Task Allocation in Volunteer Computing Networks under Monetary Budget Constraints

Huseyin Guler
Koc University
Istanbul, Turkey

B. Barla Cambazoglu
Yahoo! Research
Barcelona, Spain

Oznur Ozkasap
Koc University
Istanbul, Turkey

ABSTRACT

We propose using monetary budget constraints in volunteer computing networks so that the peers can limit the financial burden incurred on them due the usage of their computational resources by the network. Under the assumption that the price of the electricity consumed by the peers has temporal variation, we show that our approach leads to an interesting task allocation problem, where the goal is to maximize the amount of work done by the peers without violating the monetary budget constraints set by the peers. We propose various heuristics as solution to the problem, which is NP-hard. Our extensive simulations using realistic data traces and real-life electricity prices demonstrate that the proposed techniques considerably increase the amount of useful work done by the peers, compared to a baseline technique.

1. INTRODUCTION

The focus of this work is on volunteer computing networks, where the goal is to solve a computationally expensive problem by using the resources provided by a large number of geographically distributed peers. In such networks, a central authority (dispatcher) is responsible for the management of the network. Typically, the dispatcher divides a large problem instance into smaller tasks and distributes them among the peers for processing. The peers join the network on a volunteer basis and provide their computational resources to help the processing without receiving any immediate financial benefit. Since the provided resources consume energy, the peers even end up with increased electricity bills. This may create an obstacle for growing the volunteer computing network as peers will be less motivated to join the network.

In some volunteer computing networks, peers are allowed to specify an upper bound on the number of tasks they are willing to process in a given period of time or they may specify the fraction of time their resources can be used [1, 5]. In this work, we go one step beyond and propose an alternative where peers can explicitly specify the maximum amount of money they can afford to spend in a time period while their resources are used. The incentive behind this approach is that allowing such monetary constraints may motivate more peers to contribute to the network since the peers will have a guarantee that the financial overhead incurred by the network will be limited.

From the perspective of the volunteer computing network, having such monetary constraints leads to an interesting task allocation problem. Obviously, the goal of the network is still to maximize the amount of work done in a given time period. However, under the assumption that the electricity prices show temporal variation [4, 7, 8], the dispatcher now needs to decide when to assign a task to a peer for processing. Assigning tasks to a peer when the peer is consuming electricity at high prices will lead to quick exhaustion of the peer's monetary budget. Hence, it is important for the dispatcher to estimate the time periods where the peers are consuming cheap electricity and assign the tasks to them accordingly, trying to exhaust peers' monetary budgets as much as possible but without exceeding them.

Our contributions are the following. We investigate the problem of allocating tasks in a volunteer computing network under monetary budget constraints that are set by the peers. We formally state the problem and propose various heuristic solutions. Through simulations based on realistic data traces and real-life electricity prices, we investigate the performance of the proposed heuristics. The empirical results indicate significant improvement in the task processing capacity of the network relative to a realistic baseline.

2. PROBLEM SPECIFICATION

We consider a volunteer computing network consisting of N peers and a central dispatcher. The dispatcher can exploit the computational resources of the peers to perform certain tasks. We assume that the main computational overhead is in the processing of the tasks by the peers and the communication overheads are negligible (this is typically the case in practice). We also do not take into account the financial costs potentially incurred by the Internet service provider of the peer due to the allocated bandwidth.

Each peer i, when joining the network for the first time, specifies a personal monetary budget B_i, which indicates the maximum electricity bill increase that the peer can afford while it contributes to the processing of the tasks. The budgets are defined over time periods, each with a fixed length of T units of time.[1] The dispatcher keeps track of an estimate of the monetary cost incurred on each peer during the

[1]We assume that the budgets are set on a weekly basis.

current time period. The cost estimates are reset to zero at the end of every time period. The resources of a peer can be used only if its monetary budget is not yet fully consumed within the current time period.

To simplify the problem definition, we assume that the time is partitioned into unit time intervals, i.e., the time is not continuous but discrete. More specifically, the time period T is divided into T/u time slots, each of length u time unit. The dispatcher can allocate a time slot $[t, t+u)$ either entirely or may decide not to allocate it at all. A peer can process the tasks assigned to itself independent of the other peers and tasks. Therefore, the allocation problem we will describe can be optimized separately for each peer.

When subscribing to the network, the peers let the dispatcher know about the properties of their computational resources (e.g., the number of cores and the CPU clock frequency).[2] Also, the peers independently set a minimum and a maximum CPU utilization threshold, which are denoted by m_i and M_i, respectively. Moreover, at the beginning of every time slot $[t, t+u)$, through a locally running daemon, each peer i lets the dispatcher know about its expected CPU utilization $U_i(t)$ in that time slot. The CPU utilization $\widehat{U}_i(t)$ due to the tasks assigned by the dispatcher is bounded by $\max(M_i - U_i(t), 0)$, i.e., the dispatcher fully utilizes the CPU of the peer without exceeding $M_i - U_i(t)$. The tasks can be allocated on the computational resources of peer i only if $U_i(t) < m_i$. Under these constraints, we have

$$\widehat{U}_i(t) = \begin{cases} M_i - U_i(t), & U_i(t) < m_i \leq M_i, \\ 0, & \text{otherwise.} \end{cases} \quad (1)$$

We assume that the dispatcher has access to the information about the current and historical unit electricity prices of each peer and assume that the electricity prices show temporal variation [4, 7]. The dispatcher, however, has no knowledge of the future electricity prices. We denote by $E_i(t)$ the price of the electricity for peer i in time slot $[t, t+u)$. If the computational resources of peer i are used by the network during the time slot $[t, t+u)$, the increase $I_i(t)$ in the electricity bill of peer i is estimated by

$$I_i(t) = \widehat{U}_i(t) \times W_i \times E_i(t) \times u, \quad (2)$$

For a given peer and a time slot, the benefit is measured by the computational work done by the peer in the specified time slot. The computational work is defined in terms of the clock frequency of the peer and the expected CPU utilization allocated by the network in the given time slot. More formally, the computational work done at peer i in time slot $[t, t+u)$ is denoted by $J_i(t)$ and defined as

$$J_i(t) = \widehat{U}_i(t) \times F_i, \quad (3)$$

Given the above-mentioned definitions and notation, which is summarized in Table 1, our objective is to maximize

$$\sum_{i=1}^{N} \sum_{t=0}^{\lfloor T/u \rfloor} J_i(t), \quad (4)$$

subject to

$$\sum_{t=0}^{\lfloor T/u \rfloor} I_i(t) \leq B_i, \quad \text{for } 1 \leq i \leq N. \quad (5)$$

[2]We assume that each peer has a single CPU with varying clock frequencies.

Table 1: System Parameters

Parameters	Symbol
Number of peers in the network	N
Monetary budget of peer i	B_i
Time period for resetting peer budgets	T
Length of a unit time slot	u
Electricity cost of peer i in time slot $[t, t+u)$	$I_i(t)$
Electricity price for peer i in time slot $[t, t+u)$	$E_i(t)$
CPU utilization of peer i in time slot $[t, t+u)$	$U_i(t)$
Maximum CPU utilization threshold for peer i	M_i
Minimum CPU utilization threshold for peer i	m_i
CPU frequency of peer i	F_i
Power consumption of peer i's CPU in watts	W_i

3. SOLUTIONS

In order to simplify the presentation, we refer to the time slots as tasks. This is because, in our problem definition, the time is discretized into time slots and each time slot is either fully allocated for processing tasks or not allocated by the dispatcher. In particular, we assume that the dispatcher has infinitely many tasks and the goal is to determine in which time slots to allocate the computational resources of the peers, rather than how to schedule individual computational tasks for execution.

If there is a perfect knowledge of the future electricity prices, our task allocation problem can be formulated as the 0-1 knapsack problem. In this formulation, the knapsack value corresponds to the monetary budget of the peer and the items correspond to the tasks (time slots). An item's weight corresponds to the cost of allocating the respective task (see Eq. 2). The value of an item is determined according to the work done in the respective time slot (see Eq. 3). Naturally, in our scenario, the dispatcher has no information about the future electricity prices. The time slots can be allocated on-the-fly using only the past electricity price information. Consequently, our problem reduces to the online knapsack problem, where the item weights and values are not known beforehand. We also have the additional peer availability constraint in our case.

The online knapsack problem is first studied in [6] and its solution is applied to different problems, including the auction bidding problem [9, 10] and the knapsack secretary problem [2]. The problem is known to be NP-hard [2] and hence there is no polynomial-time algorithm for an optimal solution. Since the online knapsack problem is NP-hard, our task allocation problem is also NP-hard. Hence, heuristics play an important role in finding solutions.

3.1 Algorithms

Naive baseline (`Baseline`). This is a very naive approach, which does not take into account the temporal variation in the electricity prices. A time slot is allocated by the dispatcher if the peer's computational resources are available in that time slot, i.e., whenever the peer is online and the minimum CPU utilization constraint set by the peer is satisfied ($U_i(t) < m_i$ must hold).

Solutions based on expected electricity price. This class of heuristics exploit the past electricity prices to decide whether the current price of the electricity consumed by the peer is relatively high or not. The techniques compute an expected electricity price value based on the electricity price data observed in the past. If the current electricity price is lower than a certain fraction of the expected price,[3] the time slot is allocated; otherwise, it is not allocated. We evaluate three simple approaches for computing the price.

Average of yesterday (`Yesterday`). The expected electricity price is computed by averaging the sample price values observed on the previous day. This technique is based on the expectation that the prices will not differ much between two consecutive days, i.e., it exploits the recency.

Past average of today (`SameDayHistory`). The price is assumed to be the average of the prices observed on the previous occurrences of the current day of the week. The expectation here is that the electricity prices tend to be similar on the same days of the week, i.e., the periodicity is exploited.

Average of entire history (`EntireHistory`). The average price value over the entire price history is used.

History repeats (`HistoryRepeats`). This heuristic tries to exploit the weekly repetition in the electricity prices. We compress the entire price data into a single week of data by computing an average value for each hour in a week. Then time slots of the current week are sorted in increasing order and stored in a candidate list. Then, the candidate list is traversed starting from the time slot with the lowest price towards the time slot with the highest price and the time slots whose cost sum do not exceed the remaining monetary budget of the peer are marked, indicating their suitability in terms of the electricity price. The traversal stops when the sum exceeds the remaining monetary budget. Then, the current time slot is allocated for running tasks only if it is among the marked slots in the candidate list. The candidate list is periodically updated since it becomes outdated in time and needs to be refreshed.

Online knapsack (`OnlineKnapsack`). As another solution, we adapt the algorithm proposed in [9], which has two assumptions about the input data. First, the weight of each item is much smaller than the capacity of the knapsack, i.e., $I_i(t) \ll B_i$, and the second the *value/weight* ratio of each item is both upper- and lower-bounded, i.e.„ $L \leq J_i(t)/I_i(t) \leq U, \forall t$. These assumptions enable the proposed algorithm to have a constant competitive ratio of $\ln(U/L)+1$. We also have the peer availability constraint in Eq. 5. To capture this constraint, we modify the original algorithm such that the U and L values are used along with the remaining budget in order to determine a threshold value. The decision of allocating the current time slot is made based on a comparison between the gain ratio in the current time slot and the threshold value calculated by the function given in [9]. The algorithm allows the system to aggressively allocate time slots at the beginning while most of the budget is available. The system becomes more selective in time as the remaining budget gets smaller.

Oracle (`Oracle`). In order to set an upper bound on the performance of the proposed heuristics, we design an `Oracle` algorithm that has access to the future electricity prices and the future CPU usage patterns of the peers. As explained before, the optimum solution of our problem cannot be found in polynomial time. However, since `Oracle` is assumed to have access to the future electricity prices, the problem is transformed into the traditional 0-1 knapsack problem, for which there is a pseudo-polynomial algorithm that finds an optimum solution.

[3]This fraction is a parameter that needs to be tuned.

4. SIMULATION SETUP

To evaluate the performance of the heuristics, we simulate a volunteer computing network. In our simulations, we use real-time electricity prices obtained from ComEd, an electricity provider located in the USA. The historical electricity prices are sampled on an hourly basis over a period from 3 September 2012 to 2 December 2012. The simulation is run on the last week while earlier weeks provide the historical price data for some of the heuristics. We assume that the peers are located in six different countries: USA, Germany, Russia, Turkey, China, and UK. Since the obtained electricity price distribution is representative only for the USA, for each of the remaining countries, we linearly scaled this distribution by considering their average electricity prices (obtained from the U.S. Energy Information Administration).

We simulated a volunteer computing network consisting of 10,000 peers. The peers are assumed to be distributed in the previously mentioned countries such that each country receives peers proportional to its Internet user population. The minimum and maximum CPU usage constraints (i.e., m_i and M_i values) of the peers are selected from a uniform distribution within a range of 15%–20% and 60%–75%, respectively. The CPU clock frequency values (F_i) are sampled from a uniform distribution in the range of 1.7GHz and 3.2GHz. The power consumption values (W_i) are taken from Intel's website, taking into account the clock frequencies.

We also simulate the peers' availability in the network. We assume that this mainly depends on the time of the day and rely on the measurements provided in [3], where the peer availability is shown to have a diurnal pattern. In Fig. 1, we display the hourly change in the electricity prices together with the fraction of peers that are online. According to the figure, during day times, there is some positive correlation between the availability of the peers and the electricity prices. However, we observe a negative correlation in night hours, where the number of online peers increases while the electricity price starts to decrease until the midnight.

An important parameter in our problem is the monetary budgets set by the peers. In practice, the budgets may be affected by many factors, the socio-economic and cultural factors being the most prominent. In this work, since there are many factors independently affecting the peers' budgets, we set the budgets according to a normal distribution with varying mean values. In particular, we use budget values between three to ten cents. This range is obtained by a user study performed among the students of Koc University.

5. EXPERIMENTAL RESULTS

In Fig. 2, we compare the performance of our heuristics in terms of the amount of work done by the peers for varying monetary budget values (the reported results are averages of five runs). As expected, all of the proposed heuristics perform better than `Baseline` and worse than `Oracle`. All three threshold-based heuristics (`Yesterday`, `SameDayHistory`, and `EntireHistory`) achieve better performance compared to the `OnlineKnapsack` and `HistoryRepeats`. In particular, the amount of work done by the `Yesterday` heuristic is 30% higher than the work done by `Baseline` and is 7% less than the work done by `Oracle`, on average.

As the budget values increase, the performance gap between the proposed heuristics and the baseline starts to diminish. This is because higher budget values imply more

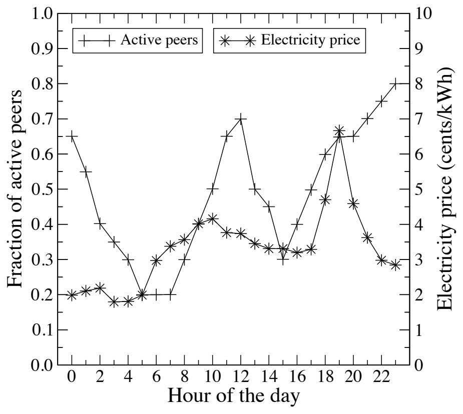

Figure 1: The electricity prices and the fraction of peers that are active at a certain hour of the day.

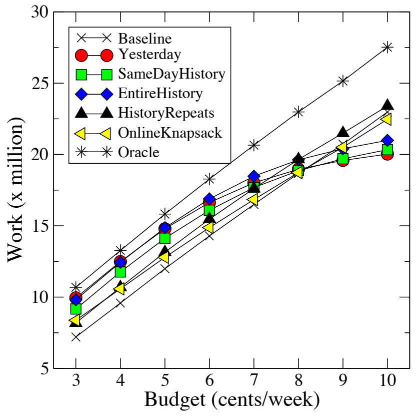

Figure 2: The computational work distribution with respect to the weekly budgets.

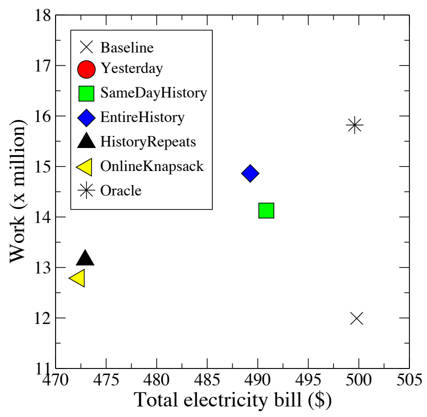

Figure 3: The total electricity bill of the peers versus the total amount of work done.

flexibility in allocating the time slots and it becomes less important to allocate time slots in which the electricity prices are low. In an extreme case, if we set the budget value to infinite, `Baseline` will attain the optimum result since it will allocate all available slots without any budget constraint.

In Fig. 3, we show the trade-off between the total amount of work done and the total electricity bill of the peers (the budget is set to three cents). As expected, `Baseline` leads to the lowest amount of work and the highest electricity bill since it is completely blind to the variation in electricity prices. As observed in Fig. 2, the `Yesterday`, `SameDayHistory`, and `EntireHistory` heuristics yield the largest amount of work. The `HistoryRepeats` heuristic, however, achieves closer performance to the heuristics while leading to a much lower electricity bill. It is interesting to note that, although `Oracle` attains good performance in terms of the amount of work done, it has a rather poor performance in reducing the total electricity bill, which is not considered as an optimization objective in our problem definition.

Although our main objective is to maximize the total computational work done by the peers, we observed that, as a by-product, some heuristics also decreased the electricity bill of the peers relative to the baseline heuristic. For example, in our simulations involving 10,000 peers, the `HistoryRepeats` heuristic led to a \$27 decrease in the weekly electricity bill of the peers when compared to `Baseline`. Assuming a network involving one million participants (e.g., SETI@Home), this implies $\$27{\times}100=\$2,700$ saving per week. Projecting this to a whole year, the saving becomes $\$2,700{\times}52=\$140,400$ per year, i.e., the total electricity bill of the peers in the network can be significantly reduced by the proposed heuristics.

6. CONCLUSION

We proposed setting monetary budgets in volunteer computing networks, limiting the financial burden incurred on the peers due to the usage of their computational resources by the network. We showed that, under the assumption of temporal volatility in electricity prices, this idea leads to a particular task allocation problem. We formally specified this task allocation problem and proposed various heuristics as potential solutions. Simulations using real-life electricity price data demonstrated that the proposed heuristics can increase the amount of useful work done in the network while respecting the peers' budget, compared to a naive baseline and a competitive online algorithm from literature. In particular, the amount of work done by our best performing heuristic is 30% higher than the work done by `Baseline` and 7% less than the work done by `Oracle`, on average.

7. REFERENCES

[1] D. P. Anderson. Emulating volunteer vomputing scheduling policies. In *Proc. 2011 IEEE Int'l Symp. Parallel and Distributed Processing Workshops and PhD Forum*, pages 1839–1846, 2011.

[2] M. Babaioff, N. Immorlica, D. Kempe, and R. Kleinberg. A knapsack secretary problem with applications. In *Proc. 10th Int'l Workshop on Approximation and 11th Int'l Workshop on Randomization, and Combinatorial Optimization*, pages 16–28, 2007.

[3] R. Bhagwan, S. Savage, and G. M. Voelker. Understanding availability. In *Proc. 2nd Int'l Workshop on Peer-to-Peer Systems*, pages 256–267, 2003.

[4] E. Kayaaslan, B. B. Cambazoglu, R. Blanco, F. P. Junqueira, and C. Aykanat. Energy-price-driven query processing in multi-center web search engines. In *Proc. 34th Int'l ACM SIGIR Conf. Research and Development in Information Retrieval*, pages 983–992, 2011.

[5] D. Kondo, D. P. Anderson, and J. M. Vii. Performance evaluation of scheduling policies for volunteer computing. In *Proc. 3rd IEEE Int'l Conf. e-Science and Grid Computing*, pages 415–422, 2007.

[6] A. Marchetti-Spaccamela and C. Vercellis. Stochastic on-line knapsack problems. *Mathematical Programming*, 68:73–104, 1995.

[7] A. Qureshi, R. Weber, H. Balakrishnan, J. Guttag, and B. Maggs. Cutting the electric bill for Internet-scale systems. In *Proc. ACM SIGCOMM 2009 Conf. Data Communication*, pages 123–134, 2009.

[8] L. Rao, X. Liu, L. Xie, and W. Liu. Minimizing electricity cost: optimization of distributed Internet data centers in a multi-electricity-market environment. In *Proc. 29th IEEE Int'l Conf. Computer Communications*, pages 1145–1153, 2010.

[9] Y. Zhou, D. Chakrabarty, and R. Lukose. Budget constrained bidding in keyword auctions and online knapsack problems. In *Proc. 17th Int'l Conf. World Wide Web*, pages 1243–1244, 2008.

[10] Y. Zhou and V. Naroditskiy. Algorithm for stochastic multiple-choice knapsack problem and application to keywords bidding. In *Proc. 17th Int'l Conf. World Wide Web*, pages 1175–1176, 2008.