OR Spectrum (2009) 31:707–725
DOI 10.1007/s00291-008-0134-7

REGULAR ARTICLE

Delivery strategies for blood products supplies

Vera Hemmelmayr · Karl F. Doerner · Richard F. Hartl · Martin W. P. Savelsbergh

Published online: 19 March 2008
© Springer-Verlag 2008

Abstract We introduce a problem faced by the blood bank of the Austrian Red Cross for Eastern Austria: how to cost-effectively organize the delivery of blood products to Austrian hospitals? We investigate the potential value of switching from the current vendee-managed inventory set up to a vendor-managed inventory system. We present solution approaches based on integer programming and variable neighborhood search and evaluate their performance.

Keywords Variable neighborhood search · Inventory routing · Fixed routes · Blood delivery

1 Introduction

Hospitals use a variety of products in the treatment of their patients. Many of these products, most notably blood products, have a short lifespan, and therefore their supply and inventory have to be managed carefully. Blood products are crucial for hospitals, as

V. Hemmelmayr · K. F. Doerner (✉) · R. F. Hartl
Department of Business Administration, University of Vienna,
Bruenner Strasse 72, 1210 Vienna, Austria
e-mail: Karl.Doerner@univie.ac.at

V. Hemmelmayr
e-mail: Vera.Hemmelmayr@univie.ac.at

R. F. Hartl
e-mail: Richard.Hartl@univie.ac.at

M. W. P. Savelsbergh
The Logistics Institute, Georgia Institute of Technology, Atlanta, GA 30332-0205, USA
e-mail: martin.savelsbergh@isye.gatech.edu

they are required for surgeries and for the treatment of patients with chronic illnesses, e.g., patients with cancer. As a consequence, blood products are delivered to hospitals on a regular basis to ensure that an adequate supply of the required blood products is available. Thus, a blood bank is faced with a situation in which a set of customers (hospitals, clinics, medical institutes) requires regular deliveries of certain products (blood conserves), which they consume at different rates. Any delivery policy should be such that no shortfalls of products occur to the customer, but at the same time spoilage of products has to be kept to a minimum. The situation is complicated by the fact that product usage varies over time. Of course, a blood bank also wants to minimize its delivery costs.

The problem outlined above was introduced to us by the largest Austrian blood bank—the blood bank of the Austrian Red Cross (ARC) for Eastern Austria. The blood bank receives about 250,000 blood donations each year. There are four types of blood: O, A, B, and AB. Blood is further distinguished by its rhesus factor, which can be positive or negative. Of the 250,000 blood donations, about 210,000 can be used for further processing into different blood products, such as plasma, platelets, granulocytes, cryoprecipitate, and erythrocyte concentrate. The shelf life of the different blood products ranges from a few days to several weeks. The blood bank serves 55 hospitals. Most of the blood products are delivered to the hospitals on regular delivery routes, but some are distributed through other channels. The delivery routes are planned manually; no routing software or geographic information system is used. The hospitals are grouped into four regions and fixed routes for visiting the hospitals in a region have emerged over time. On a given day, hospitals that requested a delivery of blood products the previous day are visited in the order in which they appear on these fixed routes.

The logistics department at the Austrian Red Cross has recently decided that they want to explore alternatives to the current system. They want to investigate whether a more flexible and dynamic routing system will reduce delivery costs, and they want to understand the cost benefits, if any, of changing from a vendee-managed inventory environment to a vendor-managed inventory system. A vendor-managed inventory system should be of interest to the hospitals, as well as it is likely to reduce costs, product spoilage, and product shortfalls. For an introduction to vendor-managed inventory systems, see Campbell et al. (2001).

Our study provides partial answers to these questions. We investigate alternative delivery strategies, where we are mindful of the fact that for practical reasons regularity in delivery patterns is desirable. We develop and evaluate two alternative delivery strategies. The first strategy retains the concept of regions and the use of fixed routes, but uses integer programming techniques to optimally decide delivery days. The second strategy combines more flexible routing decisions with a focus on delivery regularity, i.e., repeating delivery patterns for each hospital. For an initial assessment of the potential benefits of these delivery strategies, we investigate a simplified setting: we consider a single blood product and we assume known and constant daily demand for each hospital, estimated from a year's worth of historical demand data. More complex settings that can be handled using our optimization technology are discussed in Sect. 4.

It is not necessary for us to consider a reverse logistics process, because unusable blood products are considered hazardous waste and are therefore collected for disposal separately by other logistic service providers. [For a comprehensive discussion of reverse logistics, see Alshamrani et al. 2007]. Similarly, we do not consider re-distribution of blood products between different hospitals, as is done in Pierskalla (2004), because Austrian legal regulations do not allow such redistribution. Moreover, redistribution of blood products between different hospitals is such a fundamental change that officials are reluctant, at best, to even consider it. Finally, redistribution between hospitals is more relevant and will have a larger impact in stochastic settings, and our focus in this paper is purely deterministic.

Blood products are delivered to the hospitals using a small fleet of identical vehicles. As the size of a bag with a blood product is very small, vehicle capacity is never restricting and therefore ignored. On the other hand, driver duty period limits need to be respected and thus the tour length has to be restricted. An analysis of the historical demand data revealed that some hospitals require daily deliveries or deliveries every other day, whereas for other hospitals it is sufficient to have a delivery once a week or even once every 2 weeks. To handle this usage diversity, we require a delivery schedule to cover a 2 week period. Such a delivery schedule can then be repeated every 2 weeks. In a stochastic setting, a rolling horizon approach may be more appropriate.

The objective is to minimize travel cost; no inventory costs are taken into account. This is often seen in health care logistics. As stock-outs may result in loss of life, hospitals actually prefer to have high inventory, even if this results in higher costs. Obsolesce costs need not be considered either, because our approach does not allow spoilage of blood products.

The main result of our study is that these more sophisticated delivery strategies have the potential to significantly reduce delivery costs. Our computational experiments show a potential cost savings of about 30%. The difference between the two proposed delivery strategies is relatively small (less than 5% on average) and depends on the instance characteristics. In tightly constrained situations (i.e., limited storage capacity at the hospitals and short spoilage periods) focusing on optimal delivery day decisions gives slightly better results, whereas in less constrained situations focusing on regular delivery patterns gives slightly better results.

Problems related to blood delivery have also been studied by Rytilä and Spens (2006), who analyze the blood supply chain using simulation, and by Pierskalla (2004), who develops an integrated regional and central blood bank location—allocation model, in which a sweep algorithm is used for the delivery routing part. We focus completely on the delivery of blood. However, even in that part of the blood supply chain, there are opportunities for huge cost savings. In the case of the Austrian Red Cross, with expenditures around 200,000 € annually, a cost reduction of 30% translates to a savings of 60,000 € per year.

The remainder of the paper is organized as follows. In Sect. 2, we discuss three solution approaches that differ in terms of underlying philosophy, development time, and performance. In Sect. 3, we present an extensive computational study that analyzes various aspects of the proposed solution approaches and demonstrates their ability to reduce costs. Finally, in Sect. 4, we provide some conclusions and discuss future research.

2 Solution approaches

2.1 A basic heuristic approach

To mimic the current environment, we have implemented a basic heuristic that is driven by product inventory levels at the hospitals. Each day, those hospitals that need to receive a delivery on that day, because they would otherwise experience a stockout the next day, will be visited. The heuristic closely mimics the inventory policy used in practice, but is more sophisticated when it comes to constructing delivery routes. Delivery routes are constructed daily, whereas in practice fixed routes are used.

The delivery routes for a day are determined using the savings algorithm (Clarke and Wright 1964) followed by a local improvement procedure based on move, exchange, and 3-opt operators [see (Gendreau et al. 1997; Kindervater and Savelsbergh 1997) for a discussion of local search for vehicle routing problems]. When a hospital receives a delivery, its inventory is filled up to capacity. This is a reasonable policy, since we do not consider inventory holding costs and vehicle capacities. To handle product spoilage, the storage capacity at hospital is adjusted, if necessary, based on the hospital's daily usage and the product spoilage period.

2.2 An integer programming approach

The integer programmingbased approach continues to employ the current scheme in which the set of hospitals is divided into four regions and the hospitals in each region are served by a single vehicle using a fixed route, which simply skips those hospitals that do not require a delivery. Many medium-sized enterprises with a private fleet for the delivery of products, like the ARC, prefer fixed routes, because they do not want to recompute delivery routes on a daily basis.

Since the hospitals are not served every day and the hospitals are not always served together, the fixed delivery routes still differ from day to day. This flexibility allows a reduction in delivery costs, but has to be exploited carefully to ensure a sufficient supply of blood products at hospitals at all times and a minimum amount of blood product spoilage.

At the heart of the integer programming model is the observation that short cutting of a fixed route allows for the consideration of a substantial number of routes and may therefore provide adequate flexibility to achieve substantially reduced delivery costs. Consider a fixed route starting and ending at the distribution center and visiting n hospitals. Denote the fixed route by $(0, 1, \ldots, n+1)$, where 0 and $n+1$ denote the distribution center, and $1, \ldots, n$ denote the hospitals (in the order in which they are visited by the fixed route). Furthermore, let y_i denote whether hospital i is included in a route and let x_{ij} denote whether hospital i is immediately followed by hospital j in a route (as the distribution center is on any route, we have $y_0 = y_{n+1} = 1$). Then short cutting the fixed route can be modelled as

$$x_{ij} \geq y_i + y_j - 1 - \sum_{k=i+1}^{j-1} y_k \qquad i = 0, \ldots, n, \;\; j = 1, \ldots, n+1.$$

Table 1 Notation

z^t	for $t = 1, \ldots, T$, a 0–1 variable indicating whether or not a route is executed on day t
d_i^t	for $i = 1, \ldots, n$ and $t = 1, \ldots, T$, a continuous variable indicating the quantity of blood delivered to hospital i on day t
y_i	for $i = 1, \ldots, n$ and $t = 1, \ldots, T$, a 0–1 variable indicating whether or not a hospital is visited on day t
I_i^t	for $i = 1, \ldots, n$ and $t = 1, \ldots, T + 1$, a continuous variable indicating the quantity of blood in inventory at hospital i at the beginning of day t
x_{ij}^t	for $i = 0, \ldots, n$, $j = 1, \ldots, n + 1$, and $t = 1, \ldots, T$, a 0–1 variable indicating whether or not the delivery vehicle travels from hospital i to hospital j on day t (where 0 and $n + 1$ denote the Red Cross distribution center)
I_i^1	the initial inventory at the beginning of the first day at hospital i
t_{ij}	travel time between two locations i and j
s_i	service time at location i
u_i^t	the product usage at hospital i on day t
C_i	storage capacity at hospital i
D	maximum route duration
k_1	days of inventory required as safety stock
k_2	length of spoilage period

Hospital i is followed immediately by hospital j if and only if hospital i and j are both visited, but none of the hospitals in between i and j are visited. Note that the hospitals visited on a route are visited in the same order as on the fixed route. At the moment, we solve a travelling salesman problem to obtain the fixed route for visiting the hospitals in a region, but it is probably more appropriate to solve a heterogeneous probabilistic travelling salesman problem (Bianchi and Campbell 2007).

Given the fixed routes for the regions, we use an integer programming model to determine the actual routes during the 14-day planning period. Based on the demand patterns and capacity restrictions at the individual hospitals, the integer programming model optimally decides which hospitals to visit on any given day so as to ensure that none of the hospitals experiences a stock-out, spoilage is kept at a minimum, and delivery costs are minimized. The parameters and decision variables of the integer program are given in Table 1.

Under the assumption that the hospitals are indexed from 1 to n according to the order in which they are visited on the fixed route, the formulation is as follows:

$$\min \sum_t \sum_{i,j} t_{ij} x_{ij}^t$$

subject to

$$I_i^{t+1} = I_i^t - u_i^t + d_i^t \qquad i = 1, \ldots, n, \ t = 1, \ldots, T \tag{1}$$

$$I_i^t \geq \sum_{s=t}^{t+k_1-1} u_i^s \quad i = 1, \ldots, n,\ t = 1, \ldots, T \tag{2}$$

$$I_i^{T+1} \geq I_i^1 \quad i = 1, \ldots, n \tag{3}$$

$$I_i^t \leq C_i \quad i = 1, \ldots, n,\ t = 1, \ldots, T \tag{4}$$

$$I_i^t \leq \sum_{s=t}^{t+k_2-1} u_i^s \quad i = 1, \ldots, n,\ t = 1, \ldots, T \tag{5}$$

$$d_i^t \leq C_i y_i^t \quad i = 1, \ldots, n,\ t = 1, \ldots, T \tag{6}$$

$$x_{ij}^t \geq y_i^t + y_j^t - 1 - \sum_{k=i+1}^{j-1} y_k^t \quad i = 0, \ldots, n,\ j = 1, \ldots, n+1,\ t = 1, \ldots, T \tag{7}$$

$$\sum_{i=0}^{n+1} \sum_{j=0}^{n+1} t_{ij} x_{ij}^t + \sum_{i=1}^{n} y_i^t s_i \leq D \quad t = 1, \ldots, T. \tag{8}$$

$$y_0^t = y_{n+1}^t = 1 \quad t = 1, \ldots, T. \tag{9}$$

$$z_t \geq y_i^t \quad i = 1, \ldots, n,\ t = 1, \ldots, T \tag{10}$$

Constraints 1 ensure inventory balance from period to period. Constraints 2 ensure that the safety stock requirements are met. Constraints 3 ensure that the inventory at the end of the planning period is greater than the inventory at the start of the planning period. Constraints 4 ensure that inventory never exceeds the storage capacity. Constraints 5 ensure that inventory does not spoil. Constraints 6 enforce that a positive delivery quantity only occurs when a hospital is visited. Constraints 7 capture short cutting of the fixed route. Constraints 8 limit route duration. Constraints 9 simply state that the distribution center is "visited" every day. Finally, constraints 10 force the indicator variable z_t to be equal to one when a delivery route is executed on day t. These indicator variables are not necessary for the correctness of the formulation, but are convenient and useful when we introduce changes to the formulation to enhance its solvability.

To solve instances more efficiently, we make a few modifications to the formulation:

- Since there may be many feasible solutions with the same cost (performing a route a day earlier or a day later may not affect feasibility), we perturb the objective function and give a slight preference to making deliveries earlier in the planning period, i.e.,

$$\min \sum_t \sum_{i,j} t_{ij} x_{ij}^t + \sum_t \epsilon t z^t, \tag{11}$$

  where ϵ is a small positive constant. Such changes tend to speed up the search for the optimal solution.
- For each hospital i, we calculate the minimum number of deliveries M_i that have to be made to that hospital during the planning horizon (a simple calculation based on the initial inventory, the daily usage, and the storage capacity) and impose

$$\sum_{t=1}^{T} y_i^t \geq M_i \qquad i = 1, \ldots, n. \tag{12}$$

- We fix the number of route executions K during the planning period ($K \geq \max_i M_i$), i.e.,

$$\sum_{t=1}^{T} z^t = K. \tag{13}$$

  Initial experimentation revealed that the integer programs solved much more efficiently when the number of route executions during the planning period is fixed. Therefore, we solve several integer programs for various values of K.

Furthermore, we enforce that high branching priority is given to z^t variables, medium branching priority is given to y_i^t variables, and low branching priority is given to x_{ij}^t variables.

2.3 A variable neighborhood search approach

The second approach is based on viewing the problem as a periodic vehicle routing problem (PVRP) with tour length constraints (but without capacity constraints). In the PVRP, a planning horizon of T days is considered and each customer i specifies a service frequency e_i and a set C_i of allowable combinations of visit days. For example, if $e_i = 2$ and $C_i = \{\{1, 4\}, \{2, 5\}, \{3, 6\}\}$, then customer i must be visited twice during the planning period and these visits should take place either on days 1 and 4, or on days 2 and 5, or on days 3 and 6. The challenge is to simultaneously select a visit combination for each customer and to solve the implied daily vehicle routing problems.

As different visit frequencies may lead to feasible delivery patterns for hospitals, i.e., delivery patterns that do not result in product shortages and product spoilage, we allow each hospital i to have a set E_i of visit frequencies and thus of associated (periodic) visit combinations. For example, consider a planning horizon of 6 days and consider possible frequencies 2 and 3 for hospital i (i.e., $E_i = \{2, 3\}$). The following set C_i of visit combinations will be generated $\{\{1, 3\}, \{2, 4\}, \{3, 6\}, \{1, 3, 5\}, \{2, 4, 6\}\}$. Visit combinations that lead to an infeasible delivery pattern will be deleted.

A variable neighborhood search (VNS) algorithm was developed for the solution of this variant of the PVRP. The basic idea of VNS is a systematic change of neighborhoods within a local search procedure. That is, several neighborhoods are used within a local search instead of just one, which is generally the case. More precisely, VNS explores larger and larger neighborhoods of the current solution. The search jumps from its current point in the solution space to a new one if and only if an improvement has been made. The steps of basic VNS are shown in Fig. 1, where $N_\kappa (\kappa = 1, \ldots, \kappa_{\max})$ is the set of neighborhoods. The stopping condition can be a limit on CPU time, a limit on the number of iterations, or a limit on the number of iterations between two improvements. See, (Mladenovic and Hansen 1997; Hansen and Mladenovic 2001) for a more thorough description of VNS.

Initialization. Select the set of neighborhood structures $N_\kappa(\kappa = 1, \ldots, \kappa_{max})$, that will be used in the search; find an initial solution x; choose a stopping condition;
Repeat the following until the stopping condition is met:

1. Set $\kappa \leftarrow 1$;
2. Repeat the following steps until $\kappa = \kappa_{max}$:
   (a) *Shaking*. Generate a point x' at random from κ^{th} neighborhood of x ($x' \in N_\kappa(x)$);
   (b) *Local search*. Apply some local search method with x' as initial solution; denote with x'' the so obtained local optimum;
   (c) *Move or not*. If this local optimum x'' is better than the incumbent, or some acceptance criterion is met, move there ($x \leftarrow x''$), and continue the search with N_1 ($\kappa \leftarrow 1$); otherwise, set $\kappa \leftarrow \kappa + 1$;

Fig. 1 Steps of the VNS (Hansen and Mladenovic 2001)

Next, we describe the different components of the VNS that we have implemented for our variant of the PVRP.

2.3.1 Initial solution

To construct an initial solution the set of hospitals is first divided into clusters. This is done by solving a vehicle routing problem consisting of all customers. The vehicle routing problem is solved using the savings algorithm (Clarke and Wright 1964). Next, the same visit combination is assigned to all hospitals in a cluster. This implies that the hospital, which requires the highest frequency, determines the frequency for all hospitals in the cluster. Finally, the resulting daily vehicle routing problems are solved, again using the savings algorithm. The savings algorithm terminates when no two routes can feasibly be merged, i.e., no two routes can be merged without violating the route duration constraints. As a result, the number of routes may exceed the number of available vehicles. If that happens, the route with the fewest customers is identified and the hospitals in this route are moved to other routes (minimizing the increase in costs). Note that this may result in routes that no longer satisfy the duration constraints. This step is repeated until the number of routes is equal to the number of vehicles. Since the initial solution may not be feasible, the VNS needs to incorporate techniques that drive the search to a feasible solution.

2.3.2 Shaking

The set of neighborhoods used for shaking is at the heart of the VNS. Each neighborhood should strike a proper balance between perturbing the incumbent solution and retaining the good parts of the incumbent solution. Two popular and effective neighborhoods for vehicle routing are based on the move and the cross-exchange operators. In a move, a segment of a route is incorporated in a different route. In a cross-exchange, two segments of different routes are exchanged. The orientation of the segment(s) and of the route(s) is preserved by the move and cross-exchange operators.

Table 2 Set of neighborhood structures with $\kappa_{max} = 15$

κ	Operator	Minimum segment length	Maximum segment length
1	Move	1	min (1, n)
2	Move	1	min (2, n)
3	Move	1	min (3, n)
4	CROSS	1	min (1, n)
5	CROSS	1	min (2, n)
6	CROSS	1	min (3, n)
7	CROSS	1	min (4, n)
8	CROSS	1	min (5, n)
9	CROSS	1	min (6, n)
		Minimum customers	Maximum customers
10	Change combination, lower frequency	1	1
11	Change combination, lower frequency	1	2
12	Change combination, lower frequency	1	3
13	Change combination, lower frequency	1	4
14	Change combination	1	1
15	Change combination	1	2

[The cross-exchange neighborhood was successfully used in a VNS for the vehicle routing problem with time windows (Braysy 2002).] The move and cross-exchange operators are used to define a set of neighborhoods that allow the exploration of increasingly distant solutions from the incumbent to overcome local optimality and strive for global optimality.

For the periodic vehicle routing problem, it is essential to also have a neighborhood that changes the visit combinations for hospitals. We use a neighborhood in which the visit combinations of only a limited number of hospitals are changed. For each of these, a frequency is chosen randomly and then a visit combination associated with this frequency is chosen, also randomly. The selection of the visit combination is biased by the visit combinations of the other hospitals in the same cluster. That is, if a combination contains days on which the other hospitals in the cluster are visited, it is more likely that this combination is chosen.

The metric to measure the increasing size of a neighborhood is given by the maximum number of hospitals in the route segment used within the operators. Let n denote the number of hospitals in a route, then Table 2 shows for each neighborhood κ the maximum segment length considered.

In each neighborhood, all the possible segment lengths are equally likely to be chosen. However, our choice of neighborhoods is biased towards smaller segment lengths to focus the search rather close to the incumbent solution.

2.3.3 Local search

A solution obtained through shaking is afterwards submitted to a local search procedure to come up with a locally optimal solution. We use a restricted version of 3-opt

(sequence inversion is not allowed) and only improve individual routes (that way only routes that have changed during shaking have to be reoptimized). The local search restarts immediately after an improving move was found.

2.3.4 Acceptance decision

After the shaking and the local search procedures have been performed, the solution thus obtained has to be compared to the incumbent solution to be able to decide whether or not to accept it.

The acceptance criterion in the basic VNS is to accept only improvements. To avoid getting trapped in bad local optima it may be beneficial in some situations to also accept nonimproving solutions. Hansen and Mladenović (2000) propose an extension of the basic VNS called Skewed VNS. In this approach, a solution is not only evaluated by its objective value but also by its distance to the incumbent solution, favoring more distant solutions. However, we implement a scheme that is inspired by simulated annealing (Kirkpatrick et al. 1983), because the objective function is already skewed by the penalty parameters, and the use of an additional distance parameter will impair the objective function even further and is not promising.

More specifically, improving solutions are always accepted, and inferior solutions are accepted with a probability $\exp\frac{-(f(x')-f(x))}{T}$. The acceptance of inferior solutions depends on a temperature T and the difference in value of the new solution and the incumbent solution. The temperature T is decreased during the search process. The initial temperature T is set to 50 in all instances. The temperature is decreased linearly every 1,000 iterations, in such a way that it is close to 0 in the last few iterations.

As mentioned at the beginning of this section, the VNS has to be able to handle infeasible solutions. Infeasibility occurs if the tour duration exceeds the specified limit. We use a weighted, linear penalty function for violations of this limit. This penalty function is added to the objective function before the solution is evaluated for acceptance. The penalty for each unit of time over the limit is set to 1,000. Because of this high penalty, there is a strong bias towards feasible solutions.

3 Computational study

The basic heuristic, the integer programming approach, and the variable neighborhood search approach were implemented and tested on real world data from the blood bank of the Austrian Red Cross of Eastern Austria for the year 2003. Distances were obtained using the GIS software ArcGIS (version 9) and the road data for Austria from Teleatlas. All solution procedures were coded in ANSI C and compiled with the GNU C compiler version 3.4.4. The integer programs were solved using Xpress-Optimizer 2005. All experiments were performed on a 3.2 GHz Pentium 4 processor running the Windows XP operating system.

Our computational analysis uses nine data sets (see Table 3). Eight of the data sets are derived from a month's worth of demand data, where the months were chosen in such a way that they cover the spectrum from very low demand months to very high demand months. The last data set is derived from a year's worth of data and

Table 3 Real world instances

Instance number	Month	Deviation (%)	Class
1	January	−44.00	Lowest
2	February	5.14	High
	March	2.10	Medium
	April	8.00	High
	May	3.24	Medium
3	June	12.57	Highest
4	July	8.76	High
5	August	−1.71	Medium
6	September	5.71	Medium
7	October	7.62	High
	November	−3.81	Medium
8	December	−4.76	Low
9	Average	0.00	Medium

represents an average month. The data are made anonymous so that it is impossible to infer demand patterns at the different hospitals.

The average daily demand, based on a year's worth of data, is about 500 blood bags. The deviation of the average daily demand in a given month is given in the column with header "**deviation**", e.g., the average daily demand in January is 44% less than 500 blood bags. We grouped the instances in five different classes according to the deviation of the average demand.

Because of the differences in daily demand, the minimum number of deliveries required to serve a hospital during the planning horizon differs. When we allow the use of the entire storage capacity at a hospital and we assume a spoilage period of 41 days, the minimum number of deliveries during the planning period ranges from 1 to 3.

The timing of deliveries is also impacted by the initial product inventory. We have conducted experiments with different assumptions regarding the initial inventory. In the first set of experiments, we have assumed that the initial inventory is given (equal to half the storage capacity), and in the second set of experiments, we have assumed that the initial inventory is not given, but can be set as part of the solution. Recall that we do require that the final inventory is equal to the initial inventory.

We have also conducted experiments to assess the sensitivity with respect to key problem parameters such as the storage capacity at the hospitals and the product spoilage period. We investigated three levels of storage capacity: 100, 75, and 50%. We discuss the results for the case in which 75% of the storage capacity can be used for blood products. The results for the 100% and the 50% case are given in the "Appendix". We investigated two product spoilage periods: 41 days (which is the spoilage period of regular blood products) and 11 days (which is the spoilage period of some special products). Since we are investigating a deterministic setting, we did not perform sensitivity analysis with respect to safety stock levels; this becomes relevant when settings with stochastic usage are considered.

The primary performance measure is total cost (which relates linearly to the total distance traveled during the planning period). In our results tables, we present the costs

of the delivery schedules produced by the different approaches as well as their relative differences. The best solution for each instance is presented in bold face. The average CPU times (over the different instances) are reported in the last row of each table. The solution time of the integer program was limited to 2 h and the best solution found is reported. Note that the fixed routes for each region are constructed once upfront and are not included in the reported computation times. Moreover, we solve the integer program three times for each region, as we set the number of route executions K, in constraint 13, to three different values. Consequently, since there are four regions, the maximum time spent for one instance is 24 h ($2 \times 4 \times 3$). For more than half of the instances, the optimality was not proven. When reporting results for the VNS approach, we present averages as well as the minimum over five runs (because the VNS approach incorporates randomization).

3.1 Results for fixed initial inventories

In the first set of experiments, the initial and final inventory at the hospitals are given and are equal to half of the storage capacity. We analyze the case in which 75% of the storage capacity, can be used for product spoilage periods of 11 and 41 days. As can be seen in Tables 4 and 5, the basic approach (denoted by BA) results in costs, which are 37.4% higher when the spoilage period is 11 days and 46.6% higher when the spoilage period is 41 days. The quality of the delivery schedules produced by the integer programming approach and the variable neighborhood search are basically comparable, with a slight advantage for the integer programming approach when the spoilage period is short. On the other hand, we see that the integer programming approach requires a lot more CPU time than the other approaches.

One reason for the difference in solution quality between the integer programming approach and the variable neighborhood search approach when the spoilage period

Table 4 Results for a given fixed initial inventory for a case in which 75% of the storage capacity, can be used for a product spoilage period of 11 days

Instance	BA	MIP	VNS				
			Average	Minimum	MIP−BA %	MIP−VNS %	VNS−BA %
1	3,182.3	**2,572.8**	2,614.3	2,611.3	23.7	1.6	21.73
2	4,041.5	**2,929.4**	3,006.2	2,997.5	38.0	2.6	34.44
3	4,267.9	**2,973.4**	3,499.2	3,473.8	43.5	17.7	21.97
4	4,012.4	**2,824.5**	3,193.9	3,159.1	42.1	13.1	25.63
5	3,905.3	**2,843.8**	3,085.6	3,010.2	37.3	8.5	26.56
6	3,968.1	**2,931.2**	3,190.1	3,126.1	35.4	8.8	24.39
7	4,197.5	**2,992.2**	3,301.4	3,223.8	40.3	10.3	27.14
8	3,808.7	**2,867.6**	3,159.8	3,116.5	32.8	10.2	20.54
9	3,945.4	**2,754.9**	2,994.8	2,958.0	43.2	8.7	31.74
Average	3,925.4	2,854.4	3,116.1	3,075.1	37.4	9.1	26.0
CPU time	1 s	11 h 34 min	13 min				

Bold values indicate the best solution for each instance

Table 5 Results for a given fixed initial inventory for a case in which 75% of the storage capacity, can be used for a product spoilage period of 41 days

Instance	BA	MIP	VNS				
			Average	Minimum	MIP−BA %	MIP−VNS %	VNS−BA %
1	2,453.9	1,509.0	**1,496.1**	1,491.9	62.6	−0.9	64.02
2	3,435.6	**2,305.5**	2,334.4	2,273.0	49.0	1.3	47.17
3	3,807.6	**2,589.8**	2,729.1	2,711.5	47.0	5.4	39.52
4	3,390.3	**2,338.6**	2,416.8	2,376.7	45.0	3.3	40.28
5	3,109.0	2,192.6	**2,057.5**	2,037.3	41.8	−6.2	51.11
6	3,390.4	2,242.3	**2,154.6**	2,141.6	51.2	−3.9	57.36
7	3,468.3	2,491.8	**2,339.3**	2,325.9	39.2	−6.1	48.26
8	2,888.6	1,946.2	**1,931.1**	1,920.6	48.4	−0.8	49.58
9	3,090.4	2,294.0	**2,037.9**	2,029.1	34.7	−11.2	51.65
Average	3,226.0	2,212.2	2,166.3	2,145.3	46.6	−2.1	49.9
CPU time	1 s	15 h 13 min	10 min				

Bold values indicate the best solution for each instance

is short is that the VNS only considers periodic visit combinations. If capacities and spoilage periods are restrictive, the visit frequencies increase and it becomes harder to find matching visit combinations for these high frequencies. For example, if a hospital has frequency 2, the associated visit combinations in a 6-day planning period would be {1,3}, {2,4}, and {3,6}. On the other hand, if a hospital has visit frequency 3, the associated visit combinations are {1,3,5} and {2,4,6}. Even if these hospitals are near to each other, they cannot be visited together in 2 days (which the integer programming approach is likely to do).

3.2 Results for free initial inventories

By adjusting the initial inventories at hospitals, i.e., setting them at values other than half the storage capacity, the timing of deliveries may be synchronized better, which might in turn lead to reduced costs. We explore the value of this added flexibility in this section.

In the integer programming approach, we simply convert the initial inventory to a decision variable. In the variable neighborhood search approach, we first calculate a set of reasonable initial inventory levels (e.g., the usage in 1 day, the usage in 2 days, etc.). For each of these initial inventories, all possible visit combinations are determined, resulting in a larger set of visit combinations.

The results are presented in Tables 6 and 7. As we can see, the variable neighborhood search approach benefits more from the additional degree of freedom than the integer programming approach. On average, the delivery schedules produced by the integer programming approach improve by 5%, whereas the delivery schedules produced by the variable neighborhood search improve by 9%. The reason is that it becomes easier to combine deliveries to hospitals that are near to each other, as the drawback of

Table 6 Results for a free initial inventory for a case in which 75% of the storage capacity, can be used for a product spoilage period of 11 days

Instance	MIP	VNS		MIP−VNS %
	Cost	Average	Minimum	
1	2,655.2	**2,591.1**	2,589.9	−2.41
2	2,676.9	**2,637.5**	2,630.5	−1.47
3	**2,810.8**	2,893.6	2,881.9	2.95
4	**2,780.1**	2,892.2	2,861.1	4.03
5	2,774.6	**2,628.5**	2,599.9	−5.27
6	**2,779.3**	2,885.9	2,863.7	3.83
7	**2,962.8**	3,242.9	3,221.4	9.45
8	2,695.0	**2,591.1**	2,588.3	−3.86
9	2,658.0	**2,637.3**	2,619.6	−0.78
Average	2,754.8	2,777.8	2,761.8	0.7
CPU time	17 h 20 min	13 min		

Bold values indicate the best solution for each instance

Table 7 Results for a free initial inventory for a case in which 75% of the storage capacity, can be used for a product spoilage period of 41 days

Instance	MIP	VNS		MIP−VNS %
	Cost	Average	Minimum	
1	1,330.1	**1,323.5**	1,312.7	−0.50
2	2,107	**1,980.8**	1,907.5	−5.99
3	**2,455.2**	2,515.3	2,491.3	2.45
4	2,211.3	**2,170.9**	2,112.7	−1.83
5	2,069.6	**1,900.2**	1,883.3	−8.19
6	2,066.2	**1,924.2**	1,907.4	−6.87
7	2,449.8	**2,301.6**	2,251.8	−6.05
8	1,830.5	**1,765.7**	1,740.4	−3.54
9	2,084.4	**1,778.3**	1,756.1	−14.68
Average	2,067.1	1,962.3	1,929.2	−5.0
CPU time	17 h 11 min	10 min		

Bold values indicate the best solution for each instance

periodic visit combinations is not so strong any more. We still observe that the integer programming approach performs better when restrictions are tight, but the difference has become smaller.

3.3 Summary

In Table 8, a summary of the deviations between the basic approach, the integer programming approach, and the variable neighborhood search approach is presented.

Table 8 Average cost deviation

Capacity–Spoilage	Given initial inventory		Free initial inventory
	MIP−BA %	MIP−VNS %	MIP−VNS %
50%–11 days	41.80	13.70	5.80
50%–41 days	44.30	6.00	1.30
75%–11 days	37.40	9.10	0.70
75%–41 days	46.60	−2.10	−5.00
100%–11 days	32.80	3.10	−2.50
100%–41 days	57.50	−3.00	−4.30

Table 9 Average number of visits

Capacity–Spoilage	Given initial inventory		Free initial inventory	
	MIP	VNS	MIP	VNS
100%–11	111	110	110	110
100%–41	64	61	61	60
75%–11	112	111	112	111
75%–41	77	69	70	67
50%–11	121	117	118	117
50%–41	105	94	98	92
Average	98	94	95	93

Table 10 Average number of routes

Capacity–spoilage	Given initial inventory		Free initial inventory	
	MIP	VNS	MIP	VNS
100%–11	11	11	12	8
100%–41	9	7	8	7
75%–11	15	14	14	13
75%–41	15	13	13	11
50%–11	13	12	12	10
50%–41	10	9	9	8
Average	12	11	11	10

It demonstrates that the basic approach delivers poor results in all cases, the integer programming approach is effective in more tightly constrained environments, and that the variable neighborhood search approach performs well in relatively unconstrained environments.

Table 9 shows the average number of hospital visits. The VNS results in slightly fewer visits. Table 10 shows the average number of delivery routes. There too, the VNS results in slightly fewer routes.

4 Conclusion

We have investigated the potential impact of introducing VMI concepts in the blood delivery strategy of the Austrian Red Cross. We have designed and implemented two alternative solutions approaches. Both approaches try to provide some regularity in the delivery schedule, as a completely dynamic delivery strategy was deemed undesirable by the logisticians at ARC. Both approaches demonstrate a significant potential for cost savings; around 30% for the instances considered in the computational study. The differences between the two alternative approaches were relatively small. The implication of our study is that it is worthwhile for the ARC to enter into negotiations with the hospitals to see if they can be convinced to switch to a VMI strategy. This might take a considerable amount of time due to the politics involved.

To get an initial feel for the potential cost savings that may be derived from switching to a VMI strategy, Austrian Red Cross agreed that a deterministic setting with a single product was appropriate. Since the initial indications are that there are substantial cost savings opportunities, the next step is to investigate settings that more closely resemble the practical environment. Some of these can easily be accommodated in our optimization technology; others require further research.

The presented techniques can easily be adapted to handle nonstationary demand, i.e., settings in which demand varies over time. In fact, the IP presented in Sect. 2.2 already anticipated this and assumes nonstationary demand u_i^t. In the VNS approach presented in Sect. 2.3, nonstationary demand can be handled by slightly more sophisticated logic during the preprocessing stage when feasible visit day combinations for each hospital are determined.

Multiple blood products can also be accommodated quite easily in a deterministic setting. The demand of different products, even if they have different spoilage periods, can be aggregated to derive a total demand quantity. After obtaining a delivery schedule for the total demand quantity (using single product optimization technology), the total demand has to be disaggregated. This is done by choosing delivery quantities for all products such that their stock-out times are identical. If the products have different spoilage periods, then the minimum spoilage period determines the stock-out time for all products. The outlined scheme is applied for each hospital.

The above arguments show that nonstationary demand and multiple blood products can easily be handled by the optimization techniques presented. We have not performed computational experiments for these settings, because the necessary data were not available.

A setting that cannot so easily be accommodated is one in which the demand is uncertain (stochastic). This is, of course, an important and practically relevant setting and one that is currently under investigation.

Vendor-managed inventory implies that control for replenishment decisions is shifted from the vendee to the vendor. As we have shown, even in deterministic settings, this can result in significant cost savings for the vendor. However, the benefits are likely to be even greater in stochastic settings, where the vendor can exploit the flexibility even more advantageously.

Acknowledgements Financial support by grant #11187 from the Oesterreichische Nationalbank (OeNB) is gratefully acknowledged. We are grateful to experts of the Austrian Red Cross such as Franz Jelinek and Helmut Kallinger for providing us with detailed information about the processes of the blood bank. We would also like to thank Ortrun Schandl for processing the data.

Appendix

In this Appendix, we report the results for the experiments in which the storage capacity at the hospitals is set at 50% (Tables 11, 12) and at 100% (Tables 13, 14). The

Table 11 Results for a given fixed initial inventory for a case in which 50% of the storage capacity, can be used for a product spoilage period of 11 days

Instance	BA	MIP	VNS				
			Average	Minimum	MIP−BA %	MIP−VNS %	VNS−BA%
1	3,935.4	**2,709.8**	2,997.9	2,962.5	45.2	10.6	31.27
2	4,514.0	**3,310.9**	3,802.5	3,743.2	36.3	14.8	18.71
3	4,980.8	**3,569.1**	3,674.9	3,628.9	39.6	3.0	35.54
4	4,725.2	**3,352.4**	3,830.0	3,801.5	41.0	14.2	23.37
5	4,591.5	**3,178.8**	3,237.4	3,193.6	44.4	1.8	41.83
6	4,595.7	**3,180.2**	3,867.2	3,839.2	44.5	21.6	18.84
7	4,802.7	**3,553.7**	4,300.7	4,260.3	35.1	21.0	11.67
8	4,581.0	**3,110.8**	3,622.3	3,601.7	47.3	16.4	26.47
9	4,417.7	**3,088.0**	3,704.2	3,695.0	43.1	20.0	19.26
Average	4,571.6	3,228.2	3,670.8	3,636.2	41.8	13.7	25.2
CPU time	1 s	15 h 19 min	13 min				

Bold values indicate the best solution for each instance

Table 12 Results for a given fixed initial inventory for a case in which 50% of the storage capacity, can be used for a product spoilage period of 41 days

Instance	BA	MIP	VNS				
			Average	Minimum	MIP−BA %	MIP−VNS %	VNS−BA%
1	3,244.2	**2,071.1**	2,222.5	2,202.4	56.6	7.3	45.97
2	4,196.0	**3,043.2**	3,312.9	3,230.9	37.9	8.9	26.66
3	4,793.5	**3,442.7**	3,510.8	3,486.2	39.2	2.0	36.53
4	4,418.5	**3,051.89**	3,366.7	3,329.5	44.8	10.3	31.24
5	4,326.6	3,116.5	**2,749.7**	2,711.7	38.8	−11.8	57.35
6	4,353.7	**2,981.9**	3,343.4	3,291.3	46.0	12.1	30.22
7	4,432.0	**3,213.3**	3,666.8	3,646.1	37.9	14.1	20.87
8	4,230.0	**2,792.5**	2,863.6	2,843.2	51.5	2.5	47.72
9	4,282.7	**2,942.2**	3,199.0	3,146.2	45.6	8.7	33.88
Average	4,253.0	2,961.7	3,137.3	3,098.6	44.3	6.0	36.7
CPU time	1 s	16 h 42 min	12 min				

Bold values indicate the best solution for each instance

Table 13 Results for a given fixed initial inventory for a case in which 100% of the storage capacity, can be used for a product spoilage period of 11 days

Instance	BA	MIP	VNS				
			Average	Minimum	MIP−BA %	MIP−VNS %	VNS−BA%
1	3,283.8	**2,572.8**	2,585.6	2,583.6	27.6	0.5	27.01
2	3,890.9	2,776.4	**2,687.6**	2,627.8	40.1	−3.2	44.77
3	3,784.6	**2,826.4**	3,205.5	3,143.4	33.9	13.4	18.07
4	3,774.3	2,911.0	**2,904.7**	2,852.0	29.7	−0.2	29.93
5	3,734.2	**2,847.7**	2,854.9	2,851.3	31.1	0.3	30.80
6	3,833.7	**2,815.1**	2,877.2	2,875.8	36.2	2.2	33.24
7	3,973.2	**3,021.0**	3,237.9	3,216.8	31.5	7.2	22.71
8	3,498.8	**2,711.9**	2,855.4	2,846.0	29.0	5.3	22.53
9	3,777.9	**2,778.3**	2,845.3	2,768.8	36.0	2.4	32.78
Average	3,727.9	2,806.7	2,894.9	2,862.8	32.8	3.1	29.1
CPU time	1 s	8 h 14 min	14 min				

Bold values indicate the best solution for each instance

Table 14 Results for a given fixed initial inventory for a case in which 100% of the storage capacity, can be used for a product spoilage period of 41 days

Instance	BA	MIP	VNS				
			Average	Minimum	MIP- BA %	MIP−VNS %	VNS−BA%
1	2,011.5	1,534.8	**1,520.7**	1,517.5	31.1	−0.9	32.27
2	2,918.1	1,960.6	**1,933.2**	1,909.3	48.8	−1.4	50.95
3	3,289.9	2,090.0	**1,829.5**	1,815.8	57.4	−12.5	79.83
4	3,003.1	**1,812.7**	1,839.1	1,749.0	65.7	1.5	63.29
5	2,897.9	1,776.4	**1,640.2**	1,617.0	63.1	−7.7	76.68
6	2,998.0	1,820.7	**1,813.3**	1,793.9	64.7	−0.4	65.34
7	3,258.7	1,931.0	**1,920.0**	1,898.5	68.8	−0.6	69.73
8	2,605.3	1,704.0	**1,653.4**	1,641.8	52.9	−3.0	57.57
9	2,840.9	1,723.5	**1,687.6**	1,671.6	64.8	−2.1	68.34
Average	2,869.3	1,817.1	1,759.7	1,734.9	57.5	−3.0	62.7
CPU time	1 s	14 h 10 min	10 min				

Bold values indicate the best solution for each instance

conclusions from these experiments reinforce our earlier observations. The integer programming approach performs better in tightly constrained environments and the variable neighborhood search approach performs better in relatively unconstrained environments.

References

Alshamrani A, Mathur K, Ballou H (2007) Reverse logistics: simultaneous design of delivery routes and returns strategies. Comput Oper Res 34:347–368

Bianchi L, Campbell A (2007) Extension of the 2-p-opt and 1-shift algorithms to the heterogeneous probabilistic traveling salesman problem. Eur J Oper Res 176:131–144
Braysy O (2002) A reactive variable neighborhood search for the vehicle-routing problem with time windows. INFORMS J Comput 15:347–368
Campbell A, Clarke L, Savelsbergh M (2001). Inventory routing in practice. In: Toth P, Vigo D (eds) The vehicle routing problem. SIAM, Philadelphia, pp 309–330
Clarke G, Wright JW (1964) Scheduling of vehicles from a central depot to a number of delivery points. Oper Res 12:568–581
Gendreau M, Laporte G, Potvin J-Y (1997) Vehicle routing: modern heuristics. In: Aarts E, Lenstra J (eds) Local search in combinatorial optimization. Wiley, Chichester
Hansen P, Mladenović N (2000) Variable neighborhood search. In: Pardalos PM, Resende MGC (eds) Handbook of applied optimization. Oxford University Press, New York, pp 221–234
Hansen P, Mladenovic N (2001) Variable neighborhood search: principles and applications. Eur J Oper Res 130:449–467
Kindervater GAP, Savelsbergh M (1997) Vehicle routing: handling edges exchanges windows. In: Aarts E, Lenstra J (eds) Local search in combinatorial optimization. Wiley, Chichester
Kirkpatrick S, Gelatt C Jr, Vecchi M (1983) Optimization by simulated annealing. Science 220:671–680
Mladenovic N, Hansen P (1997) Variable neighborhood search. Comput Oper Res 24:1097–1100
Pierskalla W (2004) Supply chain management of blood banks. In: Brandeau ML, Sainfort F, Pierskalla WP (eds) Operations research and health care; a handbook of methods and applications. International series in operations research and management science. vol 70. Kluwer Academic, Boston, Chapter 5, p 103
Rytilä J, Spens K (2006) Using simulation to increase efficiency in blood supply chains. Manage Res News 29:801–819