J. Opl Res. Soc. Vol. 36, No. 10, pp. 925–928, 1985
Printed in Great Britain.
0160-5682/85 $3.00 + 0.00

Emptying Current-Account Buffers

R. A. PEFFER, J. D. RIJVORDT and J. TELGEN
Department of Applied Mathematics, Rabobank Nederland, The Netherlands

Rabobank is one of the largest banks in The Netherlands and ranks among the 50 largest banks in the world, with over 4 million current accounts. Normally customers receive statements on their accounts only when a certain buffer of changes, both payments and receipts, is filled up. At the end of the year, fiscal and administrative requirements are such that all buffers should be empty. Because of cost and service considerations as well as capacity limitations, all current-account buffers cannot be emptied at will or at the same time. The problem of when to empty current-account buffers can be modelled as a (surprisingly simple) integer-programming problem. Then the problem can be solved easily for a number of different situations, yielding substantial improvements. In this paper, the application of this model to the Rabobank situation at the end of 1983 is described in detail.

Key words: application, banking, integer programming

INTRODUCTION

The Rabobank organization consists of nearly 1000 independent local banks. Together they have formed a cooperative bank, called Rabobank Nederland, to be concerned with that part of their business that can be handled more efficiently by a large central organization. Rabobank Nederland's tasks include international banking, money and capital market transactions, stock exchange and automation.

The automation department operates two computer centres, one in Zeist serving 622 local banks in the northern and central parts of The Netherlands, and one in Eindhoven serving 342 local banks in the southern parts.

One of the major tasks of these computer centres is the operation of the current-account system, containing over 4 million accounts. Payments to and from the current accounts are dealt with primarily in the local banks. During office hours, local banks prepare their input for the computerized processing of all changes (payments and receipts) during that day. At the end of the day, this input is transferred (by car) to the computer centre. The changes to the current accounts are processed during the night, and statements for the individual account holders are printed in the early morning. These statements are shipped to the local banks, checked and mailed to the account holders.

Because of the time involved in preparing the input and checking the output at the local banks, strict limits are imposed on departure and arrival time of the cars transporting input and output. This, in turn, limits the amount of work that can be done in the computer centre. More specifically, there is a maximum number of statements that can be printed in the available time at the computer centre. Because more time is available on Friday nights (until Monday), the maximum on Fridays is substantially higher than on other days of the week.

A statement contains one or more changes. Because the cost of one statement is relatively high at Dfl.0.60 (printing, sorting, administration, mailing etc.), the changes are buffered. This implies that changes are not necessarily printed immediately, but stored in a buffer. Only when certain conditions are satisfied, this buffer is emptied and printed on a statement. These conditions include:

- a prespecified number of days has elapsed since the last statement was printed;
- a debit-balance on the account, which is not allowed;
- a payment to the account (larger than prespecified amount);
- a full buffer.

This process of buffering is easily seen to induce considerable cost savings.

THE PROBLEM

At the end of the year, all buffers have to be empty because of fiscal and administrative requirements. Therefore, some time in December, all current-account buffers have to be emptied; the date at which this happens is the debuffering date. After this date, no buffering is allowed, and every change in the current account induces a statement to be printed. As a consequence, more statements have to be printed and mailed, both on the debuffering date and in the period afterwards, until 31 December.

In 1982, the debuffering date for all local banks was Tuesday 21 December, on which day 1.7 million statements were printed, for a total of 12 million statements from 15 to 31 December.

Several factors should be taken into consideration when evaluating this approach:

the total cost of the solution,
the (peak) workload in the computer centres,
the independence of all local banks: it is contrary to the Rabobank culture to impose a centrally selected debuffering date.

The problem we faced was to develop an approach that paid due respect to these factors and was capable of determining a solution.

The working days of 1983 that were considered, together with the maximum number of statements printed in both computer centres, are displayed in Table 1.

Considerable effort was spent on determining the expected number of statements per local bank, both before, on and after the debuffering date. It was found that the ratios between these numbers can be properly described by constants.

THE MODEL

The problem can be modelled as an integer linear-programming problem. As a start, we minimize the total number of statements, subject to printing capacity and debuffering requirements, by selecting debuffering dates per local bank. Since there is no interaction between the two computer centres with regard to this problem, we formulate a separate model for each centre.

The indices we use are:

$I = 1, 2, \ldots, NB$ the number of the local bank,

$J = 1, 2, \ldots, ND$ the number of the working day.

The decision variables $x(I, J)$ indicate whether bank I has J as its debuffering date $[x(I, J) = 1]$ or not $[x(I, J) = 0]$.

The decision variables $y(I, J)$ indicate whether bank I has been debuffered before day $J[y(I, J) = 1]$ or not $[y(I, J) = 0]$.

It is easily seen that:

$$\begin{aligned} y(I, J) &= 0 \qquad\qquad J = 1 \\ &= \sum_{K=1}^{J-1} x(I, K), \quad J \geq 2. \end{aligned} \tag{1}$$

TABLE 1. *Working days and daily maxima in the debuffering period (1983)*

Working days	Daily maxima	
	Zeist	Eindhoven
Th. 15 December	700,000	400,000
Fr. 16 December	1,000,000	700,000
Mo. 17 December	700,000	400,000
Tu. 20 December	700,000	400,000
We. 21 December	700,000	400,000
Th. 22 December	700,000	400,000
Fr. 23 December	1,000,000	700,000
Tu. 27 December	700,000	400,000
We. 28 December	700,000	400,000
Th. 29 December	700,000	400,000
Fr. 30 December	1,000,000	700,000

The data we denote as:

$M(I)$ = the expected number of daily statements for bank I in the normal (buffered) situation;
$S(I, 1)$ = the extra number of statements printed on the debuffering date for bank I as a fraction of $M(I)$; thus the total number of statements printed on the debuffering date for bank I is: $[S(I, 1) + 1] * M(I)$;
$S(I, 2)$ = the extra number of statements printed after the debuffering date for bank I as a fraction of $M(I)$; thus the total number of statements per day printed after the debuffering date for bank I is: $[S(I, 2) + 1] * M(I)$;
$DM(J)$ = the maximum number of statements that can be printed on day J (see Table 1).

Both $S(I, 1)$ and $S(I, 2)$ can be considered as surplus factors; obviously $S(I, 1) \geqslant S(I, 2)$.

If the auxiliary variable $T(J)$ is introduced as the total number of statements printed on day J, the integer-programming model is:

$$\text{minimize } Z = \sum_{J=1}^{ND} T(J) \tag{2}$$

$$\text{subject to } T(J) = \sum_{I=1}^{NB} M(I) * [1 + S(I, 1) * x(I, J) + S(I, 2) * y(I, J)], \quad J = 1, 2, \ldots, ND \tag{3}$$

$$T(J) \leqslant DM(J), \qquad J = 1, 2, \ldots, ND \tag{4}$$

$$\sum_{J=1}^{ND} x(I, J) = 1, \qquad I = 1, 2, \ldots, NB \tag{5}$$

$$y(I, J) = \sum_{K=1}^{J-1} x(I, K), \quad J = 2, 3, \ldots, ND; \quad I = 1, 2, \ldots, NB. \tag{6}$$

Relations (2) and (3) take care of minimizing the total number of statements per day as the normal debuffered amount $M(I)$ plus possible surplus due to the debuffering date itself $[S(I, 1) * x(I, J)]$, and due to the debuffered situation $[S(I, 2) * y(I, J)]$. Constraint (4) ensures that the maximum capacity is not exceeded, whereas constraint (5) requires that every bank is debuffered exactly once in the period considered.

It is a matter of simple arithmetic to reduce models (2)–(5), using the fact that:

$$\sum_{J=1}^{ND} \sum_{I=1}^{NB} M(I)$$

is constant and

$$\sum_{J=1}^{ND} \sum_{I=1}^{NB} M(I) * S(I, 1) * x(I, J) = \sum_{I=1}^{NB} M(I) * S(I, 1) \sum_{J=1}^{NB} x(I, J)$$
$$= \sum_{I=1}^{NB} M(I) * S(I, 1)$$

is constant.

This results in:

$$\min \sum_{I=1}^{NB} M(I) * S(I, 2) \sum_{J=1}^{ND} \sum_{K=1}^{J-1} x(I, K) \tag{7}$$

subject to

$$\sum_{I=1}^{NB} M(I) \left\{ S(I, 1) * x(I, J) + S(I, 2) * \sum_{K=1}^{J-1} x(I, K) \right\} \leqslant DM(J) - \sum_{I=1}^{NB} M(I) \tag{8}$$

$$\sum_{J=1}^{ND} x(I, J) = 1, \quad \begin{array}{l} J = 1, 2, \ldots, ND \\ I = 1, 2, \ldots, NB \end{array}. \tag{9}$$

IMPLEMENTATION

The resulting model is an integer linear-programming problem with $NB \times ND$ (622×11 for Zeist) 0–1 variables and $NB + ND$ constraints. In general, these problems are rather hard to solve, but the special structure of the problem enables one to obtain the true optimum relatively easily (<3 min C.P.U. time on a DEC 2060 computer using the LINDO package).

The optimal solution resulted in a total of 5.5 million statements for the computer centre in Zeist over the entire period, which is substantially less than 8 million for 1982.

It is easily seen that in this solution the debuffering dates are chosen as late as possible. But this poses difficulties for the local banks in processing the statements. At the end of the year they are very busy anyway, and they would like to have the bulk of their statements (the debuffering date) at some other time. Since the local banks are the mother organizations of Rabobank Nederland and the computer centre is merely a service institute, it was agreed that cost considerations are secondary to the wishes of the local banks. For a procedure, it was agreed that local banks could ask for a specific debuffering date and that the computer centre was allowed to change that by one day at most. For the model, this implies that (5) and (9) are replaced by:

$$\sum_{J=D(I)-1}^{D(I)+1} x(I,J) = 1, \quad I = 1, 2, \ldots, NB, \tag{10}$$

in which $D(I)$ is the preferred debuffering date for bank I.

The model was rerun with these changes, and it is obvious that most of the banks are scheduled for debuffering on day $D(I)+1$ because of cost considerations. This meant that, for the majority of the local banks, the preferred debuffering date was not chosen. It turned out that this was not intended: debuffering *should* be done on the preferred day except for cases in which it was impossible. Now cost considerations have disappeared entirely, and we have to change the objective function (7) to account for this. There are two ways to do so:

$$\text{minimize} \sum_{I=1}^{NB} \{x[I, D(I)-1] + x[I, D(I)+1]\} \tag{11}$$

$$\text{minimize} \sum_{I=1}^{NB} M(I)\{[S(I,1) - S(I,2)] * x[I, D(I)+1] + S(I,1) * x[I, D(I)-1]\}, \tag{12}$$

both subject to constraints (8) and (10). Function (11) minimizes the number of banks that have to be assigned another than their preferred debuffering date. Function (12) minimizes the number of accounts debuffered on a non-preferred debuffering date.

The optimal solution in the model (11), (8) and (10) yields a total of 7.8 million statements for the computer centre in Zeist. Only six banks were assigned debuffering dates other than the ones they preferred.

CONCLUSION

The decision when to empty which current-account buffers can be modelled nicely as an integer-programming problem that is relatively easy to solve. We have shown how to reformulate the model with little effort for a number of situations arising in Rabobank Nederland.

Total cost savings vary from Dfl.0 to 2.2 million a year, depending on the relative importance of cost savings vs compliance with the wishes of local banks. The latter aspect is very important in the organizational structure (cooperation) of Rabobank Nederland. Therefore, a final solution was selected that was a considerable improvement over the hand-selected one in 1982, and at the same time only six banks were assigned debuffering dates other than the ones they indicated.