Towards Pareto Descent Directions in Sampling Experts for Multiple Tasks in an On-Line Learning Paradigm

Shaona Ghosh, Chris Lovell and **Steve R. Gunn**
Electronics and Computer Science Department
University of Southampton
Southampton, United Kingdom SO17 1BJ
shaona.ghosh@ecs.soton.ac.uk

Abstract

In many real-life design problems, there is a requirement to simultaneously balance multiple tasks or objectives in the system that are conflicting in nature, where minimizing one objective causes another to increase in value, thereby resulting in trade-offs between the objectives. For example, in embedded multi-core mobile devices and very large scale data centers, there is a continuous problem of simultaneously balancing interfering goals of maximal power savings and minimal performance delay with varying trade-off values for different application workloads executing on them. Typically, the optimal trade-offs for the executing workloads, lie on a difficult to determine optimal Pareto front. The nature of the problem requires learning over the lifetime of the mobile device or server with continuous evaluation and prediction of the trade-off settings on the system that balances the interfering objectives optimally. Towards this, we propose an on-line learning method, where the weights of experts for addressing the objectives are updated based on a convex combination of their relative performance in addressing all objectives simultaneously. An additional importance vector that assigns relative importance to each objective at every round is used, and is sampled from a convex cone pointed at the origin Our preliminary results show that the convex combination of the importance vector and the gradient of the potential functions of the learner's regret with respect to each objective ensure that in the next round, the drift (instantaneous regret vector), is the Pareto descent direction that enables better convergence to the optimal Pareto front.

Introduction

Power is an expensive resource in embedded devices like mobile phones, that exhibit increased computational capacity. There are rising user demands for being able to execute high performing applications on those devices without decreasing the battery lifetime of the device. This presents a challenging multiple objective trade-off problem that of maximum power or battery life savings with minimum performance or application runtime delay (Dhiman and Rosing 2009). With increasing number of cores and hyper-threading capability in such devices, more battery usage and application performance settings are available (Esmaeilzadeh et al. 2012; 2011). With such a wide range of settings, that the system can be configured to, makes it difficult to determine the optimal power-performance trade-off configuration that would enable the optimum balance between the objectives. Hyper-threading on cores also complicates the power-performance challenges by means of making it extremely difficult to analyze the power-performance situation due to the parallel execution of applications with shared locks (Cochran et al. 2011; Meisner et al. 2011). Further, this problem is also exhibited in large scale data centers with increased computational demands. These data centers are provided a power constraint which allows for operation of a certain number of server units. If the power constraint is exceeded, the circuit breaks and causes interruption (Cochran et al. 2011). Typically, the best power-runtime trade-off setting also depends on the workload (a workload indicates the applications executing together on the device or server at any point in time) and this setting may vary during execution of the workload (Cochran et al. 2011). Since the objectives are conflicting in nature, improving one objective is only possible by compromising the other objective: Pareto nature. This implies that the Pareto frontier or the optimal Pareto front contains the optimal runtime values as a function of the power usage values. What is interesting is that every workload has its own Pareto front of optimal power-performance trade-off values when it executes on the system.

Here, we consider the problem of learning and predicting the optimal Pareto front of trade-off values for a sequence of workloads executing on the device or server. We believe that once the optimal Pareto front is identified for a sequence of workloads, this automatically finds the optimal Pareto front for each workload. Since these values depend on the workloads that vary at runtime and that the environment is primarily non-deterministic, the learning happens in an on-line fashion. Also, as the the exhaustive set of power-performance settings, that the system can be possibly configured to are pre-determined, it makes sense to investigate the on-line learning with experts framework (Cesa-Bianchi and Lugosi 2006) in this context, where the learner has no control on the individual expert's prediction.

Related Work

Multiple objective optimization is a well known field. Here, more than one objectives are optimized while satisfying multiple constraints (Fliege and Svaiter 2000). Methods generally used to solve these problems include normal-boundary intersection method, normal constraint method, multiple runs of single-objective optimization for sampling the trade-off surfaces, or use of additional constraints (Singhee 2011). Usually, the additional constraints these methods use, make the optimization problem more time consuming, difficult to solve (Singhee 2011) and hence computationally expensive. This makes such methods unsuitable for the application areas of low-powered embedded devices and power restricted data centers. Evolutionary algorithms have been used to deal with multi-criteria optimization problems where the fitness function evaluates non-domination by other solutions. Typically, these methods use stochastic optimization to approximate the Pareto set or the Pareto front (Bokrantz and Forsgren 2011; Hu, Huang, and Wang 2003; Zitzler, Laumanns, and Bleuler 2004). However, these algorithms have very high computational costs due to lack of strong search directions and maintains many candidate solutions for the purpose convergence. In our problem on embedded mobile devices, we cannot use these approaches again due to the high computational costs associated with them. Adaptive light-weight machine learning methods are good candidates for such problems. On related work in the on-line machine learning community, the tracking of the best expert problem (Herbster and Warmuth 1998) deals with cases where the underlying true distribution changes with each subsequence or partition of data seen sequentially, for a limited number of times. Our work is different from their's in that we track the Pareto optimal set of experts; experts that lie on the Pareto front for a sequence of workloads seen; their best expert is or the set of experts tracked over the whole sequence are not necessarily Pareto optimal. Further, we do not know in advance how many switches are necessary to converge to the best expert for every partition seen. The recent work on on-line multi task learning (Lugosi, Papaspiliopoulos, and Stoltz 2009) does not address conflicting objectives that the learner needs to address, rather focuses on a tuple of actions that the decision maker selects. The other related work on the subset of experts in changing environments by Hazan et. al (Hazan and Seshadhri 2009)- PAC subset selection of bandits (Kalyanakrishnan et al. 2012) and learning experts by Eban et. al (Eban et al. 2012) are different from ours in that they work under different settings and assumptions of the environment.

Contributions

Our main contribution is in the formulation of the tasks of learning multiple objectives and predicting Pareto optimal trade-offs as an on-line learning with experts problem. Consequently, we show how the multiplicative weight update step of the learning algorithm can be significantly improved by exploiting Pareto descent directions that is obtained through a convex combination of the potential function of the regret of the learner with respect to each objective. We devise an importance vector $\boldsymbol{\alpha}$ that is obtained by sampling from a convex cone pointed at the origin of the vector space by exploiting the simultaneous linear inequality of the Blackwell condition (Blackwell 1956; 1954). The convex combination of $\boldsymbol{\alpha}$ and the potential function of the regret of the learner with respect to each objective, enables the learner to converge to the optimal Pareto front while minimizing its regret with respect to each objective.

Preliminaries

From the knowledge of multi-criteria optimization problems (Fliege and Svaiter 2000; Harada and Kobayashi 2006; Singhee 2011); when a point is far from the local optima, search directions can be found that simultaneously optimize the multiple objectives. However, as the point gets closer to the local optima, the search direction cannot optimize both objectives together; the solutions thus obtained are Pareto optimal (Brown and Smith 2003) and they are said to lie on the Pareto front. The Pareto optimal objective vector dominates all other objective vectors in the feasible space of solutions (Hu, Huang, and Wang 2003; Zitzler, Laumanns, and Bleuler 2004).

Definition 1. *Dominance: An objective vector $\boldsymbol{y}_i$ is said to dominate another objective vector $\boldsymbol{y}_j$, ($\boldsymbol{y}_i \succ \boldsymbol{y}_j$), if no component of $\boldsymbol{y}_i$ is worse than the corresponding component in $\boldsymbol{y}_j$ and at least one component is better.*

Definition 2. *Non-Dominated and Pareto Optimal Sets: In the set of feasible solutions Y, if the set of solutions Y' is said to dominate every other solution in Y, then the set Y' is the non-dominated set or the Pareto-optimal set.*

We re-iterate the definition of descent directions and Pareto descent directions from (Harada and Kobayashi 2006) as follows:

Definition 3. *Descent Directions: A descent direction is a direction that falls in the same half-space as the negative gradient of the function to minimize.*

Definition 4. *Pareto Descent Direction: The descent directions in which no other descent directions are superior in improving all objectives together are known as Pareto Descent directions. No Pareto descent direction is better than another Pareto descent direction and all Pareto descent directions are better than all descent directions.*

Problem Formulation

Let T denote finite number of rounds and L denote finite number of experts designated as $\xi_1, \xi_2, \ldots, \xi_L$. The shorthand "expert i" is used to refer to expert ξ_i. Let the number of objectives that the learner has to address in the system be denoted by M. At every round t, the learner is given an input and is asked to predict the appropriate trade-offs for each of the M objectives using the experts predictions that the learner has access to. The prediction of the learner as well the true outcome at round t are both objective vectors of the type $\hat{\mathbf{y}}_t = (y_1, y_2, \ldots, y_M) \in \mathcal{M}, \mathcal{M} \in R^M$. The unknown sequence $\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_T$ of elements form the true *outcome*

space $\mathcal{Y}$. The learner's predictions $\hat{\mathbf{p}}_1, \hat{\mathbf{p}}_2, \ldots, \hat{\mathbf{p}}_T$ belong to decision space $\mathcal{D}$, which is assumed to be a convex subset of a vector space. The experts predictions at round t are given by the set $\{\mathbf{f}_{i,t} : i \in \xi\}$ where, $\mathbf{f}_{i,t} \in \mathcal{D}$. We assume that the experts' predictions belong to a feasible, objective space of vectors that lie on an exhaustive spread of convex Pareto curves. The learner has no knowledge of these underlying Pareto curves where the predictions of the experts(objective vectors) lie and its goal is to converge to the best possible sequence of experts that lie on the Pareto optimal front or true Pareto front. The learner assigns weights or beliefs in the experts given by the vector $\mathbf{w}_{i,t}^m = (w_1, \ldots, w_M) \in \mathbb{R}^M$ for the expert i; each component of the weight vector is the belief the learner has in the expert towards addressing each objective m. The notation of superscript "m" in any symbol is used to indicate which objective is being addressed by the relevant operator. Conversely, the subscript "m" indicates that the operator is involved in the summation over all objectives. The loss function used to measure the learner's loss is a convex loss function **l**. The performance of the learner after T rounds is measured by the notion of regret given by:

$$\mathbf{R}_T^m = \sum_{t=1}^{T} (\mathbf{l}^m(\hat{\mathbf{p}}_t, \mathbf{y}_t) - \mathbf{l}^m(\mathbf{f}_{i,t}, \mathbf{y}_t)) \tag{1}$$

l is used to denote loss in any particular round and the cumulative loss is denoted by $\hat{\mathbf{L}_T}^m = \sum_{t=1}^{T} \mathbf{l}^m(\hat{\mathbf{p}}_t, \mathbf{y}_t)$, where $\hat{\mathbf{L}_t}$ is the learner's cumulative loss in round t and $\mathbf{L}_{i,t}$ is the expert i's cumulative loss in round t. The learning rate is denoted by η. All throughout, vector inequalities are to be understood component-wise. We deal with two objectives in our case as described in the learning algorithm below.

Learning Algorithm

The goal of the learner in this setup is to be able to predict trade-off values for the multiple objectives in the system such that the predictions are optimal. In our example problem discussed in previous sections, by optimal solutions, it is meant that the learner should be able to provide power-performance trade-off solutions such that no alternative solution achieves lower power and shorter runtime than this. The role of the learner then is to make predictions that lie on the Pareto front of solutions. The learner has no idea about the optimum Pareto front and has access only to the predictions of the experts, and the confidence or belief, the learner has in each of the experts. We have seen in the formulation of the problem, the performance of the learner in the on-line learning set-up is evaluated by the notion of regret which is the difference in cumulative loss of the learner and that of the best expert in hindsight as in (1), which should sub-linearly reach zero as shown in (2) (Cesa-Bianchi and Lugosi 2006), also known as Hannan Consistency.

$$\frac{1}{T}(\hat{\mathbf{L}_T}^m - \min_{i,\ldots,L} \mathbf{L}_{i,T}^m) \overset{T\to\infty}{\to} 0 \tag{2}$$

For keeping the learner's regret as low as possible, the instantaneous regret vector is used which is given by $\mathbf{r}_t^m = (r_{1,t}^m, \ldots, r_{L,t}^m) \in \mathbb{R}^L, m \in \mathbb{R}^M$, M is the number of objectives. The instantaneous regret vector is the vector of regret values of each individual expert in that round, which shows how each expert performs with respect to the best expert in hindsight. The corresponding regret vector of the learner is then given by $\mathbf{R}_T^m = \sum_{t=1}^{T} \mathbf{r}_t^m$. Additionally, a monotonically increasing convex potential function is used as a function of the learner's regret $\phi^m(\mathbf{R})$ given by $\phi\colon \mathbb{R}^L \to \mathbb{R}, \Phi(\mathbf{u}^m) = \sum_{i=1}^{L} \phi(u_i^m)$, where $u = (u_1, \ldots, u_L) \in \mathbb{R}^L$ (Cesa-Bianchi and Lugosi 2003) to measure the distance from origin of the learner's regret in a generalized way. With the help of the potential function it is obvious that the learner regret $\mathbf{R}_t$ at round t, will be kept close to the minimum of ϕ by the Blackwell condition.

$$\sup_{y_t \in \mathcal{Y}} \mathbf{r}_t^m . \bigtriangledown \Phi(\mathbf{R}_{t-1}^m) \leq 0 \tag{3}$$

where **u.v** stands for the inner product of two vectors defined by $\mathbf{u.v} = u_1 v_1 + \cdots + u_N v_N$. In other words (3) implies that at round t, by keeping the regret vector to point away from the gradient of the potential function, the point $\mathbf{R}_t$ can be kept close to the minimum of Φ. The potential function is used to keep the drift or instantaneous regret vector in the same half-space as the negative gradient of the potential of learner's regret. Since the learner has no control on the predictions of the experts; it can only control the belief it has in each of the experts in every round and the strategy it uses for making its prediction. This suggests that the way to achieve (3), the belief or weights the learner assigns to each expert is crucial in keeping the regret of the learner low. Intuitively, in every round t, the weight of each of the expert up to previous round: $\mathbf{w}_{i,t-1}^m$ is related to the regret of the expert $\mathbf{R}_{i,t-1}^m$ as $\mathbf{w}_{i,t-1}^m = \phi'(\mathbf{R}_{i,t-1}^m)$ for the ith expert. A larger weight $\mathbf{w}_{i,t-1}$ is assigned to expert i, if $\mathbf{R}_{i,t-1}^m$ is large and vice versa (Cesa-Bianchi and Lugosi 2006). The multiplicative weight update at every round t allows the selection of an instantaneous regret vector $\mathbf{r}_t^m$ for prediction that will keep its regret of the learner low in the next round, the prediction given by:

$$\hat{\boldsymbol{p_t}}^m = \frac{\sum_{i=1}^{L} \bigtriangledown \Phi(\mathbf{R}_{t-1}^m)_i \mathbf{f}_{i,t}}{\sum_{j=1}^{L} \bigtriangledown \Phi(\mathbf{R}_{t-1}^m)_j} \tag{4}$$

Independent Weight Update (IWU)

This is the baseline method where the on-line learning framework is used without any modification to the weight update step of the experts (Cesa-Bianchi and Lugosi 2006). The weights of the experts for each objective in the next round is independently updated based on the expert's performance in the previous round and is given by:

$$\mathbf{w}_{i,t}^m = \frac{\mathbf{w}_{i,t-1}^m e^{-\eta \mathbf{l}^m(\mathbf{f}_{i,t}, \mathbf{y}_t)}}{\sum_{j=1}^{L} \mathbf{w}_{j,t-1}^m e^{-\eta \mathbf{l}^m(\mathbf{f}_{j,t-1}, \mathbf{y}_t)}} \tag{5}$$

In our problem formulation however, as the objectives are conflicting in nature, updating the weight components of the weight vector $\mathbf{w}_t^m$ respective to each objective does not simultaneously decrease the regret of the learner for all objectives. This is because the learner has no information available in terms of how to relatively weigh each objective with

respect to the other; the components of the weight vector $\mathbf{w}_{i,t}^m$ of expert i has information on how the expert fairs individually on the objectives, but does not relate the regret component for one objective with the regret component for the other objective. Without the guidance on relative performance on both objectives as we discuss later, the learning algorithm is not as good as it can be in addressing the objectives together.

Relative Weight Update (RWU)

To avoid the problems of independent weight updates, and to relate the performance of the learner with respect to each objective, we modify the multiplicative weight update step for the experts. Initially, the weight of an expert in a weighted average forecaster is allowed to vary as a convex combination of its regret on individual objectives such that the relative performance of the learner towards both objectives can be influenced by the expert's weight updates as follows:

$$\mathbf{w}_{i,t}^m = \frac{\mathbf{w}_{i,t-1}^m \sum_{k=1}^{M} e^{-\eta \mathbf{l}_k(\mathbf{f}_{i,t},\mathbf{y}_t)}}{\sum_{j=1}^{L} \mathbf{w}_{j,t-1}^m e^{-\eta \mathbf{l}^m(\mathbf{f}_{j,t-1},\mathbf{y}_t)}} \tag{6}$$

The update step as in (6), relates performance of the expert i in terms of each of the m objectives at every step. This modification serves the basic intuition of relating the performance of the experts and hence the learner with respect to both objectives together. However, the limitation of this step is in not being able to relate the overall performance of an experts in both objectives with the desired overall performance in both objectives. The motivation being, that at the Pareto front, both objectives cannot be simultaneously improved - improving one inevitably degrades the other. An additional desirability or importance factor if associated with the process, when it converges towards the Pareto front, would enable the algorithm to meet the desired performance of minimizing regret with time while predicting Pareto optimal solutions.

Pareto Descent Weight update (PDWU)

In this method, we show how to ensure that at every round t, the instantaneous vector is chosen such that, the direction of drift does not increase the regret much on individual objective and also the drift is in the direction where both the objectives are improved optimally. From our knowledge on descent directions in multi criteria optimization, we know that if a descent direction is sought that produces Pareto optimal solutions, the best direction to choose is a Pareto descent direction (Harada and Kobayashi 2006) which gives non-dominated solutions (optimal solutions) in the best case, and in the worst case the descent directions are automatically obtained which gives dominated (sub-optimal)solutions. The way to ensure this is to have an importance vector $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_m) \in \mathbb{R}^M$ with constraints for convexity $\alpha_k \geq 0$ and $\sum_{k=1}^{M} \alpha_k = 1$ that places relative importance on the objectives. We use this notion of choosing Pareto descent directions whilst choosing the instantaneous regret vector $\mathbf{r}_t^m$. The instantaneous regret vector and hence the weight of the experts at every round, control the learner's regret by using the gradient of the potential information in keeping the regret close to minimum and the convex combination of the objective vectors control the direction of movement of the learning algorithm towards the optimal Pareto front. The convex combination is given by:

$$\mathbf{r}_t^m = \sum_{k=1}^{M} \alpha_k \bigtriangledown \Phi_k(\mathbf{R}_{t-1}) \tag{7}$$

The weight $\mathbf{w}_{i,t}^m$ of expert i at round t for objective m is then a convex combination of the importance vector and the potential of the expert's regret in the direction of individual objectives, and is given by: $\mathbf{w}_{i,t-1}^m = \sum_{k=1}^{M} \alpha_k \phi'(\mathbf{R}_{i,t-1}^m)$ which simplifies to:

$$\mathbf{w}_{i,t}^m = \frac{\mathbf{w}_{i,t-1}^m \sum_{k=1}^{M} \alpha_k e^{-\eta \mathbf{l}_k(\mathbf{f}_{i,t},\mathbf{y}_t)}}{\sum_{j=1}^{L} \mathbf{w}_{j,t-1}^m e^{-\eta \mathbf{l}^m(\mathbf{f}_{j,t-1},\mathbf{y}_t)}} \tag{8}$$

Sampling Importance Vector $\boldsymbol{\alpha}$:

Another crucial part of our algorithm is the selection of the importance vector $\boldsymbol{\alpha_t}$ at every round t. Typically, the complete set of Pareto descent directions (which includes the descent directions) give a convex combination of objective vectors; that forms a convex cone pointed at the origin (Harada and Kobayashi 2006), exploiting the simultaneous linear inequality as in (3) (Harada and Kobayashi 2006). For simplicity, we can consider the convex cone to be a simplex as well, that is a triangle in this case that can be generalized to tetrahedron or higher dimensions. We enforce the condition that this simplex or the convex cone is pointed at the origin by adding the following constraint to (7): $-1 \geq \mathbf{r}_{i,t}^m \leq 1$ for expert i. Any point sampled from the convex cone then should give an importance vector $\boldsymbol{\alpha_t}$ and the convex combination of $\boldsymbol{\alpha_t}$ and the objective vectors in terms of gradient of the potential functions with respect to to each objective $\phi(\mathbf{R}_t^m)$ gives the descent (drift) direction or instantaneous regret vector $\mathbf{r}_t^m$ in (7) for the learning algorithm to proceed. We sample $\boldsymbol{\alpha_t}$ from the convex hull at every round t of the learning algorithm.

Simulation Results

We perform preliminary simulation experiments, to model the example power performance problem we explained in previous sections. These experiments on synthetic data illustrates our results. Typically on a multi-core based embedded device (4 cores or more), the different power-performance settings can be as many as 40. The number of experts in our experiments is configured to 50. In reality, the power-performance policy experts are functions of the workloads currently executing on the device that predicts the objective vectors of trade-off values for each objective. Here, we use the objective vectors of power-performance trade-off predictions as our experts, rather than the actual functions themselves as in the context of on-line learning with experts, what matters are the predictions of the experts which can be imagined to be functions of the workloads. As the power performance trade-off values for any workload lie on a Pareto

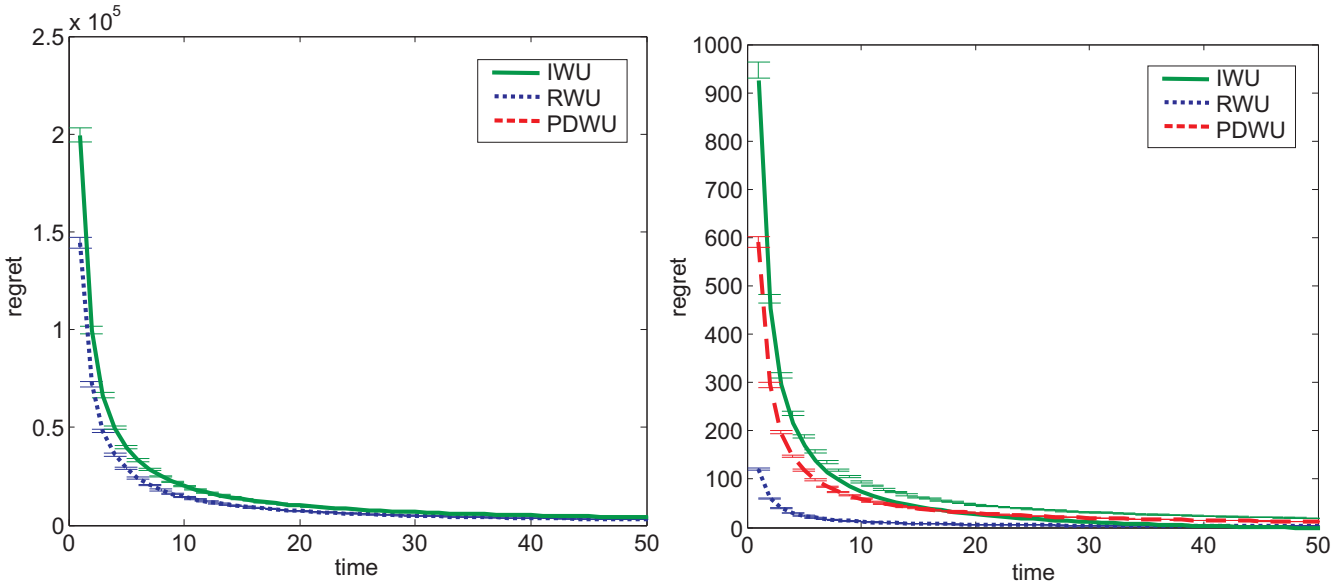

Figure 1: Instantaneous regret of learner for, (a) objective $performance$, (b) objective $power$. IWU is shown in solid GREEN, RWU in BLUE dots and PDWU in dashed RED. Each line in the graph corresponds to an average over 1000 runs of the experiment.

curve (Cochran et al. 2011), our experts are randomly chosen from a huge set of convex Pareto curves; we assume that this space constitute the feasible space of objective solutions. We designate the optimal Pareto front as the true sequence of experts in hindsight (not available to the learner), that the algorithm should converge to. The number of inputs (workloads) seen by our algorithm is preset to 20 but this is not a restriction. As mentioned before, we have two objectives, that of power savings and performance delay represented as $m = 2$ in our experiments. For most of our experiments, we vary the learning rate between 0.3 to 0.5. We also run experiments on learning rate of 0.01 and 0.8 for verifying bounds. All the on-line learning games comprise 100 rounds and the results are averaged over 1000 runs. We measure the normalized regret of the learning algorithm using (2) as the main performance evaluation metric in these experiments. Our importance vector α_t is sampled uniformly from the convex cone pointed at the origin. We perform experiments using both exponentially weighted average forecaster and weighted average forecasters. Our convex loss function is an absolute loss function.

Figure 1 shows the results of the normalized regret as explained in (2), as a performance measure for all the three methods used in our learning algorithm. This performance measure implies that the learner should have a vanishing per-round regret or the instantaneous regret. The difference in cumulative loss of the learner and the cumulative loss of the best expert in hindsight should grow sub linearly(almost surely) irrespective of the outcome the learner sees. We use the independent weight update method as discussed in (5) as our baseline method that has the on-line learning with experts framework without modification to the weight update step for the experts. The methods of relative weight update and Pareto descent weight update that use the weight updates in (6) and (8), are compared against the baseline method. Figure 1(a) shows results for objective $performance$ while figure 1(b) shows results for objective $power$. As seen from the results, RWU in Figure 1(a) performs better than the IWU. In this example, PDWU performs almost as good as the baseline method. Figure 1(b) shows that in the case of objective $power$, both methods RWU and PDWU perform better than baseline method. We see sub linear growth in the difference of cumulative loss of the learner and that of the true expert for all our methods. The graphs in Figure 1 show the instantaneous regret of the learner as it converges to zero.

Discussion and Extensions

In this paper we have seen how an on-line learning framework can be adapted to the continuous learning of multiple tasks problem. The independent weight update approach evaluates the learner in how well it addresses each of the objectives separately. The relative weight update and Pareto descent weight update methods perform a convex combination of the performance of the algorithm in addressing multiple objectives simultaneously, while making predictions from the optimal Pareto front. An importance vector is used to relatively weigh the objectives in each round. The convex combination of the importance vector and the function of learner's regret provides a Pareto descent direction for faster convergence on the Pareto optimal front.

A future direction of this work is to provide a theoretical analysis of the learning approaches proposed and to achieve

generalization bounds for the same. We would also like to perform extensive experiments on empirical data obtained from the real embedded device and the data servers to determine how the lifetime of the battery or power source is maintained while balancing the power-performance tradeoffs adequately.

References

Blackwell, D. 1954. Controlled random walks. In *Proceedings of the International Congress of Mathematicians*, volume 3, 336–338.

Blackwell, D. 1956. An analog of the minimax theorem for vector payoffs. *Pacific Journal of Mathematics* 6(1):1–8.

Bokrantz, R., and Forsgren, A. 2011. A dual algorithm for approximating pareto sets in convex multi-criteria optimization. Technical Report TRITA-MAT-2011-OS3, Department of Mathematics, Royal Institute of Technology.

Brown, M., and Smith, R. 2003. Effective use of directional information in multi-objective evolutionary computation. In *Genetic and Evolutionary Computation (GECCO-2003)*, 197–197. Springer.

Cesa-Bianchi, N., and Lugosi, G. 2003. Potential-based algorithms in on-line prediction and game theory. *Machine Learning* 51(3):239–261.

Cesa-Bianchi, N., and Lugosi, G. 2006. *Prediction, learning, and games*. Cambridge Univ Pr.

Cochran, R.; Hankendi, C.; Coskun, A.; and Reda, S. 2011. Pack & cap: adaptive dvfs and thread packing under power caps. In *Proceedings of the 44th annual IEEE/ACM International Symposium on Microarchitecture*, 175–185. ACM.

Dhiman, G., and Rosing, T. 2009. System-level power management using online learning. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on* 28(5):676–689.

Eban, E.; Birnbaum, A.; Shalev-Shwartz, S.; and Globerson, A. 2012. Learning the experts for online sequence prediction. *Arxiv preprint arXiv:1206.4604*.

Esmaeilzadeh, H.; Blem, E.; St Amant, R.; Sankaralingam, K.; and Burger, D. 2011. Dark silicon and the end of multicore scaling. In *Proceeding of the 38th annual international symposium on Computer architecture*, 365–376. ACM.

Esmaeilzadeh, H.; Cao, T.; Yang, X.; Blackburn, S.; and McKinley, K. 2012. What is happening to power, performance, and software? *Micro, IEEE* 32(3):110–121.

Fliege, J., and Svaiter, B. 2000. Steepest descent methods for multicriteria optimization. *Mathematical Methods of Operations Research* 51(3):479–494.

Harada, K., and Kobayashi, S. 2006. Local search for multiobjective function optimization: Pareto descent method. In *8th Annual Conference on Genetic and Evolutionary Computation (GECCO-2006)*, 659–666. ACM Press.

Hazan, E., and Seshadhri, C. 2009. Efficient learning algorithms for changing environments. In *Proceedings of the 26th Annual International Conference on Machine Learning*, 393–400. ACM.

Herbster, M., and Warmuth, M. 1998. Tracking the best expert. *Machine Learning* 32(2):151–178.

Hu, X.; Huang, Z.; and Wang, Z. 2003. Hybridization of the multi-objective evolutionary algorithms and the gradient-based algorithms. In *Evolutionary Computation, 2003. CEC'03. The 2003 Congress on*, volume 2, 870–877. IEEE.

Kalyanakrishnan, S.; Tewari, A.; Auer, P.; and Stone, P. 2012. Pac subset selection in stochastic multi-armed bandits. In *Proceedings of the 29th International Conference on Machine Learning*, 655–662.

Lugosi, G.; Papaspiliopoulos, O.; and Stoltz, G. 2009. Online multi-task learning with hard constraints. *Arxiv preprint arXiv:0902.3526*.

Meisner, D.; Sadler, C.; Barroso, L.; Weber, W.; and Wenisch, T. 2011. Power management of online data-intensive services. In *Proceedings of the 38th ACM International Symposium on Computer Architecture*, 319–330. IEEE.

Singhee, A. 2011. Pareto sampling using simplicial refinement by derivative pursuit. US Patent 20,110,307,430.

Zitzler, E.; Laumanns, M.; and Bleuler, S. 2004. A tutorial on evolutionary multiobjective optimization. *Metaheuristics for Multiobjective Optimisation* 3–37.