Research Notes for Chapter 8[*]

We start by discussing some sources and minor comments for the models presented in the chapter. We also mention some modeling issues related to the option of preempting a job at the cost of a repeated setup. Next, we focus on the traveling salesperson problem (TSP). We discuss the state of the art for deterministic instances and then explore some stochastic models. We also show by example how to solve a safe version of the shortest route problem.

Sources and comments

The earliest source that introduced release dates to a sequencing problem is Jackson (1955), who considered it in the context of the single-machine $T_{\max}$-problem. In the dynamic case, as in the static case, any sequence that minimizes $L_{\max}$ also minimizes $T_{\max}$ (although the converse is not true, again as in the static case). In the chapter we mentioned that the dynamic $L_{\max}$-problem is equivalent to the HBT problem. We also mentioned that the HBT problem can be solved effectively by branch and bound. Results along these lines were published by McMahon and Florian (1975), Lageweg, Lenstra and Rinnooy Kan (1976), and Carlier (1982). The latter involves a clever branching scheme and can solve instances with hundreds of jobs rapidly. That algorithm serves as the basis of sophisticated job shop algorithms that we discuss further in Chapter 14. Potts (1980) addressed the dynamic $L_{\max}$-problem, and studied the effectiveness of exploiting symmetry, as mentioned in the chapter with respect to minimizing the makespan, by solving the reversed problem (where the tail comes first and the head follows the body). He also considered other enhancements of the algorithm. In general, whenever the preempt-resume version of a problem can be solved readily, the branch and bound approach should be considered seriously for the preempt-repeat version. Indeed, the branch and bound algorithms mentioned above utilize this insight. The advantage is that the preempt-resume optimal solution serves as a lower bound. If this bound is tight, branch and bound is effective. In the same spirit, the F-problem can be solved by exploiting the preempt-resume solution (Ahmadi and Bagchi, 1990). The dynamic U-problem and the dynamic T-problem represent another level of difficulty, however, because the corresponding preempt-resume solution is not obvious.

One research question about preemption that did not receive much attention in the literature is a hybrid version of the preempt-repeat and preempt-resume models. In practice it is often possible to preempt a job and then resume it after repeating a setup. That is, we obtain a preempt-repeat setup-resume processing model. Related to this issue is the JIT technique of setup reduction known as SMED (single minute exchange of die),

[*] The Research Notes series (copyright © 2009 by Kenneth R. Baker and Dan Trietsch) accompanies our textbook *Principles of Sequencing and Scheduling,* Wiley (2009). The main purposes of the Research Notes series are to provide historical details about the development of sequencing and scheduling theory, expand the book's coverage for advanced readers, provide links to other relevant research, and identify important challenges and emerging research areas. Our coverage may be updated on an ongoing basis. We invite comments and corrections.

Citation details: Baker, K.R. and D. Trietsch (2009) Research Notes for Chapter 8 in *Principles of Sequencing and Scheduling* (Wiley, 2009). URL: http://mba.tuck.dartmouth.edu/pss/.

developed by Shigeo Shingo (Shingo, 1985). Shingo focused on repetitive manufacturing applications. In job shops, however, it is sometimes possible to set work on modular jigs that facilitate mounting them or dismounting them (Trietsch, 1992). Similarly, some modern machine tools have multiple tables that allow switching jobs almost instantaneously without actually dismounting the current job during its preemption. If so, the setup time is reduced considerably and it becomes possible to use preemption in practice. Engineering solutions aside, if the only cost is time, inserted idle time should never exceed the setup time. Then, the question is whether the benefit of preempting exceeds the time-cost of that setup. In that connection, inserted idle time is especially unattractive in stochastic environments because it implies waiting for a stochastic arrival while work is available with certainty. If the realistic cost of preemption is just lost setup time, starting work on a waiting job is a cost-free option. At most we lose setup time with some probability, but the alternative is to waste that time or more with certainty. The usefulness of quick setups is even greater in such environments.

As mentioned in the chapter, we might expect that the nondelay adaptation of SPT performs quite well for the dynamic *F*-problem even when the hypothesis of Theorem 8.4 does not hold. For example, in a limited computational study, Chandra (1979) reported that the nondelay version of SPT generates solutions that are about 3% worse than optimal for 10-job problems and less than 1% worse than optimal for 30-job problems. The observation that larger problems seem to yield better results suggests that the heuristic may be asymptotically optimal, but we don't know of any published results proving or disproving this hypothesis. For the dynamic *T*-problem, we might expect the nondelay adaptation of MDD to perform quite well, even when the hypothesis of Theorem 8.5 does not hold. If we are interested in obtaining optimal solutions, a branch and bound approach is appropriate, although the computational effort might be greater than for the static version. Results along these lines are presented by Chu and Portmann (1992). Chu (1992) also uses branch and bound for the dynamic *T*-problem and harnesses an MDD-type property in the solution algorithm. Kanet and Li (2004) reported good results for the WMDD nondelay heuristic (using the exponential distribution to simulate r_j and p_j, the TWK method to generate d_j, and several approaches to assign w_j).

To conclude our comments about dynamic models, we also mentioned in the chapter that the dynamic *U*-problem is NP-hard but Algorithm 8.2 can solve it efficiently in the case of agreeable release dates and due dates. The algorithm is due to Kise, Ibaraki and Mine (1978). One special case occurs when all jobs have the same due date. In that case, Algorithm 2.1 (the Moore-Hodgson algorithm) can be used to solve the problem if we just reverse the time line. That is, we treat the common due date as time zero, the gaps between job release dates and the due date become job due dates, and the sequence suggested by the Moore-Hodgson algorithm is reversed before implementation.[*]

In the chapter we discussed in detail how to solve the *F*-problem with series-parallel structures. The same approach can be used directly for the F_w-problem. Essentially, the series-parallel structure in combination with the decomposition tree imposes a clear partial order on individual jobs. That order can prevent scheduling the next job in the SWPT sequence, but all other sequencing decisions follow SWPT. That is, we only need to violate SWPT when forced to do so by a serial relationship. Therefore, given a series-parallel structure, non-SWPT selections are always postponed as much as possible. When

[*] This solution is an unpublished result due to Tiru Arthanari.

we consider more general network structures, it is no longer guaranteed that such decisions should be postponed as much as possible. The simplest network that is not series parallel is the *interdictive graph*, depicted in Figure 16.16. In that figure, which is presented in a project scheduling context, we depict jobs (known as activities in the project case) on the arrows, but it is straightforward to transform it into the structure used in the current chapter, where jobs are presented by nodes. Given such a non-series-parallel network, there are potential choices early on that cannot be fully assessed until later. For that reason, the series-parallel structure is the most general case of precedence structure known to yield an efficient algorithm of the type described above. For general precedence structures, we would apparently need to employ a general-purpose solution procedure. For example, we could use dynamic programming, but with precedence constraints in the role of dominance conditions. Monma (1981) describes the limitations of sequencing with general precedence restrictions, whereas Morton and Dharan (1978) explore some heuristic algorithms for this type of problem.

The study of networks with series-parallel structure versus the more general case is also important in stochastic analysis of project networks. In that context, the most prevalent objective is to minimize the makespan. Therefore, we are often interested in the cdf of the makespan. As it turns out, if activity distribution times are stochastically independent and all activities start as soon as possible, then series-parallel networks can be *reduced* in the sense that it is possible to obtain the distribution of the project duration (or of any other milestone) by performing a series of basic convolutions or maximum operations. The decomposition tree then dictates the sequence of those basic operations. By contrast, stochastic analysis of networks that are not series-parallel is quite forbidding; for that reason, the interdictive graph is also known as the *forbidden graph*. Demeulemeester and Herroelen (2002) discuss the issue thoroughly. The practical resolution, however, is by simulation. Simulation also allows removal of the impractical independence assumption. If correlation is modeled by linear association (see Appendix A), then we can still solve for series-parallel networks because it is enough to consider the common element after finding the distribution for independent elements. That result follows the same arguments that prove Theorem A.4.

More on the traveling salesperson problem

The traveling salesperson problem (TSP) is NP-hard in the strict sense, but instances with thousands of cities have been solved since the mid-1980s (Laporte, 1992). Furthermore, some special cases can be solved in polynomial time. One such algorithm has been developed by Gilmore and Gomory (1964). For details of the use of the TSP in scheduling, including a summary of the Gilmore and Gomory algorithm and a way to run it in $O(n \log n)$, see Bagchi et al. (2006). One flow shop model that utilizes the Gilmore and Gomory algorithm is the two-machine blocking case, which we discuss in Chapter 10 (pp. 244-245). Returning to the general TSP, in our own ongoing research (Mazmanyan et al., 2009, available on our web site), some instances of 100-city TSP were solved in ten to twenty seconds (on a relatively slow PC with a 1.7MHz single processor) but one instance took 14 hours! That's a differential of about 360,000%, and it is due to the fact that the difficult instance involved a search tree with numerous near-optimal tours. Whereas for the purpose of worst case analysis the quicker instances prove nothing, it is still important to note that excessive branching due to preponderance of near-optimal solutions indicates that typical problems can be solved very well in practice if we are willing to settle for a good certificate rather than demand perfection. We return to this point later (as part of our stochastic analysis), but here we should mention that we used the state-of-the-art branch-and-cut, symmetric TSP software, Concorde (Applegate et. al, URL). Concorde has been associated with several breakthrough milestones; e.g., a 24,978-city instance was solved in 2004. On the one hand, such large instances may require years or even decades of CPU time (using massive parallel processing to compress the actual duration). On the other hand, most practical applications—especially those associated with actual travel applications—are smaller by many orders of magnitude, so in a practical sense, and especially if we are willing to accept nearly-optimal tours with good certificates, we can say that the deterministic TSP is tractable.

Technically, Concorde assumes that the TSP is symmetric. In the chapter, however, we do not assume symmetry because important transportation and production applications are asymmetric; e.g., due to one-way streets. Nonetheless, any n-city asymmetric TSP instance can be readily transformed to an equivalent $2n$-city TSP (Jonker and Volgenant, 1983), so the distinction is not crucial. The transformation starts by splitting every city into two—one for inbound flows and the other for outbound flows—and then introducing asymmetric costs. Because the model must be symmetric, the "inbound city" and the "outbound city" have two connections between them, both with the same cost (zero). Therefore, it is important to force the algorithm to use precisely n of these $2n$ new connections, all in the same direction; e.g., from "inbound" to "outbound." Jonkers and Volgenant use large penalty costs to ensure compliance. These penalties apply to direct connections between any two "inbound" cities or between any two "outbound" cities, thus only allowing travel from an "outbound" city to an "inbound" one or vice versa. The use of such penalty costs is somewhat detrimental in terms of the precision that can be achieved (as a rule, mixing small and large numbers in computerized applications should be avoided if possible). In the specific case of a branch-and-cut application, however, it is conceptually easy to introduce linear constraints that enforce the correct direction. The new constraints then become part of the mathematical model. It is necessary, however, to do so within the program, whereas the generic formulation is within the purview of the user.[*]

In general, the literature on stochastic versions of the TSP is sparse, perhaps because the stochastic counterpart is equivalent to the deterministic TSP. One challenging stochastic adaptation of the TSP involves optimizing a predetermined sequence to visit a random subset of cities (Jaillet, 1988). The model exploits shortcuts to bypass cities that do not require service but without re-optimizing the sequence. As such, it concerns a dynamic problem but restricts the solution to a static sequence. Jaillet restricts the probability distribution by which customers are selected for each tour as follows: given that K customers have to be visited, any subset of K customers is equally likely. For instance, when every customer is selected with the same probability, this restriction is satisfied. He then obtains some bounds on the ratio by which the regular optimal TSP can be suboptimal, and shows by example that the regular TSP can be a very poor choice. The model is challenging, especially if we remove the restriction. See also Laporte et al. (1994), who solved the problem for up to 50 potential customers. On the one hand, given

[*] We are grateful to Bill Cook for insights with respect to those options.

the current state of the art in solving the TSP, unless it is important to publish the sequence in advance, the best practical solution of the problem may well be to obtain the list of visited customers first, and then solve (optimally or heuristically). On the other hand, there are instances where pre-publication of the sequence is desirable and yet the option to use shortcuts is viable. It may also happen that the information about whether the next customer on the pre-established tour requires a visit is provided in real time, and thus the problem is still interesting.

Whereas the pure TSP counterpart model is essentially identical to the deterministic TSP, the same is not true for safe scheduling. Furthermore, stochastic models of the related vehicle routing problem (VRP) are more akin to the safe scheduling approach, although they consider other sources of variation. The VRP problem contains n customers serviced by m ($< n$) vehicles. Each customer must be assigned to a unique vehicle, and we must find the optimal tour for each vehicle. The objective may be to minimize the maximal tour length or to minimize total travel distance subject to a load ceiling for each vehicle. Whereas it is easier to solve several smaller tours as the second stage requires, the need to find the optimal allocation of customers to vehicles (partitioning) renders this problem more difficult to solve than the TSP. Conceptually, if we treat each vehicle as a machine, the relationship of the VRP to TSP is like that of the parallel machine model to the single machine model (see Chapter 9). The analogy holds also in the sense that the parallel machine model is much more complex than the single-machine model.

The stochastic vehicle routing problem (SVRP) has received some attention in the literature, and indeed some of the early applications of safe scheduling principles are associated with it. An example is Dror and Trudeau (1986). For a survey, see Gendreau et al. (1996) or Bertsimas & Simchi-Levi (1996). Most of these models assume stochastic demands on vehicle capacities that are revealed only when the vehicle arrives at each customer. This feature gives rise to an overload risk, associated with expensive corrective actions. The objective is to minimize the expected total cost, including the cost of scheduled operations and the cost of corrective actions. With few exceptions, travel-time randomness is excluded from these models, perhaps due to the conventional focus on the mean. One such exception concerns a dynamic model that relies on real-time information for minimizing mean travel time (Gao and Chabini, 2006). However, they address only the shortest route problem (whose deterministic counterpart is not really a scheduling problem). Gao and Chabini aim to minimize the mean travel time, so theirs is a real-time counterpart model. In the text and in these Research Notes, our own models are not based on real-time data; instead, they are solved *a priori*. But *a priori* models can be used to support dispatching decisions, which is a useful real-time heuristic in general (Bertsimas et al., 1990). Here we start with the safe scheduling version of the shortest route problem. This model is conceptually similar to the safe scheduling TSP that we introduced in the chapter. However, unlike the TSP, the static shortest route problem is very easy to solve in polynomial time.[*]

[*] This is achieved by a well-known forward dynamic programming approach developed in the late 1950's by Dijkstra. In the first stage, we find the nearest city to the source by comparing all direct distances. In the second stage we find the second nearest city to the source by comparing all the previously remaining options plus the new options available through the nearest city. In stage ($j - 1$) we find the jth nearest city, and so on. The process can stop as soon as the destination is identified as the kth nearest city at stage k.

In what follows, our statements about routes can also be recast in terms of tours. The essential difference between TSP tours and routes is that a tour must visit a predetermined set of cities and a route does not. We may also use *path* to refer either to a TSP tour or to a route. For our purpose, only one important difference between tours and routes exists: to solve for a TSP tour, it may be necessary to first populate the distance matrix by appropriate shortest routes. Setting that distinction aside for a while, our task is to select a path and set a due date for it. This implies two options, associated with the two main safe scheduling objectives that we presented in Chapter 7. Model 1 requires minimizing d_1 subject to $SL \geq b$, and model 2 requires minimizing $d_2 + \gamma\mathrm{E}(T)$, both by selecting the best path and then optimizing d_j accordingly. Given a path we can compute its length distribution, and then it is straightforward to set d_1: if the cumulative distribution function (cdf) of the path length is $F(t)$, then $SL(d) = F(d)$, so the optimal d_1 is $F^{-1}(b)$—the argument for which $F(d_1) = b$. In our current models, the length distribution is normal with mean μ and standard deviation σ.[*] Let $z_j = (d_j - \mu)/\sigma$, let $\varphi(z)$ denote the standard normal density function, and let $\Phi(z)$ denote the standard normal cdf. Using an asterisk to denote optimality, $z_1^* = \Phi^{-1}(b)$ and $d_1^* = \min\{\mu + z_1^*\sigma\}$ among all paths. For model 2, If $\gamma \leq 1$ then $d_2^* = 0$ and the deterministic counterpart path is optimal; so henceforth, we assume $\gamma > 1$. For any given makespan distribution, the problem can be solved by applying the critical fractile model, which implies a service level target of $b = (\gamma - 1)/\gamma$; i.e., $z_2^* = \Phi^{-1}[(\gamma - 1)/\gamma]$. For a normal distribution with mean μ and standard deviation σ, the objective function value for $d_2 = \mu + z_2^*\sigma$ in this generalized E/T model equals $\mu + \gamma\varphi(z_2^*)\sigma$. So both models can be written as minimizing a linear function of μ and σ, $\mu + \pi\sigma$, where either $\pi = z_1^*$ or $\pi = \gamma\varphi(z_2^*)$. Although $\pi < 0$ is possible in model 1, here we restrict our attention to nonnegative π. We address π as the *price* of each standard deviation unit relative to μ. Our two models are similar, but not equivalent: when the same service level is required, the prices are different so the optimal paths may not coincide and $d_1^* \leq d_2^*$. When the prices are equal, the same path is optimal but $d_1^* > d_2^*$.

Let the travel time between cities i and j—our distance metric—have mean μ_{ij} and standard deviation σ_{ij}. We use no index or a single index for paths; we also take the liberty of allowing real numbers as path indices and using different indices for the same paths as convenient. If we start at time 0, the path completion time is a random variable denoted by C. Given a path, we can compute the distribution of C, and in our case it is normal with mean μ and standard deviation σ. Define a *virtual path* as any point in the positive quadrant of the μ-σ^2 plane—that is, the Euclidean plane where the horizontal axis corresponds to μ and the vertical axis to σ^2. A path is called *efficient in the* μ-σ^2 *plane* if it minimizes the linear function $a\mu + b\sigma^2$ for some parameters a and b such that $a, b \geq 0$ and $a + b > 0$. Alternatively, we could use the μ-σ plane for our depiction of tours, and an efficient tour in that depiction minimizes $a\mu + b\sigma$ for similar parameters a and b. It can be shown that paths that are efficient in the μ-σ plane are also efficient in the μ-σ^2 plane but the converse is not necessarily true. Henceforth, as a default, we call a path *efficient* if it is efficient in the μ-σ^2 plane. Efficient paths are also Pareto optimal, but other Pareto

---

Otherwise, by the time we reach stage $(n - 1)$, we obtain a tree of shortest paths from the origin to every other city. The algorithm is polynomial in n, even though there are exponentially many potential paths.

[*] It is conceptually possible to replace the normal distribution here by the lognormal, and rely on the lognormal central limit theorem for approximate convolutions. By the arguments we present in Appendix A, this is an attractive option. The actual details, however, require further research.

optimal paths may exist. (They do not have to be efficient.) We refer to the path that minimizes expected travel time as path μ (with mean μ_μ and variance σ_μ^2). By the independence assumption, $\sigma^2 = \Sigma\sigma_{ij}^2$, where the summation is along a path; so we can also find path σ—the path that minimizes σ^2 (with mean μ_σ and variance σ_σ^2)—by solving a deterministic shortest path problem with distances given by σ_{ij}^2. If the two solutions are identical, that path is optimal. From now on, however, we assume that paths μ and σ are distinct, so $\mu_\mu < \mu_\sigma$ and $\sigma_\mu > \sigma_\sigma$. Define *path* λ as the path that minimizes $(1 - \lambda)\mu + \lambda\sigma^2$ for $0 \le \lambda \le 1$. Paths μ and σ are special cases that correspond to $\lambda = 0$ and $\lambda = 1$; i.e., path μ is also named path 0.0 and path σ is also path 1.0. (We include decimal points to emphasize that these indices are real numbers.) For any given λ, we can find path λ by solving a deterministic shortest path problem with distances of $(1 - \lambda)\mu_{ij} + \lambda\sigma_{ij}^2$ for all pairs $i, j = 1, 2, \ldots, n$. When viewed as step functions of λ, μ_λ is monotone nondecreasing by construction and σ_λ^2 is monotone nonincreasing. Suppose that the set has m members, each associated with a range of possible λ arguments. To identify all the members of this set, assume we have already identified $k \ge 2$ members (starting with $k = 2$ as given by path μ and path σ). Using any two adjacent members as *endpoints*, we obtain a *search segment* that connects them. Let λ be the argument for which $(1 - \lambda)\mu + \lambda\sigma^2$ is constant along the search segment. Searching for the shortest path with distances of $(1 - \lambda)\mu_{ij} + \lambda\sigma_{ij}^2$ we find path λ. Path λ is either a new member of the efficient set or it merges with an endpoint of the search segment, signaling that there is no efficient path between the endpoints. When all search segments yield no new paths, the efficient set is complete. We can depict the efficient set with the final search segments connecting its adjacent members as a chain in the μ-σ^2 plane, which we call the *efficient chain*. All possible paths must reside on or above and to the left of the efficient chain. The efficient paths themselves are *extreme points* along the chain. Pairs of adjacent segments form angles of less than 180° at these extreme points. To construct the complete efficient chain requires solving the shortest path problem $\max\{2, (2m - 1)\}$ times. We illustrate the computations by calculating the efficient routes for the network in Figure RN1.

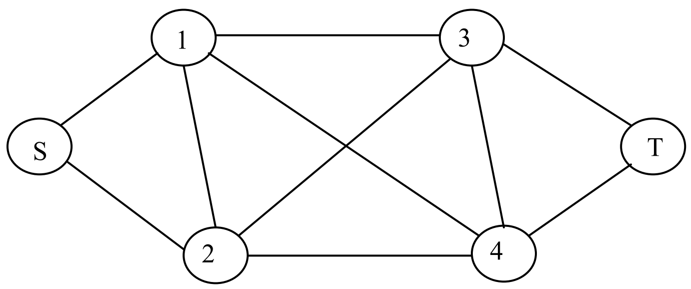


Figure RN1: An example network

Travel times are stochastically independent with the following means and variances:

Edge	S-1	S-2	1-2	1-3	1-4	2-3	2-4	3-4	3-T	4-T
Mean	8	16	11	20	24	4	8	6	6	4
Variance	4	12	4	4	5	4	4	6	10	8

For $\lambda = 0$ we use only mean data (i.e., S-1 = 8, S-2 = 16, etc.). Starting at *S*, by Dijkstra's shortest route algorithm, node 1 has a distance of 8 and no other node is closer to S, so node 1 is added to the solved set (i.e., the set of nodes for which we know the shortest route from S). Next, node 2 is added, with a distance of 16. Node 3 is solved next, via node 2, with a distance of 20. Node 4 follows, with a distance of 24. Node T comes next, via node 3, with a total distance of 26. Tracing back, the minimal route is S-2-3-T, and we should also calculate the variance: 12 + 4 + 10 = 26. The result, (26, 26), is depicted at the top of the chain of linear segments in Figure RN2. Next, for $\lambda = 1$ (i.e., when we ignore the mean), we find that route S-1-4-T is shortest in terms of variance, yielding $\mu = 36$, $\sigma^2 = 17$. To calculate the λ value for which these two routes are equivalent, we can start by calculating an interim parameter, $w = (26 - 17)/(36 - 26) = 0.9$ (that is, the absolute difference in variance divided by the absolute difference in mean between the two points). We can now find $\lambda = 1/(1 + w) = 1/1.9 = 0.5263$. For this λ, the value we need to consider for each edge is calculated by the formula $(1 - \lambda)\mu_{ij} + \lambda\sigma_{ij}^2$; e.g., for edge S-1 we have $0.4737\times8 + 0.5263\times4 = 5.895$. Given these calculated distances, we find the new shortest route as S-1-2-4-T. The numerical value associated with the tour is 25.210, which is indeed lower than 26 (the value associated with the two endpoints). Next we calculate that the new tour has $\mu = 31$ and $\sigma^2 = 20$. We now need to check whether there is another efficient route between this new route and each of the two previous ones. These are associated with $w = (26 - 20)/(31 - 26) = 1.2$, yielding $\lambda = 1/2.2 = 0.4545$ and $w = (20 - 17)/(36 - 31) = 0.6$, yielding $\lambda = 1/1.6 = 0.625$. The former yields the route S-1-2-3-T, with $\mu = 29$, $\sigma^2 = 22$. The latter yields the route S-1-3-T, with $\mu = 34$, $\sigma^2 = 18$. The insertion process stops when no new point can be identified between the current pair—so each new insertion requires two subsequent trial insertions. In the example, no additional routes emerge, so the full list is given by:

λ	0.0	0.4545	0.5263	0.625	1.0
route	S-2-3-T	S-1-2-3-T	S-1-2-4-T	S-1-3-T	S-1-4-T
mean	26	29	31	34	36
Variance	26	22	20	18	17

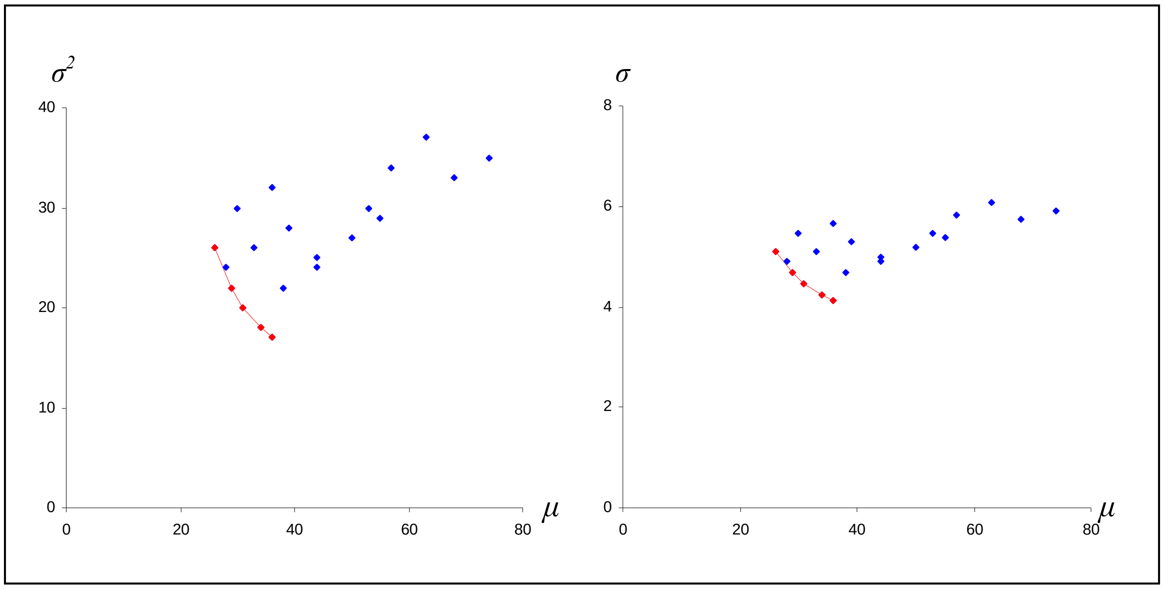


Figure RN2: The efficient set in the μ-σ^2 plane and the μ-σ plane

Figure RN2 depicts all the possible routes in this example, with the efficient set at the lower left side connected by piecewise linear segments. The left side of the figure depicts the routes in the μ-σ^2 plane and the right side depicts the μ-σ plane. In this particular example, the same tours are efficient in both depictions. It now becomes straightforward to select the best route for any given parameters. Furthermore, instead of calculating the complete set of efficient routes, it is possible to devise a search that skips most of them, but we omit the details here.

As noted above, a similar procedure applies for the TSP. However, to even pose the TSP well for some given λ, we must first solve the minimal route problem for all pairs of cities. Strictly speaking, it is not necessary to use Dijkstra's algorithm for each pair of cities for this purpose. The Floyd-Warshall algorithm can be used to solve all distances at the same time (Floyd, 1962). The complexity of Floyd's algorithm is $O(n^3)$, which is better than running Dijkstra's algorithm $n(n-1)$ times. Nonetheless, if we ignore this issue and use Dijkstra's algorithm, we can pre-solve for any λ in advance. In the example above, for any λ between 0 and 3/7, route 0.0 is optimal, and route 0.4545 is optimal for any λ between 3/7 and 1/2. The limiting values are simply those associated with the last search segments that were tested during the construction; e.g., the value 3/7 is associated with the search segment between tour 0.0 and tour 0.4545. Special attention should be accorded in case of ties: formally, we may need to consider both options because they imply different routes. Although dealing with ties makes the algorithm more tedious, it does not make a difference in terms of computational complexity.

In model 1 and model 2 we treat the due date as a decision, but alternate models assume a given due date. We discuss those in a TSP context. The model 1 alternate, *alternate 1*, calls for maximizing the service level. This problem has received some attention in the literature: Kao (1978) introduced the model but Sniedovich (1981) exposed a flaw in Kao's solution. Carraway et al. (1989) presented a generalized dynamic programming approach whose *average* complexity exceeds $O(n^2 2^n)$ and offered no evidence that the worst-case performance is better than complete enumeration. Kenyon and Morton (2003) show that for $SL \geq 0.5$ the model is convex. For this case they propose a continuous search to identify the highest possible service level by solving a series of deterministic nonlinear integer programs whose continuous relaxation is a convex program. Thus, one can approach the optimal value as closely as desired (but without actually reaching it). We can show that alternate 1 is equivalent to model 1 for any service level: they are both solved by selecting from the same finite set of efficient tours. That is, we can identify the optimal tour precisely if we can generate the set of efficient tours. Generating this set is indeed easier for $SL \geq 0.5$. Generating this part of the set is also sufficient for the solution of model 2. The model 2 alternate minimizes the penalty of late arrivals when it is proportional to the length of the delay (Laporte et al., 1992). That is, it requires minimizing $\mathrm{E}(T)$, where $T = (C - d)^+$. In our context, the model 2 alternate requires considering a larger set of Pareto optimal solutions. For instance, in Figure RN2 we observe a Pareto optimal route right below route 0.0, with mean 28 and variance 24. That route is not efficient but we cannot guarantee in advance that it is not optimal for some instance of alternate 2 (when considering the shortest route case). Therefore, model 2 and its alternate are not equivalent, and we do not consider that alternate further here.

It is possible to solve model 2 or model 1 with service level targets of at least 0.5 by iterative application of the deterministic TSP solution. The same applies to alternate 1 when the due date is not shorter than path 0.0. Furthermore, the same approach provides effective heuristic solutions for all the cases we excluded, such as model 1 with $SL < 0.5$, alternate 1 with due date smaller than tour 0.0, or alternate 2. We conjecture that the number of iterations of the TSP that might be required to identify the efficient set (and the tours required for the heuristic of the excluded cases) is bounded by a low order polynomial of n.[*] If true, the conjecture implies that the practical complexity of our problem is close to that of the deterministic TSP. Using a direct search approach (which avoids identifying the full efficient set), our numerical experience suggests that the practical complexity when expressed as a number of TSP applications does not exceed $O(n)$, with a very small coefficient. Furthermore, for a particular experimental design, we found that the suboptimality gap associated with limiting the search to precisely two TSP applications achieves about 99.98% of the potential benefit of finding the optimal tour as compared to the deterministic counterpart tour. With this approach, we were able to solve for up to $n = 1000$, using Concorde. For $\pi = 1.5$, we found empirically that as n grows large, the probability that tour μ is optimal becomes negligible, but the improvement offered by the optimal tour tends to become very small, too. Figure RN3 demonstrates that point.

[*] Bagchi (1989) addresses bi-criteria problems with a similar structure. There, the number of efficient solutions cannot exceed n. The proof does not extend to our case, however.

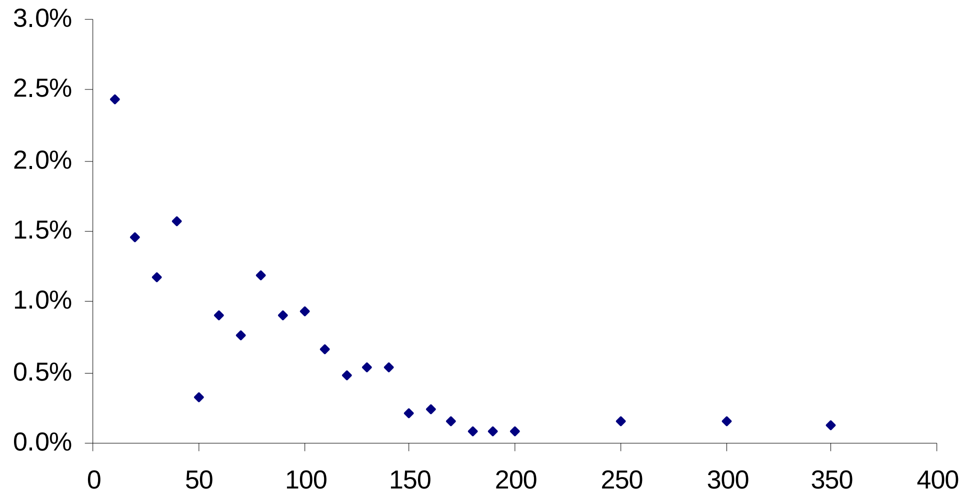


Figure RN3: The average improvement of the second TSP as a function of *n*

Practical applications often involve small *n*. When *n* is large, we usually must solve a vehicle routing problem instead, and each vehicle has a small number of cities to visit. Thus, it is useful to perform the second search, but further searches may be considered redundant. Nonetheless, we encountered no problems that were intractable in the range we tested, up to *n* = 1000. However, we did identify maximization problems with multiple near-optimal solutions that took much longer. Figure RN4 provides a clue. On the right side—marked as the Euclidean case—it shows that the right-side boundary of the convex hull of problems with *n* = 30 and *n* = 50 is almost vertical. We generated these samples by placing cities randomly on a unit square and then generating random coefficients of variation for each Euclidean distance thus obtained. (We also generated the convex hull for one *n* = 70 case, not depicted, which took more than 100 hours and yielded a similar right-side boundary.) The problem is that for the tours on the "vertical" boundary, the solver requires substantial branching, with branches that yield very little useful information on average. The left-hand side of the figure depicts cases where mean distances and variances were generated independently. That structure is less likely in practice but was easier to analyze in our experiments.

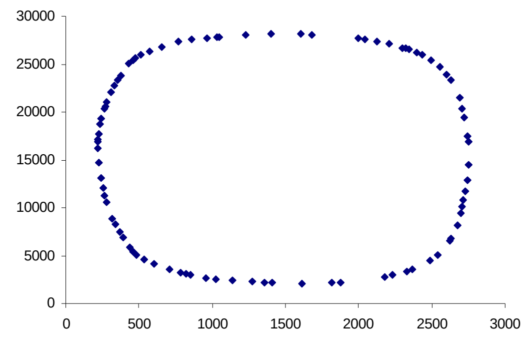
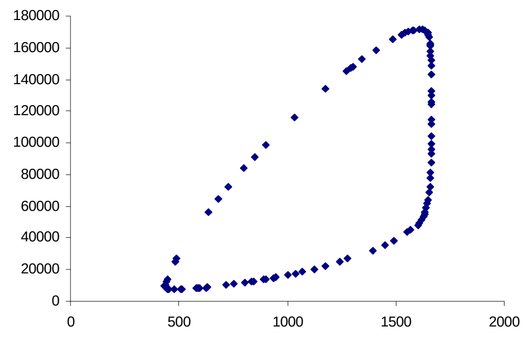
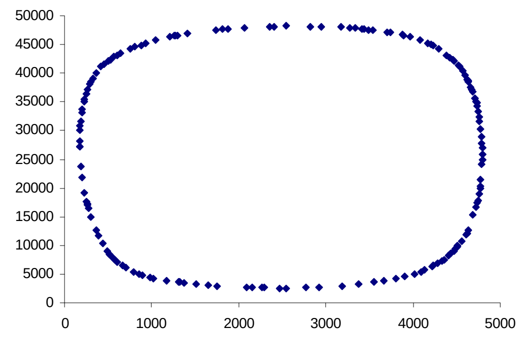
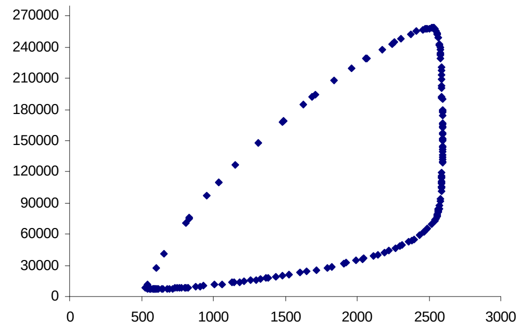
	
Random case (n=30)	Euclidean case (n=30)
	
Random case (n=50)	Euclidean case (n=50)

Figure RN4: The convex hulls of simulated TSP examples

Even if the tours on the right boundary in Figure RN4 were relevant, we could identify excellent solutions very early in these cases: the excessive time was devoted to proving optimality. For example, Figure RN5 depicts a case for *n* = 100 which took over 3000 seconds to converge to the optimal tour. If we ignore the vertical scale, it seems that for about half that time there was a sizable difference between the upper and lower bound and about half the time was devoted to the tiny improvements and verifying optimality. However, by paying attention to the scale we can see that right from the start we could stop with a certificate guaranteeing that the upper bound (our initial solution) is within 0.08%, and getting there took no more than 5 seconds. Thus, over 3000 seconds were needed to achieve 0.07% improvement and prove optimality. In a practical sense, that time was wasted. Thus the models are tractable and robust in practice.

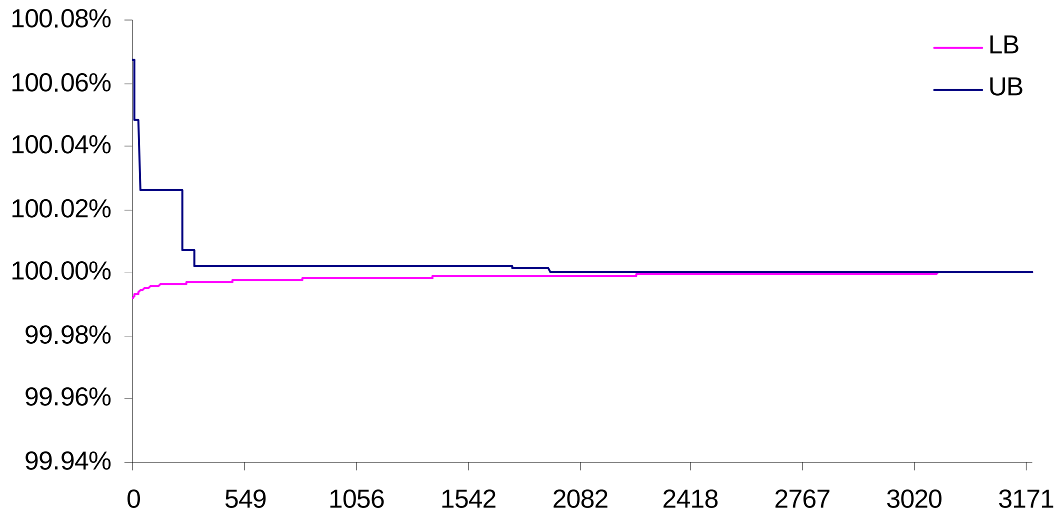


Figure RN5: The optimality gap of a 100-city example as a function of CPU time

References

Ahmadi, R. and U. Bagchi (1990) "Lower Bounds for the Single-Machine Scheduling Problem," *Naval Research Logistics* 37, 967-979.

Applegate, D., R. Bixby, V. Chvatal and W. Cook (URL accessed June 2008), http://www.tsp.gatech.edu/concorde

Bagchi, U. (1989) "Simultaneous Minimization of Mean and Variation of Flow Time and Waiting Time in Single Machine Systems," *Operations Research* 37(1), 118-125.

Bagchi, T.P., J.N.D. Gupta and C. Sriskandarajah (2006). "A Review of TSP Based Approaches for Flowshop Scheduling," *European Journal of Operational Research* 169, 816–854.

Bertsimas, D.J., P. Jaillet and A. Odoni (1990), "A Priori Optimization," *Operations Research* 38(6), 1019-1033.

Bertsimas, D.J and D. Simchi-Levi (1996), "A New Generation of Vehicle Routing Research: Robust Algorithms, Addressing Uncertainty," *Operations Research* 44, 286-304.

Carlier, J. (1982) "The One Machine Sequencing Problem," *European Journal of Operational Research* 11, 42-47.

Carraway R.L., T.L. Morin and H. Moskowitz (1989) "Generalized dynamic programming for stochastic combinatorial optimization," *Operations Research* 37, 819-829.

Chandra, R. (1979) "On $n/1/\overline{F}$ Dynamic Deterministic Problems," *Naval Research Logistics Quarterly* 26, 537-544.

Chu, C. (1992) "A Branch-and-Bound Algorithm to Minimize Total Tardiness with Different Release Dates," *Naval Research Logistics Quarterly* 39, 265-283.

Chu, C. and M.-C. Portmann (1992) "Some New Efficient Methods to Solve the $n/1/r_i/\sum T_i$ Scheduling Problem," *European Journal of Operational Research* 58, 404-413.

Demeulemeester, E.L. and W.S. Herroelen (2002), *Project Scheduling: A Research Handbook*, Kluwer Academic Publishers.

Dror, M. and P. Trudeau (1986) "Stochastic Vehicle Routing with Modified Savings Algorithm," *European Journal of Operational Research* 23, 228-235.

Floyd, R.W. (1962), "Algorithm 97: Shortest Path," Communications of the ACM 5(6), 345.

Gao S. and I. Chabini (2006) "Optimal Routing Policy Problems in Stochastic Time-Dependent Networks," *Transportation Research Part B* 40, 93-122.

Gendreau, M., G. Laporte and R. Séguin (1996), "Stochastic Vehicle Routing," *European Journal of Operational Research* 88(1), 3-12.

Gilmore, P.C. and R.E. Gomory (1964). "Scheduling a One-State Variable Machine: A Solvable Case of the Traveling Salesman Problem," *Operations Research* 12, 655–679.

Jackson, J.R. (1955) "Scheduling a Production Line to Minimize Maximum Tardiness," Research Report 43, Management Science Research Project, University of California, Los Angeles.

Jaillet, P. (1988) "A Priori Solution of a Traveling Salesman Problem in which a Random Subset of the Customers are Visited," *Operations Research* 36, 929-936.

Jonker, R. and T. Volgenant (1983) "Transforming Asymmetric into Symmetric Traveling Salseman Problems," *Operations Research Letters* 2, 161-163.

Kanet, J. and X. Li (2004) "A Weighted Modified Due Date Rule for Sequencing to Minimize Weighted Tardiness," *Journal of Scheduling* 7, 261-276.

Kao, H.P.C. (1978) "A Preference Order Dynamic Program for a Stochastic Traveling Salesman Problem," *Operations Research* 26, 1035-1045.

Kenyon, A.S. and D.P. Morton (2003) "Stochastic Vehicle Routing with Random Travel Times," *Transportation Science* 37, 69-82.

Kise, H., T. Ibaraki and H. Mine (1978) "A Solvable Case of the One-Machine Scheduling Problem with Ready and Due-Times," *Operations Research* 26, 121-126.

Lageweg, B.J., J.K. Lenstra and A.H.G. Rinnooy Kan (1976) "Minimizing Maximum Lateness on One Machine: Computational Experience and Some Applications," *Statistica Neerlandica* 30, 25-41.

Laporte, G. (1992), "The Traveling Salesman Problem: An Overview of Exact and Approximate Algorithms," *European Journal of Operational Research* 59, 231-247.

Laporte, G., F.V. Louveaux and H. Mercure (1992) "The Vehicle Routing Problem with Stochastic Travel Times," *Transportation Science* 26, 161-170.

Laporte, G., F.V. Louveaux and H. Mercure (1994) "A Priori Solution of the Probabilistic Traveling Salesman Problem," *Operations Research* 42, 543-549.

Mazmanyan, L., D. Trietsch and K.R. Baker (2009) "Stochastic Traveling Salesperson Models with Safety Time" (working paper).

McMahon, G. and M. Florian (1975) "On Scheduling with Release dates and Due Dates to Minimize Maximum Lateness," *Operations Research* 23, 475-482.

Monma, C.L. (1981) "Sequencing with General Precedence Constraints," *Discrete Applied Mathematics* 3, 137-150.

Morton, T.E. and B.G. Dharan (1978) "Algoristics for Single-Machine Sequencing with Precedence Constraints" *Management Science* 24, 1011-1020.

Potts, C.N. (1980) "Analysis of a Heuristic for One Machine Sequencing with Release Dates and Delivery Times," *Operations Research* 28, 1436-1441.

Sniedovich, M., (1981) "Analysis of a Preference Order Traveling Salesman Problem," *Operations Research* 29, 1234-1237.

Shingo, S. (1985) *A Revolution in Manufacturing: The SMED System*, Productivity Press, Cambridge, MA.

Trietsch, D. (1992) "Some Notes on the Application of Single Minute Exchange of Die (SMED)," *NPS-AS-92-019* (Technical Report, Naval Postgraduate School, Monterey, California).