SEQUENCE-DEPENDENT CHANGEOVER AND PROCESSING TIMES FOR STAGED QUEUE SCHEDULING

Robert M. Nauss, College of Business Administration, University of Missouri-St. Louis, 1 University Blvd, St. Louis, MO 63121, 314 516 6130, robert_nauss@umsl.edu
L. Douglas Smith, College of Business Administration, University of Missouri-St. Louis, 1 University Blvd, St. Louis, MO 63121, 314 516 6108, ldsmith@umsl.edu
Dirk Christian Mattfeld, Decision Support Group, University of Braunschweig, d.mattfeld@tu-bs.de
Jan Fabian Ehmke, Decision Support Group, University of Braunschweig, j-f.ehmke@tu-bs.de
Jian Li, Decision Support Group, University of Braunschweig, lijianjoin@googlemail.com

ABSTRACT

Multiple queues in which a preparatory step is required before service can begin are often found in facilities for production, transportation and logistics. The result is a staged queue with a restricted queuing discipline and sequence-dependent changeover and processing times. We compare solutions from an integer programming model with an efficient heuristic for scheduling operations to minimize average waiting times while promoting equity among users with different service requirements. We show that the heuristic generates excellent solutions when compared with the optimal solutions. We demonstrate how system performance can be improved relative to simpler scheduling rules such as FIFO or shortest processing and setup time. We utilize the heuristic (in lieu of the IP model) in a computer simulation for a transportation system.

INTRODUCTION

River locks, traffic intersections, port facilities and distribution centers often allocate resources to serve arrivals from different streams and need to switch queues periodically to avoid unduly long waiting times for some customers while trying to achieve efficiency in flow. One class of problem involves situations where the first entity to be processed from a queue needs to be positioned for access to the service facility, or may require some preparation before it is ready for processing. The result is a set of staged queues, where the entity at the head of a queue from an arrival stream must be selected for processing before any entity behind it, but flexibility remains in choosing the next entity from that stream to be prepared for processing. A schematic of such a queuing structure with two arrival streams is provided in Figure 1. In this situation, a prepared entity at the head of queue 1 or queue 2 must be chosen next for processing. When the chosen entity is removed from its preparation point, any entity in the queue behind may be chosen to be prepared next (i.e., to be placed at the preparation point while work is being completed on the item being serviced).

Previous work by the authors in [11] [12] [13] confronted such a problem in the context of scheduling traffic at locks in a river transportation system. Here upstream and downstream lockage operations had different processing times and were subject to different delays as departing entities cleared the way for oncoming traffic to enter the lock. They derived integer programming (IP) models for identifying the sequence of operations that would clear all traffic from an existing set of queues while limiting the maximum waiting time for any entity to an accepted tolerance beyond the time that the entity would have to wait if a First-In, First-Out (FIFO) processing sequence were adopted. In addition, they developed a scheduling heuristic that could be employed in a stochastic environment and tested it against the IP model on a series

of test problems that were randomly generated to conform to historical arrival patterns. In all trial problems where it was generally more efficient to continue to serve entities from the same queue, the IP and heuristic solutions were identical. In some cases where the processing environment generally favored alternating service between queues to achieve overall operating efficiency, however, the IP solution was slightly superior. In this paper, we discuss how it may be embedded in scheduling systems with the aforementioned queuing structure.

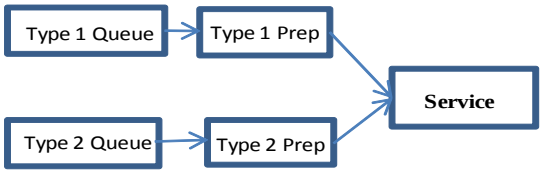

FIGURE 1. SCHEMATIC OF A SERVICE FACILITY WITH TWO STAGED QUEUES

PREVIOUS RESEARCH

Most of the published work on scheduling operations with sequence-dependent setup times and processing times assumes: (1) that the entities to be processed are in a single queue and (2) any of the queued entities can be selected to be processed first. Articles [1] [2] surveyed over three hundred papers on job-shop scheduling with consideration of the setup times required for each job. Most of the papers ignored the stochastic nature of processing and setup times to make the test problems analytically tractable. Various IP models and heuristic solution procedures were proposed to solve job-shop scheduling problems when setup times depend on job sequences ([3] [4] [5] [6] [7] [8] [9] [10]).

None of the above, however, accommodated the type of staged queuing system in Figure 1. This can be a significant oversight in situations where physical constraints impose such a limitation – as commonly encountered in applications for traffic, transportation, production and logistics.

RESTRICTED QUEUING STRUCTURE SCHEDULING

Upon service completion for an entity, a decision must be made whether to accept the next entity from the same queue as the entity just serviced, or whether to select a prepared entity from another queue. We devise a heuristic scheduling procedure for solving it by contriving a two-queue problem where the first entity to be processed from each queue must be the one positioned at the head of the queue and where the setup and processing times for each entity depend upon the queue from which the previous entity was drawn. We assume that the system serves two classes of entities (large and small) each with different setup (changeover) times and processing times that depend on the queue in which they reside.

We devise the heuristic to mimic the optimal IP solution and in many cases the heuristic scheduler matches the IP solution. For others, the heuristic solution has a relatively small gap. Large entities and entities in queue 1 requiring additional work and efficiencies can be achieved by changing queues because of smaller setup times when changeovers occur. Such situations may arise when a single resource alternates between two processing streams and positioning of

the next entity to be prepared for service in a particular stream may occur with consideration of the expected processing times of each of the queued entities, depending on whether a switch between queues occurs. We assume that the preparation time required for an entity placed at the head of the queue is always less than the smallest processing time.

The Heuristic Scheduler

The heuristic begins with the solution that selects the next item to prepare for processing as the one that has the lowest sum of preparation and processing time. Then it employs a neighborhood search process to explore for opportunities to reduce the sum of the setup and operating times by altering the processing sequence. For brevity, we omit the details of the scheduler and instead present a summary.

The first phase of the algorithm is geared to process entities in increasing order of processing times to the extent possible (fastest processing time (FPT)), with the general expectation that it is more efficient to continue to process entities from the same queue. All entities in position 2, 3, etc. of each queue are therefore first sorted according to their processing times assuming that the preceding entity was processed from the same queue.

In Phase 1, we search for opportunities to reduce total wait while processing entities successively from the same queues. Phase 2 is a neighborhood search. We complete the shift of entities with the assumption that the sequence of all those chosen from a particular queue is retained. Next, we examine if reductions in total waiting time can be achieved by alternating queues (to accommodate unusual situations where a longer setup time may be required to process entities from the same queue instead of from the opposite queue). In phase 3 we branch to test for effects of alternating the selection of the first entity. Here we do a simple branch and assess both selections to determine the preferred selection.

Implementing Priority Shifting to Encourage Equity

When entities of one type are able to be processed more efficiently than entities of another type, our solutions can result in unacceptably long waits for the latter (in both queues). To mitigate this, we can employ a priority shifting mechanism whereby entities are shifted into a higher priority class after a designated interval. If *waitlim* is the length of that interval, *tnow* is the present time and *arrivtime* is the time of arrival for an entity, the number of priority shifts that an entity experiences at *tnow* is *nshifts* = int((*tnow*-*arrivtime*)/*waitlim*)). Generally, we would like to have all entities with a given *nshifts* value precede entities with a lower *nshifts* value in each queue. That can be arranged except for the preparation stage because a small entity, for example, may arrive after a large entity has been positioned for preparation. In our implementation of the heuristic with priority sharing, we change the value of *nshifts* for an entity at the preparation stage to the highest value of *nshifts* for any entity queued behind it. Then we solve the sequencing problem only for entities with *mxnshifts* = max{ *nshifts* value in any queue}.

There are other alternatives that may be employed when shifting priorities. One could count the number of entities with the maximum current value for *nshifts* in each queue. If they differ,

processing could be assigned to the prepared entity at the head of the queue with the maximum number of entities in the queue with *nshifts=mxnsnifts*. Alternatively, the total processing and setup times could be weighted by some function of *nshifts* when evaluating solutions in Phase 2 and Phase 3 of the heuristic. The former alternative would be simpler than the one we employ; the latter would be more complex.

APPLICATION OF THE HEURISTIC SCHEDULER IN A STOCHASTIC ENVIRONMENT

The actual times that are required for setups and processing of entities are subject to uncertainty and the mix of entities in the queue can change as new arrivals occur during processing. There is a question, therefore, of whether the more complex scheduling mechanism will improve long-run performance over simpler decision rules such as FPT with priority shifting. To test this, we have applied the heuristic in a complex simulation model involving bi-directional queues at a series of locks in an inland waterway (see [12]).

Towboats with tows of barges transit the waterway navigation system through a series of locks that can accommodate up to seven barges (small tows) without reconfiguring the tow and employing auxiliary support. Large tows (usually with 15 barges), in contrast, need to be processed at each lock by decoupling a portion of the tow. The first portion is then processed through the lock and then tied up just outside the lock. Then the water level of the lock is returned to its previous state, and the second portion of the tow is processed through the lock. Both tow portions are then recoupled and the area is cleared to allow other vessels and tows to enter the lock from the opposite direction.

Average times required for lockage operations involving large tows is approximately two hours, but for small tows it is about one-half hour – depending on time of the year, direction of travel, and the specific lock at which the operation takes place.

Detailed historical records of every lockage operation at a series of five locks over an entire shipping season were used. The purpose was to assess the extent to which average waiting times for upstream and downstream river traffic could be reduced by employing more efficient scheduling procedures than the basic FIFO rules currently employed for normal operations.

It was necessary to simulate the greatly varying mixes of traffic that occur at different times (months) of the year, day of the week, and time of the day in order to test alternative scheduling procedures. The simulation model was created in Arena 10.0 with an embedded C++ routine to determine whether the vessel to be locked next is the one positioned to enter the lock on the downstream or upstream side. The decision is made as a vessel departs the lock. Which vessel to process next is determined with consideration of which vessels are positioned at the head of the queue in each direction (and therefore which vessel must first be processed from each direction). Also considered is the free choice of vessel that may next be positioned for processing in a particular direction after the vessel at the head of a queue is selected to enter the lock. The chosen solution is the one that would complete the processing of all currently queued vessels and tows with minimum total (or average) waiting time, assuming no further arrivals occur. This is exactly the staged queuing process problem described earlier.

Lock traffic at the locks was generated randomly from time-varying exponential distributions to include six classes of vessel-barge tow configurations. Lognormal distributions with systematically varying means and coefficients of variation were used for processing times. The individual time for an operation is generated after the simulated operation starts – thus introducing the type of uncertainty faced in the natural environment. Vessel itineraries were constructed with a Markovian process that determined whether the vessel should proceed to the next lock in the system or, alternatively, terminate its voyage in the section of the river most recently reached. Probabilities of terminating the voyage in the current section of the river were varied according to the month of the year and nature of the barge tow.

CONCLUSION

Results of the simulation experiments in the stochastic environment confirm that marginal improvements in system performance may be achieved by employing a heuristic queue-clearing scheduler that considers the sequence-dependent processing and setup times of all entities in staged queues even though the selection of the next entity to be processed must always occur from entities at the preparation stage (i.e., at the head of the respective queue) and the problem changes dynamically with each new arrival.

TABLE 1 – WAITING TIMES FOR ALTERNATIVE SCHEDULING RULES WITH STOCHASTIC PROCESSING AND CHANGEOVER TIMES

Entity Class	FIFO (benchmark)		FPT (no priority shift)		Heuristic (no priority shift)		FPT (with priority shift after 480 minutes)		Heuristic (with priority shift after 480 minutes)	
	Median	95th	Median	95th	Median	95th	Median	95th	Median	95th
Large	308	3,605	169	3,881	284	3,306	372	3,100	279	2,965
Small	273	3,390	106	653	147	618	153	2,893	153	2,606
Average Wait	715		561		548		632		600	
Improvement relative to FIFO			21.50%		23.30%		11.60%		16.10%	

For our simulation of an actual service system (see Table 1), FPT generated improvement relative to FIFO scheduling and the heuristic generated further improvement (with reduction in waiting times of 21.5% and 23.3% respectively in the case when no priority shifting was employed to foster equity among classes of users). Somewhat surprisingly, greater relative benefit from the heuristic was achieved from using the heuristic when priority shifting was employed. With priority shifts after every 480 minutes and restricting choice to entities in the highest priority class, FPT with priority shifting generated improvement of 11.6% and the heuristic generated improvements of 16.1% over FIFO. The system analyzed was operating at peak capacity in the summer season; so random variation in arrivals and system impairments can have substantial effects on waiting times. More research is required to test the efficacy of the heuristic in a variety of problem settings with different operational parameters.

Our heuristic scheduler can be adapted to other situations in transportation and production where sequence-dependent setup times and restricted queuing disciplines are encountered. Other applications, such as cross-docking activities, seaport operations and intermodal terminal facilities are likely to require a generalization of the heuristic to handle more than two queues.

REFERENCES

[1] Allahverdi, A., Gupta, J.N.D., & Aldowaisan, T. A review of scheduling research involving setup considerations. *Omega*, 1999, 27, 219-239.

[2] Allahverdi, A., Ng, C.T., Cheng T.C.E., & Kovalyov, M.Y. A Survey of Scheduling Problems with Setup Times or Costs. *European Journal of Operational Research*, 2008, 187, 985-1032.

[3] Balas, E., Simonetti, N., & Vazacopoulas, A. Job shop scheduling with setup times, deadlines and precedence constraint. *Journal of Scheduling*, 2008, 11, 253-262.

[4] Carroll, J. L. & Bronzini, M. S. Waterway transportation simulation models: development and application. *Waterway Resources Research*, 1973, 51-63.

[5] Dai, M.D.M. & Schonfeld, P. Metamodels for estimating waterway delays through series of queues. *Transportation Research Part B*, 1998, 32 (1), 1-20.

[6] Gagne, C., Price, W., & Gravel, M. Comparing an ACO algorithm with other heuristics for the single machine scheduling problem with sequence-dependent setup times. *Journal of the Operational Research Society*, 2002, 53, 895-906.

[7] Gendreau, M., Laporte, G. & Guimaraes, E.M. A divide and merge heuristic for the multiprocessor scheduling problem with sequence dependent setup times. *European Journal of Operational Research*, 2001, 133, 183-189.

[8] Gupta, S.R. & Smith, J.S. Algorithms for single machine total tardiness scheduling with sequence dependent setups. *European Journal of Operational Research*, 2006, 175, 722-739.

[9] Lin, S.W. & Ying, K-C. A hybrid approach for single-machine tardiness problems with sequence-dependent setup times. *Journal of the Operational Research Society*, 2008, 59, 1109-1119.

[10] Nauss, R.M. Optimal sequencing in the presence of setup times for tow/barge traffic through a river lock. *European Journal of Operational Research*, 2008, 187 (3), 1268-1281.

[11] Smith, L.D., Sweeney, D.C., & Campbell J.F. A Simulation Model to evaluate Decision Rules for Lock Operations on the Upper Mississippi River. *HICSS 2007*.

[12] Smith, L.D., Sweeney, D.C., & Campbell, J.F. Simulation of alternative approaches to relieving congestion at locks in a river transportation system. *Journal of the Operational Research Society*, 2009, 60, 519-533.

[13] Smith, L.D., & Nauss, R.M. Investigating strategic alternatives for improving service in an inland waterway transportation system. *International Journal of Strategic Decision Sciences*, 2010, 1, 2, 62-81.