OR and Simulation in Combination for Optimization

Nico M. van Dijk, René Haijema, Erik van der Sluis, Nikky Kortbeek, Assil Al-Ibrahim and Jan van der Wal

Abstract This chapter aims to promote and illustrate the fruitful combination of classical operations research (OR) and computer simulation. First, a highly instructive example of parallel queues will be studied. This simple example already shows the necessary combination of OR (queueing) and simulation that appears to be of practical interest such as for call center optimization. Next, two more 'real life' applications are regarded: (1) blood platelet production and inventory management at blood banks, and (2) train conflict resolution for railway junctions. Both applications show the useful combination of *simulation* and *optimization methods from OR*, in particular stochastic dynamic programming (SDP) and Markov decision theory (MDP), to obtain simple rules that are nearly optimal. The results are based on real-life Dutch case studies and show that this combined OR-simulation approach can be most useful for 'practical optimization' and that it is still wide open for further application.

1 Introduction

Discrete event simulation is well known as a most powerful tool for logistical process computation and performance evaluation in a vast majority of fields. Standard applications are found in the production sector, the service industry (call centers, administrative logistics), transportation (public transportation systems, road traffic, airports,

N.M. van Dijk (✉) · R. Haijema · E. van der Sluis · N. Kortbeek · A. Al-Ibrahim · J. van der Wal
Faculty of Economics and Business, University of Amsterdam, Amsterdam, The Netherlands
e-mail: N.M.vanDijk@uva.nl

N.M. van Dijk · N. Kortbeek
Stochastic Operations Research group, University of Twente, Enschede, The Netherlands

R. Haijema
Operations Research and Logistics Group, Wageningen University, Wageningen, The Netherlands

M. Mujica Mota et al. (eds.), *Applied Simulation and Optimization*,
DOI 10.1007/978-3-319-15033-8_3

harbors, maritime logistics, express delivery systems, and so on), computer networks, communication networks and, over the last decade with fast growing attention, in health care logistics. In most of these applications simulation is required, due to

- the complexity of the system,
- the uncertainties (stochastics) involved at micro up to macro level.

On the one hand, analytic techniques, most notably OR (operations research) techniques such as queueing analysis and dynamic programming, are generally restricted to simplified models and simplifying underlying assumptions that have to be made. On the other hand, simulation by itself does not standardly provide underlying insights nor techniques for optimization. Clearly, in simple situations in which an optimization problem can be parameterized by one parameter, such as by the number of staffing or storage capacity to be determined, simulation search approaches can be suggested to expedite and automate the search for an optimal value. An elegant exposé of such methods can be found in Krug [20]. Case-specific references are found in Sects. 2–4.

Unfortunately, in most realistic logistical situations there will be multiple parameters and problem aspects that complicate the optimization. In these situations, at best a number of different scenarios might be proposed to be evaluated and to be compared by simulation. Alternatively, fast and extensive simulation search approaches might be developed for optimization, as studied in the last decade. As such, simulation is to be regarded as a most practical and almost unlimited tool for scenario "optimization". But, as mentioned, it remains to be realized that simulation by itself does not provide any of these scenarios nor an automatic tool for optimization. This is where OR might help out in either of two directions

(i) To suggest scenarios based upon general OR results and insights.
(ii) To provide an OR-optimization technique for the problem included.

Clearly, analytic or OR models are generally too simplistic for realistic modeling. Simulation in contrast, hardly seems to have any limitations on the modelling complexity at all. But to the price of loosing general insights due to this complexity or at least having to simulate extensively to gain such insights. Here the simplistic OR model might play an important role of just being generic and providing essential insights. A similar statement can be made for insights on modeling the underlying stochasticity. OR models strongly rely upon distributional assumptions, most notably of exponential nature, which can easily be parameterized by an arrival or a service rate. Such strict simplifying assumptions can be relaxed by simulation, but to the price of requiring detailed input data on very specific input distributions. In addition, the outcomes of a simulation dependent on the sampled random data. For a fair comparison of slightly different scenarios of the same system requires many and long simulation runs to obtain accurate confidence intervals that allow for hypothesis testing. In contrast, to say the least, analytic or OR models, even though simplistic and whether exact or approximate, provide expressions or algorithms that are 100 % verifiable and replicable.

Table 1 Combined advantages

OR advantages	OR disadvantages
Optimization	Simple models
By techniques	Strict assumptions
Also by insights	
Scenario development	
Analytic approximate results	
SIM-advantages	SIM-disadvantages
Evaluation only	Real-life stochastics
By numbers only	Real-life complexities
By scenarios only	
Computationally expensive	
Requires highly detailed data	

In short, a combination of OR and simulation might thus become highly beneficial to compensate for the disadvantages of one another and to exploit the advantages of either

- OR for insights from simple computation and optimization,
- Simulation for more realistic evaluation and validation.

The disadvantages and combined advantages are summarized in Table 1.

This chapter aims to promote and illustrate the practical potential of the combination of simulation and optimization (OR). It therefore collects and exposes three applications based on more detailed and technical papers by the authors, as outlined in Table 2. The chapter is organized by its separate applications in Sects. 2–4. In each of these sections, the same structuring is used by its specific problem formulation and background literature, and also by a presentation of the combined OR-simulation approach and by its concrete practical numerical results and conclusions. The chapter will be concluded by an evaluation in Sect. 5.

2 Should We Pool or Not?

This section studies the question as simple as whether queues (or line ups) in front of service desks should be combined (pooled) into a single queue (line up) or not. The example in this section is based on van Dijk and van der Sluis [30] and van Dijk and van der Sluis [31].

Table 2 Combination of techniques for three applications

Sections	Topic	Combination
Sect. 2	Pooling in call centers	SIM + Q insights
Sect. 3	Blood banks	SIM + MDP
Sect. 4	Railways	SIM + Q + semi-MDP

Legend SIM: Simulation; Q: Queueing; MDP: Markov Dynamic Programming

In this section, it will be shown that both OR insights and simulation are essential for an improvement of either a pooled or an unpooled situation and further optimization.

2.1 Motivation and Literature

The question relates to a variety of daily-life situations such as at banks, information desks, ticket offices, up to manufacturing with parts and tools lined up for parallel machines. A question that seems too simple to be asked as the answer seems so obvious. In contrast, however, it appears to be surprisingly intriguing.

Capacity pooling is a common concept. The general perception seems to be in favor of pooling (e.g., see Borst et al. [7], Cattani and Schmidt [8]). From an OR-, or rather queueing-, point of view, it seems less obvious, if not highly intriguing (e.g., Bell and Williams [5]). Counterintuitive examples can already been found in the book of Wolff [34] and Smith and Whitt [26]. Particularly, motivated by present-day developments of so-called skill-based routing; over the last decade, it has been given considerable attention within the application field of call centers (see Gans and Zhou [14], Wallace and Whitt [33]). The insights and results from the field of queueing appear to be essential to steps for performance improvement and optimization, such as by overflow and threshold policies (see van Dijk and van der Sluis [30], Osogami et al. [23], Squillante et al. [27], Wallace and Whitt [33]). Here, overflow mechanisms come in for which simulation becomes necessarily required.

2.2 Queueing Insights

2.2.1 A First Queueing Insight

Indeed, the last perception seems supported by the most standard queueing expression $D = 1/(\mu - \lambda)$ with

μ: the service rate (or capacity) of the server (per unit of time)
λ: the arrival rate (per unit of time) and
D: the average (or mean) delay

with the implicit assumption of exponential service times. (This assumption can be regarded as formal justification for speaking in terms of a service rate.) Pooling two of such servers as if it becomes a twice as faster server with double arrival rate thus seems to reduce the mean delay by 50 % according to $D = 1/(2\mu - 2\lambda)$. This delay reduction seems to result from the efficiency gained by pooling individual queues. The inefficiencies in the nonpooled case are avoided, as one server can no longer be idle while there are still customers waiting at another. The intuitive reasoning above thus seems to be supported by queueing theoretic results and in-line with the general perception that pooling is beneficial.

More precisely, by straightforward calculation from standard M/M/s-expressions, we find

$$\frac{W_E(2,\rho,\tau)}{W_E(1,\rho,\tau)} = \frac{\tau\rho^2/(1-\rho^2)}{\tau\rho/(1-\rho)} = \frac{\rho}{1+\rho}$$

with $W_E(s,\rho,\tau)$: the mean waiting time for an exponential server group with s servers, (that is an M/M/s-queue) with traffic load $\rho = \lambda/s\mu$ per server, with λ the arrival rate and $\tau = 1/\mu$ the mean service time.

This shows that the effect of pooling two parallel servers depends on the traffic load ρ and will indeed lead to a reduction of at least 50 %.

2.2.2 A Second Queueing Insight: Variability Factor

This reasoning, however, relies upon the implicit assumption of statistically identical jobs, identical servers, and equal server loads; however, when services are pooled with different service means, there is also another elementary queueing result that becomes important. This is the factor of the variability of the service times (in addition to just the mean). For example, for the case of a single-server system this is expressed by Pollaczek–Khintchine's famous formula

$$W_G = (1+c^2)W_D - \frac{(1+c^3)}{2}W_E$$

where W_G, W_D and W_E are the expected (average or mean) waiting times for the situation of a general service distribution (G) with squared coefficient of variation c^2, respectively, for a deterministic (or fixed) service time D (hence with $c^2 = 0$) and for an exponential distribution E (for which $c^2 = 1$), and where

c^2 the squared coefficient of variation $= \sigma^2/\tau^2$
σ^2 the variance of the service time;
τ the mean service time.

In words, this formula tells us that also the variation (as expressed by σ^2) around (relatively to) the mean τ of the service times plays an essential factor for the average delay (as compared to the situation of fixed service times).

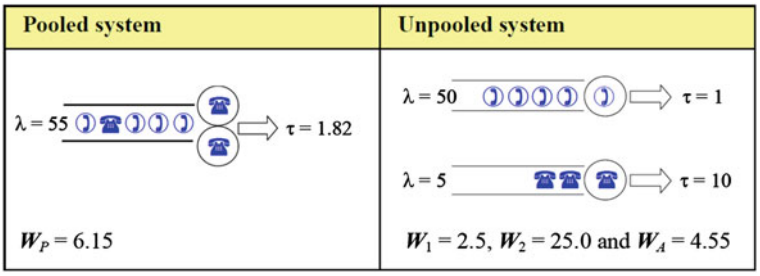

Fig. 1 Two-server example ($k = 10$, $\rho = 0.83$) by OR (queuing)

By pooling different services, due to the mix variability introduced and as by the Pollaczek–Khintchine formula, the effect will thus be less beneficial.

2.2.3 Instructive Example

Consider the situation of two arrival streams of service (e.g., call) requests, referred to as of type 1 and 2, with arrival rate λ_1 and λ_2, and mean service time τ_1 and τ_2, with equal workload $\rho = \lambda_1\tau_1 = \lambda_2\tau_2$, and two servers which can handle either type of service. Let

$k = \lambda_1/\lambda_2 = \tau_2/\tau_1$,
$\overline{\tau} = p_1\tau_1 + p_2\tau_2$, with $p_i = \lambda_i/(\lambda_1 + \lambda_2)$,
W_A Average waiting time for all jobs,
W_1 Average waiting time for type 1 jobs,
W_2 Average waiting time for type 2 jobs,
W_P Average waiting time for the pooled case.

Figure 1 then illustrates the effect of pooling, for example, with $k = 10$, $\lambda_1 = 50$ and $\lambda_2 = 5$ per hour, $\tau_1 = 1$ and $\tau_2 = 10$ min (hence 10 times more short jobs which are 10 times shorter). Furthermore, the service times are assumed to be deterministic and the waiting times are expressed in minutes.

2.2.4 A Pooling Formula

With c^2_{mix} denoting the mix coefficient for the pooled case, as computed by:

$$c^2_{mix} = \frac{p_1(\tau_1 - \bar{\tau})^2 + p_2(\tau_2 - \bar{\tau})^2}{\bar{\tau}^2} = p_1\left(\frac{\tau_1}{\bar{\tau}^2}\right)^2 + p_2\left(\frac{\tau_2}{\bar{\tau}^2}\right)^2 - 1$$

the effect of pooling these two servers is then given by

$$c^2_{mix} = \frac{k}{k+1}\left(\frac{k+1}{2k}\right)^2 + \frac{1}{k+1}\left(\frac{k+1}{2}\right)^2 - 1 = \frac{(k-1)^2}{4k}.$$

$$\frac{W_P}{W_A} \approx \frac{1/2(1+c_{mix}^2)W_E(2,\rho,\bar{\tau})}{1/2W_E(1,\rho,\bar{\tau})} = (1+c_{mix}^2)\left(\frac{\rho}{1+\rho}\right) = \frac{(k+1)^2}{4k}\left(\frac{\rho}{1+\rho}\right)$$

$$\frac{W_P}{W_1} \approx \frac{1/2(1+c_{mix}^2)W_E(2,\rho,\bar{\tau})}{1/2W_E(1,\rho,\tau_1)} = (1+c_{mix}^2)\frac{2k}{k+1}\left(\frac{\rho}{1+\rho}\right) = 1/2(k+1)\left(\frac{\rho}{1+\rho}\right)$$

These expressions directly lead to the following conclusions:

Conclusions 1 (As based on OR (queueing), for the two-server case, deterministic services and identical loads)

1. Pooling is always beneficial for type 2.
2. There can be an increase for a fraction $k/(k+1)$ of the calls for $k > 3$.
3. Pooling is not beneficial for $k > 5$.

Similar results can be just as well for larger server numbers; say instead of 2 single servers for 2 groups of servers each of size s = 5, 10, 20 servers, as of realistic interest for call center dimensioning. For more details, we refer to van Dijk and van der Sluis [30, 31].

More generally, by these rather basic but essential OR (queueing) insights and results, it would seem advantageous to combine the advantages of:

- No (or minimum) idleness as for the pooled case
- No (or minimum) service variability as for the unpooled case.

2.3 Improvement and Optimization by OR and Simulation

2.3.1 Overflow

An overflow system is proposed to further improvement for the overall mean waiting time. This is where *simulation* comes in necessarily. Overflow systems are virtually unsolvable analytically. In Fig. 2, the results by simulation are shown for a two-way overflow (2WO) and a one-way (1WO-1) scenario as specified by

- *Two-way overflow* (*2WO*): A separate queue for each type. An idle server, when there are no jobs of its own type waiting, will take a job waiting of the other type, if any.
- *One-way overflow* (*1WO-1*): A separate queue for each type. Only an idle server of type 2 and if there are no jobs waiting of type 2 will take a job from the other queue, if any.

Figure compares for $s = 1$ (two parallel servers), $k = 10$ (hence with 10 times more short jobs which are 10 times shorter) four basic scenarios of the pooled (P), the unpooled (U), a two-way overflow (2WO) and a one-way (1WO-1) scenario.

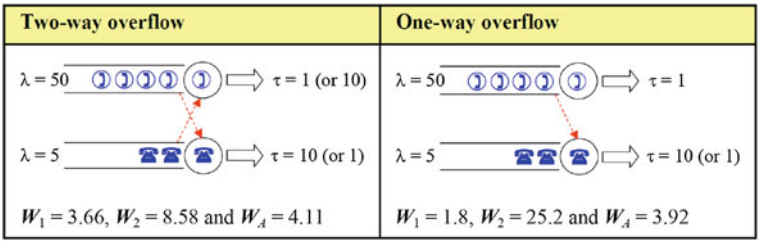

Fig. 2 Two-server example ($k = 10$, $\rho = 0.83$) by OR insights and simulation

Two more simple scenarios to improve the overall average waiting time, as also based on queueing insights, are to prioritize type 1 jobs, either without preemption (service interruption) or with preemption of type 2 jobs when a type 1 arrives. In either way, service for a type 2 job only starts (or is resumed) when no more type 1 jobs are waiting

- *Nonpreemptive-Priority-1* (*NP1*): As in the pooled case and with priority for type 1 jobs when a server idles. Type 2 jobs are served only if there is no type 1 job waiting.
- *Preemptive-Priority-1* (*PP1*): As scenario NP1. In addition, when a type 1 job arrives, a type 2 job is preempted. When no more type 1 jobs are waiting, type 2 jobs are resumed.

By simulation the possible improvement is illustrated in Fig. 3 for the situation with $k = 10$, $\rho = 0.9$ and $s = 10$ (20 servers in total). (To focus on type 1 jobs, the two-way scenario, which would rank in between the unpooled and one-way scenario for the allover average, is left out.)

Rank	Scenario	W_A
5	Pooled	0.71
4	Unpooled	0.63
3	One-way	0.52
2	NP1	0.38
1	PP1	0.20

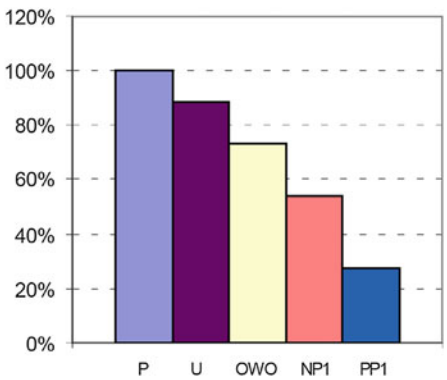

Fig. 3 Average waiting times for different scenarios

Conclusion 2 As based upon OR insights and by simulation, it appears that overflow and priority rules perform substantially better.

2.3.2 Single Threshold Optimization

As shown in Sect. 2.3.1, a simple priority rule, particularly the preemption scenario for short (type 1) jobs, generally seems to perform quite well and to be "optimal" among simple scenarios. Unfortunately, preemption (interruption) of service will generally be impractical. A further improvement step by queueing and Simulation might therefore be proposed by using threshold policies, where $T(\theta_1)$ indicates a threshold policy as specified by

$$T(\theta_1) = \begin{cases} \text{a server of either type serves jobs from queue 1,} \\ \quad \text{if either } (i) \quad m_1 \geq \theta_1 \quad \text{or} \quad (ii) \quad m_2 = 0 \wedge m_1 \geq 1; \\ \text{it serves jobs from queue 2,} \\ \quad \text{otherwise.} \end{cases}$$

More precisely, by exploiting simulation even an optimization can take place by determining

$$W^* = \min_{\theta_1} W(\theta_1).$$

The results, as had to be obtained by simulation, for some optimal threshold values compared to the results for the pooled and NP1 scenarios are shown in Table 3.

In fact, it has also been concluded by simulation that these single threshold policies, with only a threshold value θ_1 for type 1, are nearly optimal among all policies that use threshold values for both type 1 and type 2 servers.

Table 3 Optimal threshold values

s	Pooled	NP1	$\mathbf{T}(\theta_1)$	
	W_P	W_A	W_A	θ_1^*
1	11.53	4.58	4.37	3
2	5.27	2.44	2.27	4
3	3.33	2.63	1.51	4
4	2.34	1.19	1.12	4
5	1.76	0.93	0.88	5
10	0.71	0.38	0.38	1
15	0.40	0.21	0.21	1
20	0.26	0.14	0.14	1
30	0.13	0.07	0.07	1

2.3.3 Double Optimization

So far, an improvement of the average overall average waiting time was obtained by particularly improving the mean waiting time for type 1 jobs. Here the average was computed proportional to the arrival rate ratio for type and type 2 jobs. Though this averaging makes natural sense, the choice of weights for type 1 and 2 jobs is still arbitrary. In all scenarios so far, a price still had to be paid by type 2 jobs (even though for just a small percentage of the jobs).

As an another objective, and supported by insight from queueing, we can address the question whether a scenario can be found that strictly improves the pooled scenario, that is, in mean waiting time, for both type 1 and type 2 jobs. As mentioned in Sect. 2.3.2, for the overall average mean waiting time, an optimization by threshold values basically boiled down to just one threshold value θ_1. For the purpose of a strict improvement, in contrast, also a prioritization of type 2 jobs might thus be expected and be required. Instead of one threshold value θ_1, we will consider threshold rules with one threshold values θ_1 and θ_2 for either type of jobs. More precisely, let the threshold rule $\mathbf{S}(\theta_1, \theta_2) = \mathbf{Thr}(\theta_1, \theta_2, \theta_1, \theta_2, 1, 1)$, i.e., as specified by:

$$\mathbf{S}(\theta_1, \theta_2) = \begin{cases} \text{a server of type 1 serves jobs from queue 2,} \\ \quad \text{if either } (i) \quad m_2 \geq \theta_2 \wedge m_1 < \theta_1 \text{ or } (ii) \quad m_1 = 0 \wedge m_2 \geq 1; \\ \quad \text{otherwise, it serves jobs from queue 1.} \\ \text{a server of type 2 serves jobs from queue 1,} \\ \quad \text{if either } (i) \quad m_1 \geq \theta_1 \wedge m_2 < \theta_2 \text{ or } (ii) \quad m_2 = 0 \wedge m_1 \geq 1; \\ \quad \text{otherwise, it serves jobs from queue 2.} \end{cases}$$

Among these dynamic (or queue length dependent) rules $\mathbf{S}(\theta_1, \theta_2)$ by simulation a rule is sought which strictly improves the pooled scenario. An $\mathbf{S}$(Opt)-rule is determined that takes into account the waiting times of both job types by:

Step 1: Solve

$\min_{\theta_1,\theta_2} \max \{\mathbf{W}_1[S(\theta_1, \theta_2)], \mathbf{W}_2[S(\theta_1, \theta_2)]\}$

This leads to an optimal threshold combination $(\theta_1, \theta_2)^*$

and overall average waiting time under $(\theta_1, \theta_2)^*$: $\mathbf{W}_A[S(\theta_1, \theta_2)^*]$.

Step 2: Solve

$$\mathbf{W}_A[S(\theta_1, \theta_2)^{**}] = \min_{\theta_1,\theta_2} \mathbf{W}_A[S(\theta_1, \theta_2)] \quad \text{s.t.} \quad \max \{\mathbf{W}_1[S(\theta_1, \theta_2)], \mathbf{W}_2[S(\theta_1, \theta_2)]\} \leq \mathbf{W}_{Pooled}$$

Step 3: If $(\theta_1, \theta_2)^{**}$ exists, then

$$\mathbf{W}_A[S(\text{Opt})^{**}] = \mathbf{W}_A[S(\theta_1, \theta_2)^{**}],$$

otherwise

$$\mathbf{W}_A[S(\text{Opt})^{**}] = \mathbf{W}_A[S(\theta_1, \theta_2)^{*}].$$

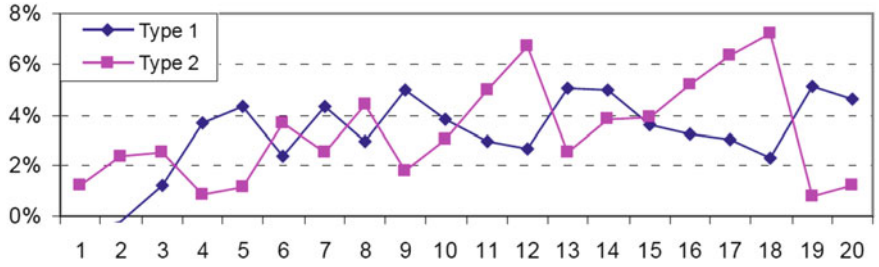

Fig. 4 Relative improvements over the pooled scenario

k \ s	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
2	(4,2)	(4,2)	(2,1)	(2,1)	(2,1)	(2,1)	(2,1)	(2,1)	(2,1)	(2,1)	(2,1)	(2,1)	(2,1)	(2,1)	(2,1)	(3,2)	(3,2)	(3,2)	(3,2)	(3,2)
3	(4,1)	(4,1)	(3,1)	(3,1)	(3,1)	(5,2)	(5,2)	(5,2)	(5,2)	(5,2)	(2,1)	(2,1)	(2,1)	(2,1)	(2,1)	(2,1)	(2,1)	(2,1)	(2,1)	(2,1)
4	(5,1)	(7,2)	(4,1)	(4,1)	(6,2)	(6,2)	(3,1)	(3,1)	(3,1)	(3,1)	(3,1)	(3,1)	(3,1)	(3,1)	(5,2)	(5,2)	(2,1)	(2,1)	(2,1)	(2,1)
5	(9,2)	(6,1)	(5,1)	(5,1)	(4,1)	(4,1)	(4,1)	(7,2)	(3,1)	(3,1)	(3,1)	(3,1)	(3,1)	(3,1)	(3,1)	(3,1)	(3,1)	(6,2)	(6,2)	(6,2)
6	(8,1)	(7,1)	(6,1)	(5,1)	(5,1)	(5,1)	(4,1)	(4,1)	(4,1)	(4,1)	(3,1)	(3,1)	(3,1)	(3,1)	(3,1)	(3,1)	(3,1)	(3,1)	(3,1)	(3,1)
7	(13,2)	(8,1)	(7,1)	(6,1)	(10,2)	(5,1)	(5,1)	(4,1)	(4,1)	(4,1)	(4,1)	(4,1)	(4,1)	(3,1)	(3,1)	(3,1)	(3,1)	(3,1)	(3,1)	(3,1)
8	(14,2)	(9,1)	(8,1)	(7,1)	(6,1)	(11,2)	(5,1)	(5,1)	(4,1)	(4,1)	(4,1)	(4,1)	(4,1)	(9,2)	(3,1)	(3,1)	(9,2)	(9,2)	(9,2)	(9,2)
9	(12,1)	(15,2)	(9,1)	(8,1)	(7,1)	(6,1)	(6,1)	(5,1)	(5,1)	(5,1)	(5,1)	(4,1)	(4,1)	(4,1)	(4,1)	(4,1)	(4,1)	(3,1)	(3,1)	(3,1)
10	(18,2)	(12,1)	(15,2)	(8,1)	(7,1)	(7,1)	(6,1)	(6,1)	(5,1)	(5,1)	(5,1)	(5,1)	(4,1)	(4,1)	(4,1)	(4,1)	(4,1)	(4,1)	(3,1)	(3,1)

(θ1, θ2)** solution (θ1, θ2)* solution (θ1, θ2)* solution, pooling average better

Fig. 5 Optimal threshold combinations $(\theta_1, \theta_2)^{**}$ or $(\theta_1, \theta_2)^{*}$ for strict improvements

The improvements are only in the order of a few % but consistently outside 95 % confidence intervals with a range of 0.5 %. Figure 4 shows the relative improvements (mean waiting time reduction) that can so be obtained for both type 1 and type 2 jobs over the pooling scenario for $k = 10$ and $s = 1$ up to 20.

Figure 5 lists optimal threshold combinations $(\theta_1, \theta_2)^{**}$ (if existing), for which the pooled scenario is improved allover, and optimal threshold combinations $(\theta_1, \theta_2)^{*}$ otherwise, for different values of s, mix ratios k and $\rho = 0.9$. It shows that $(\theta_1, \theta_2)^{**}$ does not always exist. For example, for $k = 10$ and $s = 2$, at least one of the two job types will always be worse than for the pooled case. However, for most (s, k)-values $(\theta_1, \theta_2)^{**}$ appears to exist.

Conclusion 3 By OR (queueing) insights and Optimization using Simulation a strict improvement over both short and long jobs might be feasible.

2.4 Summary of Combined Queueing and Simulation

To summarize this first application section of combined queueing and simulation, tailored to question of pooling and possible improvements and optimization, we may thus conclude that:

- Pooling is not necessarily optimal in all situations.
- Queueing insights appear to be essential.

- Simulation is required.
- Queueing insights and further optimization by simulation may lead to substantial and even strict improvements.

3 Blood Inventory Management

Blood management is of worldwide and generic concern. This includes the production (or rather acquisition of donors) and the inventory management of perishable blood platelets. No general and practical approach seems to be available. In this section, therefore, an integrated OR-simulation approach is provided.

3.1 Problem Motivation

Blood inventory management is a problem of general human interest with a number of concerns and complications. Our problem of interest will concentrate on the production and inventory management of blood platelets. Here there are a number of conflicting aspects. On the one hand, the demand is highly "uncertain" and apart from planned surgeries (if such information is used) roughly 50 % of the demand is unpredictable. Clearly, as lives may be at risk, shortages are to be minimized. On the other hand, the supply is voluntary, and also for ethical reasons blood has to be considered as highly precious. Any spill, by outdating, of blood (products) is thus highly "undesirable" if not to be avoided at all. As an extra complicating factor, blood platelets (thrombocytes) have a limited life time or rather "shelf life" of at most 6 days, this in contrast to red blood cells and plasma in all sorts of blood types that can be kept for months up to over a year. In addition, regular production of a platelet pool takes about 1 day. Hence production volumes should be set carefully. Another complicating factor is that part of the patients need the youngest platelets available, whereas other patients can be transfused with any platelets that do not exceed their shelf life of 5 or 6 days. Figure 6 shows the product of interest; to help one patient, platelets of five donors are needed.

3.2 Literature

The above perishable inventory management problem is studied in literature using various techniques. In the late 1960s and 1970s of last century, the problem is first analyzed by mathematical analysis of rather simple models that assume zero lead time, stationary demand, and that neglect the existence of different groups of patients, etc., see Nahmias [22], Prastacos [24]. More realistic studies use simulation models to gain insights in the performance of base stock policies, see o.a. Katsaliaki and Brailsford [18], Sirelson and Brodheim [25].

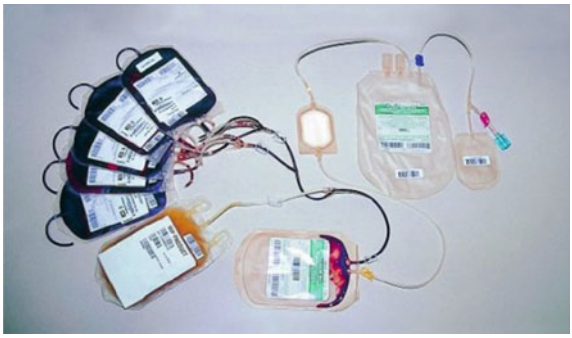

Fig. 6 A single platelet pool consists of platelets of five donors

Base stock policies set the order quantity equal to the difference between the actual stock level (or stock position) and a fixed-up-to level S. The value of parameter S is weekday dependent as the practical problem is nonstationary: Mean demand are weekday dependent and no production happens during the weekends. Optimizing the parameters of order policies is usually very time consuming as the for each day of the week an optimal parameter setting has to be found and these parameters are correlated. The number of combinations is often too large to apply enumerated search using simulation. A recent study that applies simulation based optimization using meta heuristics is Duan and Liao [12].

It is known that an optimal order policy should consider the ages of the products in stock, see Fries [13], Nahmias [22]. Base stock policies neglect stock ages but nevertheless they are commonly applied for being relatively easy to implement and to analyze. The optimality gap is hardly studied for realistic problem settings with positive lead time, nonstationary/periodic demand, and multiple types of patients who require different issuing policies. Main reason is the computational complexity involved in determining an optimal stockage-dependent policy Blake et al. [6]. This gap is investigated in the following papers: Haijema et al. [16, 17], Van Dijk et al. [32]. In these papers, optimal stockage-dependent order policies are derived and methods are presented that use simulation in combination with optimization to derive improved but simple ordering policies as well as a way for finding nearly optimal order-up-to-levels. In this chapter we summarize and integrate the findings of these studies.

3.3 Combined Optimization-Simulation Approach

In Haijema et al. [16] a combined approach for the blood platelet inventory problem has therefore been followed, which combines OR and simulation by the following steps:

Step 1: Optimization model: First, a stochastic dynamic programming (SDP) formulation is provided, which neglects the existence of blood types. This latter assumptions will be validated in Step 5.

Step 2: Optimal solution: The dimension of the (SDP) formulation is then reduced (downsized) by aggregating the state space and demands so that the downsized (SDP) problem can be solved numerically (using successive approximation). That is, the optimal value and an optimal strategy is determined for the downsized SDP.

Step 3: Simulation for investigation: Then, as essential tying step, this optimal policy is (re)evaluated and run by simulation in order to investigate the structure of the optimal strategy. In this simulation, one registers the frequency of (state, action)-pairs for the downsized problem.

Step 4: Simulation for re-optimization: The results of step 3 are used to derive practical order rules, like improved base stock policies and to obtain nearly optimal parameter values. By a heuristic search procedure parameter values of these rules are fine tuned for the full-size problem.

Step 5: Simulation for validation: The quality (near-to-optimality) of this practical simple order-up-to strategy is evaluated by detailed simulation. In this step it is also justified, for Dutch blood banks, that blood types are ignored in the previous steps.

As the technical (mathematical) details of steps 1 and 2 are somewhat 'standard' but also 'complicated' and worked out in detail in Haijema et al. [16] and related references, we present here a compact presentation of the essential OR and Simulation Steps. The results of Step 5 (validation by simulation) are reported for two cases in Sects. 3.4 and 3.5.

3.3.1 Steps 1 and 2 Optimization by SDP

To give an SDP formulation, the state of the system is described by (d, x) with

d: the day of the week $(d = 1, 2, \ldots, 7)$

and

$\mathbf{x} = (x_1, x_2, ., x_m)$ the inventory state

with x_r = the number of pools with a residual life time of r days (maximal $m = 6$ days) (A pool is one patient-transfusion unit containing the platelets of five different donations).

Let $V_n(d, x)$ represent the minimal expected costs over n days when starting in state (d, x). The optimal inventory strategy and production actions are then determined by iteratively computing (solving) the SDP-equations for $n = 1, 2, \ldots$

$$\mathbf{V}_n(d, \mathbf{x}) = \min_k \left[c(\mathbf{x}, k) = \sum_b p_d(b) \mathbf{V}_{n-1}(d, t(\mathbf{x}, k, b)) \right] \tag{1}$$

Table 4 Optimal productions by SDP for a selection of states

Production	Inventory (old . . . young)
7	(0, 0, 5, 0, 0, 9)
8	(0, 0, 6, 0, 0, 8)
9	(0, 0, 8, 0, 0, 6)
10	(0, 6, 2, 0, 0, 6)
10	(5, 0, 3, 0, 0, 6)

with

k	the production action,
$c(\mathbf{x}, k)$	the one day expected costs in state $\mathbf{x}$ under production k,
$pd(b)$	the probability for a (composite) demand b,
$t(\mathbf{x}, k, b)$	the new inventory state depending on k, b, $\mathbf{x}$, and some issuing policy,

and

$\mathbf{V}_0(d, \mathbf{x}) = 0$ to start up the iterative computations.

However, for a realistically sized problem for one of the Dutch regional blood banks the computational complexity of this SDP for a one-week iteration already becomes of an order 1014, which makes the computation times prohibitively large. Therefore, we have downsized the demands and inventory levels by aggregating the pools into quantities of four. This strongly reduces the computational complexity, so that an optimal strategy can be computed for this downsized problem by the optimizing actions of the SDP. However, in practice one needs a simple rule and this optimal strategy has no simple structure. See, for example, Table 4 which prescribes the production volumes on Tuesday for five different states, which all have the same total inventory level of 14 pools, but of varying ages.

3.3.2 Steps 3 and 4 Simulation for Investigation and Re-optimization

In order to derive a simple order-up-to strategy which only depends on the total predicted inventory, the actual platelet production-inventory process is therefore simulated for 100,000 replications so as to register how often which total predicted final inventory level (I) and corresponding action occurs under the optimal strategy (as determined by SDP) for the downsized problem. As an illustration, for a particular day of the week and the dataset of the regional blood bank, this led to the "simulation table" in Fig. 7. For example, it shows by row 15 and column 7 that during the 100,000 replications 2593 times a state was visited with a total final inventory (I) of 7 followed by a production decision of 8 (order-up-to 15). Order-up-to-level 15 occurs in 74.5 % of the states visited, however, often a higher production is optimal. The order-up-to level can be seen as a target-inventory level for Wednesday mornings.

I	2	3	4	5	6	7	8	9	10	11	12	13	14	cum.
Order-up-to														
23													4	4
22												28		28
21											96			96
20										267				267
19								2	748	3				753
18							18	1928	31	1				1978
17						6331	4490	353	26	1				11201
16					8260	2078	783	7						11128
15	3131	14123	20926	23646	10087	2593	39							74545
:														
0														
cum.	3131	14123	20926	23646	18347	11002	5330	2290	805	272	96	28	4	100000

Fig. 7 Simulation frequency table of (State, Action)-pairs for tuesdays from simulation of optimal SDP solution for 100,000 weeks

We conclude that a simple order-up-to rule might perform well. By investigating the states at which the optimal production volume is higher we have derived even better rules that closely resembles the optimal production strategy. For example, a base stock policy that first estimate the quantity that is left upon replenishment is doing better as it compensates for estimated waste during the lead time. Such a policy is called in Haijema et al. [16], the final stock rule. Another improved policy applies two order-up-to levels, one for the demand for young platelets, and one for the total demand. Both these improved policies are discovered and tested by simulation of the optimal stockage-dependent policy.

3.3.3 Step 5 Validation by Simulation

The results of Step 5 are reported for two real-life applications that differ in their motivation. Application 1 was selected for validation of the method with the premier objective of reducing waste. In Application 2 the focus is on applying the method such that one issues younger product while maintaining low levels of waste and a high product availability. The Netherlands can be divided in four regions at which blood is collected and processed, see Fig. 8. For Application 1, region North-East is considered, for Application 2 region South-East is selected.

Fig. 8 The Dutch blood banks divide The Netherlands in four regions

3.4 Application 1: Spill Reduction at Dutch Blood Bank North-East

3.4.1 Main Results

Applying this combined approach to data from the Dutch regional Blood Bank North-East, the following conclusions could be drawn

1. A simple order-up-to rule could reduce the spill from roughly 15–20 %, as a figure that also seems rather standard worldwide, to $<\mathbf{1}$ % (while also shortages were reduced and nearly vanished).
2. The combined SDP-Simulation approach led to an accuracy within **1** % of the exact optimal value for the downsized problem.

Detailed data and results are discussed below.

3.4.2 Problem Data Dutch Blood Bank North-East

The maximal shelf life of a platelet pools is five days counted from the first morning that platelet pools are released to the stock located at the blood bank. The demand for platelet pools is Poisson distributed with means as reported in Table 5.

The demand for young prefers products of at most 3-days old. The any-age demand can be met by any pools of at most 5-days old. Falling short one pool is considered to be five times as severe as wasting one platelet pool. This is a managerial trade-off

Table 5 Means of Poisson demands per weekday and per type of demand

Demand	Mon	Tue	Wed	Thu	Fri	Sat	Sun	Weekly
'Young'	20	15	26	15	20	0	0	96
'Any-age'	6	6	6	6	6	8	10	48
Total	26	21	32	21	26	8	10	144

that is reflected in a penalty costs of 150 € for spilling one pool, and 750 € for falling short one pool. Inventory costs are estimated to be only 0.1 € per day per pool. Meeting the demand for young by products with a residual shelf life of only 2 or less days is penalized by a cost of 200 per pool. The objective is to minimize the sum of these costs.

3.4.3 Results Dutch Blood Bank North-East

An optimal stockage-dependent policy can be obtained by SDP but only after scaling the demand figures as if demand happens in multiples of 4 pools (=Step 2). The resulting policy is simulated to investigate its structure (=Step 3). The result of Steps 3 and 4 are five more rules, which neglect the age of the products in stock while setting an order quantity. The performance of these rules are compared in Table 6, using the scaled or downsized demand distributions. Clearly stockage-dependent ordering (SDP/MDP) gives lowest annual costs. A *fixed-order quantity* orders every weekday a fixed weekday dependent quantity and is clearly far from optimal. The *Order-up-to S*, which is a base stock policy, provides annual costs that are 9.9 % above the optimal cost level. For the scaled problem, the weekday dependent order-up-to levels are multiple of 4; for Monday to Friday we get $S = (64, 72, 72, 64, 80)$. The new policy *Bounded Order-up-to S*, adds a minimum and a maximum order quantity to order-up-to S policy. The effect of these bounds are that are quantities are more smooth, which results in lower annual costs, primarily due to generating less mismatches, i.e., it happens less frequently that old pools are used to meet the

Table 6 Performance of optimal policy and derived rules

Rule	Outdating[a]		Shortage[a]		Mismatch[b]		Annual costs	
MDP-optimal	37	1.95 %	4.9	0.26 %	0.09	0.01 %	36,605	–
Fixed-order-quantity	157	8.41 %	1.4	0.07 %	0.00	0.00 %	98,459	+168.9 %[c]
Order-up-to S rule	36	1.95 %	5.7	0.30 %	2.17	0.17 %	40,236	+9.9 %[c]
Bounded order-up-to S	36	1.95 %	5.6	0.30 %	0.59	0.05 %	39,077	+6.8 %[c]
2D-order-up-to rule	36	1.92 %	5.2	0.28 %	2.16	0.17 %	38,779	+5.9 %[c]
Final stock rule	36	1.91 %	5.1	0.27 %	1.98	0.16 %	38,389	+4.9 %[c]

[a] In batches of 4 pools per year; % of total (1872 batches = 7488 pools)
[b] In batches of 4 pools per year; % of young-demand (1248 batches = 4992 pools)
[c] Percentage above optimal cost level (MDP)

demand for young pools. The *2D-rule*, which has both an order-up-to level for young and for total stock, and the *Final stock rule* show further cost reductions. The best policy is still about 5 % above the minimal costs level achieved by MDP-optimal.

3.4.4 Re-optimization and Validation

For the validation, we restrict ourselves to the order-up-to S policy as it is commonly used. The parameters are reoptimized by local search and simulation with the nonscaled demand distributions resulting in $S = (65, 74, 80, 64, 82)$. Validation happens in a more detailed simulation program that takes into account the blood type of both patients and donors. Donors are selected mainly form the category O and A for being the most compatible donor, see Fig. 9. The percentages indicate that the two blood types cover 89 % of the population. Most donor provide full blood donations from which three types of blood products are made: Red blood cell concentrates, plasma products, and platelet concentrates. As for the production of platelet pools only a third to a half of the donations is needed, one usually has enough platelets available of the most compatible blood types O and A. In total, we consider eight blood types by combining O, A, B, and AB with the Rhesus-D factor.

Table 7 report estimates of annual figures by two simulation models: The multi-group model simulates patients and donors of eight different blood types; the Universal-group model simulates as if all donors and patients have identical or fully compatible blood types. The result in annual performance is virtually the same, as blood of the universal blood group O is plenty available.

If instead of 33 % of the available blood is sued for platelet production, 50 % or even 67 % is used more of other blood types is used for production. This is demonstrated in Table 8. The annual production stays the same but if blood is more scarce, more of the less compatible blood type B is produced which is no problem if demand is met choosing pools of the least favorite but compatible blood type first.

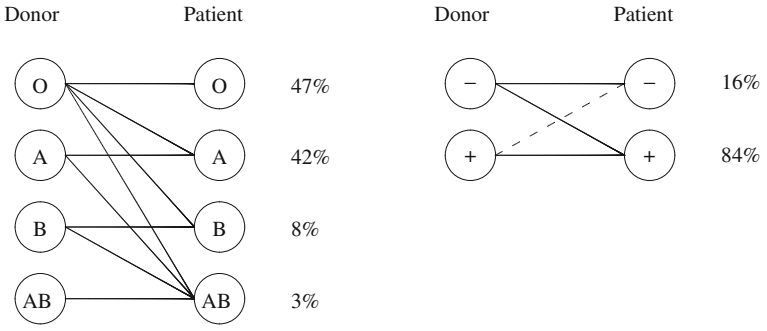

Fig. 9 Compatibility of blood types and Rhesus-D factor

Table 7 Impact of blood types on performance of order-up-to S (simulation for 100 million weeks)

Criterion	Multigroup model		Universal-group model	
	In pools	Relative (%)	In pools	Relative (%)
Production	7597		7597	
Outdating	143	1.9	142	1.9
Shortage	33	0.4	33	0.4
Quality mismatch	9	0.2	9	0.2
Annual costs	48,168		47,726	

Table 8 Production per blood group under the rescaled order-up-to S rule (over 100 simulation runs of 1 million weeks each) when on average a third, a half or two third of the whole blood donations (WBD) is used for platelet production

Scenario	Total	O^-	O^+	A^-	A^+	B^-	B^+	AB^-	AB^+
33 % of WBD used	7597								
Annual production		1582	5944	62	10	0	0	0	0
% of total production		20.8 %	78.2 %	0.8 %	0.1 %	0 %	0 %	0 %	0 %
50 % of WBD used	7597								
Annual production		1018	5268	525	786	0	1	0	0
% of total production		13.4 %	69.3 %	6.9 %	10.3 %	0 %	0 %	0 %	0 %
67 % of WBD used	7597								
Annual production		739	4201	560	1991	8	86	0	14
% of total production		9.7 %	55.3 %	7.4 %	26.2 %	0.1 %	1.1 %	0.0 %	0.2 %

3.5 Application 2: Age Reduction at Dutch Blood Bank South-East

As donated, platelets tend to clutter and the number of effective platelets within a pool decreases as time evolves; the quality of a platelet pool is directly related to its age when its transfusion takes place. Therefore, besides shortages and outdating, there is a third quality factor.

The issuing age of the platelets.

This quality factor is most important for treatment of special patients (oncology and hematology) which constitute roughly 40 % of all demand. Clearly, the SDP by itself does not take the age into account. By simulation, in contrast, issuing ages can easily be kept track of. In Kortbeek et al. [19] and Haijema et al. [16], therefore, the SDP-simulation approach was extended so that also the quality aspect of the issuing age is addressed. The extension was applied to a new Dutch Blood Bank study, the Dutch Blood Bank South-East, with a (meanwhile) extended maximal shelf life of 6 days. Below, several strategies are presented that improve the age of the platelets issued, such as by slightly relaxing the shortage performance, by introducing penalty costs for older issues or by a special "weekend" production. The results were obtained by successfully exploiting the strengths of

Table 9 Performance of base case South-East

Performance indicator	Base case
Shortage	0.04 % (in days 0.13 %)
Outdating	0.25 %
Average age	3.75
Age distribution	(2.5; 17.2; 19.1; 30.8; 25.1; 5.3) %

- SDP for optimization, and
- Simulation for evaluation.

In the first study only two cost elements are used, outdating and shortage cost. Shortage costs are taken to be five times as high as the outdating costs. In this base variant, there are no penalty costs with respect to age. The issuing policy is FIFO, that is, the oldest platelets are issued first. At Saturday there is a limited production capacity of 20 pools, and at Sunday there is no production at all.

Table 9 shows that the shortage and outdating figures are excellent. The issuing age of the platelets, however, is fairly high, with more than 30 % of the platelets being issued at shelf ages 5 and 6 days and with an average age of 3.75 days.

At this point, the SDP-Simulation approach is exploited in order to explore scenarios and strategies which can improve the issuing age. An obvious first attempt is to use the LIFO (Last In First Out) issuing policy instead of FIFO, so to always issue the youngest platelets in stock. Although the average issuing age improves to 2.26, the price with respect to outdating (11.0 %) and shortages (1.87 %) is very high. Therefore it was concluded that LIFO is not the solution. A second alternative is to allow the number of days with shortages to be relaxed to about 1 % (recall this percentage was 0.13 for the base case). 1 % amounts to 3 to 4 days per year and is considered to be acceptable by the Blood Bank. By allowing more shortages one will keep fewer inventories, which might result in issuing younger platelets. In order to find a nearly optimal order-up-to strategy giving 1 % shortage days, the ratio between shortages and outdating costs is decreased. The results are displayed in Table 10.

There is a considerable improvement of the age distribution, while outdating has become virtually zero. Only 14 % of the issued platelets is of shelf age 5 and 6 days, compared to 30 % for the base case and the average age has been decreased from 3.75 to 3.18. In the base case, the maximal shelf life is 6 days. But what happens if one decides not to use platelets of age 6 days, so that platelets become outdated at the end of day 5 or even at the end of day 4? It can be expected to lead to lower order-up-to levels, so more shortages but issuing younger platelets. With respect to the SDP this results in changes in the state space, the expected costs and the transition probabilities. Using the same cost structure as in the base case one obtains the results in Table 11. The results for a shelf life of 5 days for shortages and outdating are quite good. The percentage of the platelets issued at shelf age 5 is about 16 % and the average age of the platelets issued is 3.2 days. Although for 4 days the age distribution improves considerable, the increase in outdating makes this solution unacceptable.

Table 10 Performance South-East, when shortages are tuned to 1 % of days

Performance indicator	Shortage are tuned to 1 %
Shortage	0.35 % (in days 0.95 %)
Outdating	0.02 %
Average age	3.18
Age distribution	(8.4; 25.9; 20.3; 31.2; 13.1; 1.1) %

Table 11 Performance South-East, when shelf life is reduced

Performance indicator	5 days	4 days
Shortage	0.24 % (in days 0.66 %)	0.73 % (in days 1.88 %)
Outdating	1.22 %	5.38 %
Average age	3.23	2.70
Age distribution	(6.8; 25.8; 20.2; 31.5; 15.6, −) %	(15.9; 29.5; 23.0; 31.6, −, −) %

Another possibility is to discourage the issuing of older platelets by penalization in the cost function of the SDP. This penalty is taken to be half the outdating costs. (It is important to note that the more cost parameters are used, the more difficult it is to quantify them in such a way that the effect one is aiming for is indeed achieved.) Compared to the base case, the only change in the SDP is in the costs. Two cases are considered: In the first case day 4, 5, and 6 are discouraged, and in the second case day 5 and 6. The results are displayed in Table 12.

Both cases show almost equal results. As expected, shortages have increased, but the average age went down to 3.12, and issues of day 5 and 6 halved compared to the base case. The final scenario studies a combination of successful scenarios: a shelf life of 5 days, a penalization for issuing platelets of age 5, and shortages about 1 % in days. The results are displayed in Table 13.

The proposed combination appears to be a very satisfactory improvement, with shortages in the order of 1 % in days, outdating just below 1 %, the average age reduced from 3.75 to 3.06 and only 11.0 % issued at age 5.

Table 12 Performance South-East, when discouraging issuance of older pools

Performance indicator	Penalize day 4, 5 and 6	Penalize day 5 and 6
Shortage	0.30 % (in days 0.95 %)	0.33 % (in days 1.11 %)
Outdating	0.04 %	0.08 %
Average age	3.12	3.10
Age distribution	(9.5; 26.0; 23.1; 27.6; 12.4, 1.5) %	(9.9; 26.7; 23.5; 25.1, 12.7, 2.0) %

Table 13 Performance South-East, when combining scenarios

Performance indicator	Combination
Shortage	0.36 % (in days 1.04 %)
Outdating	0.92 %
Average age	3.06
Age distribution	(9.0; 27.0; 24.0; 29.0; 11.0; −) %

3.6 Summary Blood Inventory Management

To summarize this section it can thus be concluded that the (perishable inventory) problem is so complex that it is impossible to obtain practical results by only SDP or only by Simulation. However, substantial and practical improvements could be obtained (and have real life been implemented (see Kort et al. [11], Van Dijk et al. [32]) by their combination.

4 Rail-Track Scheduling

Railroad scheduling is highly complex as punctual and detailed scheduling at minute basis is confronted with stochastic disruptions on the other. Simple and practical rules are then required. These in turn are highly situation dependent, including time-tabling, frequencies up to (country) infrastructures. No general stochastic appears to be available approach other than simulation. In this section, again an integrated OR-simulation approach is suggested, as based upon Stochastic (Semi-Markovian) Dynamic Programming (SDP).

4.1 Motivation

An example of yet another class of stochastic decision problems for which a combined SDP-simulation approach seems most fruitful is found in rail-track scheduling. In the Netherlands, tracks are heavily used. This implies that these tracks have to be used in an intelligent way. As an example, consider the junction as depicted in Fig. 10. If two or more trains enter the junction more or less simultaneously it has to be decided which train is admitted first. To a certain extent the basis of this decision problem is deterministic but in practice it is also highly stochastic due to stochastic arrival times, delays and speed differences. Accordingly, the problem of online dynamic conflict resolution has the flavor of both a scheduling and a queueing problem.

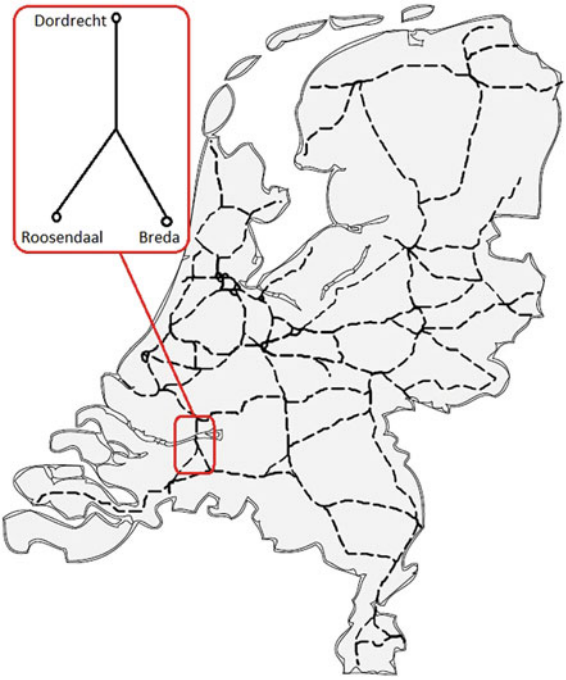

Fig. 10 The railway infrastructure of The Netherlands with selected junction for the test case

4.2 Literature

Railway scheduling problems have been extensively studied in the literature and are known to be NP-hard (Garey and Johnson [15]). Excellent overviews are given in Assad [4], Cordeau et al. [9], Törnquist [28] and D'Ariano [10]. In the overview paper Törnquist [28], the relevant literature is classified into three main categories: Tactical scheduling, operational scheduling, and rescheduling. While the Tactical and operational scheduling involved constructing the timetable from scratch, rescheduling is done when train conflicts arise due to perturbations. The online dynamic conflict resolution falls in the category of Rescheduling.

Very little literature exists on Rescheduling. The issue has been addressed only recently due to the complex nature of the problem and the very limited available computational time. The different approaches that are described in the literature minimize delay propagation by setting the train order at crossing points. Amongst these approaches is the model proposed in Adenso-Dıaz et al. [1]. The authors describe the online conflict resolution problem as a mixed integer programming model and state that solving this problem by means of the Branch-and-Bound technique is very time consuming. Instead, the authors propose a heuristic approach that intelligently reduces the search space by elimination of certain branches that are considered to be inferior. The approach is implemented at the Spanish national railway company where the tool preselects the best resolution rules and presents them to a train dispatcher.

Tornquist and Persson [29] propose a two level procedure to resolve train conflicts. The authors suggest an approximation strategy, which in most cases does well with respect to computational time and solution quality. Araya et al. [3] formulate the online scheduling problem as a 0–1 mixed integer programming problem which is solved in two steps. First, a suboptimal solution is obtained by a heuristic approach. The branch-and-bound approach is then used to find the optimal solution. A number of experiments show the efficiency of the approach in terms of computational time. Another approach is to formulate the train conflict problem as a Job-Shop problem. Here, the trains are jobs and the tracks are machines. The problem is then to find the best assignment of the trains to the tracks so that the overall delay (or some other optimization function) is minimized. Mascis and Pacciarelli [21] introduce blocking and no-wait constraints to the Job-Shop scheduling problem and use an 'Alternative graph' to solve it.

These heuristics, however, do not guarantee the optimality of the solution. Moreover they do not account for future uncertainties, such as stochastic train arrivals. In the next section, a stochastic approach is discussed. These approaches attempt to model uncertainties which are found in the real world (think of the running times, dwell times and other operations which are often stochastic).

4.3 Combined Simulation and Optimization Approach

4.3.1 OR-Approach: Optimization

This track conflict problem can partially be regarded as a 'standard' OR-scheduling problem, more precisely, as a job-shop problem with blocking. By considering trains as jobs and tracks as machines, an 'optimal train order' for a track can be found by branch-and-bound techniques. It is a job-shop problem with blocking because an occupied track section blocks a successive train to enter that section. Trains at the preceding section can thus be delayed. The usual job-shop formulation, however, uses fixed handling times without delays and variability's.

4.3.2 Simulation Approach

As delay aspects and the variability of travel times are crucial for the track conflict, a stochastic approach might be able to cover more aspects of the problem. Simulation would thus be in place, despite the fact that it does not optimize at all. Indeed, in earlier literature (see references in Al-Ibrahim [2]) simulation is used to analyze a junction. In those studies, train are assigned by dynamic priorities. The dynamic priority can be a function of the train type, its experienced delay, the delay caused by acceleration and possible other conflicts. However, optimization is not involved.

4.3.3 Combined Approach

In Al-Ibrahim [2], therefore, a more extended combination of OR and simulation is suggested. To include both queueing (time) and scheduling (optimization) aspects a Semi-Markov Decision Process (SMDP) is formulated.

4.3.4 SMDP-Formulation

The Semi-Markov Dynamic Programming (SMDP) formulation for the stochastic junction-track scheduling problem essentially takes into account the stochastic nature and different durations of transitions. It has the form:

$$V_{n+1}(A,v,d) = \min_k \left\{ \begin{array}{l} c(A,v,d) + \\ (\tau/\tau^k(A,v,d)) \sum_{(A',v',d')} P^k\left[(A,v,d);(A',v',d')\right] V_n(A',v',d') \\ + \left[1 - \tau/\tau^k(A,v,d)\right] V_{n+1}(A,v,d) \end{array} \right\} \tag{2}$$

where a state (A, v, d) represents a state of the form:

$(A, v, d) = (A_1, v_1; A_2, v_2; d_1, d_2, \ldots, d_N)$
with
A_l denoting the trains in queue $l \in \{1, 2\}$
v_l indicating whether the trains are moving ($v_l = 0$) or not ($v_l = 1$)
d_j the train type which is occupying the jth position past the junction.

The costs $c(A, v, d)$ cover the time that all trains together are spending in the subnetwork up to a next transition. Further

$P^k\left[(A, v, d); (A', v', d')\right]$ represents the transition probability from a state (A, v, d) into (A', v', d')

$\tau^k(A, v, d)$ is the average duration of a transition in state (A, v, d), when decision k is taken.

4.3.5 Simulation-SMDP Approach

A combined approach can now be suggested, which combines simulation with the SMDP optimization algorithm in a number of steps, as briefly outlined below.

Step 1: (SMDP optimization) For the junction under consideration, a semi-Markovian decision process is formulated and solved. (As shown above and argued in more detail in Al-Ibrahim [2]). For every possible, state an optimal decision is registered.

Step 2: (Simulation) Trains are generated for the junction subnetwork according to a global train schedule but with a number of stochastic elements to include initial randomness and speed differences. The trains are simulated until a conflict is detected. The simulation run is interrupted and the conflict is registered.

Step 3: (Finding the optimal SMDP decision) The train conflict situation is mapped on to a state of the SMDP model. Then the corresponding optimal decision is read and communicated to the simulation. The simulation implements this decision and the simulation continues until the next conflict occurs.

Step 4: The delays, as they occurred in the simulation with the optimal SMDP-decisions, are registered.

Step 5: Comparison. Simulation is used to also obtain the performance of a number of other heuristical rules to settle the conflicts.

Step 6: Results. The SMDP results as well as the results for the heuristics are reported.

In short, simulation is used to capture queueing, to generate conflicts and to evaluate decisions made while SMDP is used to determine the (within the model) optimal train order.

4.4 Application Results

4.4.1 Application 1: Junction Case

In cooperation with "ProRail" (the Dutch Railway operator) the approach has first been applied to a small but complicating and generic junction within The Netherlands. The junction has 12 arriving trains per hour, 6 fast passenger intercity trains (IC) and 6 slow freight trains (FR); half of the trains on each one of the arriving tracks. After the junction there are 5 positions (which reflect a distance of more than 13 km). The FR trains need 170 s to accelerate from speed 0 to speed 80 km/h, while the IC trains only need 30 s to reach the speed of 120 km/h.

To verify that the combined SMDP-simulation approach outperforms simple practical rules like the FCFS (First Come First Served) rule or a strict priority rule for passenger or for freight trains, the approach is compared with these rules by simulation. Table 14 shows the results. The values are average delays per train type over 12 days at 15 h a day. The results show that the SMDP-simulation approach almost captures the quality for passenger trains as by strictly prioritizing passenger trains and for freight trains as by strictly prioritizing freight trains.

4.4.2 Application 2: A Network Case

Next, in close cooperation with the department of "Traffic Control" of the Dutch Railway operator ProRail the approach has been applied to a more complicated network structure as shown in Fig. 11, called the corridor "Utrecht–Gouda." This is a heavily utilized corridor with frequent train conflicts. Presently these conflicts are resolved by ProRail according to so-called TAD rules. (TAD is the Dutch acronym

Table 14 Results by simulation and the SMDP-simulation rule for the FCFS, IC-FR (priority to passenger trains), and FR-IC (priority to freight trains)

12 trains per hour				
Discipline	Delay IC (s)	Delay FR (s)	Avr delay (s)	Number conflicts (per hour)
FCFS	182	86	134	2.6
IC-FR	164	109	137	2.6
FR-IC	175	48	111	2.3
SMDP	162	51	106	2.2

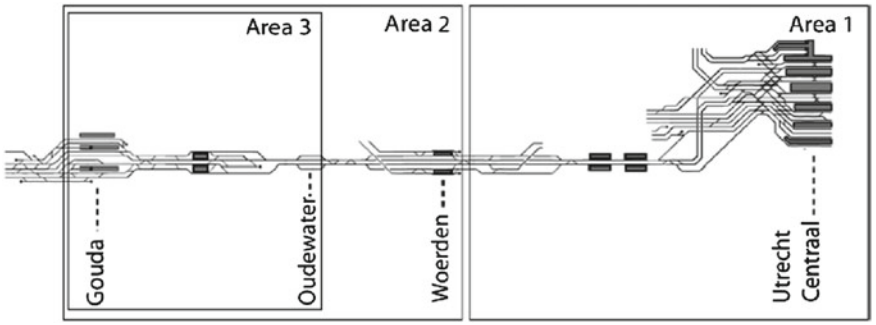

Fig. 11 Corridor Utrecht–Gouda (We here confine ourselves to the trains running from Utrecht to Gouda and do not consider the opposite direction)

for train order document.) These rules are computed offline and prescribe the train order in case a conflict arises. Table 15 shows an example of such a rule.

It is stated that train service 4000 is scheduled to be the first one to run toward the Gouda station (Gd) followed by train services 2000, 2800, and a freight train, if its delay is less than 6 min. If the train service 4000 has a delay between 6 and 10 min, the TAD rule prescribes that the train should let train services 2000, 2800 and the freight train go first.

For the corridor Utrecht–Gouda the TAD rules give unsatisfactory results. ProRail, therefore, was searching for alternative rules that improve the train punctuality for this corridor. Figure 11 shows the corridor in more detail and indicates the three areas, which in our approach will be considered separately.

Table 15 Example of a TAD rule

Train	To	Arrival time	Minimum delay	Maximum delay	Train order
4000	Gd	−0.03/−0.33	0	6	4000 − 2000 − 2800 − FR
4000	Gd	−0.03/−0.33	6	10	2000 − 2800 − FR − 4000

After inspecting the corridor and the specific elements that play a role at each conflict location, we concluded that there are three different areas where conflicts occur and where a resolution rule is needed. In Area 1, the trains leave the Utrecht station toward Gouda. When a conflict arises, one needs to establish an optimal departure order based on some optimization criterion. In Area 2, at Woerden station, there is a double track over a distance of 8 km which makes it possible for fast trains to overtake slower trains without delaying them much. Here, one needs to know if, and when, a fast train may overtake a slower one. Finally in Area 3, at the place called Oudewater, it is possible to stop a freight train so that a passenger train can overtake it. The rule here should prescribe when it is optimal to stop a freight train in favor of a passenger train. Solving the SMD model, described in the previous section, yields the so-called SMD strategy which decides about the order of the trains for each area separately. Just like the TAD strategy the SMD strategy is local and computed offline. However, while the TAD strategy assumes that only one train is delayed at a time, the SMD strategy prescribes conflict resolution rules for all possible situations. In its computation, it does not only consider trains in the direct proximity of a conflict area but it also includes information about (random) future arrivals.

The performance of the SMD strategy is compared to the performance of the TAD rules and some other simple heuristics. For this comparison simulation is necessarily required.

Within the simulation, we have applied the timetable of the year 2007 and used disturbances which are comparable to the ones recorded in 2007. By means of the "common random number" technique the different strategies are confronted one by one to the same set of events so that the differences in performance are solely related to the strategies themselves and not to the random nature of the simulation process.

Table 16 shows the punctuality in percentages at the three stations Utrecht (Ut), Woerden (Wd) and Gouda (Gd) for the different strategies. Here, a train is called "punctual" if the delay is less than 3 min. Each value represents the punctuality averaged over all trains and different train services that cross that station. Upon departure still 92 % of the trains is "punctual". As one sees, due to conflicts within the corridor,

Table 16 Punctuality (in %) of Utrecht–Gouda trajectory

Discipline	Ut	Wd	Gd
TAD	92	86	72
SMD	92	88	82
FCFS	92	85	70
IC-IR-RE-FR	92	89	83
IC-FR-IR-RE	92	89	82
FR-IC-IR-RE	92	89	80
FR-RE-IR-IC	92	86	74
RE-IR-IC-FR	92	87	78
LeastDelayedFirst	92	86	81
MostDelayedFirst	92	87	73

the punctuality decreases toward the end of the corridor. Some strategies resolve the conflicts in a more beneficial way, which translates into a higher punctuality at the end of the line. The FCFS strategy turns out to be the worst strategy. The TAD strategy improves FCFS strategy, but only a bit. The strategy MostDelayedFirst tries to minimize the delay for the most delayed trains by giving them priority over other trains, which however leads to poor overall results. Giving priority to the least delayed train turns out to be a better solution. From the train type priority rules IC-IR-RE-FR turns out to perform very well. This rule gives Intercity trains (IC) priority over all other train types. The Inter Regional (IR) trains have the second highest priority then come Regional trains (RE) and Freight trains (FR) have no priority at all. The SMD strategy improves the punctuality of the TAD strategy by 10 % points and is among the best performing strategies. When considering different scenarios, changing the percentage of freight trains or the total amount of trains, we found that the performance of the simple priority rules was quite sensitive to the number and the mix of the trains. The strategies that performed well in one case were not performing well at all in other cases. The SMD strategy performed very well in all cases, which encourages us to apply this approach to other corridors.

4.5 Summary of Rail-Track Scheduling

Summarizing the results of this section, first of all we note that there is no other way to evaluate the different train scheduling rules than by simulation. We also note that even for experienced and intelligent train schedulers, it is impossible to generate and compare all strategies. The SMDP-algorithm though, in principle computes presumably optimal decisions. In practice, these decisions can be overruled in the light of other information and expertise of schedulers, which cannot be included in the SDMP-simulation model. Nevertheless, the combined SMDP-simulation approach appeared to provide a valuable tool to support practical train scheduling.

5 Evaluation

OR (Operations Research) is well known for its value of mathematical optimization. Most famous applications considered in OR are the shortest routing problems (in route planning systems), and standard inventory optimization (e.g., by deterministic EOQ formulas). Generally, however, stochastics is involved to model uncertainty. Here the value of OR seems less famous, although simple insights and formulas from queueing theory (Q) and stochastic inventory theory (on replenishment policies) are available, next to techniques for stochastic optimization such as Markov decision theory (MDP).

In contrast, in practice OR techniques are often perceived as being too complex to apply and OR models are too simple by relying on strong underlying assumptions such as exponential distributed process times. Simulation, in particular discrete event simulation, then naturally comes as a manageable and practical tool for evaluation and

search-based optimization with virtually no restriction on either practical complexity or stochastic assumptions. The use of OR results, particularly of stochastic nature, generally seems to be skipped.

This chapter aimed to promote that even in that mathematically unsolvable practice, results from OR could still be most useful in combination with simulation, in either of two ways

(i) To provide insights so as to assist the simulation steps in search for optimization.
(ii) To provide an optimization formulation and a technique for its computational feasibility as well as its evaluation by using simulation.

The three applications discussed in this chapter, are real-life applications but their description is far from complete. These examples simply show that each practical problem description requires a tailor-made solution, for which both an OR formulation and the way it is to be integrated with simulation are to made specific and practical.

Simulation engineers might regard OR as too restricted for real-life scale and complexity. OR practitioners in contrast might regard simulation as insufficiently formally supported and specific. This chapter aimed to illustrate the opposite. That one could well benefit from the other. At practical call center scale by queueing insights and by simulation for practical improvements. For blood banks and railways, a theoretical OR solution technique for solvable systems used by simulation for expansion to simple practical rules. Accordingly, a combination appears to be highly mutually beneficial. Beyond these and other applications by themselves, this combination also seems of future research interest such as to integrate simulation for the nearly open problem of transient queueing applications on the one hand and to support simulation search approaches by OR on the other.

References

1. B. Adenso-Dıaz, M. Oliva González, and P. González-Torre. On-line timetable re-scheduling in regional train services. *Transportation Research Part B: Methodological*, 33(6):387–398, 1999.
2. A. Al-Ibrahim. *Dynamic Traffic Management for Railway Networks: A Semi Markovian Decision approach for train conflict resolution*. PhD thesis, University of Amsterdam, the Netherlands (Supervisor J.van der Wal), 2010.
3. S. Araya, K. Abe, and K. Fukumori. An optimal rescheduling for online train traffic control in disturbed situations. In *Decision and Control*, 1983. *The 22nd IEEE Conference on*, volume 22, pages 489–494. IEEE, 1983.
4. A.A. Assad. Models for rail transportation. *Transportation Research Part A: General*, 14(3):205–220, 1980.
5. S.L. Bell and R.J. Williams. Dynamic scheduling of a system with two parallel servers in heavy traffic with resource pooling: asymptotic optimality of a threshold policy. *The Annals of Applied Probability*, 11(3):608–649, 2001.
6. J.T. Blake, S. Thompson, S. Smith, D. Anderson, R. Arellana, and D. Bernard. Optimizing the platelet supply chain in nova scotia. In *Proceedings of the 29th meeting of the European Working*

Group on Operational Research Applied to Health Services (ORAHS). Prague: European Working Group on Operational Research Applied to Health Services, pages 47–66, 2003.
7. S. Borst, A. Mandelbaum, and M.I. Reiman. Dimensioning large call centers. *Operations research*, 52(1):17–34, 2004.
8. K. Cattani and G.M. Schmidt. The pooling principle. *INFORMS Transactions on Education*, 5(2):17–24, 2005.
9. J.-F. Cordeau, P. Toth, and D. Vigo. A survey of optimization models for train routing and scheduling. *Transportation science*, 32(4):380–404, 1998.
10. A. D'Ariano. *Improving real-time train dispatching: models, algorithms and applications*. Number T2008/6. Netherlands TRAIL Research School, 2008.
11. W. de Kort, M. Janssen, N. Kortbeek, N. Jansen, J. van der Wal, and N. van Dijk. Platelet pool inventory management: theory meets practice. *Transfusion*, 51(11):2295–2303, 2011.
12. Q. Duan and T.W. Liao. Optimization of blood supply chain with shortened shelf lives and abo compatibility. *International Journal of Production Economics*, 153:113–129, 2014.
13. B.E. Fries. Optimal ordering policy for a perishable commodity with fixed lifetime. *Operations Research*, 23(1):46–61, 1975.
14. N. Gans and Y.-P. Zhou. Call-routing schemes for call-center outsourcing. *Manufacturing & Service Operations Management*, 9(1):33–50, 2007.
15. M.R. Garey and D.S. Johnson. Computers and intractability: a guide to the theory of np-hardness, 1979.
16. R. Haijema, J. van der Wal, and N.M. van Dijk. Blood platelet production: Optimization by dynamic programming and simulation. *Computers & Operations Research*, 34(3):760–779, 2007.
17. R. Haijema, N.M. van Dijk, J. van der Wal, and C.Th. Smit Sibinga. Blood platelet production with breaks: optimization by sdp and simulation. *International Journal of Production Economics*, 121(2):464–473, 2009.
18. K. Katsaliaki and S.C. Brailsford. Using simulation to improve the blood supply chain. *Journal of the Operational Research Society*, 58(2):219–227, 2007.
19. N. Kortbeek, J. van der Wal, N.M. van Dijk, R. Haijema, and W. de Kort. Blood production and issuing optimization: Strategies for younger platelets. *UvA - research report*, 2008.
20. W. Krug. *Modelling, Simulation and Optimisation: For Manufacturing, Organisational and Logistical Processes*. SCS-European Publishing House, 2002.
21. A. Mascis and D. Pacciarelli. Job-shop scheduling with blocking and no-wait constraints. *European Journal of Operational Research*, 143(3):498–517, 2002.
22. S. Nahmias. Perishable inventory theory: A review. *Operations Research*, 30(4):680–708, 1982.
23. T. Osogami, M. Harchol-Balter, A. Scheller-Wolf, and L. Zhang. Exploring threshold-based policies for load sharing. 2004.
24. G.P. Prastacos. Blood inventory management: an overview of theory and practice. *Management Science*, 30(7):777–800, 1984.
25. V. Sirelson and E. Brodheim. A computer planning model for blood platelet production and distribution. *Computer methods and programs in biomedicine*, 35(4):279–291, 1991.
26. D.R. Smith and W. Whitt. Resource sharing for efficiency in traffic systems. *Bell System Technical Journal*, 60(1):39–55, 1981.
27. M.S. Squillante, C.H. Xia, D.D. Yao, and L. Zhang. Threshold-based priority policies for parallel-server systems with affinity scheduling. In *American Control Conference, 2001. Proceedings of the 2001*, volume 4, pages 2992–2999. IEEE, 2001.
28. J. Törnquist. Computer-based decision support for railway traffic scheduling and dispatching: A review of models and algorithms. In *OASIcs-OpenAccess Series in Informatics*, volume 2. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2006.
29. J. Tornquist and J.A. Persson. Train traffic deviation handling using tabu search and simulated annealing. In *System Sciences, 2005. HICSS'05. Proceedings of the 38th Annual Hawaii International Conference on*, pages 73a–73a. IEEE, 2005.
30. N.M. van Dijk and E. van der Sluis. To pool or not to pool in call centers. *Production and Operations Management*, 17(3):296–305, 2008a.

31. N.M. van Dijk and E. van der Sluis. Practical optimization by OR and simulation. *Simulation Modelling Practice and Theory*, 16(8):1113–1122, 2008b.
32. N.M. Van Dijk, R. Haijema, J. Van Der Wal, and C.Th. Smit-Sibinga. Blood platelet production: a novel approach for practical optimization. *Transfusion*, 49(3):411–420, 2009.
33. R.B. Wallace and W. Whitt. A staffing algorithm for call centers with skill-based routing. *Manufacturing & Service Operations Management*, 7(4):276–294, 2005.
34. R.W. Wolff. Stochastic modelling and the theory of queues. *Englewood Cliffs, NJ*, 1989.