Research Notes for Chapter6*

As discussed in the research notes for Chapter 1, the development of stochastic sequencing models was slow until the late 1970s. Nonetheless, the basic stochastic counterpart models covered in this chapter were discovered earlier. Theorems 6.1 and 6.2 are due to Rothkopf (1966). Theorem 6.3 is due to Crabill and Maxwell (1969), who were also the first to note that minimizing the maximal expected tardiness is not identical to minimizing the expected maximal tardiness. Theorem 6.4 is due to Hodgson (1977). Corollary 4.1 was actually published earlier than the theorem itself, by Banerjee (1965). We are not aware of earlier sources of Theorems 6.5, 6.6, 6.7 and 6.8. Theorem 6.6 could be used in a branch and bound application for finding the sequence that minimizes the expected maximal penalty, but research to test whether it is indeed effective for that purpose is lacking. A basic linear-association model was proposed by Trietsch (2005), and, because it addresses projects, we discuss it in the research notes of Chapter 18. Essentially, the results on linear association given in this chapter, Chapter 11 and Appendix A were developed to avoid the ubiquitous stochastic independence assumption, and yet maintain tractability. Characterizing the cases for which traditional results based on stochastic independence can be extended to linearly-associated distributions is an open research problem. But Theorem 6.8 is one example where Theorem 6.7 suffices for this purpose. In Chapter 7 we present another example, Algorithm 7.1 (see also Theorem7.1), where Theorem 6.7 can be used to expand the conditions of a result from stochastic independence to linear association. Extending the analysis to more elaborate associations is another area that requires further research.

Sample-based analyses are central to our computations for stochastic models. They also help us avoid the stochastic independence assumption. Furthermore, in subsequent chapters, samples are vital to our calculations of safety time. It is therefore important to discuss the theoretical underpinnings of this approach. But first, we lay to rest a common myth, according to which the use of a sample—often associated with simulation—is in some sense less precise than the use of analytic models. From a practical point of view, the opposite is true. It is the typical analytic model that is more divorced from reality. Key to any analytic model is the use of given parameters. But where do these parameters actually come from? At best, such parameters are estimated from practical samples (which, under a sample-based approach, could have been used directly), and typically the models also involve fitting theoretical distributions to the data collected. When fitting distributions, goodness-of-fit tests apply, and if the tests are passed, we can say that the fit cannot be rejected. We can never say that the fit is correct, however. So there are two sources of potential error in this approach: estimating parameters and selecting a distribution. Again, if a real-life sample is given, our approach would not require estimating anything from it: we could often simply use it as is. As a result, we would avoid both the estimation errors

and the errors in fitting theoretical distributions. An additional advantage, even if we use simulated samples (which, admittedly, are not necessarily better representations of reality), is that we don't need strong assumptions, such as stochastic independence. Thus, we achieve more realistic representations, and we do not require restrictions designed mainly to achieve tractability in analytic models.

Analytic models have their value, especially in studying the behavior of stochastic systems in a stylized framework that allows seeing fundamental issues more clearly. Indeed, we utilize such models in this text, and we do so for that very purpose. But they should not always be touted as a better approach for practical problems than using sample data. There is a caveat, however. Using a larger sample implies longer computation times. As a rule of thumb, we can estimate the relative computation time as roughly equal to r times that of the deterministic counterpart, where r is the number of realizations in the sample. So, if we can estimate distributions well and then use them efficiently without relying on simulation, we may be able to solve larger problems than we could with a sample. For instance, if computation time is proportional to 2^n, we might expect to be able to solve for 10 fewer jobs if we use a sample of 1000 realizations.

We now turn to the historical development of sample-based analysis. For this purpose we assume the sample is actually simulated: when the sample is given from real data, the issues we now discuss become moot. Essentially, using a sample is a very intuitive concept and has probably been reinvented repeatedly, making it difficult to identify the first source of this idea. For instance, in Chapter 15 we discuss job shop simulations that date back to the 1960s, and they essentially involve the use of sample-based data. Early applications of computerized simulation, however, did not address optimization. Instead, they involved comparison of few alternatives (e.g., sequences) presented by the user. For this purpose, they relied on the statistical sampling approach combined with artificial generation of samples. In the early days, fast access memory was a severe bottleneck, and in many cases, the sample was regenerated for each new test. To make such comparisons more precise, it is possible and recommended to regenerate the exact same sample for each alternative solution by using the same seed for generating random numbers. Furthermore, it is possible to calculate the mean and the variance of a sample during the process of generating it without keeping the data in memory or storage. When memory is at a premium and storage is expensive, this facility is very attractive. The same approach can also be used in sample-based optimization; i.e., it is not necessary to actually store the sample. For instance, Gurkan et al. (1994) recommend this approach. Today, it is much easier to keep a sufficiently large sample in fast access memory, so the idea of reusing a sample repeatedly is technically more attractive. This is especially true if each stored number is the result of a simulation from a complex distribution. For example, in an early experiment, we used a very large sample with data generated by an Excel distribution function. In that case, it took 75 seconds to generate a sample and write it in a spreadsheet; but afterwards it took only 0.2 seconds to optimize the precise schedule for it. There were no sequencing decisions involved, or the optimization might have taken much longer, but this case demonstrates that accessing a sample is a very quick operation relative to generating it—that is, regenerating the same sample repeatedly would be highly wasteful.

Regardless of whether we store the sample or regenerate it, sample-based optimization starts with an assumption (implicit in the main body of our text) that processing times are *stationary*. Processing times are stationary if they do not depend on

jobs' start times; i.e., they are independent of the sequence or the schedule. The processing times of two jobs may be statistically correlated and yet stationary. For optimization, we utilize this stationarity to justify using the same sample under different schedules. By contrast, if processing times were a function of their start time (the schedule), or the sequence, this approach would require modification. For instance, in problems involving highway travel time, the stationarity assumption does not hold because travel time tends to be longer during rush hour. In such a case, sample-based analysis requires modification. Such modification may involve transformations of the distribution as described by Trietsch and Quiroga (2009) in a different context. We also use this approach for stochastic crashing in Chapter 18.

Two types of optimization are relevant. One is associated with optimizing nonlinear functions; e.g., precise scheduling decisions, where the decision variable is continuous and the function has a continuous derivative, but stochastic noise is also present, necessitating the clever use of samples. Shapiro and Homem-de-Mello (1998) discuss such a case with a multivariate normal distribution. For such cases, it makes sense to use small samples when we are far from the optimum and larger samples as we approach the final solution. Indeed, these authors recommend that approach. A related subject is to find the optimal setting of a complex process; e.g., in chemical engineering. In this case, the underlying function—say, yield—is not known theoretically but must be estimated empirically by trial and error. If so, the problem calls for a statistical experimental design approach, also known as *response surface optimization* (RS—see Box and Draper, 1987). RS can also be useful when few discrete decisions are involved; e.g., whether to use ingredient *X* or ingredient *Y*. It is not likely to be useful for sequencing decisions involving many jobs, however, because each possible sequence would become a special case requiring estimation. Related papers discuss the use of sample-based analysis in the context of integer programming—which conceptually can be used for sequencing decisions; e.g., see Kleywegt et al. (2001) and Verweij et al. (2003). Nonetheless, for the purpose of making sequencing decisions, there is less evidence that it is safe to start the search with a small sample and increase the sample size as we approach the final solution. It may be a useful heuristic to do so, but this subject requires further research. The state of the art in our context is to use neighborhood searches for the optimal sequence.

A related approach is pursued by Healy and Schruben (1991). They generate and store a sample and then optimize for each repetition separately and select the solution that is correct most frequently. This measurement—the frequency at which a sequence is optimal—is also known as the *optimality index*, a term coined by Dodin (1996) in a sequencing context. The use of optimality indices may look attractive, but Portougal and Trietsch (1998) cautioned that it is not a robust criterion. They demonstrated that maximizing the optimality index may favor the selection of schedules whose distributions have a low mode but high variance and high mean. Such distributions tend to be superior very often—that is why they have high optimality indices—but when they fail, the failure is worse than it could have been. The practical conclusion is that if we need to resort to heuristics, we are likely to be better off using the deterministic counterpart sequence as a basis for scheduling than using a sequence with a higher optimality index. This choice is guaranteed to be easier computationally and likely to be at least as good once we take stochastic variation into account.

Portougal and Trietsch (1998) is also the source of the result cited in Section 6.6 that if $Y \geq_{st} X$ and Y and X are independent, then $\Pr\{Y \geq X\} \geq 0.5$. (This result can be extended to the linearly-associated case by invoking Theorem 6.7.) Therefore, when comparing two stochastically-ordered distributions, the one that is stochastically smaller will have a higher optimality index. This result is highly intuitive but the proof is not immediate. On the subject of stochastic dominance, we should mention that several stronger forms of stochastic dominance are often mentioned in the research literature. Perhaps the most important one is the strongest possible dominance, where one variable dominates the other with probability 1 (*w.p.*1), or *almost surely*. If Y is larger than X almost surely we can write $Y \geq_{as} X$. Because dominance *w.p.*1 implies stochastic dominance, every result that can be proved for distributions with stochastic dominance also applies for stochastic dominance *w.p.*1. Suppose that two random variables Y and X are independent. If $Y \geq_{as} X$ then the cdf of Y must reach 1 before the cdf of X can exceed 0. For this reason, independent random variables with this strong dominance between them are also described as having *non-overlapping* distributions. But it is easy to show that this dominance does not require non-overlapping distributions when the variables are correlated (Ross 1996). For example, if Z is a nonnegative random variable and $Y = X + Z$, then $Y \geq_{as} X$ but the two can have overlapping distributions nonetheless. In fact, the assumption of independence is so ubiquitous that sometimes results that are stated for non-overlapping distributions could actually be proven for regular dominance *w.p.*1.

Emerging Research Areas

A useful research area, suggested by our discussion above, is cataloguing classical results that assume independence and classifying them according to whether they can or cannot be generalized to linearly-associated distributions. That includes, for example, distinguishing between results that require dominance with probability one and those that actually require non-overlapping distributions. In general, doing that is a task that requires studying in detail a very large number of publications. Whereas some cases would be almost immediate to analyze, others may require careful study. In Appendix A, among other things, we discuss a case where Theorem 6.7 cannot be used because it involves due dates that are not subject to the common bias. Hence, it is clear that not every result obtained under the independence assumption can be extended. It would be especially useful to devise general rules that can help in making such decisions.

We now discuss an important open area of future research that combines stochastic analysis with conventional mathematical programming techniques, such as branch and bound (B&B) and dynamic programming (DP). We address specifically the stochastic T- and T_w-problems, but part of the challenge is to identify additional models that might be addressed that way. As a rule, we can apply B&B, DP or Integer Programming to practically any stochastic problem by sample-based optimization. For instance, Gutjahr et al. (1999) apply B&B within a sample-based optimization framework for the T-problem. Similarly, some of the references discussed before involve sample-based optimization by various mathematical programming approaches. We also remark that stochastic programming with recourse typically utilizes a set of scenarios, which we might as well call a sample. Our aim here, however, is to show the applicability of the analytical approach for some distributions *without* a sample, and thus achieve more efficient computation. To

begin, we give a streamlined proof of a slight generalization of Theorem 6.8. The generalization allows agreeable weights and we address any two jobs. That is, we actually prove a generalized version of Theorem 2.8 (see Exercise 2.8*g*). Theorem 6.8 is essentially a corollary of our new result. (In the second edition we replaced the original Theorem 6.8 by Theorem RN6.1 because it is more general and yet has a simpler proof. The previous version of Theorem 6.8 did not include weights.)

Theorem RN6.1 In the T_w-problem, let the processing times, p_j, of all jobs be linearly associated, and let jobs 1 and 2 satisfy $p_1 \leq_{st} p_2$, $d_1 \leq d_2$ and $w_1 \geq w_2$, then job 1 precedes job 2 in an optimal sequence.

Proof.

»» Again, we prove for independent processing times and then invoke Theorem 6.7 to cover linear association. In Figure RN6.1, the expected tardiness of a job is depicted as a tail to the right of its due date, above the distribution that applies to it and below the upper horizontal line of 1. The relevant distributions are either F_k if job k is scheduled first (k = 1, 2), or F_{1+2} if job k is scheduled second. These three distributions also reflect any preceding jobs that have already been scheduled, or any jobs scheduled between jobs 1 and 2. As the figure shows, job 1 is stochastically smaller and has a lower due date, per the condition of the theorem. Let $T_{F,d}$ denote the area of the tail above distribution F (where F = 1, 2 or 1+2) to the right of due date d (where d = 1, 2). $T_{F,d}$ measures an expected tardiness; for instance, $T_{1+2,1}$ is the expected tardiness of job 1 if it is sequenced second and is thus subject to the completion time distribution F_{1+2}. We start with the sequence 1-2, assuming the two jobs are adjacent. By an API, the tardiness cost of job 1 increases by $w_1(T_{1+2,1} - T_{1,1}) \geq w_1(T_{1+2,2} - T_{1,2})$, whereas the tardiness cost of job 2 decreases by $w_2(T_{1+2,2} - T_{2,2}) \leq w_2(T_{1+2,2} - T_{1,2})$. But, because $w_2 \leq w_1$, $w_2(T_{1+2,2} - T_{1,2}) \leq w_1(T_{1+2,2} - T_{1,2})$, so the gain is bounded from above by a lower bound of the loss and the change cannot decrease and may increase the total weighted tardiness. Now allow additional jobs (which need not be stochastically ordered) between jobs 1 and 2. If we interchange the two jobs, all these intermediary jobs follow a stochastically larger job so their expected tardiness cannot decrease. Hence, such jobs cannot provide incentive to perform the interchange either. ««

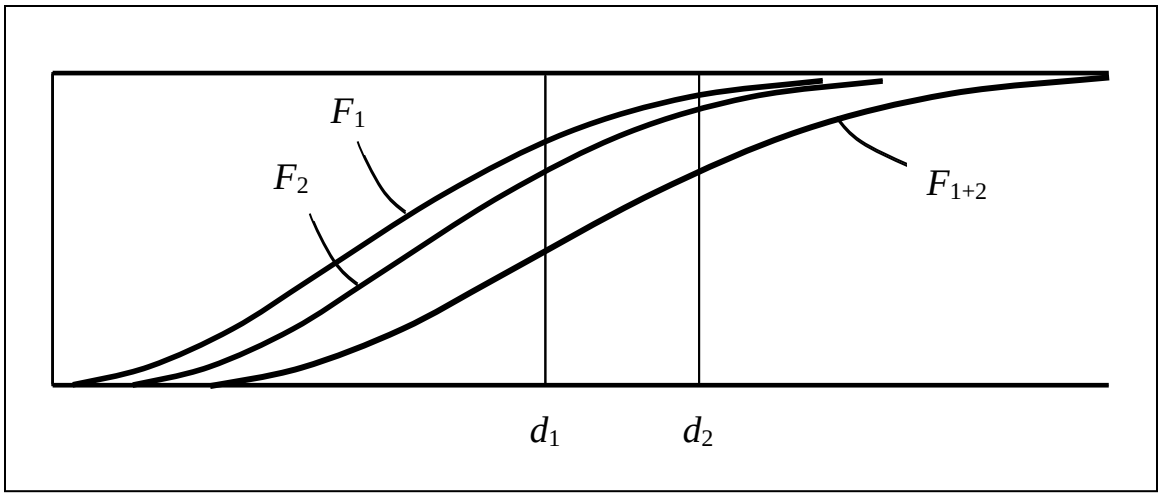

Figure RN6.1

Parenthetically, in Chapter 7, we introduce *predictive Gantt charts*. A predictive Gantt chart provides distributions for starting times and completion times and shows due date performance graphically. Essentially, Figures 6.2 and RN6.1 incorporate all the ingredients of a predictive Gantt chart. Our proof highlights the usefulness of predictive Gantt charts for stochastic analysis.

To continue, calculating expected tardiness by tail areas can sometimes be performed without simulation. For instance, when processing times are normal it is easy to obtain distributions such as F_{1+2} by convolution. In such cases, we can perform API tests by comparing the expected weighted cost of the two possible sequences. If job 1 comes first, the total weighted cost is given by: $w_1T_{1,1} + w_2T_{1+2,2}$ whereas if job 2 comes first, the total weighted cost is given by: $w_1T_{1+2,1} + w_2T_{2,2}$. Therefore, job 1 can come first if:

$$w_1T_{1,1} + w_2T_{1+2,2} \leq w_1T_{1+2,1} + w_2T_{2,2}$$

or

$$w_1(T_{1+2,1} - T_{1,1}) \geq w_2(T_{1+2,2} - T_{2,2})$$

This condition can take the place of the WMDD dispatching heuristic that we introduced in Chapter 4. Job 1 comes first if it satisfies the condition for any selection of another job as job 2. Stochastic dominance is not required. It can also be used within branches in a B&B application.

We invoked the normal distribution because it is easy to calculate convolutions for it and to calculate tail areas thereafter: by (B.14), $E(T_j) = \sigma[\phi(w) - w\Phi(-w)]$, where ϕ is the standard normal density function, $\Phi(w)$ is the standard normal cdf, $w = (d - \mu)/\sigma$, μ is the mean, and σ is the standard deviation of the processing time. The service level in this case is given by $\Phi(w)$.Although $\Phi(w)$ is elliptic, it is as good as analytic in the practical sense (because very precise calculations are available by appropriate series); e.g., the Excel function NORMSDIST(w) can be used in calculations. Therefore, it is possible to use the normal tail result for the purpose of solving tardiness problems by B&B or for DP.

Furthermore, it is equally easy to calculate tail areas for the lognormal distribution. Let μ and s be the mean of the lognormal distribution and the standard deviation of its core normal (see Appendix A for the relationship between these parameters). Define $z = \ln(d/\mu)/s + s/2$, which implies a service level of $\Phi(z)$. Then it can be shown that the expected tardiness is $\mu\Phi(s - z) - d\Phi(-z)$. The only problem with the lognormal, however, is that we don't have convenient convolutions for it. Nonetheless, there is one important special case for which we can use the lognormal distribution without resorting to approximations, and that is when processing times are linearly associated but with stochastic variation restricted to the common element, Q. Another important case that can be approximated very well is when each element is distributed lognormal with the same s, and Q is also lognormal. In that case, we can invoke the lognormal central limit theorem (see Appendix A) to obtain approximate convolutions. When all processing times are lognormal with the same s, then they are stochastically ordered.[†] In such case, we can also apply Theorem RN6.1. In Trietsch et al. (2010) we report that linearly-associated lognormal distributions provided a good fit for processing times in two project environments in Armenia. We believe that it should be useful in a much wider context as well. Thus, solving for this particular distribution has a validated practical application. (In the second edition, Appendices A and B include a much more extensive coverage of the lognormal. Appendix A and Chapter 18 also discuss extensive new evidence to the efficacy of this distribution, confirming the statement above.)

Returning to our tail-based analysis, one might think that such observations, at least with respect to the better-known normal model, would have led to the application of B&B or DP to the stochastic T-problem and the stochastic T_w-problem. Nonetheless, that does not seem to be the case (except by the sample-based approach that we cited above). Thus, we believe that the application of these tools to stochastic scheduling is a ripe area for research. (In Chapters 7 and 8 of the second edition we report some results along these lines.) Similarly, these stochastic problems can be addressed by various search heuristics with little adaptation. For example, the current best search technique for the T_w-problem seems to be *dynasearch* (Congram et al., 2002; Grosso et al., 2004). As discussed in our Research Notes for Chapter 4, Dynasearch is a neighborhood search approach that is based on searching various combinations of pair interchanges and has been shown to be much more effective than regular pair-interchange search heuristics that consume the same search time. On the one hand, in the deterministic T_w-problem case, the fastest application of dynasearch utilizes shortcuts that go beyond just using the analogue of Theorem RN6.1. On the other hand, it is conceptually easy to adapt *basic* dynasearch—without shortcuts—to the stochastic version. Specifically, the effects of independent PIs are additive, which, as we discussed in RN4, is the main requirement for dynasearch to be potentially effective. (By this criterion, dynasearch could also be applied to a sample.) We may also be able to identify useful shortcuts that apply in the stochastic case (including Theorem RN6.1). Thus, the application of dynasearch and other search techniques to our problem is highly likely to bear fruit. Nonetheless, it is also important to establish the size limit of problems that can be solved to optimality (e.g., by B&B or DP), so we should not limit our attention to search heuristics.

[†] In our Research Notes for Appendix A we show that lognormals with the same s are stochastically ordered in the likelihood ratio sense, which is a slightly stronger result.

Complexity of the Minimax Problem

New coverage in the second edition relates to the complexity status of the flowtime problem under minimax cost or minimax regret, both with exhaustive scenarios. We show that they are NP-hard. This result has already been claimed for minimax regret by Daniels and Kouvelis (1995), who formulated instances with two reversed (and therefore, related) scenarios as Generalized Assignment Problem (GAP) models. They then quoted secondary sources that GAP is NP-complete, thus concluding their proof. We could not find the original proof that GAP is NP-complete, and the reason we looked for it in the first place was that we wanted to verify that NP-completeness still applies in spite of the relationship they imposed between the two scenarios. Be that as it may, however, in addition to their own—arguably incomplete—proof, Daniels and Kouvelis (1995) also referenced a working paper by Yu for an alternative proof. This citation eventually led us to Yang and Yu (2002), who provided a proof not only for minimax regret but also for minimax cost. Hence, the complexity status of the models in question is resolved. Nonetheless, we now discuss new analysis that we developed before finding Yang and Yu (2002): We restrict a problem well-known to be NP-complete, namely Partition, to the special case presented by Daniels and Kouvelis, thus completing their original proof (and, as a by-product, providing a new proof that GAP is NP-complete). We do so because it might provide some additional insight to minimax criteria and because it serves as an introduction to a case we cover in our Research Notes for Chapter 7.

The key to the proof is letting two scenarios be reversed versions of each other. For an arbitrary first scenario, with given processing times p_j, the processing time of the kth job under the second scenario is p_{n-k+1}; that is, the processing time of the $(n - k + 1)$th job under the first scenario. (If this special case is NP-hard, then the general case is also NP-hard.) Next, they formulate minimax regret as a GAP model where the flowtime of scenario 1 is to be minimized subject to the constraint that the difference between the two flowtimes should not be negative. To see why that is a valid formulation, we first show that the flowtime problem can be cast as an assignment problem. Next, we show why the constraint is valid. Then we give an illustrative example whose optimal solution is obtained by solving a Partition instance. Finally, we show one way to restrict the problem to any given Partition instance. That restriction not only completes the published proof but also proves—again—that GAP is NP-complete.

To see that the flowtime problem is identical to the assignment problem, let x_{jk} be a {0, 1} variable such that $x_{jk} = 1$ if and only if job j is in position k (that is, $[k] = j$), and $x_{ik} = 0$ otherwise (for $i, k = 1, 2, \ldots, n$). Now consider the contribution of job $[k]$ to the objective function: it is given by $(n - k + 1)p_{[k]}$, and therefore it also equals

$$\sum_{j=1}^{n} (n - k + 1) x_{jk} p_j$$

where only the contribution of job $[k]$ is counted (because $x_{ik} = 0$ for all other jobs). Therefore, the flowtime model can be formulated as the following Assignment Problem (AP):

$$\min\left\{\sum_{k=1}^{n}\sum_{j=1}^{n}(n-k+1)x_{jk}p_j\right\}$$

subject to:

$$\sum_{j=1}^{n} x_{jk} = 1$$

$$\sum_{k=1}^{n} x_{jk} = 1$$

$$x_{jk} \in \{0,1\}$$

which is a classic AP. (It is the ability to assign a fixed cost for each cell that renders the flowtime model an instance of AP.) To obtain the GAP introduced by Daniels and Kouvelis, we add the following side constraint

$$\sum_{k=1}^{n}\sum_{j=1}^{n}(n-k+1)x_{jk}(p_j - p_{n-j+1}) \geq 0$$

In words, the flowtime of scenario 2 (where the processing time of job j is p_{n-j+1}) should not exceed the flowtime of scenario 1.

In essence, we assume regret will be maximized for scenario 1, and that is why we choose to minimize its flowtime. By symmetry, we can make this assumption without loss of generality. With that in mind, the constraint is justified because if we allow a negative right-hand side, the makespan under scenario 2 can grow excessively and the maximal regret will shift from scenario 1 to scenario 2 (and be excessive). In more detail, for any sequence, S, consider the sum of the flowtimes of the two scenarios. In this sum, the processing time of job $[k]$ is counted $(n-k+1)+k=(n+1)$ times, hence the sum is given by $(n+1)\sum p_j$, which is a constant. We refer to half of that constant as F_0. Denote the flowtime of scenario 1 under SPT as F_{min}. Suppose the flowtime of scenario 1 is $F_0 + \Delta$, where, by the side constraint, $\Delta \geq 0$, then the flowtime of scenario 2 is $F_0 - \Delta$. Under scenario 1, the regret is $F_0 + \Delta - F_{min}$ whereas under scenario 2 it is $F_0 - \Delta - F_{min}$, so the maximal regret is associated with scenario 1 and is higher than the minimal regret by 2Δ; by symmetry, exactly the same values apply under scenario 2. So the best we can do is to reduce regret to $F_0 - F_{min}$, and that is achieved by minimizing the flowtime of scenario 1 under the listed constraint.

Now consider minimax criteria: for the optimal minimax regret solution it is precisely $F_0 + \Delta$, and it, too, cannot fall below F_0. Hence, the same optimal solution applies to minimax cost as well. The only difference between the minimax cost objective and the minimax regret objective is that we don't need to subtract F_{min}, but F_{min} is effectively a constant.

To illustrate the relationship between our problem and Partition, we now ask whether we can achieve a flowtime of F_0; that is, can we drive Δ down all the way to zero. That's a decision problem with a **YES** or a **NO** answer, and we show a Partition instance with the same answer associated with it. In SPT order, let scenario 1 have processing times of 1, 2, 3, 8, 10, 14 (and hence scenario 2 has 14, 10, 8, 3, 2, 1). Furthermore, assume the scenarios are equi-probable; that is, each can manifest with probability 50%. This assumption allows us to calculate the expected processing times, as follows: (1 + 14) / 2 = 7.5, (10 + 2) / 2 = 6, (8 + 3) / 2 = 5.5, (and by symmetry) 5.5, 6, and 7.5. For reversed and equi-probable scenarios each value except perhaps one, associated with the median job, appears at least twice, and in our particular example it appears exactly twice. Now consider the SEPT sequence. There are 8 distinct SEPT sequences starting with 3 or 4, then 4 or 3 (respectively), 2 or 5 followed by 5 or 2 (similarly), and then 1 or 6 followed by 6 or 1. (In general, for even n, assuming all $n/2$ expected values are distinct, there will be $2^{n/2}$ possible SEPT sequences. For odd n there will be $2^{(n-1)/2}$ distinct SEPT sequences.) The expected flowtime for each of these eight candidates is 125, which is also F_0. Now, consider the particular SEPT sequence for which all pairs with equal expected processing times are in increasing order; that is, pick the sequence 3-4-2-5-1-6. Under scenario 1 this sequence has a flow time of 112 and under scenario 2, 138. When compared to E(F), 125, we find that the flowtime under scenario 1 falls below 125 by 13 whereas scenario 2 exceeds 125 by the same amount, yielding 125 + 13 = 138. Accordingly, in this case minimax cost and minimax regret are determined by scenario 2. Now reverse all the relevant pairs, yielding the sequence 4-3-5-2-6-1. It has a flowtime of 138 for scenario 1 and 112 for scenario 2 (which we could anticipate by symmetry). Switching jobs 3 and 4 adds 8 – 3 = 5 to the flowtime of scenario 1, switching 2 and 5 adds 10 – 2 = 8, and switching 1 and 6 adds 14 – 1 = 13. By switching all three, we increase the flowtime of scenario 1 by 5 + 8 + 13 = 26, thus going from 112 to 138, and the effect on scenario 2 is the opposite. Now note that 5 + 8 = 13 and consider one of the two sequences 3-4-2-5-6-1 or 4-3-5-2-1-6: both have the same flowtime under both scenarios, matching SEPT at 125. We now know that either one of these two sequences is optimal. But to answer the decision problem whether such a solution exists we had to partition the set {5, 8, 13} to two subsets with the same total weight (13), which is an instance of the Partition problem. Hence, finding the solution is at least as hard as solving Partition. That illustrates that the problem is NP-hard. Finally, for a more formal proof, we now show one way to restrict our GAP problem with the special constraint structure to any instance of Partition. Because GAP is clearly in NP, this restriction will also prove that GAP is NP-complete.

Given an instance of Partition, with n elements, start by indexing them in nondecreasing order. Let u_k denote the size of the kth element so u_n is the largest element in the set. Arbitrarily, let $A = u_n$. Now create scenario 1, with $2n$ processing times, constructed from both ends to the middle, as follows: $p_1 = (A + 2 + u_1)/2$, $p_{2n} = (A + 2 - u_1)/2$, $p_2 = (A + 4 + u_2)/2$, $p_{2n-1} = (A + 4 - u_2)/2$, …, $p_k = (A + 2k + u_k)/2$, $p_{2n-k+1} = (A + 2k - u_k)/2$, …, $p_n = (A + 2n + u_n)/2$, $p_{n+1} = (A + 2n - u_n)/2$. Claim: the original Partition instance yields a **YES** if and only if the GAP achieves F_0, and **NO** otherwise. To wit, if we construct scenario 2 as the reverse of scenario 1 and assume the two sequences are equi-probable, then one of the 2^n SEPT sequences is given by n consecutive pairs of jobs, sequenced per the order at which we populated the list, (1, n, 2, $n - 1$, …, k, $n - k + 1$, …, n, $n + 1$), such that the sum of indices of each pair is $n + 1$, where the first two jobs have expected

processing times of $A + 2$; the kth pair, $A + 2k$; and the last pair, $A + 2n$. That is, it is a SEPT sequence because the means are monotone nondecreasing. The sequence will remain SEPT if we interchange the jobs of a pair (whose indices sum to $n + 1$), but not if we interchange any other two jobs (whose indices do not sum to $n + 1$); that is, the expected flowtime does not exceed F_0 if and only if we avoid any interchange between jobs that are not paired. In the contribution to the GAP objective function of each such pair, the first job is counted once more than the second, and therefore the makespan of the sequence listed above under scenario 1 is $F_0 + \sum u_k$. By reversing the sequence of pair k, we reverse the sign of u_k in the makespan expression, and if we can find a subset of pairs to reverse such that it leads to F_0, then the answer to the original Partition instance must be **YES**, whereas otherwise it is **NO**. Hence the restriction to Partition is valid and GAP must be NP-complete even with a single side constraint with the special structure considered.

Incidentally, it is straightforward to extend the symmetric formulation of the GAP model presented by Daniels and Kouvelis to *any* two sequences, although in such case we will not know in advance whether the constraint should be ≥ 0 or ≤ 0. However, we can solve both versions and select the better, and thus optimal, solution. Hence, on the one hand, there is no real need to restrict the formulation to the symmetric case. On the other hand, it is not enough to show that the problem can be formulated as a GAP instance. To prove NP-hardness we must do the opposite, namely show that the problem can be reduced to a known NP-complete problem, such as Partition or GAP. Above we showed both directions (with respect to the symmetric case).

References

Box, G.E.P. and N.R. Draper (1987) *Empirical Model-Building and Response Surfaces*, Wiley.

Banerjee, B.P. (1965) "Single Facility Sequencing with Random Execution Times," *Operations Research* 13, 358-364.

Congram, R.K., C.N. Potts and S.L. van de Velde (2002) "An Iterated Dynasearch Algorithm for the Single-Machine Total Weighted Tardiness Scheduling Problem," *INFORMS Journal on Computing* 14, 52-57.

Crabill, T.B. and W.L. Maxwell (1969) "Single Machine Sequencing with Random Processing Times and Random Due-Dates," *Naval Research Logistics Quarterly* 16, 549-555.

Daniels, R.L., and P. Kouvelis (1995) "Robust Scheduling to Hedge Against Processing Time Uncertainty in Single-Stage Production," *Management Science* 41, 363–376.

Dodin, B. (1996) "Determining the Optimal Sequences and the Distributional Properties of their Completion Times in Stochastic Flow Shops," *Computers and Operations Research* 23, 829-843.

Grosso, A., F. Della Croce, R. Tadei (2004) "An enhanced dynasearch neighborhood for the single-machine total weighted tardiness scheduling problem," *Operations Research Letters* 32, 68-72.

Gurkan, G., A.Y. Ozge and S. Robinson (1994), "Sample-path Optimization in Simulation," *Proceedings of the 1994 Winter Simulation Conference*, 247-254.

Gutjahr, W.J, A. Hellmayr and G.Ch. Pflug (1999) "Optimal Stochastic Single-Machine-Tardiness Scheduling by Stochastic Branch-and-Bound," *European Journal of Operational Research* 117, 396-413.

Healy, K. and L.W. Schruben (1991) "Retrospective Simulation Response Optimization," *Proceedings of the 1991 Winter Simulation Conference* 901-906.

Hodgson, T.J. (1977) "A Note on Single Machine Sequencing with Random Processing Times," *Management Science* 23, 1144-1146.

Kleywegt, A.J., A. Shapiro and T. Homem-De-Mello (2001) "The Sample Average Approximation Method for Stochastic Discrete Optimization," *SIAM Journal of Optimization* 12, 479-502.

Ross, S.M. (1996) *Stochastic Processes*, 2nd Ed., Wiley.

Rothkopf, M.H. (1966) "Scheduling with Random Service Times," *Management Science* 12, 707-713.

Portougal, V. and Trietsch, D. (1998) "Makespan-Related Criteria for Comparing Schedules in Stochastic Environments," *Journal of the Operational Research Society* 49, 1188−95.

Shapiro, A. and T. Homem-de-Mello (1998) "A Simulation-Based Approach to Two-Stage Stochastic Programming with Recourse," *Mathematical Programming* 81, 301-325.

Trietsch, D. and F. Quiroga (2009) "Balancing Stochastic Resource Criticalities Hierarchically for Optimal Economic Performance and Growth," *Quality Technology and Quantitative Management* 6.

Trietsch, D., L. Mazmanyan, L. Gevorgyan and K.R. Baker (2010), A New Stochastic Engine for PERT (working paper). URL: <http://faculty.tuck.dartmouth.edu/images/uploads/faculty/principles-sequencing-scheduling/Engine.pdf>

Verweij, B., S. Ahmed, A.J. Kleywegt, G. Nemhauser and A. Shapiro (2003) "The Sample Average Approximation Method Applied to Stochastic Routing Problems: A Computational Study," *Computational Optimization and Applications* 24, 289-333.

Yang, J. and G. Yu (2002) “On the Robust Single Machine Scheduling Problem,” *Journal of Combinatorial Optimization* 6, 17-33.