Parallel Evolutionary Algorithms for Energy Aware Scheduling

Yacine Kessaci, Mohand Mezmaz, Nouredine Melab, E.G. Talbi, Daniel Tuyttens

Parallel Evolutionary Algorithms for Energy Aware Scheduling

Y. Kessaci, M. Mezmaz, N. Melab, E-G. Talbi and D. Tuyttens

Abstract

Reducing energy consumption is an increasingly important issue in computing and embedded systems. In computing systems, minimizing energy consumption can significantly reduces the amount of energy bills. The demand for computing systems steadily increases and the cost of energy continues to rise. In embedded systems, reducing the use of energy allows to extend the autonomy of these systems. In addition, the reduction of energy decreases greenhouse gas emissions. Therefore, many researches are carried out to develop new methods in order to consume less energy. This chapter gives an overview of the main methods used to reduce the energy consumption in computing and embedded systems.

As a use case and to give an example of a method, the chapter describes our new parallel bi-objective hybrid genetic algorithm that takes into account the completion time and the energy consumption. In terms of energy consumption, the obtained results show that our approach outperforms previous scheduling methods by a significant margin. In terms of completion time, the obtained schedules are also shorter than those of other algorithms.

Yacine Kessaci
INRIA Lille-Nord Europe. Parc Scientifique de la Haute Borne 40, avenue Halley Bât.A, Park Plaza 59650, Villeneuve d'Ascq, France. e-mail: yacine.kessaci@inria.fr

Mohand Mezmaz
UMons/FPMs, 9, rue de Houdain 7000 Mons, Belgium. e-mail: Mohand.Mezmaz@umons.ac.be

Nouredine Melab
INRIA Lille-Nord Europe. Parc Scientifique de la Haute Borne 40, avenue Halley Bât.A, Park Plaza 59650, Villeneuve d'Ascq, France. e-mail: nouredine.melab@lifl.fr

El-Ghazali Talbi
INRIA Lille-Nord Europe. Parc Scientifique de la Haute Borne 40, avenue Halley Bât.A, Park Plaza 59650, Villeneuve d'Ascq, France. e-mail: talbi@lifl.fr

Daniel Tuyttens
UMons/FPMs, 9, rue de Houdain 7000 Mons, Belgium. e-mail: Daniel.Tuyttens@umons.ac.be

1 Introduction

Computers use a significant and growing portion of the energy in the world. Therefore, energy-aware computing is crucial for large-scale systems that consume considerable amount of energy and embedded systems that utilize battery for their power. A recent study on power consumption by servers [13] shows that, in 2005, the power used by servers represented about 0.6% of total U.S. electricity consumption. That number grows to 1.2% when cooling and auxiliary infrastructures are included. In the same year, the aggregate electricity bill for operating those servers and associated infrastructure was about $2.7 billions and $7.2 billions for the U.S. and the world, respectively. The total electricity use for servers doubled over the period 2000 to 2005 in worldwide. The number of transistors integrated into today's Intel Itanium 2 processor reaches nearly 1 billion. If this rate continues, the heat (per square centimeter) produced by future Intel processors would exceed that of the surface of the sun [12].

In this chapter, we present some important works done in the literature to reduce energy consumption. These works are classified according to three criteria. The first criterion concerns the optimization method used to minimize the consumed energy. This method can be mono-objective or multi-objective approach. The second criterion concerns the level of the system on which an approach is based. Indeed, a method can be based on the hardware or the software part of a system. The third and last criterion concerns the type of system to which the approach is intended to be used. An approach can be developed for computing systems or embedded systems.

In order to give an example of a method and as a use case, we describe our new parallel bi-objective hybrid genetic algorithm that takes into account, not only makespan, but also energy consumption. Our new approach is a hybrid between a multi-objective parallel genetic algorithm and energy-conscious scheduling heuristic (ECS) [15]. The results clearly demonstrate the superior performance of ECS over the other algorithms like DBUS [2] and HEFT [25]. Genetic algorithms make it possible to explore a great range of potential solutions to a problem. The exploration capability of the genetic algorithm and the intensification power of ECS are complementary. A skillful combination of a metaheuristic with concepts originating from other types of algorithms lead to more efficient behavior. Our algorithm is effective as it profits from the exploration power of the genetic algorithm, the intensification capability of ECS, the cooperative approach of the island model, and the parallelism of the multi-start model. The island model and the hybridization improve the quality of the obtained results. The multi-start model reduces the running time of a resolution.

The remainder of the chapter is organized as follows. Section 2 gives an overview of our classification. Section 3 explains the classification according to the optimization method used to minimize the consumed energy. Section 4 describes the classification according to the level of the system on which an approach is based. Section 5 explains the classification according to the type of system to which the approach is intended to be used. Section 6 presents the application, system, energy and schedul-

ing models used in our use case. Our algorithm is presented in Section 7. The results of our comparative experimental study are discussed in Section 8. The conclusion is drawn in Section 9.

2 Energy aware approaches

This section presents an overview of the main methods published in the literature to reduce energy consumption. Our state of the art does not address the techniques used by computer system manufacturers. Indeed, manufacturers use some techniques in the design of electronic circuits and components to reduce their energy consumption. Our classification only focuses on methods intended to be exploited by the users of computing and embedded systems. As shown in Fig. 1, these methods can be classified according to three criteria.

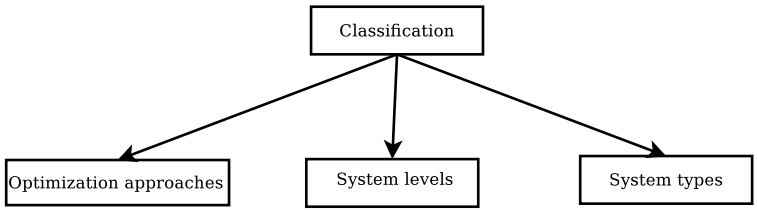

Fig. 1 The classification criteria used

The first criterion concerns the type of the used optimization. This optimization can be mono-objective or multi-objective. In multi-objective optimization, the main approaches found in the literature are based on an aggregation, a lexicographic or a Pareto fitness function.

The system level is the second criterion on which methods can be classified. To reduce energy consumption, the methods are based either on software or hardware part of a system. In the hardware level, the techniques used are generally virualisation and consolidation based approaches. While in the software level, two techniques can be identified to address the task scheduling problem: Static methods where scheduling is done before the execution of a program, and dynamic methods, where the appropriate scheduling is calculated during the execution of a program.

The third and last criterion used in our classification is the type of system to which the approach is intended to be used. In this criterion, a system can be either a computing or an embedded system. Reducing energy consumption in embedded systems aims to increase the autonomy of devices. While the goal of reducing energy consumption in computing systems is to decrease the cost of the energy and the greenhouse gas emissions. Table 2 gives some examples of approaches with their classification using our taxonomy.

Method	Level	System	Optimization
DPM&DVS[23]	Hardware(Dyn.)	E.S.	Mono-objective
DynamicAssgn[10]	Hardware(Dyn.)	C.S.(distributed comp.)	Multi-objective(Lex.)
ECS[15]	Hardware(Dyn.)	C.S.(high-performance comp.)	Multi-objective(Agg.)
ECTC and MaxUtil[14]	Software(Cons.)	C.S.(cloud comp.)	Mono-objective
Jitter[6]	Hardware(Dyn.)	C.S.(high-performance comp.)	qyut Multi-objective(Lex.)
MMF-DVFS[21]	Hardware(Stat.)	C.S.(distributed comp.)	Multi-objective(Lex.)
MO[1]	Software(Virt.)	C.S.(cloud comp.)	Multi-objective(Lex.)
PSAGA/PGA[16]	Hardware(Dyn.)	E.S.	Mono-objective
TDVAS[22]	Hardware(Stat.)	C.S.(cluster comp.)	Multi-objective(Lex.)

Table 1 Examples of approaches with their classification

3 Optimization approaches

An approach used to optimize the energy consumption can be mono-objective or multiobjective. The mono-objective approaches, such as [8, 14, 16, 17, 20, 23, 26], only aim to optimize the energy consumption with assuming some constraints, especially on the completion time. In addition to the consumed energy, multi-objective methods are also trying to optimize other objectives such as quality of service (QoS). QoS is usually expressed in terms of makespan. The energy consumption depends heavily on the QoS offered. Indeed, the minimization of energy consumption and the maximizing the QoS are conflicting objectives. The goal of a multi-objective optimization method is to find a good compromise between these two objectives. In addition, the cost of a service can be another objective to consider. More the QoS and energy consumption increase, more their cost increases. Table 3 gives some examples of the optimized objectives in different methods found in literature.

Method	Optimization	Criteria
DPM&DVS[23]	Mono-objective	Energy
DynamicAssgn[10]	Multi-objective(Lex.)	Energy and completion time
ECS[15]	Multi-objective(Agg.)	Energy and completion time
ECTC and MaxUtil[14]	Mono-objective	Energy
Jitter[6]	Multi-objective(Lex.)	Energy and completion time
MMF-DVFS[21]	Multi-objective(Lex.)	Energy and completion time
MO[1]	Multi-objective(Lex.)	Energy and QoS
PSAGA/PGA[16]	Mono-objective	Energy
TDVAS[22]	Multi-objective(Lex.)	Energy and completion time

Table 2 Examples of the optimized objectives in different methods

As shown in Fig. 2, it is possible to identify three main categories according to the multi-objective approach chosen: Aggregation, lexicographic and Pareto approaches.

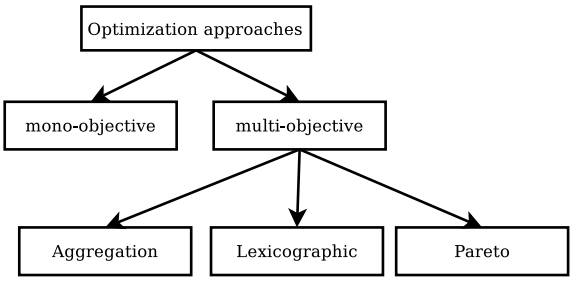

Fig. 2 Classification according to the optimization approaches

3.1 Aggregation approach

The aggregation (or weighted) method is one of the first and most used methods for generation of Pareto optimal solutions. It consists in using an aggregation function to transform a multi-objective problem into a mono-objective problem by combining the various objective functions into a single objective function generally in a linear way. The obtained results in the resolution of the problem depend strongly on the parameters chosen for the weight vector.

In [15], the authors address the task scheduling problem on heterogeneous computing systems (HCSs) and propose an energy-conscious scheduling heuristic (ECS) that takes into account the completion time and energy consumption. The heuristic proposed in [15] tries to balance these two performance goals using a novel objective function called relative superiority (RS).

3.2 Lexicographic approach

In this traditional approach, the search is carried out according to a given preference order of the objectives. This order defines the significance level of the objectives. Then, a set of mono-objective problems are solved in a sequential manner. If the problem associated with the most significant objective function has a unique solution, the search provides the optimal solution and stops. Otherwise, the problem associated with the second most significant objective function is solved. The same stopping criteria and process are iterated until the treatment of the last function.

In [21], the authors investigate the task scheduling problem, and propose a heuristic called maximum minimum frequency DVFS (MMF-DVFS). The proposed method operates in two stages. The goal of the first stage is to find a schedule of tasks that minimizes the makespan. The second stage tries to find the right setting of the processor to minimize energy consumption without changing the makespan of the schedule provided by the first stage.

3.3 Pareto approach

The Pareto approaches use the concept of dominance in the fitness assignment, contrary to the other approaches that use a scalarization function or treat the various objectives separately. The main advantage of the Pareto approaches is that they do not need the transformation of the multi-objective problem into a mono-objective problem. In a single run, they are able to generate a diverse set of Pareto solutions in the concave portions of the convex hull of feasible objective space.

For example, [18] and [24] are one of the methods that use a Pareto approach. In [18], the authors propose a multi-objective fuzzy genetic algorithm to optimize the energy saving scheduling tasks on heterogeneous chip multi-processor (CMP) system. In this algorithm, the Pareto set includes chromosomes with the shortest task execution time and the lowest system energy. In [24], the authors study the inter-relationships between energy consumption, resource utilization, and processor utilization.

4 System levels

To reduce energy consumption, various issues such as resource management in both software and hardware must be addressed. The system level (i.e. hardware or software) used by a method is another criterion on which methods can be classified. Software approaches are mainly based on virtualization or task consolidation. Virtualization consists to run on a single computer multiple operating systems as if they are running on separate computers. Hardware approaches use the opportunity offered by manufacturers in modern processors to adjust the voltage and frequency. This adjustment can vary the processor performance and thus its energy cost.

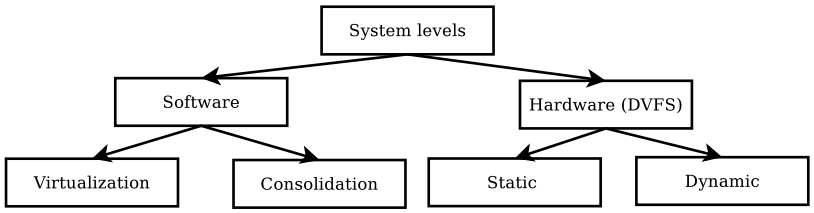

Fig. 3 Classification according to the system levels

4.1 Hardware level

Dynamic voltage frequency scaling (DVS) is a powerful management technique in computer architecture. This technique enables processors to dynamically increase or decrease voltage supply levels (VSLs). DVS to decrease voltage is known as under-volting, and DVS to increase voltage is known as over-volting. Under-volting is done in order to conserve power, and over-volting is done in order to increase computer performance. Dynamic frequency scaling (DFS) is another powerful conservation technique that works on the same principles as DVS. This technique reduces the number of instructions a processor can issue in a given amount of time, thus reducing performance. Voltage and frequency scaling are often used together to save power. When used in this way it is commonly known as dynamic voltage and frequency scaling (DVFS). Many methods, such as [5, 6, 10, 11, 15, 16, 18, 21, 22, 23, 26], try to exploit this technique to reduce energy consumption.

In [6], the authors present a system called Jitter, which reduces the frequency on nodes that are assigned less computation and therefore have slack time. This saves energy on these nodes, and the goal of Jitter is to attempt to ensure that they arrive just in time so that they avoid increasing overall execution time.

[22] proposes an energy-efficient scheduling algorithm (TDVAS) using the dynamic voltage scaling technique to provide significant energy savings for clusters. The TDVAS algorithm aims at judiciously leveraging processor idle times to lower processor voltages (i.e., the dynamic voltage scaling technique or DVS), thereby reducing energy consumption experienced by parallel applications running on clusters. Reducing processor voltages, however, can inevitably lead to increased execution times of parallel task. The salient feature of the TDVAS algorithm is to tackle this problem by exploiting tasks precedence constraints.

In [11], the authors proposed a new algorithm that reduces energy consumption in a parallel program executed on a power-scalable cluster using DVFS. Whenever the computational load is not balanced, parallel programs encounter slack time, that is, they must wait for synchronization of the tasks. This algorithm reclaims slack time by changing the voltage and frequency, which allows a reduction in energy consumption without impacting on the performance of the program.

[10] also presents a novel dynamic algorithm for remapping tasks for energy efficient scheduling of DAG based applications for DVS enabled systems.

There are a lot of work on task scheduling problem using the dynamic voltage and frequency scaling technique in heterogeneous computing and real-time embedded systems. As previously mentioned, the main idea of the methods proposed in the literature is to change the voltage level to reduce energy consumption. These voltage level changes are made by taking into account timing information of processors, such as idle or slack time, and timing information of tasks, such as task deadline, task release time and task execution time. According to the availability of timing information of tasks, it is possible to distinguish two categories of methods to solve an energy-aware task scheduling problem: Static scheduling and dynamic scheduling.

In static scheduling [5, 18, 21, 22, 26], timing information of tasks and timing information of processors are available in compile time before the deployment of tasks. This static scheduling involves most of the large-scale computational problems such as object recognition in machine vision applications, chemistry and bioinformatics. Having this information allows static schedulers to be developed by maximizing processor utilization with meeting all timing information of tasks.

In dynamic scheduling [6, 10, 11, 15, 16, 23], also called online or real-time scheduling, the deadlines of tasks are available in compile time but their release and execution times should be estimated during run time. This dynamic scheduling not only involves dynamic large-scale approximation and optimization such as in weather forecasting and search algorithms but also is used in most of power-aware devices like laptops, wireless sensors and cell phones.

4.2 Software level

In software level, two main techniques are used to reduce energy consumption: Virtualization and task consolidation. Operating system-level virtualization is a server method where the operating system allows for multiple isolated user-space instances. Virtualization can be seen as splitting an underlying hardware entity into smaller identical virtual entities running isolated from each other. Virtualization allows to reduce energy consumption

In [8], an architecture for virtualizing and sharing hardware resources in future home environments is presented. The architecture aims at utilizing existing home resources in such a way that the consumed energy is minimized and the energy is efficiently used. A fully decentralized management system is proposed, interconnecting possibly thousands of homes in a peer-to-peer like manner. Energy optimization is done in a decentralized way by converging to a global energy optimum based on energy and performance metrics that have been defined. [1] investigates the energy consumption in office environments and discusses the potential of energy savings. An energy-efficient office management approach is suggested, based on resource virtualization, power management, and resource sharing.

Like virtualization, task consolidation is an effective method to increase resource utilization and in turn reduces energy consumption. The task consolidation problem is the process of assigning a set N of n tasks (service requests or simply services) to a set R of r resources without violating time constraints. This technique aims to maximize resource utilization by minimizing energy consumption. Indeed, a resource allocation strategy that takes into account resource utilization would lead to better energy efficiency.

[24] exposes some of the complexities in performing consolidation for power optimization, and proposes some research directions to address the challenges involved. In [14], the authors present two energy-conscious task consolidation heuristics. These two heuristics aim to maximize resource utilization and explicitly take into account both active and idle energy consumption. The proposed heuristics as-

sign each task to the resource on which the energy consumption for executing the task is explicitly or implicitly minimized without the performance degradation of that task.

5 System types

As shown in Fig. 4, two types of systems can be identified: Embedded systems and computing systems. The issue of reducing energy consumption interests these two types of systems for different reasons. In embedded systems, the main goal of reducing energy consumption is to increase the autonomy of devices, while in computing systems, the main goal is to reduce the energy cost.

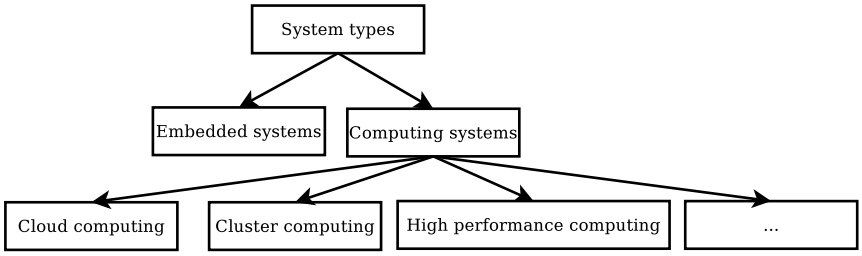

Fig. 4 Classification according to the system types

5.1 Embedded systems

An embedded system can be defined as an electronic and autonomous system which is dedicated to a specific task. The resources of an embedded system are limited. This limitation is usually spatial (limited size) and energy (consumption restricted). Many of today's embedded systems, such as wireless and portable devices rely heavily on the limited power supply. Therefore, energy efficiency becomes one of the major design concerns for embedded systems. Many methods, such as [5, 16, 23, 26], are developed to reduce the energy consumption of embedded systems.

In [5], some scheduling heuristics are presented that determine the best trade-off between three techniques: DVS, processor shutdown, and finding the optimal number of processors. Indeed, when peak performance is unnecessary, DVS can be used to reduce the dynamic power consumption of embedded multiprocessors. However, static power consumption is expected to increase significantly. Then it will be more effective to limit the number of employed processors, and use a combination of DVS and processor shutdown.

[23] presents an optimal approach based on the time-indexed semi-Maxkov decision processes (TISMDP) for optimizing power management policies of embedded systems.

In [16], the authors developed a genetic algorithm for the DVS scheduling problem. They also describe a Parallel genetic algorithm for finding better schedules with less time by parallelizing the genetic algorithm. A hybrid parallel algorithm is also developed to further improve the search ability of parallel genetic algorithm by combining the parallel genetic algorithm with the technique of simulated annealing.

5.2 Computing systems

Demand for cloud computing, high performance computing, cluster computing, etc. is growing. The industry is responding to the demand by more powerful systems, and consequently, creating highly energy-intensive products. The energy consumption of these systems is associated with various environmental, system performance and monetary issues.

Scheduling parallel applications on computing systems, especially on large-scale computing systems, is challenging due to high energy consumption and significant communication latencies. Therefore, conserving energy consumption and reducing schedule lengths are two important concerns in the design of economical and environmentally friendly systems. It is only recently that much attention has been paid to energy consumption in scheduling. [10] and [21] addresse the task scheduling problem on distributed computing systems. [15] and [18] investigate the same problem on respectively heterogeneous computing systems and chip multi-processor systems. With the chip multi-processor being more and more widespread used in the laptop, desktop and data center area, the power-performance scheduling issues are becoming challenges. In [22], the authors address the scheduling parallel applications on large scale clusters. In the past decade cluster computing platforms have been widely applied to support a variety of scientific applications. [17] proposes an energy efficient algorithm to schedule real-time tasks with data access requirements on grids. Taking into account both data locations and application properties, the authors design a distributed energy-efficient scheduler that aims to seamlessly integrate the process of scheduling tasks with data placement strategies to provide energy savings.

Consolidation of applications in cloud computing environments presents an important approach to streamline resource usage and improve energy efficiency. Based on the fact that resource utilization directly relates to energy consumption, the authors in [14] have modeled their relationship and developed two energy-conscious task consolidation heuristics. [24] outlines the challenges in finding effective solutions to the consolidation problem on cloud computing systems. In [8], an architecture for sharing computing resources among home environments in a peer-to-peer manner is proposed in order to improve the energy efficiency. [20] explores how to integrate power management mechanisms and policies with the virtualization tech-

nologies being actively deployed in large-scale datacenters to address costs and limitations in cooling or power delivery.

6 Problem modeling

In this section, we describe the system, application, energy and scheduling models used in our study.

6.1 System model

The target system used in this work consists of a set P of p heterogeneous processors/machines that are fully interconnected. Each processor $p_j \in P$ is DVS-enabled; in other words, it can operate with different VSLs (i.e., different clock frequencies). For each processor $p_j \in P$, a set V_j of v VSLs is random and uniformly distributed among three different sets of VSLs (Table 3). Since clock frequency transition overheads take a negligible amount of time (e.g., $10\mu s$- $150\mu s$ [9], [19]), these overheads are not considered in our study. The inter-processor communications are assumed to perform with the same speed on all links without contentions. It is also assumed that a message can be transmitted from one processor to another while a task is being executed on the recipient processor which is possible in many systems.

	Pair 1		Pair 2		Pair 3	
Level	Voltage (v_k)	Relative speed (%)	Voltage (v_k)	Relative speed (%)	Voltage (v_k)	Relative speed (%)
0	1.5	100	2.2	100	1.75	100
1	1.4	90	1.9	85	1.4	80
2	1.3	80	1.6	65	1.2	60
3	1.2	70	1.3	50	0.9	40
4	1.1	60	1.0	35		
5	1.0	50				
6	0.9	40				

Table 3 Voltage-relative speed pairs

6.2 Application model

Parallel programs can be generally represented by a directed acyclic graph (DAG). A DAG, $G = (N, E)$, consists of a set N of n nodes and a set E of e edges. A DAG is also called a task graph or macro-dataflow graph. In general, the nodes represent tasks partitioned from an application; the edges represent precedence constraints. An edge $(i, j) \in E$ between task n_i and task n_j also represents inter-task communication. A task with no predecessors is called an entry task, n_{entry}, whereas an exit task, n_{exit}, is one that does not have any successors. Among the predecessors of a task n_i, the predecessor which completes the communication at the latest time is called the most influential parent (MIP) of the task denoted as $MIP(n_i)$. The longest path of a task graph is the critical path.

The weight on a task n_i denoted as w_i represents the computation cost of the task. In addition, the computation cost of the task on a processor p_j, is denoted as $w_{i,j}$ and its average computation cost is denoted as $\bar{w_i}$.

The weight on an edge, denoted as $c_{i,j}$ represents the communication cost between two tasks, n_i and n_j. However, a communication cost is only required when two tasks are assigned to different processors. In other words, the communication cost when tasks are assigned to the same processor is zero and thus can be ignored.

The earliest start time of, and the earliest finish time of, a task n_i on a processor p_j is defined as

$$EST(n_i, p_j) = \begin{cases} 0 & if n_i = n_{entry} \\ EFT(MIP(n_i), p_k) + c_{MIP(n_i),i} & otherwise \end{cases} \quad (1)$$

$$EFT(n_i, p_j) = EST(n_i, p_j) + w_{i,j} \quad (2)$$

Note that the actual start and finish times of a task n_i on a processor p_j, denoted as $AST(n_i, p_j)$ and $AFT(n_i, p_j)$ can be different from its earliest start and finish times, $EST(n_i, p_j)$ and $EFT(n_i, p_j)$, if the actual finish time of another task scheduled on the same processor is later than $EST(n_i, p_j)$.

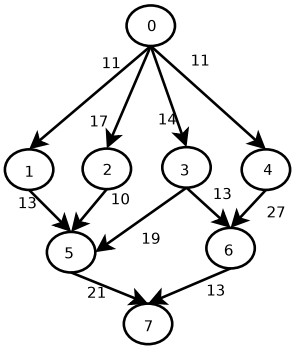

Fig. 5 A simple task graph

In the case of adopting task insertion the task can be scheduled in the idle time slot between two consecutive tasks already assigned to the processor as long as no violation of precedence constraints is made. This insertion scheme would contribute in particular to increasing processor utilization for a communication intensive task graph with fine-grain tasks.

A simple task graph is shown in Fig. 5 with its details in Table 4 and Table 5. The values presented in Table 5 are computed using two frequently used task prioritization methods, t-level and b-level. Note that, both computation and communication costs are averaged over all nodes and links. The t-level of a task is defined as the summation of the computation and communication costs along the longest path of the node from the entry task in the task graph. The task itself is excluded from the computation. In contrast, the b-level of a task is computed by adding the computation and communication costs along the longest path of the task from the exit task in the task graph (including the task). The b-level is used in this study.

The communication to computation ratio (CCR) is a measure that indicates whether a task graph is communication intensive, computation intensive or moderate. For a given task graph, it is computed by the average communication cost divided by the average computation cost on a target system.

Task	p_0	p_1	p_2
0	11	13	9
1	10	15	11
2	9	12	14
3	11	16	10
4	15	11	19
5	12	9	5
6	10	14	13
7	11	15	10

Table 4 Computation cost with VSL 0

Task	b-level	t-level
0	101.33	0.00
1	66.67	22.00
2	63.33	28.00
3	73.00	25.00
4	79.33	22.00
5	41.67	56.33
6	37.33	64.00
7	12.00	89.33

Table 5 Task Priorities

6.3 Energy model

Our energy model is derived from the power consumption model in complementary metal-oxide semiconductor (CMOS) logic circuits. The power consumption of a CMOS-based microprocessor is defined to be the summation of capacitive, short-circuit and leakage power. The capacitive power (dynamic power dissipation) is the most significant factor of the power consumption. The capacitive power (P_c) is defined as

$$P_c = ACV^2 f \tag{3}$$

where A is the number of switches per clock cycle, C is the total capacitance load, V is the supply voltage, and f is the frequency. Equation (3) clearly indicates that the supply voltage is the dominant factor; therefore, its reduction would be most influential to lower power consumption. The energy consumption of the execution of a precedence-constrained parallel application used in this study is defined as

$$E = \sum_{i=0}^{n} ACV_i^2 f.w_i^* = \sum_{i=0}^{n} \alpha V_i^2 w_i^* \tag{4}$$

where V_i is the supply voltage of the processor on which task n_i is executed, and w_i^* is the computation cost of task n_i (the amount of time taken for n_i's execution) on the scheduled processor.

6.4 Scheduling model

The task scheduling problem in this study is the process of allocating a set N of n tasks to a set P of p processors (without violating precedence constraints) that minimizes makespan with energy consumption as low as possible. The makespan is defined as $M = max\{AFT(n_{exit})\}$ after the scheduling of n tasks in a task graph G is completed. Although the minimization of makespan is crucial, tasks of a DAG in our study are not associated with deadlines as in real-time systems.

7 A parallel evolutionary algorithm

In this section, our new parallel bi-objective hybrid approach is presented.

7.1 Hybrid approach

In our approach illustrated in Fig. 6, a solution (chromosome) is composed of a sequence of N genes. The i^{th} gene of a solution s is denoted s_j. Each gene is defined

by a task, a processor and a voltage. These three parts of s_j are denoted respectively $t(s_j)$, $p(s_j)$ and $v(s_j)$. This means that the task $t(s_j)$ is assigned to the processor $p(s_j)$ with the voltage $v(s_j)$.

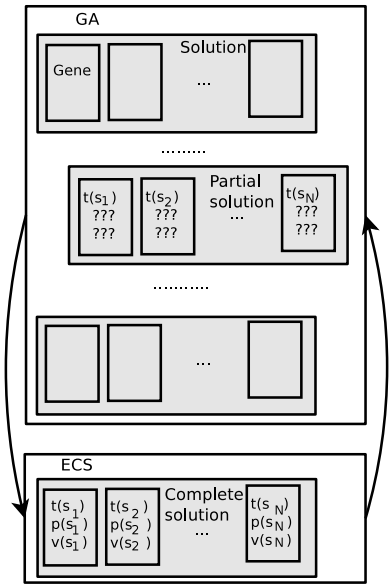

Fig. 6 Our hybrid GA (GA and ECS)

The new approach we propose is based on ECS which is not a population-based heuristic. ECS tries to construct in a greedy way one solution using three components.

- A first component to build the task parts of each gene of the solution.
- A second component to build the processor and voltage parts of these genes.
- And a third component to calculate the fitness of a solution in terms of energy consumption and makespan.

Unlike ECS, our approach provides a set of Pareto solutions. This approach is a hybrid between a multi-objective GA and the second component of ECS. The role of the GA is to provide good task scheduling. In other words, the GA builds task parts $t(s_1)$, $t(s_2)$, ..., $t(s_n)$ of a solution s. Therefore, the mutation and crossover operators of the GA affect only the task part of the genes of each solution.

The second component of ECS is called whenever a solution is modified by these two operators. The first role of this component is to correct the task order to take into account the precedence constraints in the task graph. Then the component completes the processor and voltage parts of the genes of the partial solutions provided by these operators. In other words, ECS builds the remaining parts $p(s_1)$, $p(s_2)$, ..., $p(s_n)$ and

$v(s_1)$, $v(s_2)$, ..., $v(s_n)$ of the partial solutions provided by the mutation and crossover operators of the GA.

The evaluation (fitness) operator of the GA is called once the task, processor and voltage parts of each gene of the solution are known. The role of this operator is to calculate the energy consumption and the makespan of each solution.

The mutation operator is based on the first component of ECS. This first component returns all tasks scheduled according to their b-level values. The principle of our mutation operator is also based on the scheduling of tasks according to their b-level values. The b-level concept is explained in Section 6. It should be noted that one can choose the t-level values instead of those of b-level. First, the operator chooses randomly two integers i and j such that $1 \leq i < j \leq n$ and $b-level(t(s_i)) < b-level(t(s_j))$. Then, the operator swaps the two tasks $t(s_i)$ and $t(s_j)$ (see Fig. 7).

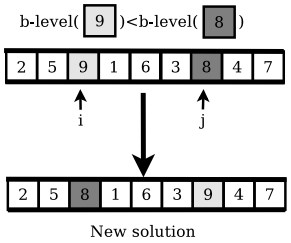

Fig. 7 The mutation operator

As illustrated in Fig. 8, the crossover operator uses two solutions $s1$ and $s2$ to generate two new solutions $s'1$ and $s'2$. To generate $s'1$, the operator:

- considers $s1$ as the first parent and $s2$ as the second parent.
- randomly selects two integers i and j such that $1 \leq i < j \leq N$.
- copies in $s'1$ all tasks of $s1$ located before i or after j. These tasks are copied according to their positions ($s'1_k = s1_k$ if $k < i$ or $k > j$).
- copies in a solution s all tasks of $s2$ that are not yet in $s'1$. Thus, the new solution s contains $(j-i+1)$ tasks. The first task is at position 1 and the last task at the position $(j-i+1)$.
- and finally, copies all the tasks of s to the positions of $s'1$ located between i and j ($s'1_k = s_{k-i+1}$ for all $i \leq k \leq j$).

The solution $s'2$ is generated with the same method by considering $s2$ as the first parent and $s1$ as the second parent.

The other elements of the GA in the new approach are conventional. Indeed, our GA randomly generates the initial population. Its selection operator is based on a tournament strategy. The algorithm stops when no new best solution is found after a fixed number of generations.

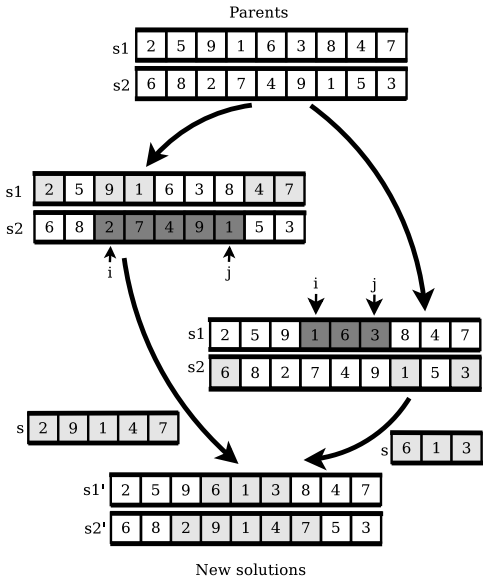

Fig. 8 The crossover operator

7.2 Insular approach

The island model [3] is inspired by behaviors observed in the ecological niches. In this model, several evolutionary algorithms are deployed to evolve simultaneously various populations of solutions, often called islands. As shown in Fig. 9, the GAs of our hybrid approach asynchronously exchange solutions. This exchange aims at delaying the convergence of the evolutionary process and to explore more zones in the solution space. For each island, a migration operator intervenes at the end of each generation. Its role consists to decide the appropriateness of operating a migration, to select the population sender of immigrants or the receiver of emigrants, to choose the emigrating solutions, and to integrate the immigrant ones.

7.3 Multi-start approach

Compared to the GA, ECS is more costly in CPU time. The different evaluations of ECS are independent of each other. Therefore, their parallel execution can make the approach faster. The objective of the hybrid approach is to improve the quality of solutions. The island approach also aims to obtain solutions of better quality. The goal of the parallel multi-start approach is to reduce the execution time. As shown in Fig. 10, our parallelization is based on the deployment of the approach using

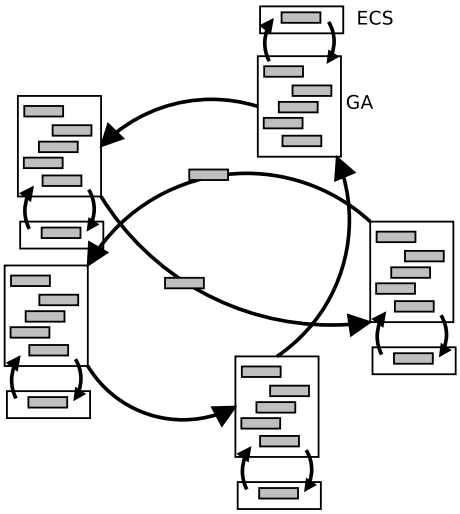

Fig. 9 The cooperative island approach

the farmer-worker paradigm. The GA processes are farmers and ECS processes are workers.

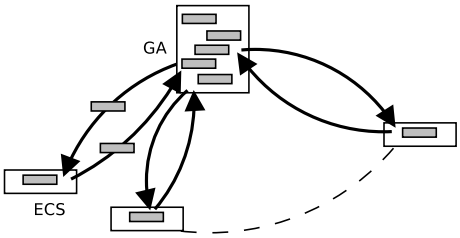

Fig. 10 Illustration of the multi-start approach

8 Experiments and results

This section presents the results obtained from our comparative experimental study. The experiments aim to demonstrate and evaluate the contribution of the hybridization, the insular approach and the multi-start approach respectively compared to ECS, the hybrid approach and the insular approach.

8.1 Experimental settings

The performance of our approach was thoroughly evaluated with the Fast Fourier Transformation [4] task graph which is a real-world application. A large number of variations were made on this task graph for more comprehensive experiments. Various different characteristics of processors were also applied. Table 6 summarizes the parameters used in our experiments.

Parameter	Value
The number of tasks	$\sim$20 $\sim$40 $\sim$60 $\sim$80 $\sim$120
The number of processors	02 04 08 16 32 64
Processor heterogeneity	100 200 random
CCR	0.1 0.2 1.0 5.0 10.0

Table 6 Experimental parameters

The new approach is experimented on about 10,000 instances distributed equitably according to the number of tasks, the number of processors, the processor heterogeneity and the CCR ($1/5$ of instances have a number of tasks equal to $\sim$20, $1/5$ of instances have a number of tasks equal to $\sim$40,..., $1/6$ of instances have a number of processors equal to 2, etc.).

Experiments have been performed on a grid of three clusters. A total of 714 cores are used. The first two clusters are located at the University of Mons in Belgium, while the third cluster is at Université de Lille1 in France.

8.2 Hybrid approach

The hybrid approach is experimented on all instances of Table 6. Each instance is solved twice. The first resolution is done with ECS, and the second resolution with the new approach. These experiments are launched by a script on one of the cores of our grid according to their availability.

Experiments show that our approach improves on average the results obtained by ECS. Indeed, the energy consumption is reduced by **47.49%** and the makespan reduced by of **12.05%**. In addition, our experiments show clearly that the more processors there are, the more the new approach improves the results of ECS.

8.3 Insular approach

The objective of the following experiments is to show that our island approach improves the quality of the solutions provided by the hybrid approach. This insular

approach is useful when solving large instances. Therefore, the experiments, presented in this section, focus only on the large instances of Table 6. The instances used are those with the number of tasks is 120, the number of processors is 64, the value of CCR is 10, and the heterogeneity of processors is 200 (20 instances). Each instance is solved using 1, 5, 10, 30 or 50 islands. An insular approach with 1 island is equivalent to the hybrid approach.

Fig. 11 illustrates the S-metric average values obtained with different numbers of islands. These values are normalized with the average value obtained by the experiments using 1 island. The S-metric measures the hyper-volume defined by a reference point and a Pareto front. It allows to evaluate the quality of a Pareto front provided by an algorithm.

Experiments show that whatever the number of used islands the insular approach improves the Pareto front obtained with the hybrid approach. As shown in Fig. 11, the use of 50 islands, instead of 1 island (i.e. the hybrid approach), improves the S-metric of the obtained Pareto front by **26%**. In Fig. 11, the more the number of islands is used, the better the results will be.

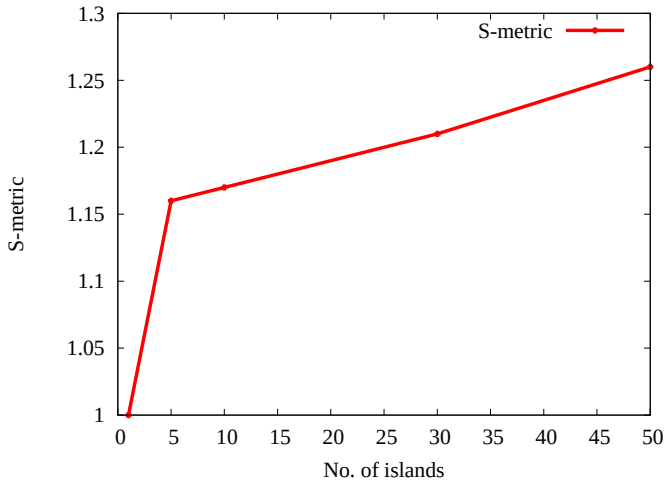

Fig. 11 S-metric value according to the number of islands

8.4 Multi-start approach

This section presents the experiments done to assess the quality of our multi-start approach. The parameters of the instances used in our experiments are: The CCR is 0.1, 0.5, 1.0, 5.0 or 10.0, the number of processors is 8, 32, or 64, and the heterogeneity of processors is 100, 200 or random. The population of the GA contains 20

chromosomes. Therefore, 21 computing cores are used to solve each instance (20 cores to run the ECSs and 1 core to run the GA). In our case, an experiment can not have a speedup greater than 21.

The average speedup obtained is 13.06. Our experiments show that the speedup increases proportionally to the number of processors on which the precedence-constrained parallel application is run, and the CCR and the heterogeneity of processors do not impact significantly the quality of the acceleration of our approach.

9 Conclusions

In this chapter, we presented a classification of different methods used in literature to reduce energy consumption. Our classification is made according to three criteria. These criteria are the optimization method used to minimize the consumed energy (i.e. mono-objective or multi-objective optimization), the level of the system on which an approach is based (i.e. hardware or software level), and the type of system to which the approach is intended to be used (i.e. computing or embedded systems).

As a use case and to give an example of a method, we presented a new parallel bi-objective hybrid genetic algorithm to minimize energy consumption and makespan. The energy saving of our approach exploits the dynamic voltage scaling (DVS). According to our classification, the new method can be considered as an optimizing multi-objective method with a Pareto approach. It uses the hardware part of the system in a dynamic way. Our method is intended to be used in computing systems.

Our new approach has been evaluated with the Fast Fourier Transformation task graph which is a real-world application. Experiments show that our bi-objective meta-heuristic improves on average the results obtained in the literature (see [25], [2] and [7]) particularly in energy saving. Indeed, the energy consumption is reduced by **47.5%** and the completion time by **12%**. The experiments of the insular approach also show that the more the number of islands is used, the better the results will be. The use of 50 islands, instead of 1 island (i.e. the hybrid approach), improves the S-metric of the obtained Pareto front by **26%**. Furthermore, the multi-start approach is on average **13** times faster than the island approach using 21 cores.

Therefore, one of the main perspectives of the work presented in this chapter is to determine the solving approach to choose among ECS, the hybrid approach, and the insular approach, according to the precedence-constrained parallel application at hand. If the insular approach is chosen, the major issue is to determine the best number of islands to be used. This future work aims to minimize the total amount of consumed energy by the chosen solving approach and by the precedence-constrained parallel application to be solved. It is clear, for example, that the insular approach is interesting for the large and resource consuming precedence-constrained parallel applications and the applications intended to be executed several times.

Acknowledgments

Experiments presented in this chapter were carried out using the Grid'5000 experimental testbed, being developed under the INRIA ALADDIN development action with support from CNRS, RENATER and several Universities as well as other funding bodies (see https://www.grid5000.fr). Clusters of University of Mons were also used. We would like to thank the technical staffs of the Grid'5000 and the clusters of University of Mons for making their clusters accessible and fully operational.

References

1. Andreas Berl and Hermann de Meer. A virtualized energy-efficient office environment. In *Proceedings of the 1st International Conference on Energy-Efficient Computing and Networking*, e-Energy '10, pages 11–20, New York, NY, USA, 2010. ACM.
2. D. Bozdag, U. Catalyurek, and F. Ozguner. A task duplication based bottom-up scheduling algorithm for heterogeneous environments. In *Proc. Int'l Parallel and Distributed Processing Symp.*, April 2005.
3. J.P. Cohoon, S.U. Hedge, W.N. Martin, and D. Richards. Punctuated equilibria : A parallel genetic algorithm. In In J.J. Grefenstette, editor, *Proceedings of the Second International Conference on Genetic Algorithms*, page 148, 1987.
4. T.H. Cormen, C.E. Leiserson, and R.L. Rivest. Introduction to algorithms. In *MIT Press*, 1990.
5. Pepijn de Langen and Ben Juurlink. Trade-offs between voltage scaling and processor shutdown for low-energy embedded multiprocessors. In *Proceedings of the 7th international conference on Embedded computer systems: Architectures, modeling, and simulation*, SAMOS'07, pages 75–85, Berlin, Heidelberg, 2007. Springer-Verlag.
6. Vincent W. Freeh, Nandini Kappiah, David K. Lowenthal, and Tyler K. Bletsch. Just-in-time dynamic voltage scaling: Exploiting inter-node slack to save energy in mpi programs. *J. Parallel Distrib. Comput.*, 68:1175–1185, September 2008.
7. M.R. Garey and D.S. Johnson. Computers and intractability: A guide to the theory of np-completeness. In *W.H. Freeman and Co.*, pages 238–239, 1979.
8. Helmut Hlavacs, Roman Weidlich, Karin Hummel, Amine Houyou, Andreas Berl, and Hermann de Meer. Distributed energy efficiency in future home environments. *Annals of Telecommunications*, 63:473–485, 2008. 10.1007/s12243-008-0045-2.
9. Intel. Intel pentium m processor datasheet, 2004.
10. Jaeyeon Kang and Sanjay Ranka. Energy-efficient dynamic scheduling on parallel machines. In *Proceedings of the 15th international conference on High performance computing*, HiPC'08, pages 208–219, Berlin, Heidelberg, 2008. Springer-Verlag.
11. H. Kimura, M. Sato, Y. Hotta, T. Boku, and D. Takahashi. Emprical study on reducing energy of parallel programs using slack reclamation by dvfs in a power-scalable high performance cluster. In *Cluster Computing, 2006 IEEE International Conference on*, pages 1 –10, 2006.
12. G. Koch. Discovering multi-core: Extending the benefits of moore's law. In *TechnologyIntel Magazine*, July 2005.
13. J. G. Koomey. Estimating total power consumption by servers in the u.s. and the world.
14. Young Lee and Albert Zomaya. Energy efficient utilization of resources in cloud computing systems. *The Journal of Supercomputing*, pages 1–13, 2010. 10.1007/s11227-010-0421-3.
15. Young Choon Lee and Albert Y. Zomaya. Minimizing energy consumption for precedence-constrained applications using dynamic voltage scaling. In *Proceedings of the 2009 9th IEEE/ACM International Symposium on Cluster Computing and the Grid*, CCGRID '09, pages 92–99, Washington, DC, USA, 2009. IEEE Computer Society.

16. Man Lin and Chen Ding. Parallel genetic algorithms for dvs scheduling of distributed embedded systems. In Ronald Perrott, Barbara Chapman, Jaspal Subhlok, Rodrigo de Mello, and Laurence Yang, editors, *High Performance Computing and Communications*, volume 4782 of *Lecture Notes in Computer Science*, pages 180–191. Springer Berlin / Heidelberg, 2007. 10.1007/978-3-540-75444-222.
17. Cong Liu, Xiao Qin, S. Kulkarni, Chengjun Wang, Shuang Li, A. Manzanares, and S. Baskiyar. Distributed energy-efficient scheduling for data-intensive applications with deadline constraints on data grids. In *Performance, Computing and Communications Conference, 2008. IPCCC 2008. IEEE International*, pages 26 –33, 2008.
18. Lei Miao, Yong Qi, Di Hou, Chang li Wu, and Yue hua Dai. Energy saving task scheduling for heterogeneous cmp system based on multi-objective fuzzy genetic algorithm. In *Systems, Man and Cybernetics, 2009. SMC 2009. IEEE International Conference on*, pages 3923 – 3927, 2009.
19. R. Min, T. Furrer, and A. Chandrakasan. Dynamic voltage scaling techniques for distributed microsensor networks. In *Proc. IEEE Workshop on VLSI*, pages 43–46, April 2000.
20. Ripal Nathuji and Karsten Schwan. Virtualpower: Coordinated power management in virtualized enterprise systems. In *Proceedings of twenty-first ACM SIGOPS symposium on Operating systems principles*, SOSP '07, pages 265–278, New York, NY, USA, 2007. ACM.
21. Nikzad Babaii Rizvandi, Javid Taheri, Albert Y. Zomaya, and Young Choon Lee. Linear combinations of dvfs-enabled processor frequencies to modify the energy-aware scheduling algorithms. *Cluster Computing and the Grid, IEEE International Symposium on*, 0:388–397, 2010.
22. Xiaojun Ruan, Xiao Qin, Ziliang Zong, K. Bellam, and M. Nijim. An energy-efficient scheduling algorithm using dynamic voltage scaling for parallel applications on clusters. In *Computer Communications and Networks, 2007. ICCCN 2007. Proceedings of 16th International Conference on*, pages 735 –740, 2007.
23. Tajana Simunic, Luca Benini, Andrea Acquaviva, Peter Glynn, and Giovanni De Micheli. Dynamic voltage scaling and power management for portable systems. In *Proceedings of the 38th annual Design Automation Conference*, DAC '01, pages 524–529, New York, NY, USA, 2001. ACM.
24. Shekhar Srikantaiah, Aman Kansal, and Feng Zhao. Energy Aware Consolidation for Cloud Computing. In *Proceedings of HotPower '08 Workshop on Power Aware Computing and Systems*. USENIX, December 2008.
25. H. Topcuouglu, S. Hariri, and M.Y. Wu. Performance-effective and low-complexity task scheduling for heterogeneous computing. *IEEE Trans. Parallel Dist. Systems*, 13(3):260–274, 2002.
26. Jianli Zhuo and Chaitali Chakrabarti. Energy-efficient dynamic task scheduling algorithms for dvs systems. *ACM Trans. Embed. Comput. Syst.*, 7:17:1–17:25, January 2008.