Combining Queueing Theory and Scheduling for Dynamic Real World Systems

Student: Tony T. Tran Supervisor: J. Christopher Beck
Department of Mechanical and Industrial Engineering
University of Toronto, Ontario, Canada
{tran,jcb}@mie.utoronto.ca

Abstract

The central thesis of my dissertation is that stochastic reasoning from queueing theory can be integrated with combinatorial scheduling to provide high quality schedules for real world, dynamic systems. This thesis builds on work that showed the potential of such a combination for dynamic systems found in the queueing and scheduling literature. My study strives to extend this concept to handling real world problem areas which are often very time sensitive and restrictive in available system control and information. In such systems, I believe proficient reasoning about system dynamics and combinatorial optimization is essential for creating schedules that are robust and have desirable long-run performance. I will study the performance improvements possible by integration as well as the effects of control, time, and information restrictions of a system.

Introduction

Dynamic and combinatorically complex properties are often present in real world scheduling and resource allocation applications. Dynamism can occur because requests arrive to the system over time and the system manager does not have prior knowledge of how many or when these requests arrive. Decision making must occur online as requests arrive to the system. To further complicate matters, proper control of when and where a request is processed has significant impact on the overall performance of a system. These complications arise because of non-homogeniety in resources or because of performance dependencies in the sequence in which requests are processed. Our aim is to create models for a number of different systems that are dynamic and combinatorically complex - data centre (DC) scheduling and electric vehicle (EV) charging being two such systems.

The DC scheduling problem is concerned with management of a DC comprised of tens of thousands of non-identical machines. The objective is to schedule arriving jobs to decrease response time and increase system throughput. The scale of the system makes optimization very difficult since a system manager must decide which machine should process a job as well as at what time. The resulting system can be thought of as similar to the knapsack problem with tens of thousands of knapsacks, a dynamically changing set of jobs, and added temporal considerations. The optimization of DCs for improved quality of service can result in substantial profits when many of today's largest companies, such as Google, Amazon, and Facebook, rely heavily on their ability to provide fast and reliable service.

The second system we study involves the charging of EVs. Owners and potential owners of EVs are often concerned that recharging a battery is inconvenient when compared to refuelling a combustion engine vehicle. Unlike refuelling, recharging a battery can require hours to complete. As such, making a stop to recharge in the same manner as one would refuel is often not a possibility. Recharging should be accomplished during periods when a driver would be occupied with other activities. One example of convenient charging is a parking lot. People naturally leave their cars for extended periods in parking lots when they are at a shopping mall, work, or flying out at an airport. These locations provide a convenient place in which an EV can be charged while the drivers are occupied. However, due to power and space limitations, ensuring everyone receives their requested charge may not be possible. We wish to explore these parking lot systems and develop models, using queueing theory and scheduling, to manage the strategic and operational issues which are present.

To properly handle systems that are both dynamic and combinatorially complex, we propose the use of queueing theory and scheduling. Queueing theory and scheduling are two research areas that have developed independently. One can describe both queueing theory and scheduling research as the pursuit of understanding and optimizing the use of scarce resources over time. In a general sense, all systems in these two research areas are comprised of jobs which must be processed by servers. The two approaches diverge fundamentally on the problem characteristics that form their core challenges. In queueing theory, systems are dynamic and uncertain, but rarely have combinatorics. Scheduling, in contrast, generally studies static systems with complex combinatorial properties. The emphasis on these different characteristics leads to distinct theoretical and problem solving techniques. Thus, we believe that queueing theory and scheduling have developed the tools necessary for handling different aspects of the systems we are interested in, but the work on combining the two research areas have just begun.

The primary aim of this research is to investigate how the techniques developed in scheduling and queueing theory can be combined to better represent and manage dynamic combinatorially complex systems. We believe that high level analysis of queueing theory and low level decision making of scheduling can lead to improved performance in comparison to traditional methods. Through proper integration of the tools from the two areas of study, both the combinatoric and dynamic properties of the system can be addressed. The systems we study are ones with promise for improvement under a hybrid queueing theory and scheduling scheme.

Background

Uncertainty in scheduling is often present because the processing times are not known or machines fail (Bidot et al. 2009). For uncertainties because of dynamism, where jobs are added or removed, scheduling is generally viewed as a collection of static problems which are solved one after the other. Within such a framework, the sophisticated scheduling tools developed for deterministic systems can be applied to a set of these dynamic ones. While static scheduling in such a manner enables the application of traditional scheduling technology to dynamic systems, there tends to be little emphasis on long-run performance and considerations of the dynamics of the problem; notable exceptions being works on anticipatory scheduling (Branke and Mattfeld 2002) and online stochastic combinatorial optimization (Van Hentenryck and Bent 2006).

On the queueing theory side, combinatorics have not been of great interest. The body of research is generally concerned with providing optimal policies in some *probabilistic* sense over a long time horizon. Given the difficulties of providing optimal policies under highly combinatorial systems, the queueing community often ignores such structures. Yet, scheduling in queueing has been shown to greatly affect the performance of the system (Wierman, Winands, and Boxma 2007). However, the work only considers a set of simple systems and the analysis of a more combinatorially complex problem would be much more difficult.

Queueing theory and scheduling can be seen as complementary. The two share the same high-level goals, but the core research problems are very different. Research on the integration of queueing theory and scheduling has recently appeared. Terekhov et al. (2012) looked at stability guarantees of dynamic two machine flow shop environments for a scheduling model. They also showed the performance gains one could achieve when sacrificing optimal short term solutions in place of achieving long term metrics. Another recent paper looked at hybrid queueing and scheduling algorithms for scheduling jobs on alternative resources with setup times (Tran et al. 2013). They show that guidance from queueing analysis can help scheduling algorithms find high quality schedules that perform well under long-term metrics. These works provide an indication of the potential of integrating queueing theory and scheduling, yet, the works are on small, stylized systems.

Data Centre Scheduling

In a DC, jobs are processed on servers housed on racks. There are thousands of racks which each contain a number of servers. The set M represents the different server configurations (types) in the system. Each configuration has N_m identical servers defined by their total resource capacity. Based on a study of the trace data in Google's compute clusters, the system is constrained by CPU and memory (Mishra et al. 2010). Therefore the configurations are distinguished based on these two resources.

The jobs that arrive to the system are comprised of one or more tasks. Each task is to be processed on one of the servers where machine resources are consumed for the task's total duration. Between the tasks of a single job, there may exist precedence constraints that enforce completion of certain tasks before other tasks can begin processing. A job is considered complete when all tasks belonging to it are serviced.

Uncertainty is assumed to be present only in the fact that future jobs are not known; complete information of a job is available upon arrival to the system. However, it is assumed that there is access to information about the dynamics of the system. Specifically, the distribution of the arrival rate of jobs along with probability distributions for the resource requirements of tasks are given. Mishra et al. (Mishra et al. 2010) classify jobs on Google's compute clusters into eight different job classes, defined by the arrival rate together with the CPU and memory requirements of a class. The job class information allows us to analyze the system behaviour.

The objective of the problem is to schedule jobs to minimize the mean time-in-system of the jobs and maximize the system throughput. Given that the machines have different resource capacities and the tasks have different resource consumption profiles, an efficient matching of tasks to machines can have a positive effect on resource utilization. Thus, our belief is that queueing theory accommodates the higher level issue of matching job classes and machine configurations whereas scheduling ensures proper matchings from the queueing analysis is realized during execution.

Scheduling Algorithm

A hybrid queueing and scheduling algorithm with three-stages is proposed to handle the scheduling of jobs to machines in a DC. This hybrid algorithm is called long-term evaluative scheduling (LoTES). The first stage aims to provide high level analysis to understand the dynamics of the system and find the best mix of jobs for machines. Stage two then creates an interpretation of the analysis for a low level scheduler. Finally, in the last stage, the interpreted model is used to dispatch arriving jobs as they enter the system. The break down of the three stages shows where one would benefit from using either queueing theory or scheduling. At the first stage, queueing theory is well adapted to provide understanding of the system dynamics. In contrast, scheduling is better equipped for the two lower level stages which attempt to realize the high level analysis.

Full details of the first stage are presented. In the interest of space, only a brief description of the latter two stages is given

First Stage In the first stage, jobs are treated as a continuous fluid where the rate of arrival is the inflow and the throughput is the outflow rate. The allocation linear program (LP), used in queueing networks, is adapted to the DC system (Andradóttir, Ayhan, and Down 2003). However, the allocation LP has previously only been used for unary resources. Therefore, the LP must be modified to provide a bound on performance when there are multiple multi-capacity resources. The solution of the allocation LP is an efficient matching of job classes to machine configurations. The allocation LP is as follows,

$$\max \quad \lambda$$

$$\text{s.t.} \quad \sum_{j\in M}(\delta_{jkl}c_{jl}n_j)\mu_k \geq \lambda\alpha_k r_{kl}, \quad k \in K, l \in R \quad (1)$$

$$\frac{\delta_{jkl}c_{jl}}{r_{kl}} = \frac{\delta_{jk1}c_{j1}}{r_{k1}}, \qquad j \in M, k \in K, l \in R \quad (2)$$

$$\sum_{k\in K} \delta_{jkl} \leq 1, \qquad j \in M, l \in R \quad (3)$$

$$\delta_{jkl} \geq 0, \qquad j \in M, k \in K, l \in R \quad (4)$$

where,

Decision Variables

- λ: Arrival rate of batch jobs
- δ_{jkl}: Fractional amount of resource l machine j devotes to class k

Parameters

- α_k: Probability of a job being in class k
- μ_k: Service rate of tasks that belong to jobs in class k
- c_{jl}: Capacity of resource l on machine type j
- r_{kl}: Expected requirement of resource l for tasks belonging to jobs of class k
- n_j: Total number of machines of configuration type j

The LP assigns the fractional amount of resources that each machine configuration type should allot to classes with the goal of maximizing the capacity of the system. Constraint (1) guarantees that enough resources are allocated for the requirements of each class. Constraint (2) ensures that the resources that are assigned to a job match the resource requirement profiles. The allocation LP prevents assigning more resources than available with constraint (3). Finally, constraint (4) ensures the non-negativity of assignments.

Solving the allocation LP provides δ^*_{jkl} values which tell us how to efficiently allocate jobs to machine configurations. However, without accounting for fragmentation, the allocation LP only acts as an upper bound on the achievable capacity of a system rather than a tight bound which Andradóttir et al. (2003) are able to achieve for the unary resource problem. Fragmentation occurs because the actual system must handle discrete tasks. For example, if a there is a single machine with 4 GBs of memory available and 4 tasks are present, each requiring 1.1 GBs of memory, then only 3 tasks can be processed with 0.7 GBs of memory idle. With discrete tasks, idle resources are always expected to be present and the arrival rate λ^* found is not actually achievable.

Second Stage The next stage takes the solution of the fluid representation and attempts to discretize the tasks. It begins by defining *bins* as snapshots of a mix of tasks on a machine type where the sum of resources required by the tasks does not exceed the resource capacity of the type. Using a LP model, the optimal set of bins to use in the DC is solved. Since the LP model is solved offline when the set of tasks to be scheduled are not yet known, we assume task resource requirements are equal to the expectation for the class.

The solution from the allocation LP of the first stage helps restrict the job classes that each machine must consider. This greatly reduces the search space that must be considered and allows for consideration fewer bins.

Third Stage The final stage occurs online as jobs arrive. A dispatcher assigns tasks to machines based on the system status and bins from the second stage. In general, the dispatcher assigns tasks in a way to ensure machines partially mimic the bins. However, the dispatcher is aware of system load and deviates from the stochastically optimal solution if there are idle machines. In this way, the dispatcher tracks what is deemed to be good long term decisions, while using online information to capitalize whenever anomalies occur.

To maintain fairness between tasks and ensure that no task will be starved, first come, first served (FCFS) ordering is enforced once a task is assigned to a machine.

Initial Results

We simulate a system with 10,000 machines comprised of 10 different machine configurations. Taking job classes from Mishra et al. (2010) we solve the allocation LP and test varying system loads based on λ^*. To compare performance against the LoTES model, we make use of a greedy scheduler that assigns tasks to an available machine arbitrarily or in the case that no machines are available, to the machine with the smallest queue of tasks. Similar to LoTES, once assigned to a machine, a task will be served in FCFS order.

Figure 1 presents simulation results of comparing the LoTES model against the greedy heuristic. We see that large performance increases is possible when using the LoTES model. At 90% loads, we see that the greedy scheduler will not be able to serve tasks in any reasonable amount of time (hours), but using LoTES only results in a few seconds wait on average.

Scheduling an Electric Vehicle Charging Facility

We consider an EV charging facility with $N \in \mathbb{N}$ docks, each with $K \in \mathbb{N}$ cables. A cable connects a dock to a car and enables charging. However, being connected does not mean that the car is able to immediately start charging. Each dock is limited to charging a single car at a time.

We assume cars arrive dynamically following a Poisson process with rate λ. The charging time each EV requires is exponentially distributed with mean μ^{-1}. In order for a vehicle to leave the system, two conditions must be met: 1) the

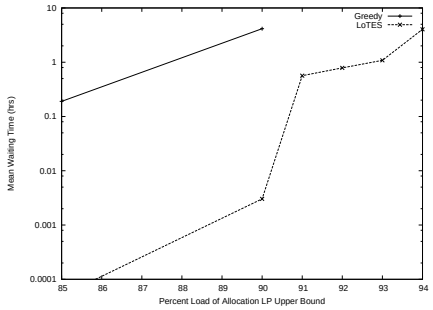

Figure 1: Mean over 20 Runs - Mean Waiting Time Performance

required charge is completed and 2) the deadline specified by the driver is reached. We assume the deadline is exactly L time units after the arrival of the EV and represents the time at which the customer has agreed to return to remove the EV. If a vehicle is charged before the deadline, it must wait until the deadline before it can exit because, typically, the driver will not return for the EV before the deadline. On the contrary, if charge completion occurs after the deadline, the EV is delayed and must wait until the charge completes before exiting the system.

We assume three information conditions for our system. The first is referred to as the *cardinality* condition: it is known how many EVs are at each dock and of those, which vehicles are delayed or charged. Under the second, *stochastic*, condition, the arrival (λ) and charging (μ) rates and their distributions are available. It corresponds to assumptions common to queueing theory (Gross and Harris 1998). Finally, we wish to consider information natural to the scheduling community (Pinedo 2005) which tends to include deterministic information about job durations and, often, arrival times. While deterministic arrival times are unrealistic in our application, it is reasonable to assume that charging time information is known upon an EV arrival. For example, the charging time can be found from either requesting the customer give the charge level they wish to purchase or by having a wireless transmitter from the vehicle broadcast this information.[1] Therefore, our third condition, which we term *observable*, assumes that the remaining charging times of every EV present in the system can be observed. For an EV j, the charging time p_j is available upon arrival.

The system manager makes two decisions. The first is whether to accept or reject an incoming vehicle. If rejected, then there is a finite cost $c_r \geq 0$ for losing a customer. The second decision is how to schedule an accepted EV. Scheduling comprises of the decision of assigning a dock for an EV and determining the order that EVs are charged. If accepted, an EV is immediately assigned to an available cable and cannot be switched. When the owner returns to pick up his/her EV, if charging is not yet complete, the delay is penalized. If T_j is the tardiness of a late EV j, that is, the time between the EVs deadline and when its charge is completed, then the delay cost is $c_d T_j$ where c_d is finite and non-negative.

The system manager wants to find an admission and scheduling policy to minimize the overall system cost. However, the control a system manager has over a parking lot may vary. One can see in most common parking lots, customers arrive and choose a spot themselves. Here, a system manager would have no direct control over customers. Thus, an indirect method to control the system is by limiting the available spots (docks and cables). Although we do not explore the capacity planning problem, we will observe some of the effects of adding cables and docks to the system in our experiments. We can also imagine a facility where customers must purchase a spot first and will then be assigned to a specific location. In this way, complete control over the admission and scheduling of a vehicle is possible.

Continuous-Time Markov Decision Process

We present a CTMDP model of a charging facility when only cardinality and stochastic information is available. Our current definition of deadlines being a fixed L time units after arrival does not adhere to the memoryless requirement of a CTMDP. Therefore, we assume that deadlines are not deterministic, but exponentially distributed with mean L. We further simplify the CTMDP by enforcing first-come, first-served (FCFS) ordering of EVs once assigned to a dock.[2]

The state of the system at time t is represented by, $\mathbf{S}(t) = \{\mathbf{Q}(t), \mathbf{W}(t), \mathbf{D}(t)\}$. Here, $\mathbf{Q}(t)$, $\mathbf{W}(t)$, and $\mathbf{D}(t)$ are vectors of size N. $\mathbf{Q}(t)$ indicates the number of EVs in the system at time t that are waiting for a charge and not yet due on each of the N docks. $\mathbf{W}(t)$ represents the number of vehicles that have completed their charge, but are waiting for the deadline and $\mathbf{D}(t)$ is the number of vehicles that are not yet charged but have already reached their deadline, on each of the N docks. We represent the element in each of the vectors using a lowercase letter with index n to indicate the dock (e.g., the nth dock is fully described by $q_n(t)$, $w_n(t)$, and $d_n(t)$).

There are $N+1$ possible actions when an EV arrives. An action, $a \in A$, can either assign the EV to one of the N docks or reject the vehicle. Therefore, $A \in \{0, 1, \ldots, N\}$ where 0 represents rejection. The cost function, $C(\mathbf{S}(t), a)$ defines the expected cost associated with action a in state $\mathbf{S}(t)$. When a vehicle is rejected, independent of the current state, the cost is c_r. If a vehicle is admitted, then we must calculate the expected cost associated with the additional vehicle for each particular state. We denote the time that an EV j completes its charge as ϕ_j. Thus, the delay cost of a vehicle is $\max\{0, (\phi_j - L)c_d\}$.

We calculate the expected delay of an accepted vehicle by conditioning on the state of the system at time t and the dock n that will be assigned the vehicle. Since there are $q_n(t) + d_n(t)$ vehicles not yet charged on dock n, admission

[1] Such transmitters are already available (Botsford 2012), but not used widely.

[2] We found numerically through simulation that a system with deterministic deadlines does not behave differently from the calculated CTMDP with exponentially distributed deadlines under FCFS. Due to space restrictions, we do not present these details.

of a new vehicle requires a total of $B = q_n(t) + d_n(t) + 1$ exponentially distributed charges until the arriving vehicle has completed its charge. Thus, the expected delay given a state i and assignment to dock n is,

$$\mathrm{E}[\text{delay}|\mathbf{S}(t), a = n] = \frac{\left[\mu^{-1}\Gamma(B+1, \mu L) - L\Gamma(B, \mu L)\right]}{\Gamma(B)}.$$

Here, $\Gamma(b)$ is the gamma function and $\Gamma(b, \mu L)$ is the upper incomplete gamma function. Therefore, for any action a, we know the expected delay, which we multiply by c_d to obtain the expected delay cost.

The transition rates depend on the system state, $\{\mathbf{Q}(t), \mathbf{W}(t), \mathbf{D}(t)\}$. Transitions occur because of three types of events: EV arrival, charge completion, and meeting a deadline. We define an N-sized vector $\mathbf{e}_n$ that has 1 as the nth element and the rest 0. A deadline can occur on any dock which has $q_n(t) + w_n(t) > 0$. If $w_n(t) > 0$, then a transition occurs with rate $\frac{q_n(t)+w_n(t)}{L}$ and will change the state to $\{\mathbf{Q}(t), \mathbf{W}(t), \mathbf{D}(t) - \mathbf{e}_n\}$. If $w_n(t) = 0$, a transition occurs with rate $q_n(t)L$ to state $\{\mathbf{Q}(t) - \mathbf{e}_n, \mathbf{W}(t), \mathbf{D}(t) + \mathbf{e}_n\}$. Charge completions can occur on any dock which has $q_n(t) + d_n(t) > 0$. If $q_n(t) + d_n(t) > 0$, a transition occurs with rate μ to $\{\mathbf{Q}(t), \mathbf{W}(t), \mathbf{D}(t) - \mathbf{e}_n\}$ if $d_n(t) > 0$, and $\{\mathbf{Q}(t) - \mathbf{e}_n, \mathbf{W}(t) + \mathbf{e}_n, \mathbf{D}(t)\}$ otherwise. For an arrival event, we must consider the action taken. If an EV is rejected, then there is no transition. If we decide to assign an arriving vehicle to the nth dock, then there is a transition rate of λ to $\{\mathbf{Q}(t) + \mathbf{e}_n, \mathbf{W}(t), \mathbf{D}(t)\}$.

The CTMDP suffers from the curse of dimensionality: solving the CTMDP for real life problems is intractable as the number of states grows exponentially. The number of states for any particular system is $(K+1)^{2N}$. With five cables and five docks, we see that there are more than 60 million states. Thus, such a model is intractable for parking facilities of even moderate capacity. Nevertheless, this model can guide us to heuristics that use stochastic information which we present in the following section.

System Control

Three different admission and scheduling policies are shown, each corresponding with one of the three information conditions. Depending on the level of control and amount of information available, a combination of an admission and scheduling policy can be used to mange the system.

Admission Policy The admission policy decides whether to accept an arriving EV. The policies also decides which docks are able to be assigned the EV. We present three policies which represent systems that, respectively, use cardinality, stochastic, and observable information:

- Free Cable - A vehicle is admitted if there are available cables - i.e., if $\exists n \ : \ q_n(t) + w_n(t) + d_n(t) < K$. Any dock with an available cable may be assigned the EV.
- CTMDP1 - Consider a single-dock version of the system with $\frac{\lambda}{N}$ rate of arrival. Solve for the optimal single-dock policy using CTMDP. If any of the docks are in a state which would accept an EV, accept the EV and restrict scheduling to one of these docks.
- Myopic - Using the charging times, calculate the delay cost of scheduling an EV on each dock. If the cost of accepting the EV on the dock is less than the cost of rejecting the EV, then accept and assign to one of these docks.

Although the admission policy will limit how one can assign an EV, it does not assign a dock.

Scheduling Policy The scheduling policy assigns an EV to a dock once admitted. Again, each policy represents systems that, respectively, use cardinality, stochastic, and observable information.

- Random - Randomly choose among one of the possible docks determined by the admission policy.
- CTMDP2 - Similar to CTMDP1, restrict the CTMDP model to a single dock and solve the Bellman equations to find the expected cost of being in each state. From the possible docks as defined by the admission policy, choose the dock in a state that yields the minimum expected cost.
- Earliest - From the set of possible docks defined by the admission policy, choose the dock that will result in the earliest completion time for the EV if all other already assigned EVs complete charge first.

These policies represent different levels of control from no involvement, where we expect customers to choose a cable randomly, to complete control where customers are sent to particular docks to maximize performance. Once assigned to a dock, EVs are charged in FCFS order.

A system manager couples an admission policy with a scheduling policy to control the facility. However, the information availability limits the choice of policies; for example, CTMDP1-Earliest can only be used if cardinality, stochastic, and observable information conditions are met.

Initial Results

We simulate the charging facility with 10 instances of 100,000 time units for every admission-scheduling policy pair. We use $L = 1$ and $\mu = 6$. For example, if our time unit is 3 hours, the parameters represent a parking lot which customers park for 3 hours and request on average 30 minutes charging time. In this system, there are ten identical docks with between one and ten cables each. Vehicles arrive at a rate of $\lambda = 50$ costs are $c_r = 1$ and $c_d = 5$.

The results of our simulation, found in Figure 2, illustrates the importance of information availability. The performance of Myopic-Earliest shows the clear advantages of having observable information. Further, obtaining some control of the system is quite important as Free Cable-Random is found to perform very poorly once there are seven or more cables. In fact, even Myopic-Random suffers when K increases since there is less control over the scheduling of EVs.

Although observable information is found to be the most important information condition for obtaining high quality performance, we have witnessed scenarios where using stochastic information has led to better performance than Myopic-Earliest. Figure 3 fixes the number of cables to 10 per dock and varies c_d from 1 to 200. We see that as

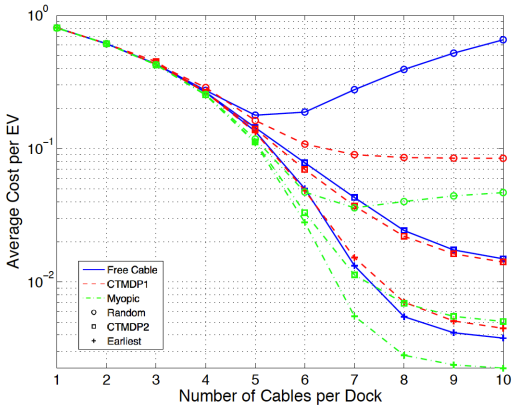

Figure 2: Experiment 2 - Multiple Dock Charging Facility.

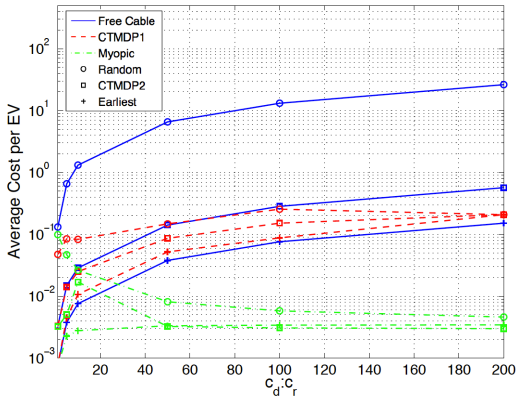

Figure 3: Experiment 3 - Multiple Dock Charging Facility.

c_d increases, Myopic-CTMDP2 becomes the best performing pair. The improvements are only marginal, however we believe a more sophisticated hybrid that better integrates stochastic reasoning and combinatorics, such as those proposed by Tran et al. (2013), could lead to better performance.

Contribution

The contributions of this work is in developing a hybrid queueing and scheduling algorithm to handle the dynamic and combinatorial aspects of scheduling in real world applications. The two systems of interest are motivated by industry and provide problem characteristics that are outside the scope of the classical scheduling or queueing literature. The integration of techniques from these two literatures can help provide richer modelling tools to capture the dynamics and the combinatorics often seen in real life. Further, the research aims to take advantage of the fact that stochastic and observable information is available in many systems and by developing the appropriate algorithmic models, long term performance and short term goals can be optimized.

This work will extend the initial hybridization of queueing and scheduling by Terekhov et al. (2012) and Tran et al. (2013) to look at real world systems with complex problem structures, highly dynamic processes, and time sensitive requirements in decision making. For such systems, it is critical to use stochastic reasoning provided by queueing theory and combinatorial optimization as found from scheduling.

References

Andradóttir, S.; Ayhan, H.; and Down, D. G. 2003. Dynamic server allocation for queueing networks with flexible servers. *Operations Research* 51(6):952–968.

Bidot, J.; Vidal, T.; Laborie, P.; and Beck, J. C. 2009. A theoretic and practical framework for scheduling in a stochastic environment. *Journal of Scheduling* 12(3):315–344.

Botsford, C. 2012. The economics of non-residential level 2 EVSE charging infrastructure. In *Proceedings of the 2012 EVS26 International Battery, Hybrid and Fuel Cell Electric Vehicle Symposium*, 1–10.

Branke, J., and Mattfeld, D. C. 2002. Anticipatory scheduling for dynamic job shop problems. In *Proceedings of the ICAPS'02 Workshop on On-line Planning and Scheduling*, 3–10.

Gross, D., and Harris, C. 1998. *Fundamentals of Queueing Theory*. John Wiley & Sons, Inc.

Mishra, A.; Hellerstein, J.; Cirne, W.; and Das, C. 2010. Towards characterizing cloud backend workloads: insights from google compute clusters. *ACM SIGMETRICS Performance Evaluation Review* 37(4):34–41.

Pinedo, M. L. 2005. *Planning and Scheduling in Manufacturing and Services*. Springer.

Terekhov, D.; Tran, T. T.; Down, D. G.; and Beck, J. C. 2012. Long-run stability in dynamic scheduling. In *Proceedings of the 22nd International Conference on Automated Planning and Scheduling (ICAPS 2012)*, 261–269.

Tran, T. T.; Terekhov, D.; Down, D. G.; and Beck, J. C. 2013. Hybrid queueing theory and scheduling models for dynamic environments with sequence-dependent setup times. In *Proceedings of the 23nd International Conference on Automated Planning and Scheduling (ICAPS 2013)*, to appear.

Van Hentenryck, P., and Bent, R. 2006. *Online Stochastic Combinatorial Optimization*. MIT Press.

Wierman, A.; Winands, E.; and Boxma, O. 2007. Scheduling in polling systems. *Performance Evaluation* 64:1009–1028.