Information-Theoretic Task Scheduling

Patrick Shields
Email: pshields@gmail.com

December 7, 2012

***Abstract*—I examine the applicability of information theory to task scheduling by constructing a utility-accrual stochastic scheduling problem in which a task's utility is a function of a task-specific random variable which is updated according to a known Markov process. Schedulers in this problem are given the option to use a time step to observe the value of a task's random variable, rather than schedule a task to run in that step. In this manner, a scheduler may trade schedulable time for Shannon information, which it may then use to make better scheduling decisions in the future. I analyze an information-theoretic approach to scheduling tasks in this problem, and ground the research in a real-world example. As far as I can tell, providing schedulers with the option to trade schedulable time for Shannon information is a novel addition to classical scheduling problems. The motivating application is software which suggests tasks to a human user and must balance the need for up-to-date information with the cost of asking users to update a task's parameters. I find that observing state variables (which decreases entropy) increases expected accrued utility on average, but that entropy is not as effective as value-of-information as a scheduling heuristic.**

I. Introduction

I have written a task manager for myself which suggests recurring tasks in descending order of utility. In its simplest form, my system assesses tasks' utility by applying the heuristic function

$$U = \alpha \log_2 (\beta \Delta t + 1),$$

where α and β are task-specific parameters and Δt is the amount of time since I last completed the task.

I do not have a systematic method of updating tasks' parameters. I update a task's parameters if I notice that its assessed utility seems too high or too low, or if it does not seem to be making it to the top of the list at an appropriate interval, relative to my other recurring tasks.

In pursuit of a more systematic and efficient way to update my task's parameters, I turn to information theory. I wish to examine if my task management system could use information-theoretic reasoning to assess the value of asking me to update a task's parameters, and to, accordingly, do so when it would exceed the value of suggesting a particular task. I would also like to relate my work to the fields of scheduling and information theory and see if information-theoretic concepts such as entropy can be used to improve the efficiency of scheduling algorithms.

I formulate a simple scheduling problem where schedulers may obtain Shannon information at the cost of some schedulable time, and implement various schedulers to examine their performance. I find entropy and expected accrued utility to be related, but entropy to be less effective of a heuristic than value-of-information. Augmenting VOI-based heuristics with other information-theoretic information in order to reduce computational requirements may still be useful. I leave that for future research.

II. Motivating Example

Consider the stochastic, sequential scheduling problem of scheduling recurring, non-preemptible tasks $\mathbb{T} = \{T_0, T_1, \ldots, T_{m-1}\}$ of single unit duration into time slots $t_0, t_1, \ldots, t_{n-1}$. A task executed at t_i achieves a reward defined by the utility function $U(i) = \alpha_i$, where α_i is a random variable associated with the task. α_i is initialized to some known value at time t_0 and then sampled independently at each subsequent time step according to

$$\alpha_i \sim \begin{cases} \alpha_{i-1} + 1, & \text{with probability } \frac{1}{2} \\ \alpha_{i-1} - 1, & \text{with probability } \frac{1}{2} \end{cases}$$

Each task's α_i is a Markov process. Conveniently, if the probability distributions after Δt time steps without observation are precomputed in order of increasing each $\Delta t \in \{1, 2, \ldots\}$, each distribution can be found in $O(\Delta t)$ time; the Δtth row of Pascal's triangle represents the numerators of the probabilities, and the denominators are all $2^{\Delta t}$.

A scheduler in this problem does not inherently know any task's α_i for $i > 0$ and must reason about them probabilistically. As time goes on, a scheduler's uncertainty (measured by entropy in bits) of each task's α_i increases.

Up until this point, I have formalized a fairly standard utility-accrual scheduling problem, if not simpler than the average problem. Now I break from that trend to introduce a novel variation: I provide a method for a scheduler to learn information about a task's random variable. At any time step t_i, rather than scheduling a task, the scheduler may instead observe the value of some task's α_i. That is, time step by time step, the scheduler either:

1) schedules a task to execute in the current time slot, or
2) observes a particular task's α_i.

In the second case, upon observing the task's α_i, the scheduler knows with certainty that task's α_i as of the current time step. Such an observation reduces the entropy of α_i to 0.

Per the standard in utility-accrual scheduling problems, an algorithm for this problem is optimal if it maximizes the expected value of the accrued utility, which is the sum of the utility accrued at each time step.

Each time slot, we must consider the trade-off between gaining information and accruing utility.

III. Algorithms and Problem Variations

In this section, I construct a few variants on the problem from the motivating example, as well as various algorithms for solving these problems, in the interest of providing a context in which to analyze the performance of a scheduler that uses entropy-minimization as a heuristic.

A. Unobserved variant

Consider a variant of the initial problem in which schedulers do not have the option to observe a task's α (call this the unobserved variant.) The scheduler now faces a fairly standard utility-accrual scheduling problem where tasks have stochastic utilities. Also consider a deterministic version of this variant, in which each α remains constant at each time step.

The expected accrued utility for any algorithm running on the (stochastic) unobserved variant is equal to the accrued utility under the deterministic version. I offer an informal proof. The expected utility of executing a task is linear with respect to the task's α at the time of task execution. For every instance that an α is decremented, another instance occurs with equal probability in which the α is incremented by the same amount. Thus the expected utility of executing the task associated with α equals the average utility under each of those cases, which averages to the utility if α were to remain the same. Expanding this to every α at every time step, we can see that there is no room for the expected utility of executing a task to change between the stochastic and determinstic versions of the unobserved variant. Accordingly, as the accrued utility is the sum of the utilities accrued each time step when executing a task, it follows that the expected value of the accrued utility in the stochastic case equals the accrued utility in the deterministic case.

All of this is to show that if we port to the original problem an algorithm suitable for the deterministic case of the unobserved variant, the expected accrued utility of the algorithm in the original problem equals its its actual accrued utility in the deterministic version of the unobserved variant. Such an algorithm simply doesn't exercise its option to observe a task's α.

An optimal algorithm for the deterministic case is one which can be shown to always achieve the maximum possible accrued utility. A simple optimal algorithm is to schedule the task with the highest α_0 (α_0 is the initial value of the task's α, which is known) at every time step. This algorithm can be implemented in $O(m+n)$, where again m is the number of tasks and n is the number of time slots.

An optimal algorithm in the stochastic case is one which achieves, for any assignment of α_0 values to tasks, the maximum possible expected accrued utility. Per the Markov process from which each task's α_i is sampled (for $i > 0$), as a scheduler loses information about a task's α over time, the expected value $E(\alpha_i)$ remains the same, since the probility mass function of α_i is symmetric about α's last known value. An optimal algorithm in the stochastic case will schedule whichever task has the highest a_0 at each remaining time step.

B. Clairvoyant variant

For purposes of comparison, I introduce a deterministic variant of the initial problem in which all α values are precomputed (through the Markov process) and made

known to the scheduler at t_0. As is common in scheduling theory, I use the term "clairvoyant" to refer to this variant since it provides the scheduler with all of the information it needs, even though such information may not be available in practice.

An algorithm for the clairvoyant variant is optimal if it maximizes accrued utility in each problem instance.

It is safe to assume that when comparing a problem instance from the initial problem and one from the clairvoyant variant in which each α parameter in one instance is equal to the corresponding α parameter in the other, the accrued utility by an optimal algorithm in the clairvoyant variant serves as an upper bound of the possible accrued utility in the problem instance from the initial problem.

A simple optimal algorithm for the clairvoyant variant is to, in each time slot t_i, schedule the task with the highest α_i at that time. This algorithm can be implemented in $O(mn)$.

C. Value of information

In the initial problem from the motivating example, we can quantify the value of an observation of α_i as the expected accrued utility for time steps $i+1, \ldots, n-1$ under the case that we observe α_i at time i, minus the expected accrued utility for those time steps if we do not observer α_i. This quantity is called the *value of information* [1]. Since we do not know what α_i would turn out to be, but we know its distribution, we calculate the expected accrued utility in time steps $i+1, \ldots, n-1$ as the expected value of the expected accrued utility over the probability distribution of α_i.

One algorithm for the initial problem is to, at each time step i, calculate the actual value of perfect information of each task's α_i. If observing the α_i which had the highest assessed value-of-information is worth more than $E(U(i))$, observe that α_i. Otherwise, schedule for execution the task that the optimal algorithm for the unobserved variant of the problem would schedule. I believe this algorithm would be optimal.

Calculating the expected accrued utility for time steps $i+1, \ldots, n-1$ is complicated by the option of the scheduler to, at later time steps, observe α values as well. I think calculating the actual value of information might be intractable. I avoid this complication by using a heuristic of the expected accrued utility. For the heuristic, I use the optimal algorithm described for the deterministic case of the unobserved variant of the problem. We can use it to schedule the remaining tasks and easily calculate the expected accrued utility for time steps $i+1, \ldots, n-1$ for the deterministic unobserved variant. That value is a lower bound on the expected accrued utility in time steps $i+1, \ldots, n-1$ for the initial problem. I include a variant of the previous potentially-optimal algorithm which uses this heuristic in the experiments under the label "VOIHeuristicScheduler". It is important to note that this algorithm is not actually calculating the value of information, but rather approximating it through a heuristic.

Calculating the true value of perfect information of a task's α_i might not be feasible. It is calculated by

$$V = \sum_x \left(p(\alpha_i{=}x) E\left[\sum_{i+1}^{n-1} U(i) | \alpha_i{=}x \right] \right) - E\left[\sum_{i+1}^{n-1} U(i) \right]$$

but we can approximate it using the lower-bound expected accrued utility in the deterministic unobserved case:

$$E\left[\sum_{i+1}^{n-1} U(i) | \alpha_i{=}x \right] \geq (n-i-2) \cdot \max_\alpha \alpha$$

$$E\left[\sum_{i+1}^{n-1} U(i) \right] \geq (n-i-2) \cdot \max_\alpha \alpha$$

A scheduler following this algorithm and the previous approximation should weakly dominate the deterministic unobserved algorithm. As a result of the approximation, the scheduler might observe a task's α before it would be optimal, since it does not account for the possibility of observing a task's α in future time slots.

D. Entropy-minimization heuristic

For the experiments, I implement an algorithm which observes the variable that would minimize entropy the most, whenever the total entropy is above a certain level. The level I selected in bits is equal to the number of tasks. This seemed to work well, as smaller or larger values seemed to achieve lower accrued utilites on average.

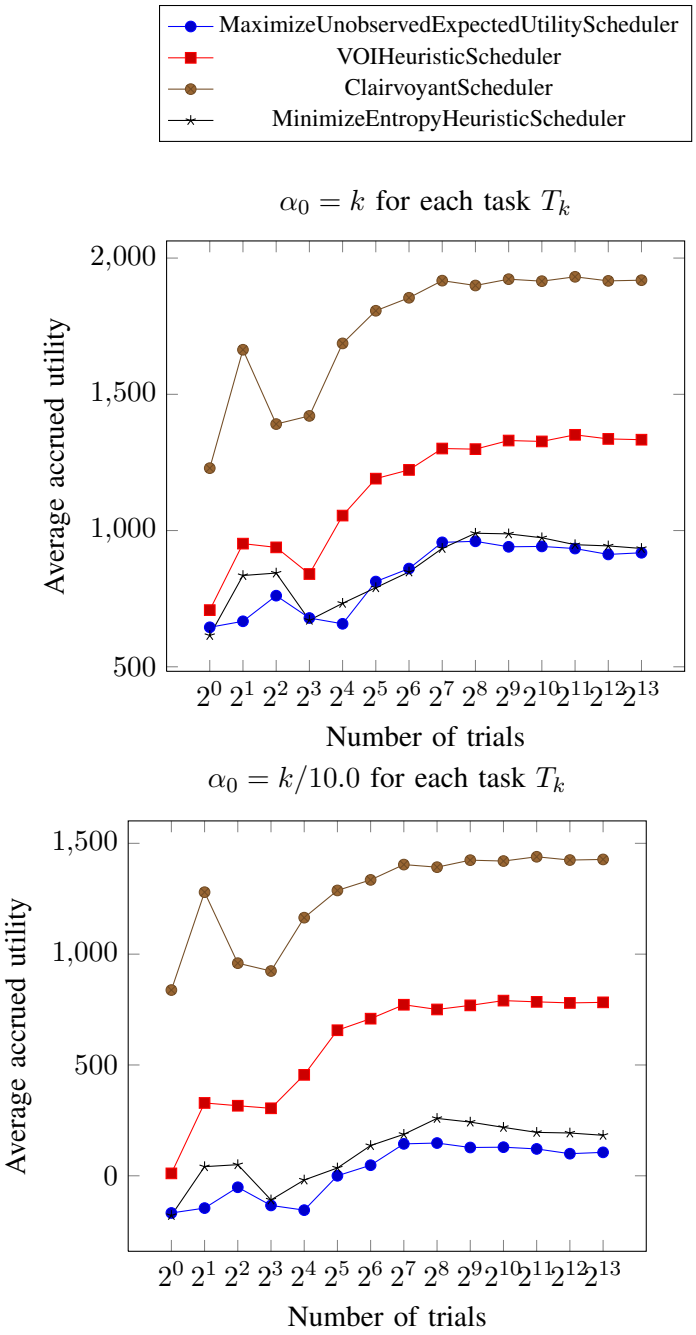
IV. Experiments

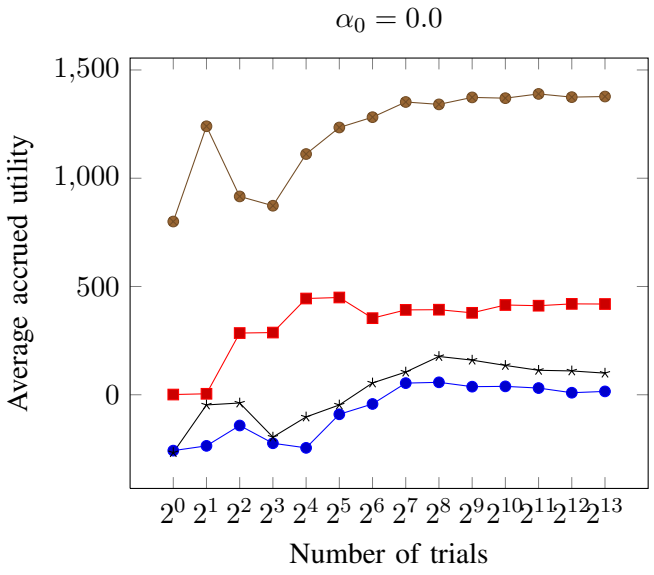
I implemented the initial problem in the Go programming language and ran four algorithms on several different initializations of α_0 values. I ran thousands of trials for each pairing of initializations and schedulers to confirm the results were converging towards an average expected utility.

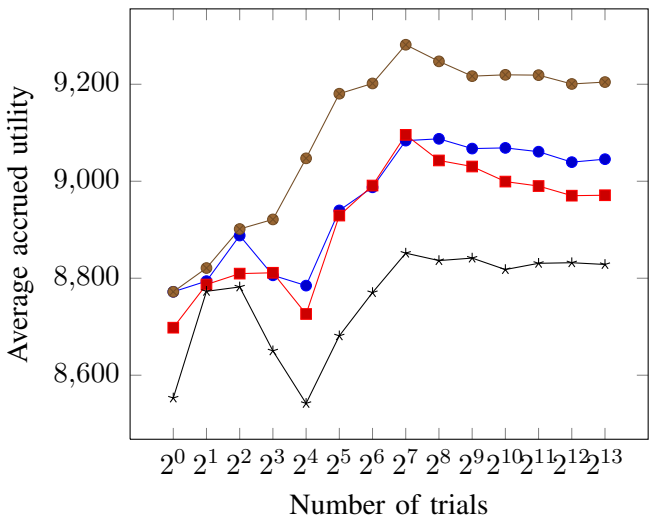
I test the following algorithms:

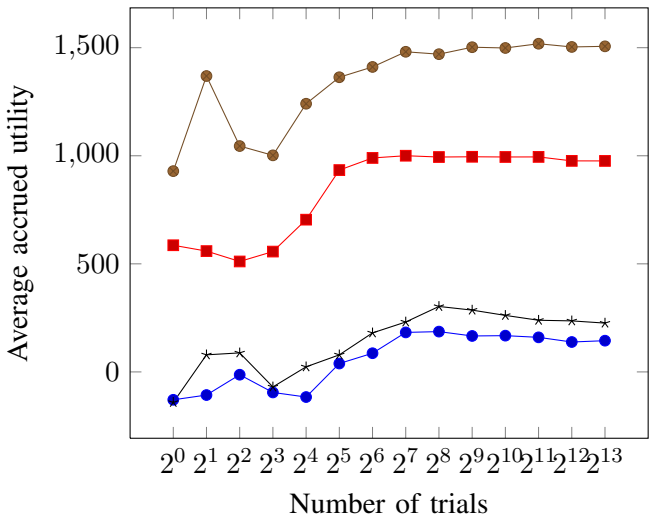
1) **MaximizeUnobservedExpectedUtilityScheduler** - Maximizes expected accrued utility assuming observation of a task's α is not allowed.
2) **VOIHeuristicScheduler** - At each step, assess the value of information for inquiring about each task's parameters. If the maximum VOI is greater than $E(U(i))$, inquire about that task's parameters. Otherwise, schedule the task with the highest expected utility.
3) **ClairvoyantScheduler** - Maximize accrued utility by scheduling the task with the highest α_i at each time step i.
4) **MinimizeEntropyHeuristicScheduler** - Observe the α that will result in the largest decrease in entropy, whenever the total entropy for the α_i values at time i in bits exceeds the number of tasks. Otherwise, schedule the task with the highest expected utility.

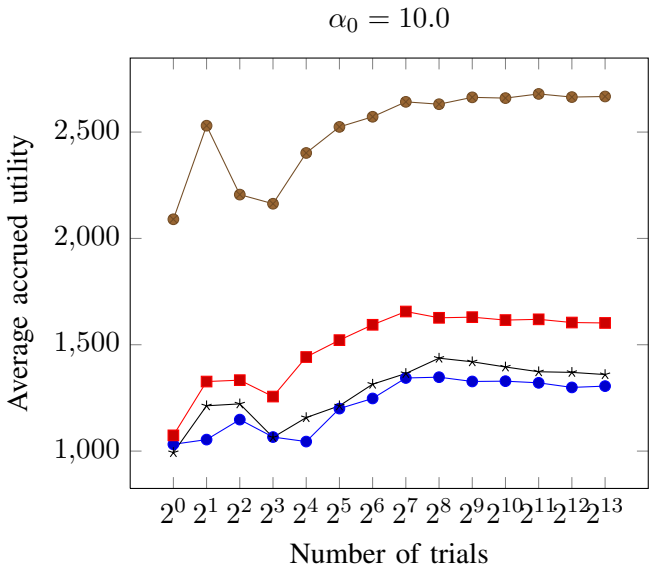
The experiments were conducted on 8 tasks over 128 time steps. Results were averaged at the end of trial 2^k for each $k \in \{0, 1, \ldots, 13\}$. The results are shown in the following graphs.











V. Discussion

A. Experiment results

The average accrued utilities seem to approach the expected values. My closed-form version of the expected accrued utility in the observed case was experimentally validated. Comparing $n \cdot \max_\alpha \alpha$ to the results for the *MaximizeUnobservedExpectedUtilityScheduler*, we see only a small difference:

Initial conditions	$n \cdot \max_\alpha \alpha$	Avg. U (2^{13} trials)
$\alpha_0 = k$ for each task T_k	896	918.499512
$\alpha_0 = k/10.0$ for each task T_k	89.6	105.799512
$\alpha_0 = 0.0$	0	15.499512
$\alpha_0 = 10.0 * k$ for each task T_k	8960	9045.499512
$\alpha_0 = 1.0$	128	144.499512
$\alpha_0 = 10.0$	1280	1305.499512

This is also evidence of correct implementation of experiment code.

The optimal clairvoyant algorithm outperforms the optimal deterministic variant algorithm and the value-of-information heuristic algorithm. It ran on the same α values as the other algorithms, but always knew the value of each α_i. Another way of putting it is that the algorithm benefitted from having complete certainty (0 entropy) about the α values. Clearly, decreasing entropy is worth something. However, bits of entropy are not all equal. Observing an α which was last observed to be low relative to other observed α values is worth less than observing an α of equal entropy which was last observed to have a high value, since the probability is lower that the α will become the highest-observed α in the former case. Entropy does not account for how valuable information is. In that sense, value-of-information seems

to be a better fit. Value-of-information is a statistical concept which is related to information theory.

B. Utility-accrual scheduling

Scheduling is an area of research with applications to many industries. Operating systems use scheduling to schedule software processes to processing units in an efficient manner while simultaneously giving the impression of responsiveness to users of the system. Manufacturing firms use scheduling to improve manufacturing processes. Telecommunications infrastructure uses scheduling to meet objectives (such as fairness) in communications.

Traditionally, tasks are modeled as having static weights and deadlines, but as described in [2], weights can fail to incorporate both importance and urgency, and deadlines can fail to represent soft time constraints. The introductory example problem is an instance of utility-accrual scheduling, where the objective is to maximize $\sum U(i)$, the sum of the accrued utilities. Utility-accrual scheduling traces back to 1985 [3]. Allowing arbitrary utility functions, utility-accrual scheduling is NP-hard with respect to the number of schedulable tasks.

Locke's best-effort scheduling algorithm [4] works on preemptive tasks and assumes utility function values reach some "critical point" (a discontinuity in the function or its first or second derivative, or arbitrarily selected if no such discontinuity exists) and decreases to zero (perhaps non-monotonically) afterward.

Many algorithms such as Li's General Utility Scheduling algorithm [5] implement polynomial-time heuristic algorithms. Heuristic algorithms for non-preemptive scheduling have been proposed [6]. Scheduling of non-preemptive scheduling with stochastic task durations has been examined in detail [7]. A polynomial-time heuristic-based scheduling algorithm for tasks with utility functions and variable cost functions (where task duration is a function of time) has been proposed [8]. Finding efficient, well-performing heuristic functions for utility-accrual schedulers appears to be an open problem and active research area.

The heuristic-based scheduling algorithm from [8] makes some different assumptions than the problem in my motivating example. It:

1) provides for a maximum time bound between multiple executions of a particular task,
2) provides that a task's execution time is the result of a variable cost function, and
3) operates only on tasks whose utilities are non-increasing.

Many of the utility-accrual schedulers in the literature focus on tasks with non-increasing utility functions. Originally, I found this counterintuitive, since my task management software represents each task's utility as a logarithmic function of time. Thinking about this further, I have realized that many of the tasks I store in my system could be viewed as higher-level tasks from which low-level task instances with non-increasing utility can be created. For example, a task statement such as "Go to the gym" might describe a high-level task which has logarithmically-increasing utility, but specific instances of the task such as "Go to the gym this morning" could have non-increasing utility once the morning has come. I am not sure how worthwhile such a system would be to implement. Non-increasing task utilities seem to facilitate the paradigm of schedulers as scheduling short-term, immediate tasks, rather than thinking about *what* to schedule in the first place.

C. Sensor management

Providing schedulers with the option to observe values of random variables introduces the field of sensor management into the problem. Sensor management has information-theoretic connections. Perhaps insights from sensor management can be applied to scheduling problems when schedulers face trade-offs between information-gathering and schedulable time.

In fact, scheduling and sensor management intersect in the subfield of sensor scheduling.

D. Implementation

I struggled to implement my initial formulation of task utilities.

Originally, I set task's utilities to be

$$U(i) = \begin{cases} 0, \text{ if } a_i \leq 0 \text{ or } b_i \leq 0 \\ \alpha_i \log_2 (\beta_i \Delta t + 1), \text{ otherwise} \end{cases}$$

where $\Delta t = i - j$ and j is the time step the task was last scheduled (or 0 if it has never been scheduled), and α_i and β_i are random variables associated with the task which

$$\delta_i \in \{a_i, b_i\} \sim \begin{cases} \delta_{i-1} + 1, & \text{with probability } \frac{1}{4} \\ \delta_{i-1} - 1, & \text{with probability } \frac{1}{4} \\ \delta_{i-1}, & \text{with probability } \frac{1}{2} \end{cases}$$

but I encountered several problems. Because the function is bounded below by 0, the expected value was influenced by how far away the last observed α was from 0. This increased the cost of computing $E(\alpha_i)$. Removing the ≥ 0 piece and the logarithmic component allowed me to assume that $E(\alpha_i)$ equals whatever α_i was last observed to be.

With the logarithmic component removed, I still had the option of making task's utilities a function of the time since they were last done (e.g. $U(i) = \alpha\Delta t$.) I chose not to set up task utilities this way for simplicity.

At one point, I attempted to give tasks non-quantum durations, but I did not think I would be able to comprehend the scheduling dynamics in that case well enough to complete the project in time.

I encountered rounding errors due to the use of 64-bit floating point numbers. The errors seemed to compound over time, since I was calculating the probability distributions using a dynamic programming table. Switching to Go's infinite-precision rational data type improved precision at a cost in performance.

I was not able to convert everything to the rational data type. Go's base-2 logarithm function operates on 64-bit floating point numbers. I used that function for entropy calculations. Accordingly, probability values were converted from rationals to 64-bit floats in that process. As I began to calculate $p(\alpha_i)$ for Δt values into the thousands range, some of the probabilities were truncated to zero. This caused errors in entropy calculations. For that reason, my experiments consider ranges of time steps on the order of hundreds, not thousands.

Changing the Markov process to a binomial distribution enabled significantly faster computation of the distributions at each Δt.

Originally, I profiled my code extensively and rewrote it to make it more efficient. While I achieved signficant speedups, performance was still too poor for my satisfcation. Greatly simplifying the initial problem enabled the greatest improvement in computational performance. It also reduces the applicability of my results, since my initial problem is intentionally easy to compute. Real-world scheduling problems have many characteristics that the initial problem does not consider.

E. Avenues for further research

Further research could ask: Is there a way to encode information to either:

1) Maximize the value-per-bit of the information (such that a utility-accrual scheduler would pay maximum price-per-bit?)
2) Equalize the value-per-bit so that an agent would be willing to pay same price-per-bit of encoded information, so that even an adversary choosing which bits of encoded information to send could decrease their value?

Other information-theoretic measures have been suggested, such as the mutual information between a state space X and the event of an optimal action being taken from state X. [9] Future research could examine the effectiveness of these other information-theoretic measure in scheduling.

F. Comparison to other scheduling problems

The scheduling problem from the motivating example has some differences with some common real-time scheduling problems. Specifically, it

1) does not address dependent tasks, and
2) does not allow preemption.

G. Extensions

The motivating example features a simple model of task utility to facilitate analysis and assessment. To return to the task management system in my introduction, I might make the following modifications:

1) Each task might have a random variable parameter β. The task's utility function could be updated to be $\alpha \log_2(\beta\Delta t + 1)$.
2) Each task's parameters might be sampled from different distributions. This could occur if the system collects data on the variability of tasks' parameters.
3) The system should be able to schedule tasks of non-quantum, stochastic length.

VI. Conclusion

While value-of-information-based heuristics seem promising, directly incorporating entropy-minimization into a scheduler was not very effective in the scheduling problem I constructed.

I discovered that not all Shannon information is equally valuable; some bits of information are worth more than others. The problem of quantifying the relevance of information goes beyond entropy. Literature on this topic looks promising and offers opportunities for further research.

In my very simple problem, the value-of-information heuristic worked well. But more complex real-world problems may not have good heuristics for value-of-information. Perhaps value-of-information-based heuristics could be augmented with information-theoretic concepts or relevance measures to lower computational requirements.

References

[1] S. J. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach (3rd Edition)*. Prentice Hall, 2010.

[2] B. Ravindran, E. Jensen, and P. Li, "On recent advances in time/utility function real-time scheduling and resource management," in *Object-Oriented Real-Time Distributed Computing, 2005. ISORC 2005. Eighth IEEE International Symposium on*, may 2005, pp. 55 – 60.

[3] E. D. Jensen, C. D. Locke, and H. Tokuda, "A Time-Driven Scheduling Model for Real-Time Operating Systems," in *IEEE Real-Time Systems Symposium*, 1985, pp. 112–122.

[4] C. LOCKE, "Best-effort decision making for realtime scheduling," *Ph. D. thesis, Department of Computer Science, Carnegie Mellon University*, 1986.

[5] P. Li, H. Wu, B. Ravindran, and E. Jensen, "A utility accrual scheduling algorithm for real-time activities with mutual exclusion resource constraints," *Computers, IEEE Transactions on*, vol. 55, no. 4, pp. 454–469, 2006.

[6] T. Tidwell, C. Bass, E. Lasker, M. Wylde, C. Gill, and W. Smart, "Scalable utility aware scheduling heuristics for real-time tasks with stochastic non-preemptive execution intervals," in *Real-Time Systems (ECRTS), 2011 23rd Euromicro Conference on*. IEEE, 2011, pp. 238–247.

[7] T. Tidwell and C. Adviser-Gill, "Utility-aware scheduling of stochastic real-time systems," Ph.D. dissertation, WASHINGTON UNIVERSITY IN ST. LOUIS, 2011.

[8] H. Wu, U. Balli, B. Ravindran, and E. Jensen, "Utility accrual real-time scheduling under variable cost functions," in *Embedded and Real-Time Computing Systems and Applications, 2005. Proceedings. 11th IEEE International Conference on*. IEEE, 2005, pp. 213–219.

[9] D. Polani, T. Martinetz, and J. Kim, "An information-theoretic approach for the quantification of relevance," *Advances in Artificial Life*, pp. 704–713, 2001.