Energy-aware stochastic scheduler for batch of precedence-constrained jobs on heterogeneous computing system

Mohammad Sajid, Zahid Raza

Abstract

The problem of optimal scheduling of precedence-constrained jobs as well as finding the Pareto-optimal sets for multi-objective scheduling problem have been proven to be nondeterministic polynomial time (NP)-complete. The growing consumption of energy has compelled the researchers to consider energy consumption as an important parameter along with other parameters in multi-objective scheduling problem. Accordingly, many energy-aware precedence-constraints scheduling algorithms have been reported in the literature. Most of the algorithms have a limitation of treating this problem as a single objective optimization problem modeling with deterministic execution times rather than stochastic execution times. This work proposes energy-aware stochastic scheduler to schedule the batch of precedence-constrained jobs on dynamic voltage frequency scaling-enabled processors in order to optimize the energy consumption and the turnaround time. The execution and inter-communication times are stochastic which are drawn from independent probability distributions. A novel encoding for batch of precedence-constrained jobs, stochastic turnaround time and energy models are also proposed. Experimental results show that, compared with other algorithms, the proposed scheduler offers reduced turnaround time and reduced energy consumption.

1. Introduction

The two important issues encountered in many real-world scheduling problems are higher energy consumption of computing servers and a certain degree of uncertainty about the execution times. The issue of higher energy consumption of computing servers is consistently gaining prominence. It is confirmed by the fact that the three million data centres in United States having 12 million computing servers consumed an estimated 91 billion kilowatt-hours of electricity in 2013. Expectedly, the energy consumption will reach to 140 billion kilowatt-hours by 2020, which leads to $13 billion per year cost and emission of nearly 150 million metric tons of carbon pollution annually [1]. The high energy consumption of computing servers has become an unavoidable issue due to high monetary cost, reduced reliability of computing devices and negative effects on the environment. These statistics encourage the development of energy-efficient hardware, software and scheduling algorithms. The second issue is the stochastic execution time of jobs and inter-communication times. This issue is adequately addressed by stochastic scheduling where the execution times of jobs are not known beforehand but the probability distributions of execution times are known [2]. The actual execution time becomes known only after the job has finished its execution. For stochastic scheduling, shortest expected processing time first (SEPT), weighted shortest expected processing time first (WSEPT) and longest expected processing time first (LEPT) policies are very popular scheduling policies. These policies work based on expectation of execution times of the jobs, however, Möhring; Schulz and Skutella; Uetz pointed out that the performance of the stochastic scheduling is affected by the expectation as well as variance of the jobs [3]. Moreover, these policies are single objective that optimize the expected makespan without considering the energy consumption of processors. Therefore, it is required to develop the algorithms that deal with multi-objective optimization as well as randomness of execution times to optimize energy consumption and expected execution time.

The successfullness of multi-objective optimization methods in various domains inspires its use to develop multi-objective scheduler to minimize energy consumption and execution time.
processors introduces new challenges to scheduling algorithms. In response to the above issues, this work proposes a multi-objective energy-aware stochastic scheduler to schedule a batch of stochastic precedence-constrained jobs on heterogeneous DVFS-enabled processors. The proposed scheduler optimizes the expected turnaround time and energy consumption using fast hypervolume-based evolutionary optimization (HypE) algorithm [7]. HypE stochastic scheduler works based on non-dominating sorting and hypervolume indicator to determine the approximation of the Pareto-optimal set. The performance of the proposed scheduler has been compared with other multi-objective algorithms as well as single objective algorithms.
The organization of the remaining paper is given as follows. Next section presents an overview of the related research in the domain. Section 3 presents the problem formulation including notations used, HCS model, batch model, energy consumption model along with problem statement. Section 4 details the proposed stochastic scheduler whereas section 5 presents the experimental study with observations. Section 6 presents the concluding remarks and the possible future research directions.

2. Related Work

We observed three basic approaches for energy aware scheduling in the literature: energy-aware heuristics based scheduling algorithms; energy-aware evolutionary scheduling algorithms and stochastic scheduling algorithms. In the former two approaches, the problem is considered as deterministic scheduling problem. In the latter approach, the scheduling policies/ algorithms have been developed considering non-deterministic/stochastic problems.

Two heuristic algorithms i.e. fixed-order list scheduling with shared slack reclamation (FLSSR) and global scheduling with shared slack reclamation (GSSR) algorithms have been proposed and evaluated for precedence-constrained jobs and independent jobs respectively in Ref. [8]. Both the algorithms employ the concept of slack sharing to scale down the frequency of processors to reduce the energy consumption and to meet the deadlines. Simulation study observes that both algorithms can save up to 44% of energy consumption.

Two heuristic algorithms as energy conscious scheduling (ECS) and ECS + Idle are investigated and evaluated for the precedence-constrained jobs on DVFS-enabled processor with exploitation of each voltage level on each processor in Ref. [9]. Both algorithms reveal superior performance in terms of energy consumption and turnaround time. In Ref. [10], the problem of minimizing the schedule length with energy constraint and the problem to minimize the energy with schedule length constraint have been addressed. Three types of energy aware heuristic algorithms i.e. pre power-determination, post power-determination and hybrid of both algorithms have been proposed. The algorithms report superior performance as compared with the optimal algorithms. The performance ratio of hybrid algorithm almost approaches 1 for precedence constrained jobs having large number of tasks.

The energy-aware scheduling problem has been formulated as the problem of task slack reclamation by using combinations of processors’ frequencies in Ref. [11]. The newly developed multiple voltage frequency selection algorithm determines the best combination of frequencies to schedule the tasks. The study observes that optimal energy consumption can be achieved only by two adjacent voltage levels if power consumption is a convex function of frequency. In another such work, Energy-aware scheduling algorithm with equalized frequency has been proposed and evaluated for precedence-constrained jobs on heterogeneous computing systems. This algorithm works in three phases aiming to minimize finish time and energy dissipation and offers better results than peer algorithms considered for comparison [12]. The research study [13] addresses the energy-aware scheduling and data allocation problem on heterogeneous computing systems. The proposed algorithms, task assignment considering data allocation (TAC-DA) and task ratio greedy scheduling (TRGS) generate an efficient schedule in such a manner that time constraints can be met and the energy consumption can be minimized. The simulation based performance analysis observed that TAC-DA and TRGS offers 13.72%, 15.76% and 19.76%, 24.67% less energy consumption in comparison to the greedy algorithm.

Evolutionary algorithms have also been utilized to address the energy aware scheduling problem. In Ref. [14], three evolutionary algorithms, multi objective cellular genetic algorithm, Non-dominated sorting genetic algorithm-II (NSGA-II) and Indicator based evolutionary algorithms are employed. Using these algorithms, three energy aware schedulers have been proposed for precedence-constrained jobs for realizing on DVFS-enabled processors. It was observed that main difficulty to solve the energy efficient scheduling problem by evolutionary algorithms is dependent mainly on processor number and communication-to-computation ratio with minor influence of the number of tasks. Bat intelligence (BI) optimization technique for single, bi and tri-objective scheduling problems while considering makespan, energy and tardiness as objective functions was introduced in Ref. [15]. The study observed clear conflicting relationships between makespan, energy and tardiness. It study shows that the increase in energy consumption results in decrease in execution times which translates into decrement in makespan and tardiness. A quantum-inspired hyper-heuristic has been employed to address the energy-aware scheduling problem for precedence constrained jobs on heterogeneous multiprocessor systems with dynamically variable voltage. Simulation results indicate that the proposed method exhibits outstanding performance in comparison to heterogeneous earliest finish time (HEFT) algorithm, ECS and genetic algorithm (GA) [16]. Another study in Ref. [17] addresses the problem of optimizing the energy consumption and system reliability. A novel bi-objective genetic algorithm has been proposed to schedule parallel jobs on DVFS-enabled systems. It is indicated through experiments that bi-objective genetic algorithm offers fine Pareto front in comparison to multi-objective HEFT and multi-objective differential evolution.

Various approximation scheduling policies e.g. SEPT, WSSEPT and LEPT have also been reported in the literature aiming to minimize the expected weighted turnaround time [18]. Linear programming-based approximation technique has been investigated in Ref. [3] to minimize the weighted completion time of stochastically independent jobs with individual release dates on m homogeneous processors. The linear programming-based scheduling policy has been proposed with performance guarantee of $3 - ((1/m) + max(1, ((m − 1)/m) \Delta))$ where $\Delta$ is upper bound on squared coefficients of variation. The research study in Ref. [18] proposed constant-factor approximation algorithms for stochastic precedence constrained jobs to optimize the total weighted completion time on homogeneous processors. For precedence-constraints scheduling with and without release dates, the proposed algorithm $(k > 0)$ has been proven with performance guarantee of $(1 + k) \cdot (1 + ((m − 1)/m) \Delta)$ and $(1 + k) \cdot (1 + ((m − 1)/m) \Delta)$ respectively [19]. An average case analysis investigated for the problem of scheduling stochastic jobs on parallel homogeneous processors to minimize the completion time has been reported in Ref. [20]. The work proves that SEPT does not minimize the expected competitive ratio for execution time of general probability distribution. In Ref. [21], the problem of scheduling the stochastic jobs on heterogeneous processors to minimize the expected weighted completion time has been proposed. By using time-indexed linear programming relaxation, a polynomial time scheduling policy with performance guarantee $(3 + \Delta)/2 + \epsilon$ is given, where $\epsilon > 0$ and $\Delta$ is the upper bound on the squared coefficient of variation of the execution times.

For stochastic precedence constrained job, stochastic heterogeneous earliest finish time (SHEFT) algorithm has been proposed for heterogeneous computing system. This algorithm employs the average execution time of tasks based on expectation and variance. It is observed that SHEFT algorithm performs better than HEFT and dynamic critical path scheduling algorithm in terms of expected makespan and standard deviation [22]. A stochastic dynamic level scheduling (SDLS) algorithm has been proposed and evaluated for
precedence constrained jobs in Ref. [23]. It is proved that if execution and communication times of tasks are replaced by random variables, the expected turnaround time is greater than or equal to the turnaround time of scheduling deterministic tasks. The SDLS algorithm performs better than HEFT, dynamic level scheduling algorithm and SHEFT in terms of makespan, speedup and makespan standard deviation. Another work in Ref. [24] investigates the energy-efficient stochastic scheduling algorithm (ESTS) to schedule the stochastic independent tasks on DVFS-enabled processors with deadline and energy constraints. The ESTS algorithm is reported to perform better than MinMin and MaxMin algorithms. In Ref. [25], energy and makespan aware stochastic robust scheduling algorithms have been proposed based on MinMin, GA, local search and Tabu search. Experimental study shows that Tabu search based algorithm offers the best results for the stochastic independent tasks on DVFS-enabled processors. The research study in Ref. [26] investigates stochastic energy aware scheduling and proposes energy aware stochastic HEFT (ESHEFT) algorithm to schedule the precedence constrained jobs on DVFS-enabled processors while, incorporating the slack time of tasks to minimize energy and turnaround time. The proposed algorithm is reported to show better results than SHEFT and ECLS in terms of turnaround time and energy consumption.

It is to be noted that mainly either stochastic algorithms/scheduling policies have been proposed for single stochastic precedence-constrained job to optimize the turnaround time or multi-objective algorithms being used to address the deterministic precedence-constrained scheduling problem to optimize the turnaround time and energy consumption. This work proposes a multi-objective energy aware stochastic scheduler to execute a batch of stochastic precedence constrained jobs on DVFS-enabled HCS to optimize both the expected turnaround time and energy consumption.

3. Problem formulation

This section presents an insight into the various components related to energy-aware stochastic precedence constrained scheduling problem.

3.1. Heterogeneous computing system

The execution environment is considered to be heterogeneous computing system (HCS) $H$ consisting of $M$ different processors represented by $H = \{p_i : 1 \leq i \leq M\}$. All processors are DVFS-enabled i.e. each processor has multiple discrete frequency levels.

The processor $p_i \in \bar{H}$ consists of FL$_k$ frequency levels and the set of frequency levels is given by $\text{Freq}_k = \{f_{k,z} : 1 \leq z \leq \text{FL}_k\}$ such that if $y < z, f_{k,y} > f_{k,z}$. To sustain an operating frequency $f_{k,z}$, the supply voltage $v_{k,z}$ is required and the set of voltage levels corresponding to each frequency level is given by $\text{Volt}_k = \{v_{k,z} : 1 \leq z \leq \text{VL}_k\}$. For example, Table 1 represents two DVFS-enabled processors that can operate on five different frequency and voltage levels [11]. The total number of frequency/voltage levels ($M_{z}^{-1}$) in computation system $H$ will be equal to the sum of all the voltage/frequency levels of all the processors i.e. $M_{z}^{-1} = \sum_{k=1}^{M} |\text{VL}_k|$. The voltage $(v_{k,z})$ and the corresponding frequency $(f_{k,z})$ of processor $p_i$ does not switch during execution of any task. All of the $M$ processors are connected via a high-speed interconnection network e.g. Myrinet or InfiniBand. The intra-communication cost is zero whereas inter-communication cost will be non-zero. The HCS $H$ model can be employed by clusters, cloud computing and easily extendable to multi-core DVFS-enabled HCS or multi-core per-Core DVFS-enabled HCS.

3.2. Batch of stochastic precedence-constrained jobs

A batch $\beta$ of $I$ stochastic jobs $\beta = \{j_i : 1 \leq i \leq I\}$ is executed on HCS $H$. In batch $\beta$, all jobs $j_i$ are independent to each other but each job $j_i$ consists of parallel and dependent tasks. The job $j_i$ is represented by Directed Acyclic Graph (DAG) in vector form $(T_i, E_i)$ where $T_i = \{t_{i,j} : 1 \leq j \leq l_i\}$ is the set of $l_i$ dependent tasks and $E_i = \{e_{i,j,k} : 1 \leq j < k \leq l_i\} \subset T_i \times T_i$. It is the set of communication edges that maintains precedence-constraints relationship between tasks. Each task $t_{i,j}$ is an indivisible unit and can be executed only on one processor at a time without preemption. The edge $e_{i,j,k} \in E_i$ represents the precedence-constraints such that task $t_{i,j}$ must complete its execution before task $t_{i,k}$ starts its execution. The task $t_{i,k}$ is the successor task of task $t_{i,j}$ and the set of all successor tasks of task $t_{i,j}$ is given by $\text{Succ}_{i,j} = \{t_{i,k} : e_{i,j,k} \in E_{j}\}$. The edge $e_{i,j,k} \in E_i$ $(e_{i,j,k} \neq e_{i,j,0})$ represents that task $t_{i,k}$ is the predecessor of task $t_{i,j}$ and the set of predecessors of task $t_{i,j}$ being $\text{Pred}_{i,j} = \{t_{i,k} : e_{i,j,k} \in E_{i}\}$. A task $t_{i,j}$ having zero predecessor tasks i.e. $\text{Pred}_{i,j} = \emptyset$, is called the source task and a task $t_{i,j}$ having zero successor tasks i.e. $\text{Succ}_{i,j} = \emptyset$, is called the sink task.

The tasks of all jobs are distributed over levels starting from 1. The tasks $t_{i,j}$ with zero predecessor tasks i.e. $\text{Pred}_{i,j} = \emptyset$, stretch out on the level 1. On the next level, the successor tasks of first level tasks exist and so on. The level of task $t_{i,j}$ is determined recursively as follows,

$$L_{i,j} = \begin{cases} 1 & \text{if } t_{i,j} \in \text{Pred}_{i,j} = \emptyset \\ 1 + \max \{L_{k,j} : \forall t_{k,j} \in \text{Pred}_{i,j}\} & \text{otherwise} \end{cases}$$

(1)

The level of batch $(BL_{\beta})$ is the maximum level of any task $t_{i,j} \in \beta$ and is computed as

$$BL_{\beta} = \max_{1 \leq i \leq I} \{L_{i,j}\} = \max \{L_{i,j} : \forall t_{i,j} \in \beta\}, \text{ Succ}_{i,j} = \emptyset$$

(2)

A sample batch $\beta_1$ of 3 jobs, i.e. $\beta_1 = \{j_1, j_2, j_3\}$ is depicted in Fig. 1. Each of the job $j_i (1 \leq i \leq 3)$ is independent to each other and consists of 7, 9 and 10 dependent tasks respectively. The batch $\beta_1$ consists of one source, one sink task; one source, two sink tasks and two source tasks, three sink tasks in jobs $j_1, j_2$ and $j_3$ respectively. In job $j_1$, the set of predecessor and successor tasks of $t_{1,1}$ is given by $\text{Pred}_{1,1} = \{t_{1,1}\}$ and $\text{Succ}_{1,1} = \{t_{1,5}, t_{1,6}\}$ respectively. The tasks $t_{1,1}, t_{2,1}, t_{3,1}, t_{3,2}$ have level 1 and their successor tasks have level 2. The level of batch $\beta_1$ is 4 i.e. $BL_{\beta_1} = 4$.

3.3. Scheduling preliminaries

The execution and communication times for each task and edge $t_{i,j}, e_{i,j,k} \in \beta$ are drawn from independent normal probability
distributions \[14\]. Let \(N(\mu, \sigma^2)\) be the normal probability distribution with mean \(\mu\) and variance \(\sigma^2\). Then, \(t_{ij} \sim N(\mu, \sigma^2)\) and \(e_{t(j,k)} \sim N(\mu', \sigma^2').\) Since, the processors employed in HCS are DVFS-enabled, the execution time of task \(t_{ij}\) have different parameters for normal probability distribution corresponding to different frequency levels. The execution time of task \(t_{ij}\) w.r.t. \(v_{xz}\) follows normal probability distribution with mean \(\mu_{i,j,x,z}\) and variance \(\sigma^2_{i,j,x,z}\) i.e. \(t_{ij} \sim N(\mu_{i,j,x,z}, \sigma^2_{i,j,x,z}).\) The \([\text{MET}_{rs}]\) and \([\text{VET}_{rs}]\) matrices of order \(I_p \times M_H\) consist of mean and variance of execution times of all task \(t_{ij} \in \beta\) w.r.t. each \(v_{xz} \in \text{Vol}_x.\) The mean \((\mu_{i,j,x,z})\) and variance \((\sigma^2_{i,j,x,z})\) of task \(t_{ij} \in \beta\) w.r.t. \(v_{xz} \in \text{Vol}_x\) are stored at \([\text{MET}_{rs}]\) and \([\text{VET}_{rs}]\). The mappings between \(i, j, x, z\) and \(r, s\) indices are computed as

\[
\begin{align*}
\rho & = \begin{cases} 
    j_i, & i = 1 \\
    \sum_{w=1}^{i-1} t_{iw} + j_i, & 2 \leq i \leq I 
\end{cases} \\
\kappa & = \begin{cases} 
    z_x, & x = 1 \\
    \sum_{w=1}^{x-1} \text{Vol}_w + z_x, & 2 \leq x \leq M 
\end{cases}
\end{align*}
\]  

(3)

(4)

Since, the execution times follow the normal distribution; therefore, by using closure property of linear transformation and summation property of normal distribution, the average execution time of task \(t_{ij} \in \beta\) i.e. \(AE(t_{ij})\) also follows the normal distribution. The expectation and variance of the average execution time of task \(t_{ij} \in \beta\) w.r.t. all voltage levels can be computed as

\[
\begin{align*}
\mathbb{E}(AE(t_{ij})) & = \frac{\sum_{s=1}^{M_H} \text{MET}_{rs}}{M_H} \\
\text{Var}(AE(t_{ij})) & = \frac{\sum_{s=1}^{M_H} \text{VET}_{rs}}{M_H}
\end{align*}
\]

(5)

(6)

where \(\rho\) has a unique value for each task \(t_{ij} \in \beta\) and can be computed using \(i, j\) and eq. (3).

The tasks of the same job i.e. \(t_{ij} \in T_i\) are likely to be scheduled on different processors. The communication times between tasks are also random and drawn from independent normal distribution. The mean of the communication edge between two tasks represents the average communication time irrespective of the processors involved whereas variance contributes towards heterogeneity of the processors. Similar to execution times, the mean and variance of the communication times between tasks is stored in \([\text{MCT}_{rs}]\) and \([\text{VCT}_{rs}]\) of order \(I_p \times (\max(I_i : 1 \leq i \leq I))\) respectively. Further, the mean \((\mu_{i,j,k})\) and variance \((\sigma^2_{i,j,k})\) of edge \(e_{t(j,k)}\) between task \(t_{ij}\) and \(t_{ik}\) are stored as \([\text{MCT}_{rs}]\) and \([\text{VCT}_{rs}]\) respectively. The mapping between \(r\) and \(i, j\) can be computed using equation (3) whereas mapping between \(s\) and \(k\) can be obtained as

\[
s = k, \quad \forall k
\]

(7)

Fig. 2 depicts the job \(J_1 \in \beta_1\) with average execution and communication time's w.r.t. HCS \(H.\)

The allocation matrix \([\lambda_{rs}]\) of order \(I_p \times M_H\) represents the
mapping between tasks to processors with the assigned voltage level. If task $t_{ij}$ is allocated to processor $p_x$ at voltage level $v_{xz}$, then $|\Delta_{xz}| = 1$ otherwise it is zero. The mapping between $r, s$ and $i, j, x, z$ indices is computed using eqs. (3) and (4).

The Procs($t_{ij}$) gives the processor on which the task $t_{ij}$ is scheduled. The expected ready time ($Pr_{T_x}$) of each processor $p_x \in H$ are updated after previously scheduled tasks have finished their execution. Each task $t_{ij}$ is assigned expected ready time ($RT_{ij,x}$) and finish time ($FT_{ij,x}$) w.r.t. the assigned processor $p_x$. The computation method for $P_{RT_x}$, $RT_{ij,x}$ and $FT_{ij,x}$ corresponding to the proposed algorithm has been presented in Section 4. For every task $t_{ij}$, the following property must hold

$$RT_{ij,x} < FT_{ij,x} < RT_{k,y} \quad \forall \; t_{ik} \in Succ_{ij}, \; \forall \; p_x, p_y$$

(8)

Each processor $p_x$ executes tasks belonging to multiple jobs. Therefore, for all tasks $t_{ij} \in \beta$ scheduled to processor $p_x$, one of the following properties must hold

$$RT_{ij,x} < FT_{ij,x} < RT_{kj,x} \quad \text{or} \quad RT_{kj,x} < FT_{kj,x} < RT_{ij,x}$$

(9)

The turnaround time ($TAT_{ij}$) of job $j_i$ will be equal to the maximum of expected finish time ($FT_{ij}$) of all the tasks $t_{ij} \in T_i$ as

$$TAT_{ij} = \max_{j=1}^{j'} \{FT_{ij}\} = \max \{FT_{ij} : \forall \; t_{ij} \in T_i, \text{Succ}_{ij} = \phi\}$$

(10)

Since, batch $\beta$ consists of more than one job and the tasks of these jobs are merged for exploiting the maximum parallelism, the turnaround time ($TAT_{\beta}$) of batch $\beta$ is the maximum of finish times of all the tasks and can be computed as

$$TAT_{\beta} = \max_{j=1}^{j'} \{\max_{j=1}^{j'} \{FT_{ij}\}\} = \max \{FT_{ij} : \forall \; t_{ij} \in \beta, \text{Succ}_{ij} = \phi\}$$

(11)

3.4. Energy consumption model

The high energy consumption increases the temperature of the devices leading to negative effects on the reliability of systems and therefore, the controlling the high energy consumption is highly required [27]. The total energy consumed by an HCS $H$ depends on all its components viz. processors, interconnection network, memory, cooling systems and so on [28]. The contribution by the processors towards the total energy plays a major role. The power consumption ($P^\text{Pow}_{k,x}$) of processor comprises of static power ($P^\text{Static}$) and dynamic power ($P^\text{Dynamic}$). The total power consumed by a processor $p_x$ at voltage $v_{k,x}$ can be written as [24],

$$P^\text{Pow}_{k,x} = P^\text{Static}_{k,x} + \eta^\text{c} P^\text{Dynamic}_{k,x}$$

(12)

Here, $\eta^\text{c} = 1$ represents the busy mode i.e. processor $p_x$ is working at voltage $v_{k,x}$ and $P^\text{Pow}_{k,x}$ represents the power consumption in the busy mode. The value of $\eta^\text{c} = 0$ represents the idle mode of the processor and is equivalent to static power consumed ($P^\text{Static}_{k,x}$). For processor ($p_x$), the static power ($P^\text{Static}$) is constant irrespective of the voltage and frequency applied.

The dynamic power ($P^\text{Dynamic}$) depends on frequency $f_{x,z}$ and voltage $v_{k,x}$ as

$$P^\text{Dynamic}_{k,x} = \gamma_x \cdot v_{k,x}^2 \cdot f_{x,z}$$

(13)

where $\gamma_x$ is a constant representing the activity factor and the physical capacitance. The constant $\gamma_x$ is design and manufacturing constant whereas frequency $f_{x,z}$ and $v_{k,x}$ are decided at the compile or run time.

The energy consumed ($E^2_{k,x}$) by processor $p_x$ at voltage $v_{k,z}$ for given time period $W^m$ can be computed using equations (12) and (13) as

$$E^2_{k,x} = \eta^\text{c} P^\text{Pow}_{k,x} W^m$$

(14)

If $\eta^\text{c} = 0$, processor is idle and $W^m$ represents the length of the idle slot on the processor. For $\eta^\text{c} = 1$ and $|\Delta_{xz}| = 1$, $W^m$ is replaced by the execution time of task $t_{ij}$ and $E^2_{k,x}$ represents the energy consumed by processor $p_x$ at $v_{k,z}$ voltage level.

3.5. The problem statement

The scheduling problem under consideration can be represented using Graham et al. notation as $\text{QM} \left(\left[ t_{ij} \right] \sim \text{stoc min} TAT_{ip} \min EN_{ip} \right)$ [2]. The first field $\text{QM}$ represents an HCS $H$ of $M$ heterogeneous parallel DVFS-enabled processors as explained in Section 3.2. The second field represents that the task $t_{ij} \in \beta$ follows the precedence constraints whereas time requirements are stochastic as explained in Section 3.3. The third field represents the objective of the problem i.e. minimization of the turnaround time and energy consumption of the batch $\beta$. The solutions that minimize both the objectives are Pareto-optimal and are non-dominated by other solutions. The set $P^*$ represents the possible approximation of true Pareto optimal solutions. The solution $S \in P^*$ is non-dominated and minimizes the turnaround time ($TAT_{ip}$) as well as energy consumption ($EN_{ip}$). The set $P^*$ of approximate Pareto optimal solutions is given by,

$$P^* = \{ S \in \Omega : \exists \; S' \in \Omega, \; F(S') \prec F(S) \} \subseteq \Omega$$

(15)

where $\Omega$ is the set of solutions with each $S \in \Omega$ representing the scheduling policy of the batch $\beta$ on HCS $H$ and $F(S) \prec F(S)$ represents that solution $S'$ dominates the solution $S$ [5].

Since, $TAT_{ip}$ and $EN_{ip}$ depends on $|\text{MCT}_{r,s}|, |\text{VCT}_{r,s}|, |\text{MCT}_{r,s}|, |\text{VCT}_{r,s}|$ and $|\Delta_{xz}|$ both quantities are random and known at the run time. The mean and variance of turnaround time ($TAT_{ip}$) are $Z_1$ and $Z_2$ whereas for energy ($EN_{ip}$) are $Z_3$ and $Z_4$ respectively. Accordingly, the statement of the problem can be written as

$$\begin{align*}
Z_1 &= \mu(TAT_{ip}) \\
Z_2 &= \sigma^2(TAT_{ip}) \\
Z_3 &= \mu(EN_{ip}) \\
Z_4 &= \sigma^2(EN_{ip})
\end{align*}$$

(16)

Minimize

$$\begin{align*}
Z_1 &= \mu(TAT_{ip}) \\
Z_2 &= \sigma^2(TAT_{ip}) \\
Z_3 &= \mu(EN_{ip}) \\
Z_4 &= \sigma^2(EN_{ip})
\end{align*}$$

s.t.

$$\begin{align*}
\frac{\sum_{r=1}^{M} \sum_{s=1}^{N} |A_{r,s}|}{|A_{r,s}|} &= 1 \quad \forall \; s \\
\frac{\sum_{r=1}^{M} \sum_{s=1}^{N} |A_{r,s}|}{|A_{r,s}|} &= \frac{\sum_{s=1}^{N} |A_{r,s}|}{|A_{r,s}|} \quad \sim 0 \quad \forall \; r \in [1, K]
\end{align*}$$

(17)

The mapping between $r, s$ and $i, j, x, z$ indices is computed using eqs. (3) and (4).
4. Proposed stochastic scheduling model

The involvement of more than one objective demands to find the Pareto-optimal solutions i.e. set of solutions that aim to find the good trade-offs of all involved objectives rather than a single optimized solution. Evolutionary multi-objective algorithms are far more efficient to generate the approximation of Pareto-optimal solutions instead of traditional methods e.g. weighted sum method due to their well-known drawbacks [5]. The primary reason to apply evolutionary multi-objective algorithms is their ability to offer multiple approximate Pareto-optimal solutions in a single run. This section details the proposed stochastic scheduler based on HypE algorithm [7].

4.1. HypE

HypE proposed by Bader et al. is an evolutionary multi-objective algorithm for arbitrary number of objectives. It features hypervolume indicator, non-dominated sorting, \((d + d)\) selection scheme for \(d\) population size and a fast search method based on Monte Carlo simulation. The hypervolume represents the volume of the objective space dominated by the Pareto optimal solutions with respect to a given set of reference points and is strictly monotonic with respect to Pareto dominance i.e. a set of solutions that entirely dominates another has a higher value of the hypervolume indicator. HypE works as the standard evolutionary algorithm and takes population size, number of sampling points and a reference set of mutually non-dominating objective vectors as inputs. Initially, HypE generates a random population of candidates and then executes the loop for a given number of generations. For each generation, it consists of successive application of mating selection, variation, and environmental selection. For mating selection, it computes the fitness value based on the hypervolume and then uses binary tournament selection to choose a population of \(d\) candidates for variation. Next, the variation is applied on chosen candidates for variation. Finally, the environmental selection method is used to select the most promising \(d\) solutions for the next iteration. The candidates are chosen using fast non-dominated sort algorithm and hypervolume based iterative approach [29].

4.2. Encoding and decoding

The proper representation of the chromosomes in phenotype and genotype space can lead to more efficient and effective performance of evolutionary algorithms. Further, one-to-one mapping in encoding (phenotype-to-genotype) and decoding (genotype-to-phenotype) can also accelerate the speed of the algorithm. The batch \(\beta\) is the set of \(I\) mutually independent jobs but each job \(j \in \beta\) consists of \(I_{ij}\) precedence-constraints tasks. Therefore, the representation of chromosomes must exploit the parallelism between and within the jobs. This work proposes a novel structure of chromosomes for batch \(\beta\) that fulfills the above desired properties consisting of scheduling policy for batch \(\beta\) having execution order of tasks. The structure ensures that all tasks become ready to schedule level-by-level and lower level tasks on queue come before the tasks of higher levels in order to follow the precedence-

constraints in the jobs.

The encoding algorithm and corresponding structure are given in Algorithm 1 and Fig. 3 respectively. The size of the chromosome is taken as the number of tasks in the batch \(\beta\) i.e. \(I_\beta\). As shown, Algorithm 1 divides the whole batch into levels and uses permutation of the same level tasks to create the structure of chromosome in the phenotype space. Next to create the genotype structure, Algorithm 1 assigns the consecutive numbers to chromosome’s genes and a look-up table is created consisting of unique index for each task. Next, it creates the look-up table for the processors. Each processor \(P_x \in H\) consists of different discrete voltage levels. Therefore, the random selection of voltage level increases the selection probability of processor with more voltage levels that results in an unbalanced load distribution with possibly high turnaround time and low system utilization. In order to tackle this issue, the proposed scheduler randomly selects a processor and then randomly picks a voltage level to execute the task. To assign a value to the gene, a Look-up Table for voltage levels is created that consists of a voltage index corresponding to each voltage level. Table 2 presents the Look-up Table for two processors as explained in Table 1 in Section 3.2.

For batch \(\beta_1\) in Fig. 1, Algorithm 1 generates a random permutation of tasks at level 1 to 4 as given by \((t_{2,1}, t_{3,2}, t_{1,3}, t_{3,1}), (t_{2,2}, t_{3,3}, t_{3,5}, t_{2,3}, t_{1,4}, t_{3,4}, t_{1,2}, t_{2,4}, t_{1,4}, t_{2,5}), (t_{3,6}, t_{5,2}, t_{8,2}, t_{1,5}, t_{2,6}, t_{3,7}, t_{1,6})\) and \((t_{2,9}, t_{3,8}, t_{1,7}, t_{3,10}, t_{3,9})\) respectively. Next, the algorithm generates the phenotype structure for batch \(\beta_1\) as shown in Fig. 4. Algorithm 1 then creates the look-up tables for tasks as well as processors. Following this, the tasks are assigned to randomly chosen processors and then the voltage levels are selected randomly using processor look-up table. The created genotype structure is shown in Fig. 5.

4.3. Stochastic turnaround time model for batch \(\beta\)

In this section, a novel algorithm (Algorithm 2: Stochastic Turnaround time model) to compute the expected turnaround time \((TAT_\beta)\) of batch \(\beta\) is presented. The proposed algorithm works based on predecessor count \((\text{predCount}_{ij})\) of tasks, the order of tasks assigned to processors, Clark’s equations [30] and the summation property of normal distribution [31]. It takes one chromosome and Look-up tables as inputs and returns the expected turnaround time of batch \(\beta\) \((TAT_\beta)\) as output. First of all, Algorithm 2 creates the phenotype structure of chromosome using tasks and processors.

![Fig. 3. Structure of chromosome with \(I_{\beta}\) genes in Phenotype Space.](image-url)
Look-up tables. The algorithm scans the phenotype structure and generates a queue Q(x) by adding tasks allocated to each processor p x ∈ H. It then assigns the predecessor count (predCount ij ) to each task t ij ∈ β. The predecessor count is zero for all the tasks at level 1. Since tasks are scheduled level-by-level, therefore, the following procedure is repeated B nd times. The algorithm inspects the queue Q(x) corresponding to each processor p x and schedules each task t ij with zero predCount ij . If a task t ij has zero predecessors (Pred ij = φ), the expected ready time of task t ij at processor p x (RT ij ) will be equal to the expected ready time of the processor p x (PRT x). If task t ij has one predecessor task t k i then it has already been scheduled on processor p x i.e. Proc i (k) = p x, then RT ij,k will be equal to PRT x. The expected finish time (FT ij,k ) of task t ij will be equal to the sum of expected ready time (RT ij,k ) and expected execution time [MET x, i ] [VE T r, s]. The mapping between i, j, x and r, s can be obtained by using equations (3) and (4). For other tasks, the expected ready time will be equal to the maximum of the processor ready times(PRT x) and the sum of expected finish time FT ij,k for all tasks t k i ∈ Pred ij and expected communication time(e ij,k ). After scheduling each task t ij, the communication data is sent to each successor task t k i ∈ Succ ij and the predecessor count ( predCount ij ) is reduced by one. After assigning expected ready and finish times to all tasks, the algorithm computes and returns the expected turnaround time of batch β (TAT β) using Clark’s equations [30].

Due to random execution and communication times, the computation of parameters RT ij,k, FT ij,k and PRT x employs the theory of randomness. To compute TAT β, the model uses three equations (18)–(20) as,

$$RT_{ij} = PRT_x \text{ if } \text{Pred}_{ij} = \phi \text{ or } |\text{Pred}_{ij}| = 1 \& \& t_{ij} \in \text{Pred}_{ij}; \text{Proc}_{ij} = p_x$$  \hspace{1cm} (18)

$$RT_{ij} = \max\left\{PRT_x, \max\left\{FT_{ij,k} + e_{ij,k} : t_{ij} \in \text{Pred}_{ij}, \text{Proc}_{ij} = p_x \right\}\right\} \text{otherwise}$$  \hspace{1cm} (19)

$$FT_{ij} = RT_{ij} + [MET_{rs}][VE T_{rs}]$$  \hspace{1cm} (20)

Here, for computing equation (19) requires the maximum of random numbers whereas equation (20) needs the sum of two random numbers. The summation property of normal distribution says,

$$\text{if } X_i \sim N\left(\mu_i, \sigma^2_i\right) \text{for } i = 1 \ldots r \text{ then } X = \sum_{i=1}^{r} X_i \sim N\left(\sum_{i=1}^{r} \mu_i, \sum_{i=1}^{r} \sigma^2_i\right) \hspace{1cm} (21)$$

Further, to compute the maximum of independent normally distributed numbers, Clark’s recursive equations can be used. In general, the task with more than one predecessor tasks (join task) is required to handle equation (19) to compute the expected ready time for join tasks. Fig. 6 depicts a partial DAG consisting of a join task t jm with n predecessor tasks. It is assumed that the expected ready time (RT jm,x ) for join task t jm is required to compute which is scheduled at processor p x and that n predecessor tasks of t jm i.e. t j1, t j2, t j3, ..., t jn are already scheduled on p x, p y, p z, ... p z respectively. Further, if task t jm ∈ β is the last scheduled task on processor p x, the expected ready time (RT jm,x ) of task t jm on p x can be stated as

$$RT_{jm,x} = \max\left\{PRT_x, \max\left\{FT_{ij,y} + e_{ij,(k,m)} : t_{ij} \in \text{Pred}_{jm}, \text{Proc}_{ij} = p_x, k = 1 \ldots n\right\}\right\}$$  \hspace{1cm} (22)

Clark’s equations can be applied to independent normally distributed numbers. It is clear from equation (22) and DAG structure that the ready and finish time of tasks are not independent although the inter-task communications times are independent to each other and follows the normal probability distribution. The correlation coefficient between any pair of inter-task communication times is zero. Since, the scheduling model is interested to
know the order of scheduled tasks and approximate probability distribution for the max operator; this work still adopt this assumption, so that Clark’s method can be applied. According to this assumption and equation (21), the ready and finish times of all tasks \(RT_{i,j,x}\) as well as processor ready time \(PRT_x\) follows normal distribution and these quantities can be computed easily.

Clark’s equations recursively compute the parameters (expectation and variance) of the greatest of normally distributed numbers i.e. it computes the parameters in pairs and reduces the variables one-by-one. The ready time of task \(t_{ij}\) (eq. (19)) consists of either the processor ready times or the sum of finish time and communication times of tasks, both of them following the normal distribution using eq. (21). Consider these quantities are communication times of tasks, both of them following the normal distribution stated as

\[
RT_{i,m,x} = \max\{C_1, C_2, C_3, \ldots, C_n\} \\
= \max\{\ldots, \max\{\max\{C_1, C_2\}, C_3\}, \ldots, C_n\} \tag{23}
\]

For eq. (23), Clark’s equations compute the expectation and variance of the maximum of \(C_1\) and \(C_2\) i.e. \(\max\{C_1, C_2\}\). The expectation can be written as

\[
E[\max\{C_1, C_2\}] = E[C_1]\Phi(\gamma) + E[C_2]\Phi(-\gamma) + a^2\Phi(\gamma) \tag{24}
\]

\[
\Phi(\gamma) = \int_{-\infty}^{\infty} \phi(t)dt, \quad \phi(t) = \frac{1}{\sqrt{2\pi}} e^{-t^2/2}, \quad \rho_{C_1, C_2} = 0 \tag{25}
\]

\[
a^2 = Var(C_1) + Var(C_2), \quad \gamma = (E(C_1) - E(C_2))/a \tag{26}
\]

The variance of \(\max\{C_1, C_2\}\) can be computed as

\[
Var[\max\{C_1, C_2\}] = \left(E^2[C_1] + Var(C_1)\right)\Phi(\gamma) + \left(E^2[C_2] + Var(C_2)\right)\Phi(-\gamma) + B \tag{27}
\]

\[
B = (E[C_1] + E[C_2])a^2\Phi(\gamma) - E^2[\max\{C_1, C_2\}] \tag{28}
\]

Similarly, Clark’s equations can be applied to \(\max\{\max\{C_1, C_2\}, C_3\}\) and mean and variance of the ready time of task \(t_{n,x}\) can be computed recursively.

After computation of finish times for all tasks \(t_{ij}\), \(TAT_{i,j}\) can be computed using eq. (11) with mean \(\mu(TAT_{i,j})\) and variance \(\text{Var}(TAT_{i,j})\). As equation (11) uses max operator, Clark’s method can be used to obtain the parameters of turnaround times i.e. expectation and variance.

4.4. Stochastic energy model for batch \(\beta\)

In this section, the expectation and variance of total energy consumed \(\langle E_N \rangle_{\beta}\) are computed. This computation procedure works based on the summation property (eq. (21)) and closure property of linear transformation (eq. (28)) due to the stochastic nature of jobs with the closure property of linear transformation w.r.t normal distribution stated as

\[
\begin{align*}
if \quad X_i &\sim N(\mu_i, \sigma_i^2), \quad a, b \in \mathbb{R} \text{ then } Y = aX_i + b \\
&\sim N(a\mu_i + b, (a\sigma_i)^2) \tag{28}
\end{align*}
\]

The total energy consumption by batch \(\beta\) depends on idle slots \(l_k\) and voltage levels of busy mode on all processors. Assume \(S_x\) is the sum of execution times on processor \(p_x\) with assigned voltage levels corresponding to all scheduled tasks. As execution times follow the normal distribution, their sum will also be following the normal distribution using (21) and (28). Therefore, the expectation and variance of \(S_x\) gets computed as

\[
E[S_x] = \sum_{s=1}^{M_n} \left[ MET_{x,s} \right] \star \left[ \mathcal{A}_{x,s} \right] \quad \forall p_x \in \tilde{H}, \quad \left[ \mathcal{A}_{x,s} \right] \neq 0 \tag{29}
\]

\[
\text{Var}[S_x] = \sum_{s=1}^{M_n} \left[ VET_{x,s} \right] \star \left[ \mathcal{A}_{x,s} \right] \quad \forall p_x \in \tilde{H}, \quad \left[ \mathcal{A}_{x,s} \right] \neq 0 \tag{30}
\]

Since, each processor has different number of idle slots due to the precedence-constraints and merged execution of many jobs simultaneously, the sum of idle slots \(l_k\) on processor \(p_k\) becomes equal to the difference between the processor ready time \(PRT_{x,s}\) after execution of all scheduled tasks and the sum of the execution times \(S_x\) on processor \(p_x\). As, \(RT_{i,j}\) and \(S_x\) both follows the normal distribution, the distribution of the difference of both random numbers will also follow the normal distribution [31]. Therefore, \(l_k\) also follows the normal distribution with expectation and variance using eq. (21) being written as

\[
E[l_k] = E[PRT_{x,s}] - E[S_x] \tag{31}
\]

\[
\text{Var}[l_k] = \text{Var}[PRT_{x,s}] + \text{Var}[S_x] \tag{32}
\]

Further, the length of the idle slots \(l_k\) follows the normal distribution. Accordingly, the energy consumption due to idle slots also follows the normal distribution using equation (28).

In order to compute the energy consumed by task \(t_{ij}\), \(W^m\) is replaced with the execution time of task at allocated voltage level. According to the closure property of the linear transformation (eq. (28)), the energy consumption of task \(t_{ij}\) follows the normal distribution indicating that the sum of energy consumption of all tasks \(t_{ij} \in \beta\) will also follows the same. Therefore, the expectation and variance of the energy consumption \(\langle E_N \rangle_{\beta}\) can be computed using eqs. (14), (21), (28), (31) and (32) as

\[
E[\langle E_N \rangle_{\beta}] = \sum_{r=1}^{l_k} \sum_{t_{s=1}^{M_n}} \sum_{x=1}^{M} \sum_{z=1}^{Vol_x} \text{Pow}_{xz}^0 \star [MET_{r,s}^x] \star \mathcal{A}_{x,s} \\
+ \sum_{x=1}^{M} \text{Pow}_{xz}^0 \star E(l_k) \tag{33}
\]

\[
\text{Var}[\langle E_N \rangle_{\beta}] = \sum_{r=1}^{l_k} \sum_{t_{s=1}^{M_n}} \sum_{x=1}^{M} \sum_{z=1}^{Vol_x} \text{Pow}_{xz}^0 \star [VET_{r,s}^x] \star \mathcal{A}_{x,s} \\
+ \sum_{x=1}^{M} \text{Pow}_{xz}^0 \star \text{Var}(l_k) \tag{34}
\]

4.5. Stochastic HypE scheduler

The Stochastic HypE Scheduler is presented in Algorithm 3. Algorithm 3 takes batch \(\beta\), HCS \(H\), maximum number of generations \(g_{\text{max}}\), number of sampling points \(M\) and population size \(q\) as inputs and returns the approximate Pareto-optimal set of solutions \(\{F_1(g_{\text{max}})\}\) with minimum expected turnaround time and
energy consumption. Algorithm 3 uses algorithm 1 to generate genotype structure, Task & Processor Look-up tables and a random population Q(t) initialized at t = 0 with $q$ chromosomes. To assign the voltage level to each task $i$ in $Q(t)$, first a processor $p_i \in H$ is chosen arbitrarily and then one of the voltage levels $v_{p_i} \in \text{Volt}_i$ is selected randomly. For each chromosome $q(t) \in Q(t)$, stochastic turnaround time and energy consumption is computed using genotype structure, Look-up tables, Algorithm 2 and equations (33) and (34). Next, the population $Q(t)$ is divided into disjoint fronts using principle of non-dominating sort [29]. The last front $(F_l(t))$ of the population consists of the chromosomes that are dominated by all fronts $F_k(t)$, $k < l$. The hypervolume of Pareto-optimal front $F_k(t)$ with respect to $F_l(t)$ and distant reference set R will have different values. As front $F_k(t)(l < k)$ approaches the true Pareto-front, the hypervolume of $F_k(t)$ will be increasing with respect to both $F_l(t)$ and reference set R. Thus, the objective space of last fronts $F_l(t) \subset Q(t)$ can be used to generate the reference set R. Let $P^R(t) \subset Q(t)$ consists of $q/5$ worst chromosomes and $Z_0^R$ represents the objective space of $P^R(t)$. Assume, $Z_0^R$ to be another objective space defined as

$$Z_0^R = \{(z_1, z_2, ..., z_q) : z_i \geq u_i\}$$  \hspace{1cm} (35)

where $u_i = \max_{(z_1, z_2) \in Z_0^R} z_i, i = 1, (q/5)$

The axis-aligned minimum bounding box containing $Z_0^R$ and $Z_0^R$ are used to generate the reference set R as,

$$R = \{(z_1, z_2, ..., z_q) : l_i \leq z_i \leq u_b\}$$  \hspace{1cm} (36)

where $l_i = \min_{q(t) \in P^R(t)} z_i(t)$ and $u_b = \max_{(z_1, z_2) \in Z_0^R} z_i(t)$

Next, Algorithm 3 employs the successive application of mating selection, variation and environmental selection for $g_{\text{max}}$ generations. It selects a population $Q(t+1)$ with size $(q/5)$ from $Q(t)$ by using tournament selection based on hypervolume indicator of the chromosomes. It then performs the variation operator to form $Q(t+1)$ using single point crossover with crossover probability $(\text{Prob}_c)$ and mutation operator with mutation probability $(\text{Prob}_m)$. Following this, the stochastic turnaround time and energy consumption are computed for each $q(t) \in Q(t)$ and $q(t) \in Q(t+1)$. The environmental selection is performed to select best individuals in $Q(t+1)$ from $Q(t) \cup Q(t+1)$ by using principle of non-dominating sort based on hypervolume indicator. Next, $|F_l(t+1)|$ or the one-tenth worst chromosomes of the population $(q/10)$ are rejected and are replaced with $|F_l(t+1)|$ or $q/10$ chromosomes of the first Pareto-front.

5. Performance analysis

In order to evaluate the performance of the proposed scheduler, simulation was done using MATLAB over Sun Fire X4470 server and Lenovo-THINK machine with Intel® Core™ i5-3470. The codes for NSGA-II and HypE are taken from Refs. [32,33] respectively. The performance of the proposed stochastic scheduler is compared with ECS [9], HEFT [34], SHEFT [22] and NSGA-II [35]. This section is divided into two sections experimental tuning and experimental results.

5.1. Experimental tuning

This section describes the execution environment employed, random batch creation method, compatibility of HEFT and ECS with stochastic scheduling, different versions of ECS [9], SHEFT [24] and HEFT [34] and parameter control mechanism for HypE [7] and NSGA-II [35].

5.1.1. Execution environment

In order to realize the proposed scheduling model, six types of processors Intel® Core™2 Extreme with 4 frequencies, Intel® Core™2 Duo with 4 frequencies, Intel® Core™2 Quad with 5 frequencies, advanced micro devices (AMD) Athlon™ accelerated processing units (APUs) with 3 frequencies, AMD Sempron™ APUs with 3 frequencies and TI digital signal processor (DSP) with 2 frequencies have been simulated. Using these processors, two execution environments, Environment-1 and Environment-2 were created. Environment-1 consists of 18 processors with 3 processors of each type whereas Environment-2 includes 48 processors with 8 processors of each type.

5.1.2. Randomly generated batch

To evaluate the performance of the proposed scheduler, a batch $\beta$ is created consisting of random number of jobs (DAGs) as well as tasks with the execution time and communication times being normally distributed. The determination of probability distribution function of execution and communication time involves two steps i.e. profiling and deriving the probability distribution function for which a number of techniques exist based on Pearson distribution, using code profiling and statistical techniques. This work employs statistical prediction technique which has been used to get the normal distribution of execution and communication times [24]. The procedure to create the batch $\beta$ takes seven fundamental characteristics as,

\begin{itemize}
  \item $I$ - The number of stochastic jobs in a batch $\beta$
  \item $[L_{\text{min}}, L_{\text{max}}]$ - The range of number of stochastic tasks in each job $J \in \beta$
  \item $[B_{\text{min}}, B_{\text{max}}]$ - The range of height/levels of any job $J \in \beta$
  \item $[E_{\text{min}}, E_{\text{max}}]$ - The range of values of communication times of tasks
  \item $[C_{\text{min}}, C_{\text{max}}]$ - The range of values of communication times of tasks
  \item $[C_{\text{min}}, C_{\text{max}}]$ - The range of values of communication times of tasks
\end{itemize}

For batch $\beta$, all I jobs $J_1, ..., J_I$ are created iteratively. In order to generate job $J_j \in \beta$, first of all, the maximum number of levels in job $J_j$ is created randomly and the tasks as well as being distributed to the levels randomly. Next, the following two steps are followed [36].

\textbf{Step 1-Generate Edges}

For each pair of task $(t_{ij}, t_{ik}) \in J_j$, generate random probability $(\text{Prob}_{(ij,k)})$ of having edges between $t_{ij}$ and $t_{ik}$ tasks. If the probability $(\text{Prob}_{(ij,k)})$ is greater than the probability of not having edge $(1 - \text{Prob}_{(ij,k)})$, then there exists an edge $e_{(i,j,k)}$ between $t_{ij}$ and $t_{ik}$ tasks, otherwise not.

\textbf{Step 2-Remove Redundant Edges}

In the first step, some edges are created between tasks at non-consecutive levels. These edges are redundant there because data can be communicated if a level-by-level path exists between these tasks. These edges also make the scheduling problem more and more complex. If an edge $e_{(i,j,k)}$ exists between $t_{ij}$ and $t_{ik}$ tasks at two non-consecutive levels i.e. $L_{ik} \geq (L_{ij} - 1)$, then edge $e_{(i,j,k)}$ is removed. If there does not exist a path $(t_{ij}, ..., t_{ik})$ between tasks $t_{ij}$ and $t_{ik}$, then a level-by-level path $(t_{ij}, ..., t_{ik})$ is created.
After generating the structure of whole batch $\beta$, the execution and communication times of tasks are generated using input parameters.

5.1.3. Compatibility of HEFT and ECS with stochastic scheduling

In order to schedule stochastic jobs using deterministic scheduling algorithms viz. HEFT [34] and ECS [9], it is required to make these algorithms adaptable to stochastic scheduling problem by re-considering the two metrics i.e. bottom level (b-level) and relative superiority metric (RS). The b-level of task $t_{ij}$ is the longest path from $t_{ij}$ to any of the sink task [37]. For different execution environments, three types of b-level of tasks have been defined viz. b-level of deterministic time tasks over homogeneous system [37], heterogeneous system [34] and b-level of stochastic tasks on heterogeneous system [22]. The b-level of stochastic task $t_{ij} \in B$ over DVFS-enabled HCS is the random longest path from a task to a sink task and is recursively defined as,

$$b^{s - \text{level}}(t_{ij}) = \text{AE}(t_{ij}) + \max \{ e_{i(j,k)} + b^{s - \text{level}}(t_{i,k}) : t_{i,k} \in \text{Succ}_j \}$$  \hspace{1cm} (37)

where $\text{AE}(t_{ij})$ is the average execution time as given in eqs. (5) and (6).

As, average execution time of task ($\text{AE}(t_{ij})$) and communication times are random and both follow the normal distribution, $b^{s - \text{level}}(t_{ij})$ also approximately follows the normal distribution and its expectation and variance are computed using eqs. (24) and (27). To generate the order of tasks w.r.t. $b^{s - \text{level}}(t_{ij})$, the well-established stochastic results are used. It states that the performance of the stochastic scheduling is affected by the expectation as well as variance [20]. Therefore, the approximate deterministic value $\text{DV}(X)$ of a random number $X$ using mean and variance can be determined as

$$\text{DV}(X) = \begin{cases} E[X] + \sqrt{\text{Var}[X]} & \text{if } \frac{\text{Var}[X]}{E[X]^2} \leq 1 \\ E[X] \left(1 + \frac{1}{\sqrt{\text{Var}[X]}}\right) & \text{otherwise} \end{cases}$$  \hspace{1cm} (38)

The $\text{RS}$ measure the degree of efficiency relative to the task execution time and effectively deals with trade-off between energy consumption and execution time [9], equation (38) has been used to determine the $\text{DV}$ values of each task $t_{ij} \in B$ w.r.t. each $p_{rs} \in P_k$ and communication times. Using these values, the computation of stochastic RS metric is straightforward and the scheduling decision for each of the ECS and HEFT can be taken easily.

5.1.4. HEFT-1, HEFT-2, ECS-1, ECS-2, SHEFT-1, SHEFT-2

HEFT [34], SHEFT [22] and ECS [9] algorithms schedule only one single job (DAG) on HCS H. Analogous to HEFT, SHEFT and ECS, HEFT-1, SHEFT-1 and ECS-1 algorithms schedule jobs from batch $\beta$ one-by-one randomly whereas HEFT-2, SHEFT-2 and ECS-2 converts the whole batch $\beta$ into single DAG by adding the pseudo entry and exit tasks with zero execution and communication times and schedule tasks of the pseudo job.

5.1.5. Parameter control mechanism for HypE and NSGA-II

The meta-heuristic algorithms are sensitive to the involved parameters due to interdependencies that affect the exploration, exploitation capabilities of the meta-heuristic algorithms and quality of the solutions obtained [38]. Therefore, it is required to either control or fine tune the involved parameters in order to find the better solutions with low computational efforts. As presented in Table 3, five batches with different number of jobs, tasks, levels, execution times and communication times have been generated to adjust the parameters of HypE and NSGA-II. The algorithms are employed to solve each of the five batch instances using mutation probability ($\text{Prob}_m$) either fixed to $1/|B|$ or randomly chosen from $[0.5, 1/|B|]$, whereas the crossover probability ($\text{Prob}_c$) is randomly chosen from $[0.7, 0.9]$. For repetitive simulations, the responses in each run offers different results for four performance parameters.

- Hypervolume indicator ($h(F_t(s_{\text{max}}))$) – This measure is originally proposed by Zitzler and Thiele that compares the outcomes of evolutionary multi-objective algorithms [39]. A large value of the indicator for first non-dominated front after $g_{\text{max}}$ generation is required.
- NPr–This measure represents the mean to number of Pareto-optimal solution in $F_t(s_{\text{max}})$ for $r$ runs of the algorithms.
- Expected Turnaround time ($\text{TAT}_{\beta}$) – As computed by Algorithm 2.
- Expected Energy Consumption($\text{EN}_p$) - As defined in equations (33) and (34).

From simulation results, it is observed that decreasing the mutation probability ($\text{Prob}_m$) from 0.5 to $1/|B|$ offers the best results. The parameters of both algorithms based on the results obtained are the population size ($q$) belonging to [100, 250], the optimal number of generations ($g_{\text{max}}$) fitting in [30, 200], the crossover probability ($\text{Prob}_c$) ranging in $[0.7, 0.9]$. The adaptive mutation probability at generation $t$ can be written as

$$\text{Prob}_m(t) = \left\{ \begin{array}{ll}
2 + \frac{\log t - 2}{\text{Max}}^{-1} & \text{if } h(F_t(t)) \geq h(F_t(t-1)) \text{ or } ACD(F_t(t)) \geq ACD(F_t(t-1)) \\
\text{Prob}_m(t - 1) & \text{otherwise} 
\end{array} \right. $$  \hspace{1cm} (39)

To decrease the mutation probability, the hypervolume indicator $h(F_t(t))$, is used by HypE whereas average crowding distance $\text{ACD}(F_t(t))$ is used in NSGA-II. The $\text{ACD}(F_t(t))$ is the average of

<table>
<thead>
<tr>
<th>Batch No.</th>
<th>$\beta_1$</th>
<th>$\beta_2$</th>
<th>$\beta_3$</th>
<th>$\beta_4$</th>
<th>$\beta_5$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\beta_1$</td>
<td>5 20 5</td>
<td>[10, 20]</td>
<td>$[2^2, 3^2]$</td>
<td>[5, 10]</td>
<td>$[1^2, 2^2]$</td>
</tr>
<tr>
<td>$\beta_2$</td>
<td>10 30 8</td>
<td>[10, 40]</td>
<td>$[2^2, 4^2]$</td>
<td>[8, 12]</td>
<td>$[1^2, 2^2]$</td>
</tr>
<tr>
<td>$\beta_3$</td>
<td>15 40 10</td>
<td>[10, 50]</td>
<td>$[2^2, 3^2]$</td>
<td>[10, 18]</td>
<td>$[1^2, 2^2]$</td>
</tr>
<tr>
<td>$\beta_4$</td>
<td>20 40 12</td>
<td>[20, 60]</td>
<td>$[1^2, 5^2]$</td>
<td>[8, 19]</td>
<td>$[2^2, 4^2]$</td>
</tr>
<tr>
<td>$\beta_5$</td>
<td>25 40 12</td>
<td>[10, 70]</td>
<td>$[2^2, 4^2]$</td>
<td>[6, 20]</td>
<td>$[2^2, 3^2]$</td>
</tr>
</tbody>
</table>
crowding distance of the solutions in the first front [35]. The increment in both \( h(F_1(t)) \) and \( ACD(F_1(t)) \) represents growth in either the number of solutions or the quality of solutions.

5.2. Experimental results

To evaluate the performance of the stochastic HypE scheduler, the experimental tuning is achieved as discussed in Section 5.1 and the tuned parameters required for the simulation are shown in Table 4. For the simulation study as discussed earlier, two environments E1 and E2 consisting of 18 and 48 processors and the batches of 10, 50 and 200 jobs are considered with tasks ranging between 32 and 256. Table 5 presents the specifications of 6 randomly generated problem instances with number of tasks and the environment considered. Problem instances P1 and P2 represent the batches of small size, P3 and P4 of middle size whereas P5 and P6 of large batch size. The results (Pareto-fronts) corresponding to 6 problem instances are shown in Figs. 7–12 respectively. For each of the problem instance, ECS-1, ECS-2, SHEFT-1, SHEFT-2, HEFT-1 and HEFT-2 are run for 100 times and the corresponding data point is the average of the data obtained from 100 experiments whereas the results obtained for HypE and NSGA-II are obtained from the first front using same population size (\( \bar{N} \)), crossover (Probc) and mutation (Probm) probabilities and the same number of generations (\( g_{\text{max}} \)).

Figs. 7 and 8 show the Pareto-fronts of HypE, NSGA-II, ECS-1, ECS-2, SHEFT-1, SHEFT-2, HEFT-1 and HEFT-2 on DVFS-enabled HCS of 18 and 48 processors i.e. E1 and E2 for the batches of 1458 and 1425 tasks respectively. It is observed from Figs. 7(a) and 8(a) that HypE and NSGA-II, which combines the jobs to exploit the parallelism and take the random attributes such as expectation and variance into account are far better than single job scheduling algorithms ECS-1, ECS-2, SHEFT-1, SHEFT-2, HEFT-1 and HEFT-2 in terms of turnaround time and energy consumption. Further, it is observed from the Pareto-fronts given in Figs. 7(b) and 8(b) that HypE and NSGA-II offers more stable performance due to the smaller standard deviations of turnaround time and energy consumptions. Further, as can be seen from Fig. 7(a) for problem instance P1 that the proposed stochastic HypE scheduler significantly outperforms NSGA-II by 9.9%, ECS-1 by 11.65%, ECS-2 by 10.37%, SHEFT-1 by 10.37%, SHEFT-2 by 10.04%, HEFT-1 by 11.675% and HEFT-2 by 10.02% in terms of average turnaround time. Even in terms of average energy consumption, it outperforms NSGA-II by 11.25%, ECS-1 by 12.28%, ECS-2 by 13.34%, SHEFT-1 by 13.5%, SHEFT-2 by 11.64%, HEFT-1 by 13.83% and HEFT-2 by 11.27%. For problem instance P2 also as shown in Fig. 8(a), it is seen that the proposed stochastic HypE scheduler exhibits a much better performance over NSGA-II by 10.33%, ECS-1 by 10.99%, ECS-2 by 10.35%, SHEFT-1 by 10.67%, SHEFT-2 by 10.24%, HEFT-1 by 13.34% and HEFT-2 by 10.64% in terms of average turnaround time and outperforms NSGA-II by

<table>
<thead>
<tr>
<th>Serial No.</th>
<th>System parameters</th>
<th>Notation used</th>
<th>Value/Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>#Jobs in Batch</td>
<td>#I</td>
<td>[10, 50, 100]</td>
</tr>
<tr>
<td>2</td>
<td>Number of tasks in job</td>
<td>([I_{\text{Min}}, I_{\text{Max}}])</td>
<td>[32–256]</td>
</tr>
<tr>
<td>3</td>
<td>Number of levels</td>
<td>([L_{\text{Min}}, L_{\text{Max}}])</td>
<td>(5, 50)</td>
</tr>
<tr>
<td>4</td>
<td>Execution times</td>
<td>([E_{\text{m Min}}, E_{\text{m Max}}]), ([E_{\text{s Min}}, E_{\text{s Max}}])</td>
<td>[1, 3000][20, 1000]</td>
</tr>
<tr>
<td>5</td>
<td>Communication times</td>
<td>([C_{\text{m Min}}, C_{\text{m Max}}]), ([C_{\text{s Min}}, C_{\text{s Max}}])</td>
<td>[1, 500][20, 100]</td>
</tr>
<tr>
<td>6</td>
<td>Population Size</td>
<td>(d)</td>
<td>[100, 250]</td>
</tr>
<tr>
<td>7</td>
<td>#Generations</td>
<td>(X_{\text{max}})</td>
<td>[30, 80]</td>
</tr>
<tr>
<td>8</td>
<td>Crossover Probability</td>
<td>Probc</td>
<td>[0.7, 0.9]</td>
</tr>
<tr>
<td>9</td>
<td>Mutation probability</td>
<td>Probm</td>
<td>Eq. (39)</td>
</tr>
<tr>
<td>10</td>
<td>Number of Samples</td>
<td>(M_{\text{HypE}})</td>
<td>10000</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Problem No.</th>
<th>Number of jobs in ( \beta )</th>
<th>Environment</th>
<th>Randomly generated tasks((I_b))</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>10</td>
<td>E-1</td>
<td>1458</td>
</tr>
<tr>
<td>P2</td>
<td>10</td>
<td>E-2</td>
<td>1425</td>
</tr>
<tr>
<td>P3</td>
<td>50</td>
<td>E-1</td>
<td>7145</td>
</tr>
<tr>
<td>P4</td>
<td>50</td>
<td>E-2</td>
<td>7269</td>
</tr>
<tr>
<td>P5</td>
<td>200</td>
<td>E-1</td>
<td>28808</td>
</tr>
<tr>
<td>P6</td>
<td>200</td>
<td>E-2</td>
<td>28768</td>
</tr>
</tbody>
</table>
The main reason behind the good performances of both HypE and NSGA-II in scheduling is that both algorithms combine all the jobs into a single chromosome to exploit the parallelism between and within the jobs of the batch. Further, both algorithms consider the expected communication and expected execution times with respect to all voltage levels and produce better scheduling strategies using non-dominated sorting for the stochastic scheduling.
However, remaining six algorithms schedules either one-by-one or using pseudo source/sink tasks that result in less parallelism exploration and exploitation between and within the jobs. It is to be noted that, HEFT-1 and HEFT-2 are non-energy aware and deterministic scheduling algorithms whereas SHEFT-1 and SHEFT-2 are non-energy aware and single job algorithms making them less suitable for batch of stochastic jobs scheduling problem. Further, the same is true for ECS-1 and ECS-2 being deterministic and single job scheduling algorithms. In general, HypE and NSGA-II are better than other six algorithms due to the following reasons:

- The use of newly defined chromosomes structure (Section 4.2) helps in better exploitation of the parallelism within and between the jobs.
- Both HypE and NSGA-II employs novel stochastic turnaround time model and parameter tuning mechanism which assists in finding the good solutions in the search space with lower time complexity.
- Both the meta-heuristics uses the Clark's equations that handle normally distributed numbers in an efficient manner resulting in better scheduling policies.

Between HypE and NSGA-II, it is observed from Figs. 7(a) and 8(a) that the proposed HypE based scheduler performs better than NSGA-II in terms of turnaround time, energy consumption and the number of solutions in the first front. This is primarily because HypE employs the hypervolume indicator for non-dominated sorting and iterative approach to remove the solution strategies from the population in comparison to NSGA-II which employs greedy approach (one shot approach) to discard the solutions.

The improvement in the performance by employing HypE in comparison to NSGA-II, ECS-1, ECS-2, SHEFT-1, SHEFT-2, HEFT-1 and HEFT-2 can also be seen from Figs. 9–12. Figs. 9 and 10 depicts the simulation results for batches of stochastic jobs of size 50 over environments E1 and E2 respectively with 7145 and 7269 tasks. Figs. 11 and 12 also represent the Pareto-fronts for batches of stochastic jobs of size 200 over environments E1 and E2 respectively with 28808 and 28768 tasks. The Pareto-fronts shown in Figs. 9(a), 10(a), 11(a) and 12(b) clearly expresses the better performance of the HypE based scheduler over NSGA-II, ECS-1, ECS-2, SHEFT-1, SHEFT-2, HEFT-1 and HEFT-2 in terms of expected energy consumption and turnaround time. The Pareto-front for standard deviation of energy consumption and turnaround time given in Figs. 9(b), 10(b), 11(b) and 12(b) too indicate a more stable
performance of the HypE over the peers due to the smaller values of standard deviations. The Pareto-fronts in Fig. 9(a) and (b) demonstrate that the use of HypE outperforms NSGA-II by 10.11%, ECS-1 by 10.90%, ECS-2 by 10.03%, SHEFT-1 by 10.45%, SHEFT-2 by 10.27%, HEFT-1 by 12.90% and HEFT-2 by 10.54% in terms of average turnaround time whereas HypE outperforms NSGA-II by 11.04%, ECS-1 by 11.66%, ECS-2 by 11.47%, SHEFT-1 by 11.7%, SHEFT-2 by 11.50%, HEFT-1 by 11.78% and HEFT-2 by 11.56% in terms of average energy consumption. The performance improvement for P4, P5 and P6 is also significant and encouraging. It can be observed from Figs. 9(a), 10(a), 11(a) and 12(a) that stochastic HypE scheduler offers more number of solutions and stable performance due to smaller standard deviation of energy consumption and turnaround time. From the results given in Figs. 9—12 it can be seen that as the size of batch increases, the performance of HypE becomes even better than its peers further improving the scheduler’s performance.

From the experimental results, it can be concluded that combining the jobs into batches with incorporation of random variables into consideration has major impact on the performance of large providers such as data centres for the stochastic scheduling problem. It has been confirmed by the experiments that the proposed stochastic HypE scheduler significantly outperforms NSGA-II by 11.36%, ECS by 14.89% and SHEFT by 13.67% in terms of average turnaround time and outperforms NSGA-II by 11.85%, ECS-1 by 13.74%, ECS-2 by 12.48%, SHEFT-1 by 13.11%, SHEFT-2 by 11.24%, HEFT-1 by 13.48% and HEFT-2 by 12.46% in terms of average energy consumption.

(6) As the number of DVFS-enabled processors increases from 18 to 48 (Problem instance P2, P4 and P5), it is expected that the turnaround time and energy consumption will decrease. Although, the energy consumption and execution time of task remains constant, the precedence constraints between tasks results in the delay in execution that results in larger turnaround time and more energy consumption in longer idle slots.

(7) It is also observed from the population of HypE and NSGA-II that most of the genes in the chromosomes employ the adjacent voltage levels. This observation supports the observation reported in Refs. [8,11].

(8) For six problem instances given in Table 5, Table 6 represents the hypervolume contribution and average number of solutions for 30 simulation runs employing HypE and NSGA-II. As can be seen, HypE offers larger hypervolume and more number of solutions in comparison to NSGA-II.

(9) It is observed from simulation study that the order of time complexity for the employed algorithms is HEFT-2, SHEFT-2 ECS-2, HEFT-1, SHEFT-1, ECS-1, NSGA-II and HypE. To mitigate the time overheads of HypE, the quantum-inspired population and some local search method viz. variable neighbourhood search can also be applied e.g. Memetic Algorithms.

6. Conclusion

The work proposes an energy aware stochastic HypE scheduler for a batch of stochastic precedence-constrained jobs on Dynamic Voltage and Frequency Scaling-enabled processors. The work incorporates a stochastic precedence-constrained encoding model, turnaround time and energy models which helps in generating the higher quality schedules while exploiting the available job and sub-job level parallelism in the batch. It has been confirmed by the experiments that the proposed stochastic HypE scheduler significantly outperforms NSGA-II by 11.36%, ECS by 14.89% and SHEFT by 13.67% in terms of average turnaround time and outperforms NSGA-II by 11.85%, ECS-1 by 13.74%, ECS-2 by 12.48%, SHEFT-1 by 13.11%, SHEFT-2 by 11.24%, HEFT-1 by 13.48% and HEFT-2 by 12.46% in terms of average energy consumption.

<table>
<thead>
<tr>
<th>Problem No.</th>
<th>$h(F_i(S_{\text{hypE}}))$</th>
<th>$N_{\text{P}}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>100 (90.69)</td>
<td>8.12/7.2</td>
</tr>
<tr>
<td>2</td>
<td>100 (93.71)</td>
<td>50.2/49.48</td>
</tr>
<tr>
<td>3</td>
<td>100 (92.87)</td>
<td>12.98/11.8</td>
</tr>
<tr>
<td>4</td>
<td>100 (87.15)</td>
<td>12.56/10.34</td>
</tr>
<tr>
<td>5</td>
<td>100 (89.31)</td>
<td>8.14/7.7</td>
</tr>
<tr>
<td>6</td>
<td>100 (95.64)</td>
<td>51.68/46.6</td>
</tr>
</tbody>
</table>
Algorithms

Algorithm 1: Encoding (Phenotype to Genotype).

<table>
<thead>
<tr>
<th>Algorithm 1: Encoding (Phenotype to Genotype)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Input:</strong> Batch β</td>
</tr>
<tr>
<td><strong>Output:</strong> Chromosome with size $I_p$</td>
</tr>
<tr>
<td><strong>Begin</strong></td>
</tr>
<tr>
<td>Divide batch β into the levels from 1 to $L_p$</td>
</tr>
<tr>
<td>Starting from level 1 to $L_p$, put a random permutation of tasks of each level in chromosome</td>
</tr>
<tr>
<td>Assign consecutive number to each task $t_{ij} \in \beta$ in the chromosome from 1 to $I_p$</td>
</tr>
<tr>
<td>Create Task Look-up Table</td>
</tr>
<tr>
<td>Create Processor Look-up Table</td>
</tr>
<tr>
<td><strong>End</strong></td>
</tr>
</tbody>
</table>

Algorithm 2: Stochastic Turnaround time model.

<table>
<thead>
<tr>
<th>Algorithm 2: Stochastic Turnaround time model.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Input:</strong> Chromosome S, Tasks &amp; Processors Look-up tables</td>
</tr>
<tr>
<td><strong>Output:</strong> Expected turnaround time of batch β, $TAT_3$</td>
</tr>
<tr>
<td><strong>Begin</strong></td>
</tr>
<tr>
<td>Scan S and use task and processor Look-up tables to create the phenotype structure</td>
</tr>
<tr>
<td>From phenotype structure, add task $t_{ij}$ to queue $Q(t)$ assigned to processor $p_x \in \beta$</td>
</tr>
<tr>
<td>Assign $predCount_{ij} = \lfloor Pred_{ij} \rfloor$ to each task $t_{ij}$</td>
</tr>
<tr>
<td>For each level from 1 to $L_p$</td>
</tr>
<tr>
<td>For each processor $p_x \in \beta$</td>
</tr>
<tr>
<td>Do</td>
</tr>
<tr>
<td>Find $r, s$ for $i, j, x$ using $S, {A_{ijk}}$, eqs. (3) and (4)</td>
</tr>
<tr>
<td>$RT_{ijx} = PRT_x$ if $Pred_{ij} = \emptyset$ or $\lfloor Pred_{ij} \rfloor = 1 &amp; t_{ik} \in \lfloor Pred_{ij} \rfloor \cup \lfloor Proc_{ik} \rfloor = p_x$</td>
</tr>
<tr>
<td>$RT_{ijx} = \max{\text{PRT}<em>x, \max{FT</em>{ijy} + e_{ijy}; t_{ik} \in \lfloor Pred_{ij} \rfloor \cup \lfloor Proc_{ik} \rfloor = p_x}}$ otherwise</td>
</tr>
<tr>
<td>$FT_{ijx} = RT_{ijx} + [\text{MET}_{ij}</td>
</tr>
<tr>
<td>$\forall t_{ik} \in \text{Succ}<em>{ij}$, Send data and reduce $predCount</em>{ik}$ by 1</td>
</tr>
<tr>
<td>Until there is task $t_{ij} \in Q(t)$ with $\lfloor predCount_{ij} \rfloor = 0$</td>
</tr>
<tr>
<td>End for</td>
</tr>
<tr>
<td><strong>End for</strong></td>
</tr>
<tr>
<td>$TAT_3 = \max{FT_{ijx}; \forall t_{ij}, \lfloor Succ_{ij} \rfloor = \emptyset}$</td>
</tr>
<tr>
<td>Return $TAT_3$</td>
</tr>
<tr>
<td><strong>End</strong></td>
</tr>
</tbody>
</table>


<table>
<thead>
<tr>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Input:</strong> Batch β, HCS H, #generations $g_{max}$, #sampling points $M_{HypE}$, population size $q$</td>
</tr>
<tr>
<td><strong>Output:</strong> Set of approximate Pareto optimal solution ($F_1(g_{max})$)</td>
</tr>
<tr>
<td><strong>Begin</strong></td>
</tr>
<tr>
<td>1. Generate genotype structure, Task &amp; Processor Look-up tables using Algorithm 1</td>
</tr>
<tr>
<td>2. Initialize random population $Q(t) = {q_1(t), q_2(t), q_3(t) \ldots q_q(t)}$ at $t = 0$ using genotype structure, Task and Processor Look-up tables</td>
</tr>
<tr>
<td>3. For population $Q(t)$, compute stochastic turnaround time and energy consumption for each $q_i(t) \in Q(t)$ using Algorithm 2 and eqs. (33) and (34)</td>
</tr>
<tr>
<td>4. Find Pareto-fronts $F_i(t), i = 1 \ldots l$, using non-dominated sort</td>
</tr>
<tr>
<td>5. Generate reference set $R$ for batch β using last front $F_1(t)$ and eq. (36)</td>
</tr>
<tr>
<td>6. Do steps 7–13</td>
</tr>
<tr>
<td>7. $t = t + 1$</td>
</tr>
<tr>
<td>8. Perform mating selection to form $Q'(t + 1)$ using $Q(t), R, M_{HypE}$ and hypervolume indicator</td>
</tr>
<tr>
<td>9. Perform variation operator to form $Q'(t + 1)$ by single point crossover with crossover probability (ProbC). Apply mutation operator with mutation probability (ProbM)</td>
</tr>
<tr>
<td>10. Compute stochastic turnaround time and energy consumption using Algorithm 2 and eqs. (35) and (36) for $q_i(t) \in Q(t)$ and $q_i'(t) \in Q'(t + 1)$</td>
</tr>
<tr>
<td>11. Perform environmental selection to form $Q(t + 1)$ on $Q(t) \cup Q'(t + 1)$ by using non-dominated sort with hypervolume indicator</td>
</tr>
<tr>
<td>12. Reject $F_i(t + 1)$ or one tenth worst candidates (i.e. $q/10$) of $Q(t + 1)$ and add randomly selected $F_i(t + 1)$ or $q/10$ candidates from first Pareto-front $F_1(t + 1)$</td>
</tr>
<tr>
<td>13. if $t &lt; g_{max}$ go to step 5 else terminate</td>
</tr>
<tr>
<td>14. Return $F_1(g_{max})$</td>
</tr>
<tr>
<td><strong>End</strong></td>
</tr>
</tbody>
</table>
References


Dr. Mohammad Sajid is a faculty in the Department of Computer Science, Aligarh Muslim University, India. He received Ph.D., M. Tech. and M.C.A. degrees in computer science from the School of Computer and Systems Sciences, Jawaharlal Nehru University, New Delhi, India. His research interests include high performance computing, scheduling/stochastic scheduling, evolutionary algorithms and multi-objective evolutionary algorithms.

Dr. Zahid Raza is a faculty in the School of Computer and Systems Sciences, Jawaharlal Nehru University, India. He is M.Sc. in Electronics, M. Tech in Computer Science. He did his Ph.D. in Computer Science from Jawaharlal Nehru University, India. Prior to joining JNU, he served as a Lecturer in Banasthali Vidyapith University, Rajasthan, India. His research interests include parallel and distributed systems, evolutionary algorithms and multi-objective evolutionary algorithms. He has published many peer-reviewed articles and has proposed various scheduling models for computational grid, cloud and cluster systems.