

SIMULATION AND OR (OPERATIONS RESEARCH) IN COMBINATION FOR PRACTICAL OPTIMIZATION

Nico van Dijk
Erik van der Sluis
René Haijema
Assil Al-Ibrahim
Jan van der Wal

University of Amsterdam
Faculty of Economics and Econometrics
Roetersstraat 11, 1018 WB
Amsterdam, THE NETHERLANDS

ABSTRACT

Should we pool capacities or not? This is a question that one can regularly be confronted with in operations and service management. It is a question that necessarily requires a combination of queueing (as OR discipline) and simulation (as evaluative tool) and further steps for *optimization*. It will be illustrated that a combined approach (SimOR) of Simulation (techniques and tools) and classical Operations Research (queueing, linear programming and scheduling) can be most beneficial. First, an instructive example of parallel queues will be provided which shows the necessary and fruitful combination of queueing and simulation. Next, the combined approach will also be illustrated for the optimization of: call centers, checking-in at airports, blood platelet production, and train scheduling for railways. Whether we should pool or not is thus just one simple question for which this SimOR approach can be most fruitful if not necessary for *practical optimization*.

1 INTRODUCTION

Simulation, or more precisely as will be meant throughout this paper: discrete event simulation, is well known as a most powerful tool for process and performance evaluation in a vast majority of fields such as to evaluate

- process flows and bottlenecks,
- throughputs and delays,
- capacities.

Standard applications are found in the production sector, the service industry (call centers, administrative logistics and hospitals) and transportation (public transportation

systems, road traffic, airports, harbours, maritime, express delivery systems and so on).

But also numerous other non-standard applications can be referred to (e.g. recreational parks, the hospitality sector, fire and ambulance systems, penitentiary institutions, personnel management, parking management, evacuation models and so on).

This success of simulation can be attributed to a number of factors such as:

- (i) the continuous drive and need for process improvement and performance objectives (such as profits to be maximized or costs to be minimized),
- (ii) the potential of simulation to model real-life complexities with almost no limitations,
- (iii) the availability of both general purpose and special purpose (application oriented) simulation software.

In most applications simulation is necessarily required, as analytic techniques, most notably OR (Operations Research)-techniques such as queueing analysis and mathematical programming, are insufficient due to:

- the complexity of the system,
- the underlying simplifying assumptions required,
- the various types of (non-exponential) stochastics involved.

And in line with (i) simulation is indeed also generally seen and used for *optimization*. Here the 'optimization' standardly relies upon executing the simulation for different scenarios such as typically for infrastructure and process layout, capacities and planning.

However, though this scenario ‘optimization’ is a major step, it remains to be realized that simulation is *no* optimization tool by itself. Clearly, if the optimization problem can be parameterized, such as typically for capacity determination, different search approaches can be suggested to expedite and automate the search for optimal values. An elegant exposé of such methods can be found in Krug (2002). This is where OR might contribute in either of two directions:

1. To suggest candidate scenarios to choose for in the first place, as based upon OR-results and insights.
2. To provide OR-optimization techniques in situations in which no common sense guesses for a ‘nearly optimal scenario’ can be made.

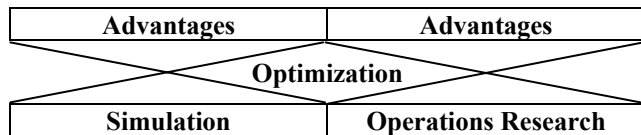
A combination of OR and simulation might then become most beneficial.

- Simulation for its evaluation,
- OR for its optimization.

The advantages of this combination for optimization are schematically represented in Table 1.

Table 1: Combined Advantages

Simulation	OR
Advantages	Disadvantages
Real-life complexities	Simple models
Real-life stochastics	Strict assumptions
Disadvantages	Advantages
Evaluation	Optimization
By scenarios	By techniques
By numbers only	Also by insights



Both directions of this combination will be illustrated in this paper by specific practical applications: direction 1 in section 2 and direction 2 in section 3. In section 4, also a third direction is shown. In this case an optimizing OR-technique (dynamic programming) and the evaluation of its optimal solution by simulation will be combined to conclude a practical nearly optimal solution. Finally, in section 5 another combined use of OR and simulation will be described briefly.

The illustrations all rely upon practical and recent research and schematically concern the topics and combinations with simulation shown in Table 2.

Table 2: Topics and Method Combinations

Section	Topic	Combination
2	Pooling in call centers	SIM + Q
3	Check-in planning	SIM + LP
4	Blood banks	SIM + DP
5	Railways	SIM + Q + DP
Legend:		
SIM: Simulation; Q: Queueing		
DP: Dynamic Programming		
LP: Linear Programming		

2 CALL CENTERS: TO POOL OR NOT?

Should we pool service capacities or not is a question of general interest in a variety of practical situations ranging from counters in postal offices, check-in desks at airports, physicians within hospitals up to whole agent groups within or between call centers. The general perception seems to exist that pooling capacities will always be advantageous, at least from a performance or capacity point of view. For call centers that is, to merge agent groups into one agent group so that the workloads can be balanced more. Indeed, when two separate agents would be pooled none of the agents can ever be idle while there are still calls waiting.

2.1 An instructive example

This perception also seems supported by the standard delay formula for a single (exponential) server with arrival rate λ and service rate μ : $D = 1 / (\mu - \lambda)$ as by pooling two of these servers, each with arrival rate λ and service rate μ , as if they are merged into one single server which is twice as fast, the mean delay would thus be reduced to by a factor 2 as by $D = 1 / (2\mu - 2\lambda)$. In other words, at first glance pooling two servers thus seems to lead to a mean delay reduction of roughly 50%.

This reasoning however relies upon the implicit assumption of two identical servers or rather identical service characteristics, with identical means. When different services are involved in contrast, the advantage of pooling remains questionable. More precisely, here a second basic result from queueing theory is to be realized: a result that seems hardly realized in practical call center environments. For the simple case of a single server this result is known as the Pollaczek-Khintchine (PK) formula. This formula, which is exact for the single server case,

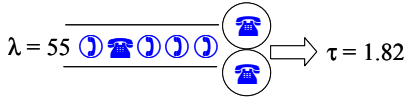
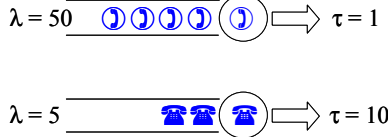
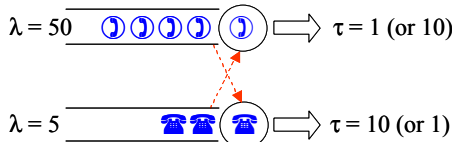
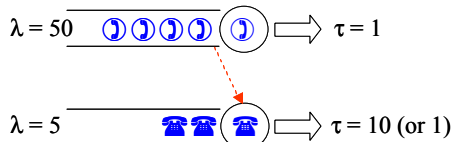
By	Pooled system	$W_A = 6.15$	Unpooled system	$W_A = 4.55$
Q		$W = 6.15$		$W_1 = 2.50$ $W_2 = 25.0$
By	Two-way overflow	$W_A = 4.11$	One-way overflow	$W_A = 3.92$
S		$W_1 = 3.66$ $W_2 = 8.58$		$W_1 = 1.80$ $W_2 = 25.2$

Figure 1: Pooling Scenarios by Queueing (Q) and Simulation (S)

expresses the effect of service variability, as made explicit by:

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

with $c^2 = \sigma^2 / \tau^2$ and

W_G : mean waiting time under a general G (and

E for an exponential) service distribution

with mean τ and standard deviation σ .

As a consequence, if we would pool two separate servers, say for services of type 1 and type 2, as if it becomes one twice as faster server, we should also realize the variability due to mixing. By mixing different services (call types) extra service variability is brought in as each consecutive service can be of either type 1 or 2. As made explicit by the PK-formula this may lead to an increase of the mean waiting time (and delay).

This is illustrated in Figure 1 for the situation of two job (call) types 1 and 2 with mean service (call) durations $\tau_1 = 1$ and $\tau_2 = 10$ minutes but arrival rates $\lambda_1 = 10$ λ_2 . The traffic load $\rho = \lambda_1\tau_1 = \lambda_2\tau_2$ is set at 83%, so that both call types bring in an equal workload.

The results show that the unpooled case is still superior, at least for the average waiting time. These results, for both the strictly pooled and unpooled case as well as the underlying insight thus purely rely upon queueing theory (as a standard discipline of Operations Research (OR)).

Based on these queueing insights, an alternative scenario can now be thought of that may combine the advantages of both scenarios:

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

To this end, the servers should still be kept devoted for type 1 and type 2 jobs (calls). But in addition, server 2 can also take a type 1 job (call) if there is no type 2 job (call) waiting and vice versa. The results for this two-way overflow system, as shown in Figure 2, already indicate an improvement over both the pooled and the unpooled case. As a further improvement, a one-way overflow is therefore suggested in which type 1 calls may but type 2 calls may not overflow. This scenario not only shows a further reduction of the waiting time allover but also improves the unpooled scenario for type 1 (91%) calls.

2.2 Combined Approach

However, in order to achieve these improvements simulation became necessarily required as there are no sufficient analytic results to capture overflow features. A combination of the queueing insights for finding improving scenarios and of simulation for evaluating these scenarios may thus turn out to be fruitful and illustrated by the results in Figure 2.

In addition, by simulation all sorts of performance aspects can be evaluated at the same time. In this case, most notably, a remarkably small percentage of overflow.

The instructive results for this simple example may seem of purely academic nature. But in contrast, similar results also appear to apply at larger scale, that is for larger number of servers as of realistic order for call centers. First, by standard queueing formula the following approximate expression has been derived in (Van Dijk and Van der Sluis 2004) for the effect of pooling two agent groups, each of equal size of s agents with one group separately for type 1 calls and one group separately for type 2 calls with arrival rate λ_i and mean call duration τ_i for type i calls at group i , $i = 1,2$,

W_A = mean waiting time for the unpooled case,
 W_p = mean waiting time for the pooled case.

$$\frac{W_p}{W_A} \approx \left[\frac{\rho}{(1+\rho)} - \frac{1}{4}(1-\rho)\sqrt{s-1} \right]^+ (1+c_{mix}^2)$$

with

$$c_{mix}^2 = \frac{p_1(\tau_1 - \bar{\tau})^2 + p_2(\tau_2 - \bar{\tau})^2}{\bar{\tau}^2}$$

$$\bar{\tau} = p_1\tau_1 + p_2\tau_2 \text{ with}$$

$$p_i = \frac{\lambda_i}{\lambda_1 + \lambda_2}$$

This new (OR)-formula shows that also for values s larger than $s = 1$ (as in the example) the reduction effect by pooling of at least 50% when pooling identical services, might completely vanish due to the mixing effect when pooling non-identical services (with means τ_1 and τ_2).

Next, by simulation this insight and approximate formula as well as the overflow scenarios have been evaluated more precisely for different numbers of agents s and mix situations.

2.3 Results (Larger Number of Servers)

The results by both the OR-formula and simulation showed that pooling only becomes superior over the strictly unpooled case beyond some sufficiently large number of agents. However, the one-way overflow scenario as could only be shown by simulation, turned out to be generally superior to both the pooled and unpooled case, though again up to some sufficiently large number of agents.

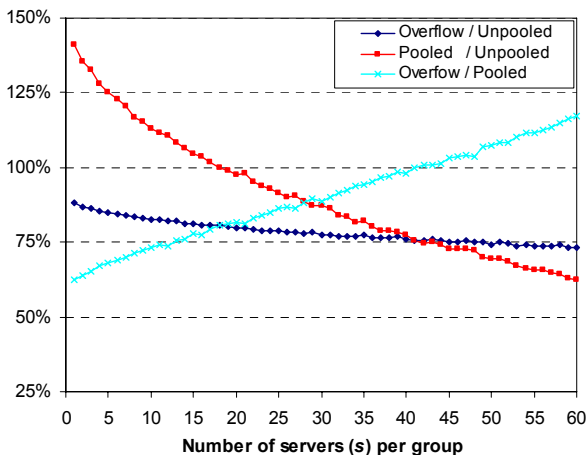


Figure 2: Comparison of Pooling Scenarios by Simulation with $2s$ servers

This is illustrated in Figure 2 with s the number of servers (agents) in the unpooled case for each of two server groups (hence with a total number of $2s$ servers). By

simulation, it illustrates that the mean waiting time for the overflow system, with just a small % of overflow, improves both the pooled and unpooled scenarios with up to 80 agents ($s = 40$).

In Van Dijk and Van der Sluis (2005) also other scenarios are suggested, again as based upon queueing insights, which, by simulation, appear to lead to further improvement.

3 CHECK-IN PLANNING

3.1 Problem Formulation and Approach

Check-in capacity, in terms of desks and desk-(labour)-hours, can be a scarce resource at airports. To minimize the required number of desks two essentially different optimization problems are involved:

- P1. A minimization of the required number of desks for a given flight during the opening hours for that flight in order to meet a given service level (% of travelers with less than so many minutes waiting time) which is of a stochastic nature.
- P2. A minimization and scheduling in order to minimize the total number of desks for a number of flights during a day, which is of a pure deterministic nature.

A two-step procedure has therefore been proposed as will be discussed and illustrated more detailed below.

- Step 1: For P1 as based upon simulation
- Step 2: For P2 as based upon LP
(Linear Programming)

3.2 Step 1: Simulation

Despite the fact that a check-in process for just a single flight (or group of flights) with just a small fixed number of desks can be thought of as a simple multi-server queueing model for which standard M/M/s-formulas are available, the opposite appears to be true. Simulation will necessarily be required for a number of reasons:

1. Most notably, as the arrival process, roughly from 50-300 passengers during 1 up to 4 hours for a single flight, is far from homogeneous over the interval of opening hours, so that a Poisson assumption is far from realistic.
2. As the arrival process in relation to the check-in times, from a few minutes up to a quarter of an hour, does not justify a steady state analysis or ‘averaging’ at all.
3. As there can be a strong ‘initial’ bias when the desks are opened prior to the flight.

4. As the check-in times themselves are far from exponential.
5. As there is a short fixed time frame of opening interval.

Each of these aspects by itself violates an analytic steady state analysis (by standard queueing results). In contrast, terminating simulation will necessarily be required.

Instructive Example As an instructive, yet realistic, example an arrival curve was used as by realistic data for an intercontinental flight from Schiphol (Amsterdam) airport. The service level was set at 90% of all travelers to have a waiting time of no more than 10 minutes. The ‘standard’ planning as based upon M/M/s-calculations would lead to the left figure. By simulation the right figure was obtained.

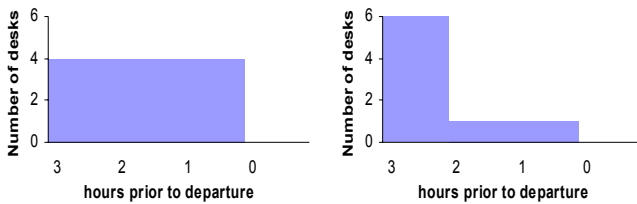


Figure 3: Constant and Dynamic Desk Capacities by Queueing (left) and by Simulation (right)

The simulation shows a triple win:

1. in the total number of desk hours (from 12 to 8),
2. in mean waiting times (from 20 to 12 min),
3. in excessive waiting times (more than 30 min in the first opening hour) for the left case.

Animation As an additional appealing feature of simulation of practical purpose in this case, insight in how long an initial burst of travelers would last or how fast it would vanish could well be visualized by animation.

3.3 Step 2: OR Approach (Linear Programming)

Once the required number of desks are determined by the hour in step 1 for a number of flights during the day a tough optimization problem remains of how to schedule the desks so as to minimize the total number of desks during any hour as well as the total number of desk-(labour)-hours. Here additional practical conditions may have to be taken into account such as most naturally that desks for one and the same flight should be adjacent.

Example As a simple (fictitious) example consider the desk requirements as shown in Figure 4 for 5 flights during 9 hours (periods), as determined by step 1. The total number of desks required then never exceeds 4.

flight	1	2	3	4	5
starting period	1	3	4	5	7
ending periode	3	5	6	7	9
# desks required	3	1	2	1	3

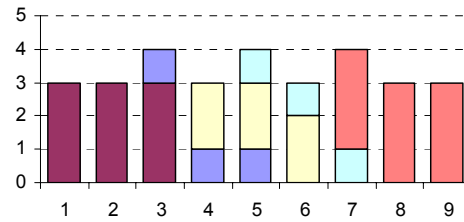


Figure 4: Desks Requirements of 5 Flights

However, a straightforward Earliest Release Date (ERD) desk allocation as shown in Figure 5, would lead to an unfeasible solution as the desks for flight 5 are not adjacent. (This could be resolved by using two more desks 5, 6 and assigning desks 4, 5 and 6 to flight 5). However, in this example a feasible solution with 4 desks is easily found as shown in the lower figure.

Infeasible schedule											
4				2	2	2			5	5	5
3	1	1	1				4	4	4		
2	1	1	1			3	3	3	5	5	5
1	1	1	1			3	3	3	5	5	5
d \ t	1	2	3	4	5	6	7	8	9		

Feasible schedule											
4				2	2	2			5	5	5
3	1	1	1				3	3	3	5	5
2	1	1	1				3	3	3	5	5
1	1	1	1				4	4	4		
d \ t	1	2	3	4	5	6	7	8	9		

Figure 5: An (In)feasible Schedule

As shown in Van Dijk and Van der Sluis (2003) also, for more realistic orders with hundreds of flights an optimal solution can be found by solving the following LP-formulation. (In Duin and Van der Sluis (2004), it is also shown that this problem is NP-hard so that heuristics may be more efficient to use).

$$\begin{aligned}
 \min \quad & D \\
 \text{s.t.} \quad & n_f \leq d_f \leq D \quad \forall f \\
 & \left. \begin{aligned} d_f + n_g \leq d_g \text{ or} \\ d_g + n_f \leq d_f \end{aligned} \right\} \quad \forall f, g \text{ with } I_f \cap I_g \neq \emptyset
 \end{aligned}$$

where

- D : Total number of desks required (indexed 1 to D);
- I_f : Check-in time interval of flight f (with $f=1, \dots, F$);
- d_f : Largest desk number assigned to flight f ;
- n_f : Number of desks required for flight f .

As shown in Van Dijk and Van der Sluis (2003), a similar LP-formulation can also be given for the optimization problem of variable allocation in which the number of desks, as determined by step 2, may vary by the hour which may lead to further savings. This is illustrated in Figure 6 for an example data set of 10 flights which leads to a further reduction in desks (from 17 to 15) and desk hours (from 117 to 92). The combination of (terminating) simulation and LP-optimization so turned out to be most beneficial.

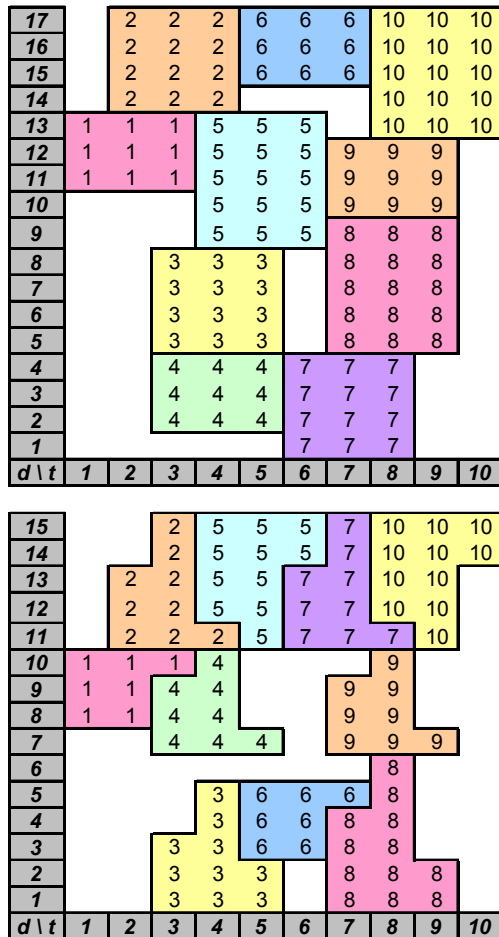


Figure 6: Constant and Dynamic Optimal Desk Schedules for 10 flights

Remark: In Atkins et al. (2003) another application of a combined simulation - linear programming approach for check-in planning was reported and awarded the best practice prize at the 2003 conference of CORS (the Canadian Operations Research Society).

4 BLOOD MANAGEMENT

4.1 Problem Motivation

Blood management is a problem of general human interest with a number of concerns and complications. Clearly, by far the most dominant concern is the risk for infections and contamination such as by HIV and hepatitis. Secondly, the necessity of natural blood will remain, despite technological developments for substitute blood products. This supply has to rely upon voluntary donors. Particularly, in underdeveloped countries (70% of the entire population) there is still a shortage. Furthermore, there are a number of complicating aspects of logistical nature: roughly to be distinguished in the transport to and from blood banks for blood processing within limited time frames and the inventory management at both blood banks and hospitals to meet the demand. Our problem of interest will concentrate on the last aspect: 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 despite planned surgeries (if such information is used) roughly 50% (at week basis) still remains to be unpredictable. And clearly, as lives may be at risk, shortages are to be minimized.

On the other hand, as the supply is voluntary while such shortages may take place, blood is to be considered as highly precious. Any spill, by outdated, of blood (products) is thus highly ‘un’desirable 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, while red blood cells and plasma in all sorts of blood types can be kept for months up to over a year. In addition, regular production of a platelet pool takes about one day. Hence production volumes should be set carefully.

Using OR-techniques, we derive an optimal strategy, but it requires simulation to investigate its structure. From that, a practical rule is derived and optimal parameters are found via a simulation-based search algorithm.

4.2 Simulation and OR Approaches

Early studies and reviews on inventory management of perishable products, such as by Nahmias (1982) and Prastacos (1984) and various references therein, therefore already do contain applications of and focus on blood inventory management.

The ‘experimental’ studies reported in these references (e.g. see Pinson et al. 1972, Katz et al. 1983) generally concern the evaluation and comparison of different order-up-to inventory strategies by means of simulation.

However, the question to which extent these strategies are even ‘near-to-optimal’ remains open, especially when

taking into account several complicating aspects such as distinguishing two types of patients: patients (mainly for Hematology) which prefer the ‘freshest’ pools available, and patients (mainly for general surgery) which can be treated by pools of any age up to the maximal shelf live.

As a relatively standard OR-approach, in contrast, also ‘simple’ dynamic programming formulations for perishable inventory management date back to these early references (e.g. Pierskalla and Roach 1972, Cohen and Pierskalla 1974). These however generally fail to be implemented due to the dimensionality curse. The number of states and computational effort becomes prohibitively large (easily in the order of 10^{50} computations). Experimental blood studies by dynamic programming have therefore not been reported other than recently presented (see Blake 2003), under a number of restricting simplifying assumptions.

4.3 Combined Approach

In Haijema, Van der Wal and Van Dijk (2004 and 2006) a combined ‘new’ approach for the blood platelet inventory problem has therefore been followed, which combines OR and simulation by the following steps:

- Step 1: First, a stochastic dynamic programming (SDP) formulation is provided.
- Step 2: 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: 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. Therefore we ‘register’ the frequency of (state, action)-pairs for the downsized problem.
- Step 4: By a heuristic search procedure a simple practical near to optimal order-up-to strategy is then derived for a rule that resembles the structure at the ‘simulation table’.
- Step 5: The quality (near-to-optimality) of this practical simple order-up-to strategy is then also evaluated by simulation.

4.4 OR and Simulation Step

As the technical (mathematical) details of steps 1 and 2 are somewhat ‘standard’ but also ‘complicated’ and worked out in detail in Haijema, Van der Wal and Van Dijk (2006), let us just restrict to an illustration of the essential OR and Simulation steps 1 and 3.

Step 1 OR Approach (SDP)

As for step 1 the state of the system is described by (d, \mathbf{x}) with

- d : the day of the week ($d = 1, 2, \dots, 7$) and
- $\mathbf{x} = (x_1, x_2, \dots, x_m)$ 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 5 different donations).

And let

$V_n(d, \mathbf{x})$: represent the minimal expected costs over n days when starting in state (d, \mathbf{x}) .

The optimal inventory strategy and production actions are then determined by iteratively computing (solving) the SDP-equations for $n = 1, 2, \dots$

$$V_n(d, \mathbf{x}) = \min_k [c(\mathbf{x}, k) + \sum p_d(b) V_{n-1}(d+1, t(\mathbf{x}, k, b))]$$

with

- k the production action,
- $c(\mathbf{x}, k)$ the one day costs in state \mathbf{x} under production k ,
- $p_d(b)$ 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
- $V_0(d, \mathbf{x}) \equiv 0$.

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 10^{14} , which makes the computation prohibitively large.

Therefore, we have downsized the demands and inventory levels by aggregating the pools in quantities of 4. 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. For example, it prescribes the following production volumes on Tuesday at the following states, which all have the same total inventory level of 14 pools, but of varying ages:

Table 2: Part of the Optimal Strategy of the SDP

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)

Table 3: Simulation Frequency Table of (State, Action)-Pairs on Tuesdays from Simulation of Optimal SDP Solution for 100,000 Weeks

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

Anticipating outdated we aggregate the predicted final inventory, after subtracting average demand according to some issuing policy for the two types of demands.

Step 3 Simulation

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 (in this case Tuesday) and the dataset of the regional blood bank, this led to the ‘simulation table’ in Table 3.

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 expected 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.

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 an even better rule that closely resembles the optimal production strategy.

The search for the best values of the order-up-to parameters for the original, full-sized problem is done via a simulation-based search algorithm, as reported in Haijema, Van der Wal and Van Dijk (2006). Finally we have used simulation models to verify the results, when to take into account the multiple blood types and the uncertainty at the supply side, see Haijema, Van der Wal and Van Dijk (2004).

4.5 Results

Applying this approach to data from a Dutch regional blood bank, we could draw the following conclusions:

1. The simple order-up-to rule reduces the spill from roughly 15 to 20%, as a figure that also seems rather standard worldwide, to less than 1% (while also shortages were reduced and nearly vanished).
2. The combined SDP-Simulation approach led to accuracy within 1% of the exact optimal value for the downsized problem.

For the detailed results we refer to Haijema, Van der Wal and Van Dijk (2004 and 2006).

5 RAIL-TRACK SCHEDULING

5.1 Motivation

An example of yet another class of decision problems is found in rail-track scheduling. Despite their length rail-tracks are a scarce resource for railways and train scheduling. Occasionally two or more trains arrive more or less at the same time at some point where it has to be decided which train goes first, e.g. at the junctions depicted in Figure 6.

Also, trains may occupy tracks for a stretch of miles while trains with different speeds cannot simply pass by at one and the same track. Particularly due to freight trains, which have a large variability in speed, track conflicts may arise not only due to junctions, but primarily due to speed differences. Resolutions of track conflicts can be sought in temporarily rerouting at sidetracks to let trains pass by or by deceleration and acceleration of trains. Clearly, such resolutions require insight, experience and continuous monitoring and are ad hoc. In addition, the resolutions may lead to delays of other trains that have spread through the network. These in turn may have cascading effects for



Figure 7: The Railway Infrastructure of The Netherlands and Sub-network for Test Case

further delays. A more structural resolution approach with locally resolving and optimizing procedures, that could be automated, would thus be highly welcome.

To a certain extent the basis of this decision process is deterministic but practice proves that it is also highly stochastic. The problem thus has the flavor of both a scheduling and a queueing problem.

5.2 Approaches

OR Approach

Partially this track conflict problem can be regarded as a ‘standard’ OR-problem; more precisely as a job shop problem with blocking and no wait constraints (see Mascis and Pacciarelli 2002). By identifying the trains as jobs and the tracks as machines, an ‘optimal’ train order for a track can be found by a branch-and-bound technique. It is a job-shop with blocking and no-waiting as an occupied track section blocks a successive train to enter that section so that the waiting takes place at the preceding section. This formulation however has two shortcomings.

First of all, it assumes pre-allocated fixed track routes for each train. This limits the number of possible solutions substantially and can lead to sub-optimal solutions. This restrictive assumption can partially be relaxed by a flexible job-shop model as handled by Mastrolilli and Gamberdella (2000). The second shortcoming, however, is of a more serious and fundamental nature. The job-shop problem uses fixed handling times without delays and variability. If, in contrast, one would describe the states of the decision process as a multi-dimensional Markov Decision Problem to include stochastic aspects, the number of states would be tremendously large while only a small fraction of these states will ever appear.

Simulation Approach

As these delay aspects and variabilities in travel times are crucial for the track conflict problem in the first place, a stochastic approach is necessarily required. Clearly, simulation would thus be in place despite the fact that it would not optimize at all. Sahin (1999) extends the simulation in this direction. In this study the dispatcher can dynamically adjust the train order. However, the effects of accelerations and decelerations cannot be included. Furthermore, the study is executed for a line-section and cannot be extended to a network level.

5.3 Simulation-OR Approach

In Al-Ibrahim and Van der Wal (2005), a somewhat different approach will therefore be suggested. Roughly it works as follows.

Just mimic the process by simulation and wait for the moments upon which some decision has to be taken. For those decision moments one may think of different decision techniques. A natural candidate would be to solve some sort of stochastic scheduling problem. Another option would be to choose from a small number of queueing scenarios, the simplest one being FCFS, but there may also be various priority rules that will outperform FCFS. More precisely, a combined approach is suggested, which combines simulation with a conflict resolution and optimization algorithm in a number of steps, as briefly outlined below, as based upon three modules:

- A simulation module,
- A conflict-prediction module,
- A resolution and optimization module.

Step 1: (Initialization) Trains are generated for some network infrastructure according to some global schedule but with a number of stochastic elements such as to include initial randomness and speed differences. The dispatching-rule is standardly initialized by FCFS (in order of arrival) train order scheduling.

Step 2: (Simulation) The train process starts to be simulated by the simulation module until a conflict is detected by the conflict-prediction module is activated.

Step 3: (Resolution and optimization) The simulation run is interrupted; the conflict registered and the algorithm module activated. This module first collects all train information (the positions, types, speeds and routes assigned). Next, all feasible train orders are detected and enumerated. A cost function is defined which leads to a cost value for a given order decision. Next, by a Finite horizon Dynamic

Programming type algorithm the ‘best’ train order is determined which leads to a minimal local delay for the conflict situation in order.

- Step 4: The steps 2 and 3 are repeated under the new train orders and iterated up to some stop criterion (a given number of iterations or convergence limit).
- Step 5: The delay process is (re)evaluated by simulation under the ‘best’ found dispatching rule (train orders).

In short, simulation is thus used as a platform:

- to generate conflicts and
- to evaluate decisions made.

While an OR-type algorithm is used to detect-resolve-and locally optimize these conflicts. Though this conflict resolution and optimization is local, simulation studies executed, as illustrated below, seem to indicate that this combined approach substantially improves the FCFS-train order scheduling as is commonly used by railway operators.

Example In cooperation with ProRail (the Dutch Railway operator) the approach has been applied to a small but nevertheless complicating sub-network within The Netherlands, as shown in Figure 6, with one crucial conflicting junction and track conflicts by different types of trains (passenger and freight trains), as illustrated in Figure 8 (trains 6 and 8 are freight trains).

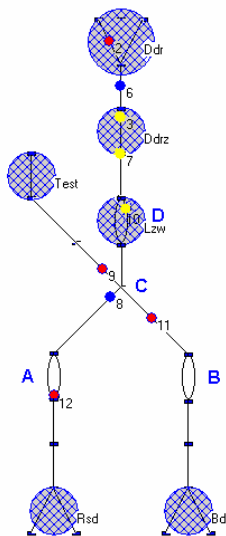


Figure 8: Train Conflicts around the Junction

By the combined approach for a range of realistic cases, the average delay was reduced by a factor ranging from 0.4 to 0.7 depending on the frequencies and number of freight trains. The FCFS-order rule was adjusted in roughly 30 to 40% of conflicts.

6 EVALUATION

Simulation is standardly used and known for evaluation purposes of process performance. Its application for optimization purposes, however, remains limited to a simple comparison or what-if analyses of scenarios.

In this paper in contrast, it is illustrated that simulation can also be used more sophisticated in combination with OR-techniques and results for ‘optimization’ purposes.

To this end, four illustrations are provided in each of which simulation is used in a non-standard way more than just for evaluation purposes in combination with different OR-techniques such as

- queueing
- linear programming
- dynamic programming and
- heuristic dynamic scheduling

The results, as based upon different practical applications, seem to indicate that this combined Simulation-OR (SimOR) approach can be most fruitful. Further application and research of this approach is therefore suggested.

REFERENCES

- Al-Ibrahim, A., and J. van der Wal. 2005. Conflict Management in Railways using Finite Horizon Dynamic Programming-type Algorithm. Working paper, forthcoming.
- Atkins, D., M.A. Begen, B. Lucny, A. Parkinson, M.L. Puterman. 2003. Right on Queue. In *OR/MS Today 30*, issue 2.
- Blake, J. T. 2003. Using dynamic programming to Optimize the Platelet Supply Chain in Nova Scotia. To appear in *Proceedings of the 29th Meeting of the European Working Group on Operational Research Applied to Health Services*. Edited by M. Dlouh’y. Prague, Czech Republic: ORAHS.
- Cohen, M. A. and W. P. Pierskalla. 1974. Perishable Inventory Theory and Its Application to Blood Bank Management. Working Paper, Department of Industrial Engineering and Management Sciences, Northwestern University, Evanston, Illinois..
- Duin, C. W., and H. J. van der Sluis. 2004. On the complexity of Adjacent Resource scheduling. To appear in *Journal of Scheduling*.
- Haijema, R., J. van der Wal and N. M. van Dijk. 2004. Blood Platelet Production: a Multi-type Perishable Inventory Problem. In *Operations Research Proceedings 2004: Selected Papers of the International Conference on Operations Research (OR 2004)*, ed. H. Fleuren, D. den Hertog and P. Kort, 84-92. Springer.

- Haijema, R., J. van der Wal and N. M. van Dijk. 2006. Blood Platelet Production: Optimization by Dynamic Programming and Simulation. To appear in a special issue on OR and Health Care Management of *Computers and Operations Research*.
- Krug, W. 2002. Modelling, Simulation and Optimisation for Manufacturing, Organisational and Logistical Processes. Erlangen, Deutschland, Gruner Druck GmbH.
- Katz, A. J., C. W. Carter, P. Saxton, J. Blutt and R. M. Kakaya. 1983. Simulation analysis of platelets production and inventory management. In *Vox Sang* 44: 31-36.
- Mascis, A., and D. Pacciarelli. 2000. Job-shop scheduling with blocking and no-wait constrains, In *European Journal of Operational Research* 143: 498-517.
- Mastrolilli, M., and L. M. Gambardella. 2000. Effective Neighborhood Functions for the Flexible job Shop Problem. In *Journal of Scheduling* 3: 3-20
- Nahmias, S. 1982. Perishable Inventory Theory: A Review. In *Operations Research* 30: 680-708.
- Pierskalla, W. P. and C. D. Roach. 1972. Optimal Issuing Policies for Perishable Inventory. In *Management Science* 18: 603-614.
- Pinson, S. D., W. P. Pierskalla and B. Schaefer. 1972. A computer Simulation Anaysis of Blood Bank Inventory Policies. Technical Report, Department of Industrial Engineering and Management Sciences, Northwestern University, Evanston, Illinois.
- Prastacos, G. P. 1984. Blood Inventory Management: an Overview of Theory and Practice. In *Management Science* 30: 777-800.
- Sahin, I. 1999. Railway traffic control and train scheduling based on inter-train conflict management. In *Transportation Research Part B*, 33: 511-534.
- Van Dijk, N. M., and H. J. van der Sluis. 2003. Check-in computation and optimization by simulation and IP in combination. To appear in *European Journal of Operations Research*.
- Van Dijk, N. M., and H. J. van der Sluis. 2004. To pool or not to pool in Call Centers. Working paper, Department of Quantitative Economics, University of Amsterdam, The Netherlands.
- Van Dijk, N. M., and H. J. van der Sluis. 2005. Pooling is not always the answer. Working paper, Department of Quantitative Economics, University of Amsterdam, The Netherlands.

AUTHOR BIOGRAPHIES

NICO VAN DIJK is a full professor of Operations Research leads this OR-research group at the University of Amsterdam, and he is also affiliated to the simulation company *Incontrol Enterprise Dynamics*.

ERIK VAN DER SLUIS is an associate professor at the University of Amsterdam within the Faculty of Economics and Econometrics.

RENÉ HAIJEMA is a Ph.D. student at the University of Amsterdam.

ASSIL AL-IBRAHIM is a Ph.D. student at the University of Amsterdam.

JAN VAN DER WAL is a full professor of Operations Research at the University of Amsterdam, within the Faculty of Economics and Econometrics.