@uni.lu

Samee U. Khan  
North Dakota State University e-mail: samee.khan@ndsu.edu

hundreds of megawatts. Thus, not counting the raising environmental concerns, the trend inflicts important economical consequences. The scope of this chapter is therefore focused on this last aspect, i.e. minimizing energy consumption while delivering a high performance level, and addresses several difficult issues in the *energy-efficient dynamic* and *autonomous* management of distributed computing resources. The intricate interplay of factors shaping the problem calls nevertheless for solutions at the crossing of, among others, distributed computing, scheduling and dynamic optimization [16, 26]. Answering part of these aspects, we propose an agent-based dynamic evolutionary multi-objective approach dealing with the time-dependent dynamic and stochastic factors defining a distributed computing environment.

Without entering into details, a few key characteristics of large scale systems, of interest for the aims of this chapter, are introduced hereafter. We first focus on HPC core aspects, gradually extending the discussion to grid and cloud systems. Existing implementations rely on expensive institution-centered high-end computers, clusters or grids, with a high degree of complexity [4, 9]. Moreover, these resources may potentially be shared across administrative domains or multiple geographically-distributed centers. Therefore, as main points, one has to deal with complexity for performance, energy consumption and execution deadlines. The DOE/NNSA/LANL (Los Alamos National Lab) 100 million US\$ BladeCenter Cluster, ranked first in Top500 in 2009, was wired through more than 10000 Infiniband and Gigabit Ethernet connections extending over almost 90 kms of fiber optic cable [18]. The system was estimated to deliver, according to IBM, 444.94 Megaflops/watt, ranking seventh in Green500<sup>1</sup>. Over the same period, the first Green500 system, a BladeCenter QS22 Cluster, delivered 536.24 Megaflops/watt. We have therefore two ends, opposing energy and performance, where an improvement in one of them leads to a degradation of the other. And the gap forcing energy requirements and performance apart does not cease to extend with time. At one year distance, in June 2010, the Oak Ridge National Laboratory's Cray XT5-HE was ranked first in Top500 with 2.331 Petaflops of computational power and only 56th in Green500 with 253.07 Megaflops/watt. At the same time, an ascending Flops per Watt tendency can be identified. Under development at IBM and to start running at the Lawrence Livermore National Laboratory (LLNL) in 2012, the Sequoia system, designed to include 1.6 millions of power-processors providing a peak performance of more than 20 Petaflops, will supposedly sustain 3000 Megaflops/watt for an input power of 6 Megawatts [19].

Cloud computing, while still dealing with complexity and energy consumption issues, brings into focus business and additional privacy constraints [1, 28]. As an outline, cloud computing can be described in terms of computational demand and offer where entities (individuals, enterprises, etc.) negotiate and pay for access to resources administered by a different entity that acts as provider. Here, demanding parties may have diverging requirements or preferences, given in contractual terms that stipulate data security, privacy or quality of service levels. Performance metrics include in this case occupation of resources, users' satisfaction or service

---

<sup>1</sup> <http://www.green500.org>

level agreement violations [29, 30]. What is more, dynamic and risk-aware pricing policies may apply where predictive models are used either in place or through intermediary brokers to assess the financial and computational impact of decisions taken at different moments in time. Furthermore, legal enforcements may restrict access to resources or data flow, e.g. data crossing borders or transferred to a different resource provider. As common examples, one can refer to Amazon Web Service or Google Apps Cloud Services.

Summarizing, various scenarios have to be dealt with in order to construct an energy-efficient optimization paradigm for distributed computing systems that offers scalability and resilience capabilities in an autonomous, transparent manner. An ideal approach would have to cope with time evolving performance and cost constraints while anticipating the long term effects of the decisions taken at specific moments in time. National security, surveillance and defence applications for which reaction time is critical impose performance as single and only criterion. Sharing computational power among high-priority applications running inside such a system may nonetheless require to make use of strategies which take into account not only the current contextual state but also future possible evolutions. At the opposite end, for the academic and public domains, reducing energy consumption may stand as a main factor [6].

We propose a decentralized, agent-based, dynamic multi-objective anticipative EA. The chapter identifies the main components one deals with in energy-efficient autonomic computing and, through abstraction and conceptualization, advances the idea of a generic application framework. Evolutionary Algorithms (EAs) represent a natural option to consider given their capability of dealing with highly multi-modal functions in both mono and multi-objective cases as well as their notorious success for various applications [7, 13]. Furthermore, the adoption of an agent oriented paradigm is clearly adapted to address the autonomous and decentralized management of energy in distributed systems. As mentioned in [25], multi-agent systems can be used as an approach to the construction of robust, flexible and extensible systems and as a modeling approach. Also, autonomic computing and, therefore, distributed systems management, is a “killer app” for multi-agent systems [11].

The remainder of this chapter is organized as follows. Related work is discussed in Section 2, followed by a brief introduction to basic optimization notions in Section 3 and a description of the model and concepts later used for experimentation in Section 4. Dynamic energy-efficient optimization aspects are discussed in Section 5, followed by results in Section 6 and conclusions.

## 2 Related Work

In the literature, energy-efficient approaches for large-scale distributed systems are typically divided into two classes: centralized and distributed. Centralized approaches [33] are historically the first ones, which allow close to optimum results but imply scalability and fault-tolerance problems as soon as the number of nodes in-

creases. To answer those limitations, distributed approaches were introduced, which in turn brought new challenges such as ensuring maintainability and global performance.

In [23] Khargharia presented a holistic theoretical framework for autonomic power and performance management exploiting different components (CPU, memory, network and disks) and their interactions. Their autonomic resource management framework optimises the power/performance ratio at each level of the hierarchy using a mathematical approach for which simulation results on static and dynamic scenarios are provided. Similarly, Bennani *et al.* in [27] proposed a hierarchical approach to the data center resource allocation problem using global and local controllers. It uses a prediction model based on mutliclass queuing network models combined with a combinatorial search technique (i.e. the Beam search algorithm). Berral *et al.* in [5] introduced another predictive model for power consumption and performance for task scheduling based on a machine learning approach (i.e. linear regression algorithm). In this approach, the learning algorithm permits to model data for a given management policy whereas in the previous approaches it is used to learn management policies.

Bio-inspired algorithms have also been investigated by Barbagallo *et al.* in [3] who optimized energy in data centers using an autonomic architecture. However this approach is limited to single-objective optimization. Another natural approach to model autonomic power management is the multi-agent paradigm, as demonstrated by Das *et al.* in [10], in which power and performance in a real data center setting is managed by agents autonomously turning on/off servers. This approach also uses reinforcement learning to adapt the agents' management policies.

To summarize, it appears that various distributed approaches dedicated to energy optimization in large-scale distributed systems have already been investigated. However these only considered standalone approaches, such as static mathematical models, single-objective bio-inspired algorithm or prediction models. The contribution proposed in this chapter intends to combine and extend these through an anticipative dynamic multi-objective evolutionary algorithm for agent-based energy-efficient computing.

### 3 Optimization Introductory Notions

For an energy function  $F$ , defined over a decision space  $X$  and taking values in an objective space  $Y$ ,  $F : X \rightarrow Y$ , one may consider the  $\min_{x \in X} F(x)$  minimization problem. With no restriction for the topics of this chapter, we can additionally assume that  $F$  is nonlinear and that  $Y \subseteq \mathbb{R}$ . If  $F$  is continuous and double differentiable,  $x^+$  is considered to be a local optimum point if the following relations stand:

$$\frac{\partial F}{\partial x^+} = 0, \frac{\partial^2 F}{\partial^2 x^+} > 0.$$

No straightforward formulation can be given for dynamic functions. Different classes can be designated here, with functions including time-evolving factors or enclosing random variables, functions depending on past states, etc. For the general case and assuming a minimization context, the goal is to identify a sequence of solutions  $\mathbf{x}(t)$ , i.e. solution  $\mathbf{x}$  to be applied at the  $t$  time moment, with  $t \in [t_0, t^{end}]$  leading to the following:

$$\min_{\mathbf{x}(t)} \int_{t_0}^{t^{end}} F(\mathbf{x}(t), t) dt$$

In a more explicit form, this implies providing a sequence of solutions which, for the given time interval, minimize the cumulative objective function obtained by integration (for discrete time intervals the problem can be formulated by summing over all time moments). Assuming  $\mathbf{x}(t)$  to be the sequence that minimizes the above relation, for a different sequence  $\mathbf{x}^\Delta(t)$ , with  $t \in [t_0, t^{end}]$ , the following stands:

$$\min_{\mathbf{x}(t)} \int_{t_0}^{t^{end}} F(\mathbf{x}(t), t) dt \leq \min_{\mathbf{x}^\Delta(t)} \int_{t_0}^{t^{end}} F(\mathbf{x}^\Delta(t), t) dt$$

For the multi-objective case,  $F$  is extended to define a vector of objective functions  $F : X \rightarrow \mathbb{R}^k$ ,  $F(x) = [f_1(x), \dots, f_k(x)]$ . The set  $X \subset \mathbb{R}^d$  of all the *feasible* solutions defines the *decision space*, while the function  $F$  maps *feasible solutions* into an *objective space*. In addition, the Pareto optimality concept is used, based on partial order relations defined as follows.

**Definition 1.** Let  $v, w \in \mathbb{R}^k$ . We say that the vector  $v$  is *less than*  $w$  ( $v <_p w$ ), if  $v_i < w_i$  for all  $i \in \{1, \dots, k\}$ . The relation  $\leq_p$  is defined analogously.

**Definition 2 (Dominance).** A point  $y \in X$  is *dominated* by  $x \in X$  ( $x \prec y$ ) if  $F(x) \leq_p F(y)$  and if  $\exists i \in \{1, \dots, k\}$  such that  $f_i(x) < f_i(y)$ . Otherwise  $y$  is called non-dominated by  $x$ .

**Definition 3.** A point  $x \in X$  is called a *Pareto point* if there is no  $y \in X$  which dominates  $x$ . The set of all Pareto solutions forms the *Pareto Set*.

In the multi-objective case, by extension, a scalarization function, given in explicit or implicit form, e.g. weighted sum of the objective functions, preference or expert based decisions, has to be used at each time step for selecting solutions out of the approximate Pareto front. Note that due to the dynamic nature of the problem *one solution and only one has to be selected and applied* per time step – this does not represent an option. The modeled problem can occupy a single state at a given time moment and no reset to a previous state can be done. In this case minimization is considered over a vector of cumulative objective function realizations and can be defined to obey Pareto optimality laws. In addition, the semantics of the minimization operator may be subject to context, e.g. deviation from a specified target. As a general model, the problem can be therefore stated as follows:

$$\min_{\mathbf{x}(t)} \left\{ \int_{t_0}^{t^{\text{end}}} f_i(\mathbf{x}(t), t) dt \right\}_{1 \leq i \leq k} = \min_{\mathbf{x}(t)} \left\{ \int_{t_0}^{t^{\text{end}}} f_1(\mathbf{x}(t), t) dt, \dots, \int_{t_0}^{t^{\text{end}}} f_k(\mathbf{x}(t), t) dt \right\}$$

Last but not least, if solutions are dependent, e.g. with  $x(t) = H(x(t-c))$ , where  $H(\cdot)$  is an arbitrary function and  $c < t$ , anticipation has to be considered. If only the current *context* is regarded, a solution minimizing the function  $F$  at the moment  $t$  is optimal, but this may not stand for the complete set of following solutions, i.e. a suboptimal overall fitness is attained. The function  $F$  must be consequently optimized by taking simultaneously into account the complete set of solutions (over time). A more detailed discussion is given in Section 5 also introducing algorithm and simulation related notions like strategy or scenario [6].

## 4 An Anticipative Decentralized Oriented Dynamic Multi-Objective Evolutionary Approach

A framework for an anticipative decentralized dynamic multi-objective evolutionary optimization of energy has been designed for simulation purposes. Modeled using the agent paradigm, an example of the instantiation of the framework is illustrated in Fig. 1, with one agent assigned per computing node.

The *General Scheduler* assigns tasks to the different available nodes, i.e. in this case through the corresponding *Local Schedulers*. In addition, each node is provided to enclose decision mechanisms capable of autonomously managing local energy-efficiency with respect to the overall or partially observed states of the system. Nodes are independently controlled through dynamic frequency and voltage scaling, directly driven by local parameters such as local system load (SL), node overhead (NO), idle time (IT) or current node state (CS), and by global indicators which describe system load (SL), operation cost (OC) and energy vs performance ratio (EvP). The scaling is also indirectly driven by the type and rate of loaded and offloaded tasks exchanged among nodes. Note that although exclusive use of simulation is made in this work, the passage to a real-world environment would still require having simulation mechanisms in order to be able to carry anticipation over the future states of the system.

The next two subsections describe the computing environment in terms of tasks attributes, e.g. life-time and states, followed by computing nodes properties. The local scheduler agent is detailed in the last subsection.

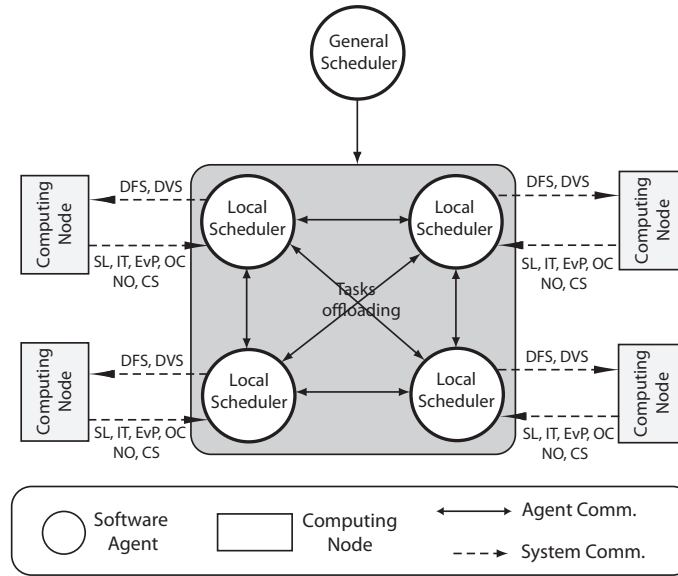


Fig. 1: Example of a *per node* system instance.

#### 4.1 Computing vs Communication Intensive Tasks

For a realistic modelling of the execution environment, the system is analyzed by considering computing and communication intensive tasks. We mainly focus on studying the behaviour of the simulated system under intense computational load and in the presence of communication or memory oriented tasks. Different scenarios can be thus envisaged where distinct policies apply, e.g. computing intensive tasks being assigned to independent machines, or where clusters of jobs are dynamically created in order to maximize occupancy and resource utilisation. As a side note, a heterogeneous environment has been preferred to a system running tasks with both computing and communication traits exclusively in order to study extreme situations. Finally, energy consumption is assessed by relying on load indicators with no direct account for communication.

The specifications defining a task are given in either invariant form, fixed for the entire lifetime of the task, or dynamic, subject to random variation at execution time. The first category includes due time, i.e. the latest point in time by which the task has to be completed, and an offloading flag. Migrating idle virtual machines to a secondary server may be coherent with load-balancing policies while offloading an application which depends on real-time processing or that requires access to security critical databases is not desirable. The offloading flag therefore serves for marking tasks which can not be offloaded once assigned to a specific resource. Estimations of the required execution time, computational load and average network transfer time

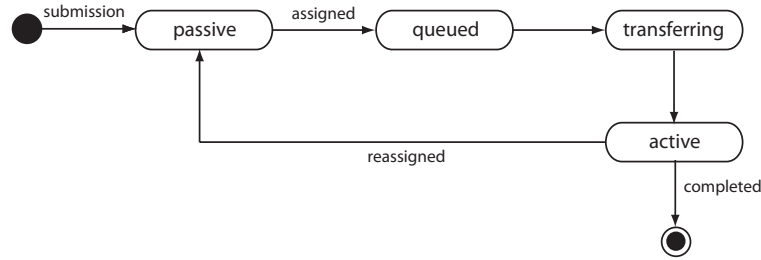


Fig. 2: Task life cycle – reassignments imply a transfer time overhead leading to potential delay penalties.

Table 1: Power states (p-states) for the Intel®Pentium®M processor (A) and a stock 3.2 GHz, 130W processor (B) with afferent frequencies, voltage and approximate power requirement.

**A: Intel®Pentium®M**

P-State	Frequency	Voltage	Power
P0	1.6 GHz	1.484 V	24.5 Watts
P1	1.4 GHz	1.420 V	17 Watts
P2	1.2 GHz	1.276 V	13 Watts
P3	1.0 GHz	1.164 V	10 Watts
P4	800 MHz	1.036 V	8 Watts
P5	600 MHz	0.956 V	6 Watts

**B: 3.2 GHz, 130W stock processor**

P-State	Frequency	Voltage	Power
P0	3.2 GHz	1.45 V	130 Watts
P1	3.06 GHz	1.425 V	122 Watts
P2	3.0 GHz	1.4 V	116 Watts
P3	2.8 GHz	1.375 V	107 Watts
P4	2.66 GHz	1.35 V	100 Watts
P5	2.6 GHz	1.325 V	94 Watts
P6	2.4 GHz	1.3 V	85 Watts
P7	2.2 GHz	1.275 V	77 Watts
P8	2.0 GHz	1.25 V	68 Watts
P9	1.8 GHz	1.175 V	55 Watts
P10	1.7 GHz	1.15 V	50 Watts
P11	1.5 GHz	1.125 V	42 Watts
P12	1.4 GHz	1.1 V	38 Watts

fall in the second category. In addition, task completion and, if the case, transfer percentage indicators can be observed at run-time. Please note that none of these dynamic factors are seen as an *a priori* information, e.g. as for use with scheduling algorithms, being available for the purpose of simulation only.

The complete life cycle of a task can be described by several different states. A schematic representation is given in Fig. 2. After submission, the task is placed in a passive state, waiting to be assigned. Active state is reached after all afferent data is transferred to the target resource and processing starts. Subsequently, depending on the nature of the task and the employed optimization strategy, multiple reassignments may occur. Before migrating to a different resource, a task is placed back into a passive state, subsequently moving to queued and transferring states. Note that all transitions requiring to move a task from a prior passive state to an active one, imply a network transfer time overhead. As a last aspect, the explicit failure of a node

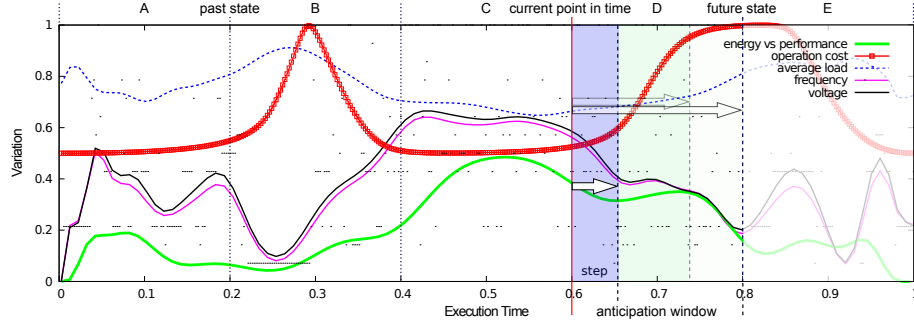


Fig. 3: Example of voltage and frequency scaling for a single node – normalized average values. A specified performance target (traced in the lower part of the figure) and operation cost direct the dynamics of the node. Other factors like overall system load or network traffic (not drawn here) may intervene.

leads to all assigned tasks to be reinitialised and queued back into the system, i.e. any derived results or application states are considered erroneous.

## 4.2 Environment and System Nodes

The computational environment is defined over a collection of abstract nodes which can scale to approximate the behavior of a single processing core or that of hierarchical resource, referred to as “Computing Node” in Fig. 1. As detailed hereafter, we focus on the interaction connecting external factors, provided in a descriptive, informative form, with state indexes, internal to nodes and imperative in nature. No information is known at system level with respect to the functioning of the nodes, assumed nonetheless autonomous and capable of transiting different performance states. An example is given in Table 1, where, for the Intel® Pentium® M 24.5W processor, the frequency and voltage associated to each state is given along with the resulting power consumption [20]. For simulation and experimentation purposes we also rely on a 3.2 GHz, 130W stock processor, not discussed in detail here. All nodes are in addition subject to a system-wide synchronization, ensuring execution context coherency over time-dependent factors.

For the purpose of this study we consider that nodes define interconnected single processing units. Different techniques may apply at core level, including Dynamic Frequency Scaling (DFS) and Dynamic Voltage Scaling (DVS) [14, 22]. Extrapolating to clusters, Dynamic Power Management (DPM) schemes can be implemented or machines can be dynamically assigned to different states or roles, e.g. communication front-ends, dispatching or storage machines. A graphical representation capturing the evolution of a single node over a one-day span, normalized as an execution time varying from 0.0 to 1.0, is given in Fig. 3 (discussed in more detail

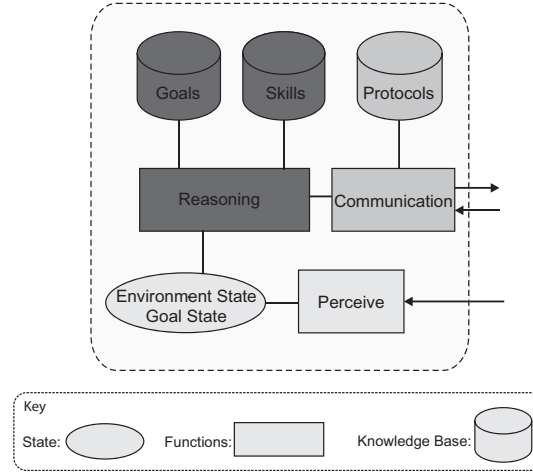


Fig. 4: A conceptual basic view of a node's architecture.

later in this section). Energy vs performance and operation cost identify the balance to ensure between consumption and computational performance, respectively the cost associated with the operation of the node. Low energy vs performance values imply a request for minimizing consumption whereas values close to 1.0 demand performance. Definite semantics can be formulated upon the incurred delay, emergency level or load percentage, where high values lead to an increased performance demand. Analogously, cost can be associated to direct financial indicators, e.g. variation expressed as a function of operational expenses. Alternative definitions may relate to carbon dioxide emissions footprint, energy supply levels (low battery translating into high costs), or environmental temperature where, for example, thermal sensors placed at different locations in a center room may limit execution from entering into a hazardous functioning condition. For the herein case we consider that both energy vs performance and operation cost terms are specified by an external independent source with arbitrary associated (unknown) semantics. No explicit dependency is assumed, i.e. energy vs performance and operation cost may vary independently.

The fundamentals defining a node at an internal level, for this study, relate to the different associated performance states, operating frequency, voltage and power specifications. Energy consumption, for a given activity factor, input power and effective capacitance, denoted respectively by  $\alpha$ ,  $C$ ,  $f$  and  $V$ , is considered to scale linearly with frequency and to be quadratically proportional to voltage. An approximation can be obtained by integrating over  $P \approx \alpha C f V^2$  (power dissipation) [24].

### 4.3 Local Scheduler

After the detailed description of the computing nodes provided in the previous subsection, we now describe the architecture of the local scheduler. A conceptual view of its architecture is given in Fig. 4, based on a cognitive agent model. The *goals* of a local scheduler are to minimize through local decisions the system's energy consumption and the overall delay.

To achieve these goals, the agent has a set of *skills* and a *reasoning* function. Skills include dynamic voltage and frequency scaling (DVS, DFS) and tasks offloading capabilities. A fuzzy inference system implements the reasoning function. A multi-objective evolutionary algorithm is used to evolve the weights of the fuzzy system while relying on an anticipation mechanism (details provided in the following). As an additional skill, tightly linked with the communication function, nodes have the possibility to offload tasks to other local schedulers. To communicate with other local scheduler agents, an agent relies on a set of *communication protocols* (assumed transparent here and not further detailed). Through the *perceive* function, the agent is aware of the environment state (i.e. the global system's performance) and of its local goals' status. The following provides a detailed description of the instantiation of the aforementioned local scheduler architecture.

As previously mentioned, a first effective energy-management skill consists in dynamically scaling frequency and voltage [14, 17]. A linear decrease in energy consumption is attained by downscaling frequency, respectively quadratic by reducing voltage. Nonetheless, while a load-dependent scaling would stand as a straightforward approach, no direct formulation is possible here due to the additional energy vs performance and operation cost specifications. High load should result in a frequency and voltage increase but this has also to obey performance and operation constraints. Additional factors taken into account here define local and average system load, node overhead, idle time and current state (active or sleeping).

A second skill consists in enforcing local and system wide loading and offloading policies along with task assignment and distribution mechanisms [2]. The dynamics driving the loading and offloading of tasks, in this case, have a direct impact on a node's load and on the balance of the system while indirectly affecting the context from which scaling decision criteria are drawn. Energy-aware load-balancing techniques have to be consequently defined, controlling not only the distribution of tasks across resources but also the type of tasks assigned to or offloaded from nodes, e.g. regrouping one computational intensive task with multiple communication oriented ones on the same node. As an example, an Intel®Pentium®M processor operating at full load in High Frequency Mode (HFM), P0 state, is expected to drive energy consumption at a higher level than three identical processors operating in Low Frequency Mode (LFM), state P5, under an equivalent load.

Summarizing, we opted for a design where frequency and voltage scaling is driven (1) in direct manner by taking into account the average load over a short term period, idle time, energy vs performance and operation cost, and (2) indirectly by controlling the type and rate of loaded or offloaded tasks. Furthermore, given that a centralized approach would not stand the scaling requirements of a large dis-

tributed system, a *by node driven* paradigm has been preferred. The effective control of the systems is conducted upon the emergent result of all nodes' decisions where each node incorporates an independent functioning logic. For the herein case all nodes use a zero degree Takagi-Sugeno [32] fuzzy inference system composed of 41 disjunctive rules (13 rules for deciding when and how to increase voltage and frequency, 16 rules for downscaling control and 12 rules for task loading and off-loading conditions). An excerpt of these rules is given hereafter with the title of example.

```

if energyVsPerfTarget is energy  $\wedge$  operationCost is high then
  decrease voltage and frequency by a large percentage

if energyVsPerfTarget is perf.  $\wedge$  consumption is low  $\wedge$  load is high then
  increase voltage and frequency by a large percentage

if energyVsPerfTarget is energy  $\wedge$  consumption is high then
  offload a high number of tasks

if energyVsPerfTarget is perf.  $\wedge$  load is low  $\wedge$  sysLoad is high then
  accept a large number of tasks to be loaded

if energyVsPerfTarget is balance  $\wedge$  consumption is low then
  accept a moderate number of tasks to be loaded

```

A brief functioning insight is offered hereafter by referring to the different sections of Fig. 3. The simulation starts from a LFM mode with a moderately high load (section A of the graph). Despite the slight operation cost increase, it is straightforward to observe the raise in voltage and frequency, in correlation with energy vs performance – note that scaling also depends on load. Analogously, for approximately the same energy vs performance level (transition between section A and section B) but for an increased operation cost and in spite of the load's escalation, voltage and frequency drop, given the steep ascension of the operation price. In the following steps, with the sustained energy vs performance increase, scaling reacts to raise voltage and frequency, also reflected by the load decline due to the completion of a larger number of tasks.

## 5 Dynamic Energy-Efficient Optimization

This section provides a detailed description of the last skill of the local scheduler agent, i.e. the dynamic multi-objective anticipative evolutionary algorithm. As described in the previous sections, the simulation model is designed to capture the dynamic nature of a real-life distributed environment. Tasks arrive at different moments in time, not known in advance, performance and cost parameters vary independently, load overhead may affect all performance factors, e.g. due to external users accessing the system, etc. Therefore, an energy-efficient approach, in this case, has to cope not only with instantaneous, static factors, but also needs to adapt over

time to a continuously changing environment and, what is more, to perturbations induced by multiple stochastic sources. An additional aspect to consider is that different evolution paths are possible from any given point in time. Moreover, as load overhead, operation cost and failure are all modeled in stochastic form, the system can be defined as a random variable where, given an initial configuration, all possible realisations at the next step or over multiple steps have to be assessed. Having taken into account all possible outcomes, a decision can be formulated with respect to changes to perform in order to maintain the system in an optimal state (maximal performance at lowest possible energy consumption). Nonetheless, as a complete assessment of all possible realisations of the system is not feasible from a computational point of view, decisions have to be taken only upon a reduced number of samples. Furthermore, as a multi-objective approach is adopted, with energy consumption and delay as objectives to be minimized, solution selection and decision problems have to be addressed. We therefore rely on concepts like *scenario*, *strategy* and *anticipation window* [6], discussed in more detail in the following. Different open questions arise and solutions to the afferent problems are subject to further research – an outline of the main concepts and implications is given hereafter.

### 5.1 Strategies and Scenarios

The evolution of the distributed system as a whole is an emergent effect of all nodes' transitions. As we desire to minimize energy consumption and delay, the control parameters of each node have to be dynamically modified at each step. We therefore speak of decisions acting on frequency and voltage scaling, e.g. putting the node in a higher state than the current one, or local energy vs performance variation. Decisions can be formulated with respect to the factors defining the context of a node at a given moment in time and the optimality of these decisions can be heuristically assessed over their effective application life-time. The main difficulty is that a decision taken at a specific moment  $t$ , although optimal with respect to the outcomes induced over its effective processing time window, i.e.  $[t, t + p]$ , may not be optimal at long term. For a graphical illustration refer to Fig. 3. Otherwise stated, for a given configuration  $\gamma$  of the system to simulate at a given time  $t$ , and given two decision  $d_t^1, d_t^2$ , e.g. performing transitions towards a lower, respectively upper power state, the short term overall outcome of the first decision may be suboptimal with respect to the second decision, i.e.  $\int_t^{t+st} F_{d_t^1}(\gamma)dt \leq \int_t^{t+st} F_{d_t^2}(\gamma)dt$ , whereas long term overall effects may prove to be exactly the opposite with  $\int_t^{t+lt} F_{d_t^2}(\gamma)dt \leq \int_t^{t+lt} F_{d_t^1}(\gamma)dt$ , where  $st < lt$  represent short and long term corresponding time window lengths. Therefore, instead of relying on punctual decisions, long term strategies are used as basis for constructing a dynamic approach capable of anticipating future consequences of current decisions. A first problem to be addressed, once the strategy concept developed, relates to the effective evaluation of these objects. For a system following a predictable evolution, which can be simulated in exact manner, i.e.

with no stochastic factors, it is sufficient to carry the simulation over the specified anticipation time interval.

At the opposite end, when stochastic sources are part of the model, for any given number of processing steps, we can only approximate the most probable states or the average state of the system. Therefore, for a set of different candidate strategies and for a given configuration of the system at a specified moment in time, samples have to be drawn, termed hereafter as scenarios, and evaluated independently. A realization of the system is thus obtained for each scenario, with respect to the constraints and dynamics dictated by each strategy. For coherence and consistency, strategies are evaluated over the same set of scenarios, the final outcome of each strategy being expressed as the average energy consumption and delay.

Note that several questions arise, subject to further research, with respect to processing and anticipation time windows, strategy evaluation procedures, etc. First, we face two opposing directions. Long simulation times lead to an escalation of the anticipation error – simulating stochastic factors over a long period of time potentially leads to configurations which no longer follow or represent reality. One would therefore want to limit the extent over which anticipation is carried. At the opposite end, short anticipation windows may not capture correctly the effects of a current decision over the future state(s) of the system. What is more, having a strategy evaluated as the average result of different scenarios may not always be consistent, e.g. when clustered or sparse states are obtained as a result.

## 6 Experimentation and Results

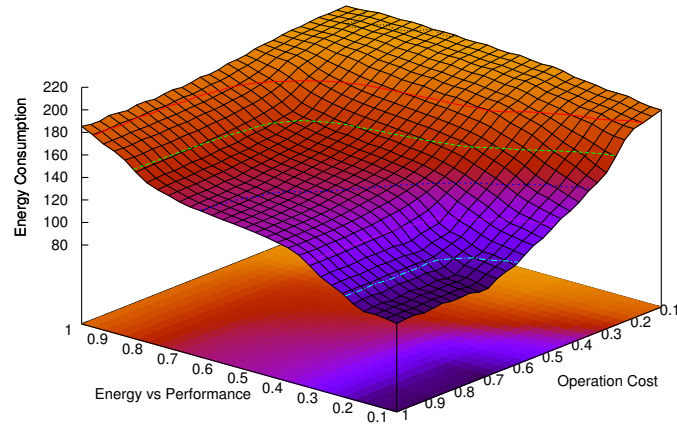
In the remainder of this chapter we present several results, first by analyzing the behaviour of the fuzzy system at a node level, subsequently moving to a system-wide focused discussion.

A comparative view of energy consumption and overall delay is given in Fig. 5. For fixed energy vs performance and operation cost values, constant over the entire one-day simulation time, the corresponding energy consumption and delay are respectively illustrated in Fig. 5a and Fig 5b. Note that, for illustration purposes, axes are mirrored between the two figures. Each point, e.g. energy vs performance and operation cost set at 0.3, respectively 0.5, represents the average value of 30 independent simulations. Different plateaus can be identified for energy consumption and, correspondingly, for delay. A substantial energy minimization is obtained for an operation cost above 0.6, i.e. the equivalent of stating that energy is an important asset, and performance levels below 0.3. Recalling the performance states of Intel® Pentium® M, this is the equivalent of operating the processor in the P4, P5 power states. Moderate consumption is attained for performance levels between 0.3 and 0.8 (equivalent of P1, P2 and P3 states) when operation cost is superior to 0.6. For all other cases, energy consumption raises to reflect either a high performance demand (energy vs performance superior to 0.8, i.e. equivalent of P0 for Intel® Pentium® M) or a low operation cost which, in turn, allows a high voltage

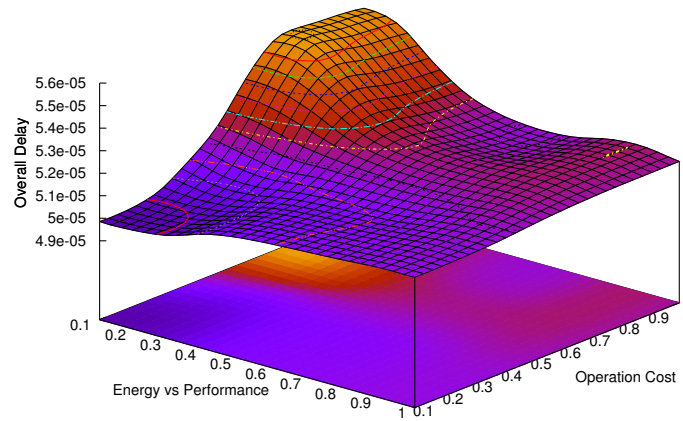
and frequency to be maintained. The behaviour of the system with respect to extreme values is also of interest, the following cases being possible:

- lowest possible energy consumption at the highest operation cost (energy vs performance and cost set at 0.0 respectively 1.0). The outcome of drastic frequency and voltage downscaling, e.g. processor constantly operated in lower performance states, close to or in LFM, is obvious and straightforward to anticipate: energy consumption is at its lowest while delay is driven to a maximum.
- energy vs performance and operation cost both set to 0.0, i.e. knowing that the price to pay for energy, for example, is highly affordable, minimize energy consumption. A first remark to be made here is that operation cost constrains the variation amount allowed for scaling. Nonetheless, as a second aspect, scaling does also consider the load of a node. Therefore, as the system is executed at an affordable cost or at no costs at all, we only have to cope with load. As a consequence, this leads to a separation in two subclasses: 1) execution under high load – the node is put in a high execution state due to the affordable cost but this in turn drives energy consumption over a positive slope (case exposed in Fig. 5); 2) low load – scaling only copes with load; minimal energy consumption is attained for minimal load.
- maximal performance, knowing that operation cost is at its highest (energy vs performance and cost both set to 1.0). As specifications demand performance, with no regard for energy consumption, and given the high load, the node is operated in HFM, state P0. This reflects symmetrically in consumption and delay graphics (Fig. 5a and Fig. 5b).
- maximal performance with lowest operation cost (values set to 1.0, respectively 0.0). Scaling is performed by following energy vs performance and load indicators. The system is hence driven to respond to load stress, where from the high energy consumption illustrated in Fig. 5.

A less explicit aspect of the presented graphs is the influence of task scheduling and load-balancing. The order and priority associated to tasks (at node level) have a direct impact on delay whereas load-balancing (system level) can result in perturbations of both energy consumption and delay. The assignment of highly computational intensive tasks to low performance nodes may be a first source of delay. Similarly, the exclusive use of a high power consumption node to run communication intensive tasks may not be optimal. As the purpose of this study is to analyse the use of a dynamic optimization algorithm for distributed systems in the presence of different stochastic sources, we consider a minimal definition for scheduling and balancing policies. Furthermore, as a centralized approach does not fulfill the requirements of a large scale distributed environment, e.g. where administration policies may have only a limited acting power across domains, we preferred enforcing node-based balancing. Each node is thus responsible for deciding upon acceptance rates, e.g. for loading tasks, bias between computational and communication intensive tasks, as well as on offloading parameters. The process is controlled by a subset



(a) Energy Consumption



(b) Delay

Fig. 5: Energy consumption and delay for various energy vs performance and operation cost levels. A high energy consumption reflects increased voltage and frequency values which in turn lead to a reduced delay – pairwise points, axes mirrored between 5a and 5b. Symmetrically, a reduced energy consumption translates to increased delay levels.

of rules within the fuzzy inference system, taking into consideration local and system load, energy vs performance and operation cost specifications. Subsequent to loading, all tasks receive equal priorities, the amount of dispatched active processing time varying in concordance with the specified computational load footprint.

The complete simulation environment, as described in the previous sections, relies on a collection of autonomous nodes. These nodes although capable of adjusting their state as dictated by the internal base of rules may not optimally scale to all possible scenarios. Or, otherwise put, the weights associated to the rules may not be optimal. A possible solution, as mentioned earlier in this chapter, consists in using an evolutionary algorithm to modify the firing weights of the rules dynamically. Thus, for each node, besides the enclosed fuzzy inference system, a multi-objective parallel NSGA-II [12] evolutionary algorithm is deployed. For each algorithm, individuals code the strategy to use for modifying rules' weights as an array of real transition values (initialized in uniform random manner in  $[0, 1]$ ). The fitness function is expressed as the energy and delay average values obtained over three different independent scenarios through simulation and anticipation in a specified time frame. A scenario is obtained as a simulation carried over a given length of time where random and stochastic factors are seeded by a given initial value. This allows for different strategies to be analyzed against an identical set of scenarios. Simulations and anticipation are performed in parallel on remote nodes to speedup the process. Please refer to Fig. 3 for an intuitive illustration of simulation and anticipation related notions.

Strategies are iteratively evolved by the algorithm having as performance measure the same anticipation mechanism, at the end of each iteration, weights being exchanged between the nodes. At following steps the simulation carried inside each node uses the updated weights hence maintaining context coherence. At every 15 iterations, for each node, out of the locally obtained front of solutions, a set of weights, best approximating the currently specified energy vs performance and operation cost constraints, is selected and made active inside the node. Having weights set for all nodes a processing step is performed, the algorithm being restarted from the new execution point. The duration of a processing step is set to approximately 30 minutes (the system being run under the direction of the inference system with the determined weights) with an anticipation time of 6 hours.

Besides the difficulty of static multi-objective optimization, the resulting online dynamic case demands, as previously described, selecting a solution out of the Pareto set at fixed discrete time moments. The selected solution is used to advance the system to a new state (processing step illustrated in Fig. 3). As the intervention of a decision maker does not represent a feasible nor a practical solution, we propose the use of an approach inspired from the interactive EMO context. The classic way of handling the preferences of an user in the interactive evolutionary multi-objective context is by means of an *achievement scalarizing function*, initially proposed by Wierzbicki [34] and defined as follows:

$$\sigma(z, z^0, \lambda, \rho) = \max_{j=1, \dots, d} \left\{ \lambda_j (z_j - z_j^0) \right\} + \rho \sum_{j=1}^d \lambda_j (z_j - z_j^0)$$

where,

- $\sigma(\cdot, \cdot, \cdot, \cdot)$  is an application of  $Z$  into  $\mathbb{R}$ ;
- $z = (z_1, z_2, \dots, z_j, \dots, z_d)$  is an objective function vector;
- $z^0 = (z_1^0, z_2^0, \dots, z_j^0, \dots, z_d^0)$  is a reference point vector;
- $\lambda = (\lambda_1, \lambda_2, \dots, \lambda_j, \dots, \lambda_d)$  is a weighted vector;
- $\rho$ , is an arbitrary small positive number ( $0 < \rho \ll 1$ ).

These functions have as basis Chebyshev definitions and have the role of projecting a given reference point  $z^0 \in \mathbb{R}^d$  (feasible or infeasible solution) into the optimal Pareto set. Note that through the achievement scalarizing functions the multi-objective problem formulation together with the reference point coordinates are incorporated in a mono-objective optimization model. The reference point for the herein case is given by the coordinates of the ideal point and provides the localization of the focus region on the Pareto-optimal front, while the weights  $\lambda_i$ ,  $i \in \{1, \dots, d\}$  used in the scalarization process supply the localization of the Pareto solutions of interest in the aforementioned region.

A graphical illustration of an example execution is given in Fig. 6 (partial view of the nodes). All simulations have been carried inside Grid'5000<sup>2</sup> [9] using a parallel decentralized parallel NSGA-II algorithm over an average number of 300 cores. Note that energy vs performance and operation cost are identical for all nodes as an expression of system-wide constraints. Nodes start from an idle state with no tasks assigned, the first steps showing a massive offload towards the third and fourth depicted nodes. Once these nodes reach a high load level, tasks are accepted only at low rates, resulting a first equilibrium point (time line from 0.1 to 0.15). As frequency and voltage are simultaneously increased, in concordance with the demanded energy vs performance level, a load decrease is recorded (close to 0.2). Next, the ascending trend of the operation cost with a peak at 0.35, leads to lower performance states for most of the nodes, where from the escalation of node load indicators. The remaining time frame follows similar dynamics.

## 7 Conclusions

A general agent-based paradigm relying on fuzzy inference enabled nodes and a multi-objective evolutionary algorithm has been presented in this chapter, outlining several key points and open questions. The results provided by the algorithm show that it is possible to cope with a highly stochastic and dynamic environment. Nodes adapt not only to load specifications but also follow arbitrary energy vs performance and operation cost constraints. A large number of extensions are possible where concrete cases are analyzed independently, e.g. as for thermal-aware systems or as with market rules driven environments. Additional insight has to be gained with

---

<sup>2</sup> <https://www.grid5000.fr>

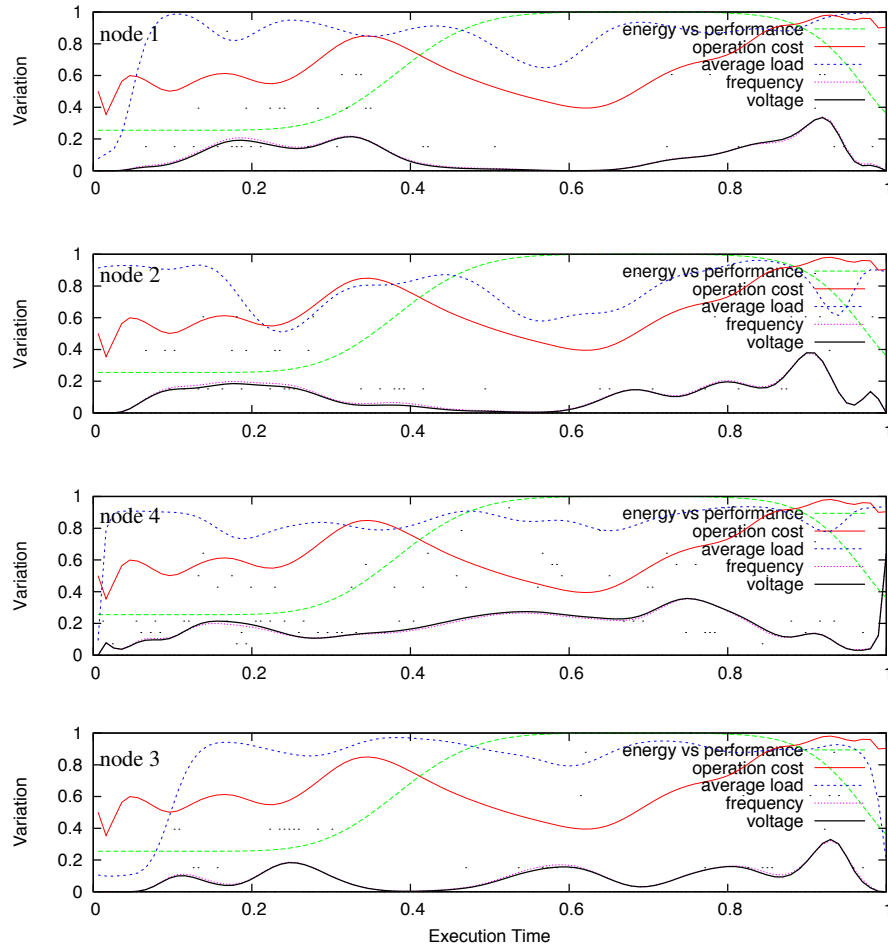


Fig. 6: Dynamic evolution of different fuzzy-enabled nodes (subset of the setup). For each node and for each processing step, weights are optimized by a dedicated, per-node multi-objective dynamic EA. Depending on the specific context, e.g. load, operation cost and energy vs performance levels, each node adapts its frequency, voltage and the acceptance or the offloading rates in order to assure an energy-efficient execution (outcome effects of varying exchange rates visible near 0.6). Please note that scenarios are determined by the operation cost, energy vs performance and the task arrival patterns, being generated and analyzed at execution time.

regard to how to exploit anticipation in an efficient, effective manner, as well as on how to select a strategy out of a given set of points, e.g. potentially using decision-making derived techniques.

## References

1. Armbrust, M., Fox, A., Griffith, R., Joseph, A.D., Katz, R.H., Konwinski, A., Lee, G., Patterson, D.A., Rabkin, A., Stoica, I., Zaharia, M.: Above the clouds: A Berkeley view of cloud computing. Tech. Rep. UCB/EECS-2009-28, EECS Department, University of California, Berkeley (2009)
2. Aydi, H., Mejía-Alvarez, P., Mossé, D., Melhem, R.: Dynamic and aggressive scheduling techniques for power-aware real-time systems. In: Proceedings of the 22nd IEEE Real-Time Systems Symposium, RTSS '01, pp. 95–. IEEE Computer Society, Washington, DC, USA (2001)
3. Barbagallo, D., Di Nitto, E., Dubois, D.J., Mirandola, R.: A bio-inspired algorithm for energy optimization in a self-organizing data center. In: Proceedings of the First international conference on Self-organizing architectures, SOAR'09, pp. 127–151. Springer-Verlag, Berlin, Heidelberg (2010)
4. Basney, J., Welch, V., Wilkins-Diehr, N.: TeraGrid science gateway AAAA model: implementation and lessons learned. In: Proceedings of the 2010 TeraGrid Conference, TG '10, pp. 2:1–2:6. ACM, New York, NY, USA (2010)
5. Berral, J.L., Goiri, I.n., Nou, R., Julià, F., Guitart, J., Gavaldà, R., Torres, J.: Towards energy-aware scheduling in data centers using machine learning. In: Proceedings of the 1st International Conference on Energy-Efficient Computing and Networking, e-Energy '10, pp. 215–224. ACM, New York, NY, USA (2010)
6. Bosman, P.A.N., Poutré, H.L.: Learning and anticipation in online dynamic optimization with evolutionary algorithms: the stochastic case. In: GECCO, pp. 1165–1172 (2007)
7. Branke, J.: Evolutionary Optimization in Dynamic Environments. Kluwer Academic Publishers, Norwell, MA, USA (2001)
8. Brown, D.J., Reams, C.: Toward energy-efficient computing. *Commun. ACM* **53**, 50–58 (2010)
9. Cappello, F., Caron, E., Dayde, M., Desprez, F., Jegou, Y., Primet, P., Jeannot, E., Lanteri, S., Leduc, J., Melab, N., Mornet, G., Namyst, R., Quetier, B., Richard, O.: Grid'5000: A Large Scale and Highly Reconfigurable Grid Experimental Testbed. In: Proceedings of IEEE/ACM International Workshop on Grid Computing (GRID'05), pp. 99–106. IEEE Computer Society, Washington, DC, USA (2005)
10. Das, R., Kephart, J.O., Lefurgy, C., Tesauro, G., Levine, D.W., Chan, H.: Autonomic multi-agent management of power and performance in data centers. In: Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems: industrial track, AAMAS '08, pp. 107–114. International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC (2008)
11. Das, R., Whalley, I., Kephart, J.O.: Utility-based collaboration among autonomous agents for resource allocation in data centers. In: Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems, AAMAS '06, pp. 1572–1579. ACM, New York, NY, USA (2006)
12. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE Transactions on Evolutionary Computation* **6**, 182–197 (2002)
13. Eiben, A., Schut, M.: New ways to calibrate evolutionary algorithms. In: Advances in Metaheuristics for Hard Optimization, Natural Computing Series, chap. 8, pp. 153–177. Springer (2008)
14. Flautner, K., Reinhardt, S., Mudge, T.: Automatic performance setting for dynamic voltage scaling. In: Proceedings of the 7th annual international conference on Mobile computing and networking, MobiCom '01, pp. 260–271. ACM, New York, NY, USA (2001)
15. Gropp, W.: Mpi at exascale: Challenges for data structures and algorithms. *Lecture Notes in Computer Science* **5759**, 3–642 (2009)
16. Hatzakis, I., Wallace, D.: Dynamic multi-objective optimization with evolutionary algorithms: a forward-looking approach. In: Proceedings of the 8th annual conference on Genetic and evolutionary computation, GECCO '06, pp. 1201–1208. ACM, New York, NY, USA (2006)

17. Herbert, S., Marculescu, D.: Analysis of dynamic voltage/frequency scaling in chip-multiprocessors. In: Proceedings of the 2007 international symposium on Low power electronics and design, ISLPED '07, pp. 38–43. ACM, New York, NY, USA (2007)
18. IBM: Fact sheet & background: Roadrunner smashes the petaflop barrier (2008)
19. IBM: 20 petaflop sequoia supercomputer (2009)
20. Intel: Enhanced Intel Speedstep Technology for the Intel Pentium M Processor, Intel White Paper (2004)
21. Kaplan, J., Forrest, W., Kindler, N.: Revolutionizing data center energy efficiency (2009)
22. Kaufmann, R.K.: The mechanisms for autonomous energy efficiency increases: A cointegration analysis of the us energy/gdp ratio. *The Energy Journal* **25**(1), 63–86 (2004)
23. Khargharia, B., Hariri, S., Yousif, M.S.: Autonomic power and performance management for computing systems. *Cluster Computing* **11**, 167–181 (2008)
24. Le Sueur, E., Heiser, G.: Dynamic voltage and frequency scaling: The laws of diminishing returns. In: Proceedings of the 2010 Workshop on Power Aware Computing and Systems (HotPower'10). Vancouver, Canada (2010)
25. Mearthur, S.D.J., Davidson, E.M., Catterson, V.M., Dimeas, A.L., Hatziaargyriou, N.D., Ponci, F., Funabashi, T.: Multi-Agent Systems for Power Engineering Applications - Part I: Concepts, Approaches, and Technical Challenges. *Power Systems, IEEE Transactions on* **22**(4), 1743–1752 (2007)
26. Mehnen, J., Wagner, T., Rudolph, G.: Evolutionary optimization of dynamic multiobjective functions. Tech. rep. (2006)
27. N. Bennani, M., A. Menasce, D.: Resource allocation for autonomic data centers using analytic performance models. In: Proceedings of the Second International Conference on Automatic Computing, pp. 229–240. IEEE Computer Society, Washington, DC, USA (2005)
28. Parkhill, D.F.: The challenge of the computer utility. Addison-Wesley Pub. Co. Reading, Mass., (1966)
29. Stantchev, V.: Performance evaluation of cloud computing offerings. In: Proceedings of the 2009 Third International Conference on Advanced Engineering Computing and Applications in Sciences, ADVCOMP '09, pp. 187–192. IEEE Computer Society, Washington, DC, USA (2009)
30. Stantchev, V., Schrpfer, C.: Negotiating and enforcing qos and slas in grid and cloud computing. In: N. Abdennadher, D. Petcu (eds.) *Advances in Grid and Pervasive Computing, Lecture Notes in Computer Science*, vol. 5529, pp. 25–35. Springer Berlin / Heidelberg (2009)
31. Sterling, T., Messina, P., Smith, P.H.: Enabling technologies for petaflops computing. MIT Press, Cambridge, MA, USA (1995)
32. Takagi, T., Sugeno, M.: Fuzzy Identification of Systems and Its Applications to Modeling and Control. *IEEE Transactions on Systems, Man, and Cybernetics* **15**(1), 116–132 (1985)
33. White, R., Abels, T.: Energy resource management in the virtual data center. In: Proceedings of the International Symposium on Electronics and the Environment, pp. 112–116. IEEE Computer Society, Washington, DC, USA (2004)
34. Wierzbicki, A.: The use of reference objectives in multiobjective optimization. *Lecture Notes in Economics and Mathematical Systems* **177**, 468–486 (1980)