Delivery Strategies for Blood Products Supplies

Vera Hemmelmayr[(1)], Karl F. Doerner[(1)],
Richard F. Hartl[(1)], Martin W. P. Savelsbergh[(2)]

(1) Department of Business Administration, University of Vienna,
Bruenner Strasse 72, 1210 Vienna, Austria
{Vera.Hemmelmayr, Karl.Doerner, Richard.Hartl}@univie.ac.at
(2) The Logistics Institute, Georgia Institute of Technology, Atlanta, GA 30332-0205, U.S.A.
martin.savelsbergh@isye.gatech.edu

Abstract

We introduce a problem faced by an Austrian blood bank: 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.

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 has to be managed carefully. Blood products are crucial for hospitals as they are required for surgeries and for the treatment of patients with chronic illnesses, e.g., cancer patients. As a consequence, blood products are delivered to hospitals on a regular basis in order 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 at the customer, but at the same time spoilage of products has to be

kept at 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 for Eastern Austria. The blood bank serves 60 hospitals and makes regular deliveries to these hospitals. About 250,000 blood products are sold and delivered every year. 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. Hospitals that have requested a delivery of blood products the previous day are visited in the order of 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. ([3]).

Our study provides partial answers to these questions. We investigate alternative delivery strategies, where we are mindful of the fact 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. To obtain an indication of the potential of these delivery strategies, we investigate a somewhat simplified problem 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. As the size of a bag with a blood product is very small, vehicle capacity is never restricting and therefore ignored. 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 two weeks. To handle this usage diversity, we require a delivery schedule to cover a two week period. Such a delivery schedule can then be repeated every two weeks, or a rolling horizon approach can be developed.

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 cost reduction of about 30%. On the other hand, 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, where as in less constrained situations focusing on regular delivery patterns gives slightly better results.

The remainder of the paper is organized as follows. In Section 2, we discuss three solution approaches that differ in terms of underlying philosophy, development time, and performance. In Section 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 Section 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 delivery routes for a day are determined using a vehicle routing algorithm. 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. In order 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. The vehicle routing problems are solved using the savings algorithm ([4]) followed by a local improvement procedure based on move, exchange, and 3-opt operators (see Gendreau et al. [11] and Kindervater and Savelsbergh [12] for a discussion of local search for vehicle routing problems).

2.2 An Integer Programming Approach

The integer programming based 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. However, the hospitals are not served every day and the hospitals are not always served together, i.e., the delivery route may 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 the distribution center and visiting n hospitals. Denote the fixed route by $(0, 1, ..., n + 1)$, where 0 and $n + 1$ denote the distribution center, and $1, ..., n$ denote the hospitals (in the order in which they are visited by the fixed route).

Furthermore, let y_i denote whether hospital i is include 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

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

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 ([1]).

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.

To be more precise, let the number of hospitals be denoted by n and the number of days in the planning period be denoted by T. Let z^t for $t = 1, ..., T$ be a 0-1 variable indicating whether or not a route is executed on day t, let y_i^t for $i = 1, ..., n$ and $t = 1, ..., T$ be a 0-1 variable indicating whether or not a hospital is visited on day t, let d_i^t for $i = 1, ..., n$ and $t = 1, ..., T$ be a continuous variable indicating the quantity of blood delivered to hospital i on day t, let I_i^t for $i = 1, ..., n$ and $t = 1, ..., T + 1$ be a continuous variable indicating the quantity of blood in inventory at hospital i at the beginning of day t, and let x_{ij}^t for $i = 0, ..., n, j = 1, ..., n + 1$ and $t = 1, ..., T$ be 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). Furthermore, let the travel time between two locations i and j be denoted by t_{ij}, the service time at location i be denoted by s_i, the product usage at hospital i on day t be denoted by u_i^t, the initial inventory at the beginning of the first day at hospital i be denoted by I_i^1, and the storage capacity at hospital i be denoted by C_i. A maximum route duration is given by D. Finally, assume that the safety stock policy at every hospital is to ensure that at least k_1 days of inventory is available at the beginning of the day, that blood products spoil in k_2 days, and that the inventory at every hospital at the end of the planning period has to be at least as large as the initial inventory at the beginning of the planning period.

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

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

subject to

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

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

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

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

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

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

$$z_t \geq y_i^t \quad i = 1, ..., n,\ t = 1, ..., T \tag{7}$$

$$x_{ij}^t \geq y_i^t + y_j^t - 1 - \sum_{k=i+1}^{j-1} y_k^t \quad i = 0, ..., n,\ j = 1, ..., n+1,\ t = 1, ..., T \tag{8}$$

$$\sum_{i=0}^{n+1} \sum_{j=0}^{n+1} t_{ij} x_{ij}^t + \sum_{i=1}^{n} y_i^t s_i \leq D \quad t = 1, ..., T. \tag{9}$$

$$y_0^t = y_{n+1}^t = 1 \quad t = 1, ..., T. \tag{10}$$

Constraints (1) ensure inventory balance from period to period. Constraints (2) ensure the safety stock requirements. 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) enforce that we can easily identify the days on which a delivery route is executed. Constraints (8) capture short cutting of the fixed route. Finally, constraints (9) limit route duration.

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

- We perturb the objective function and give a slight preference to making deliveries earlier in the planning period, i.e.,

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

where ϵ is a small positive constant.

- For each hospital i, we compute the minimum number of deliveries M_i that have to be made to that hospital and impose

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

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

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

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 Figure 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 [5] and Hansen and Mladenovic [7] 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 $\kappa \leftarrow \kappa + 1$;

Figure 1: Steps of the VNS (c. f. [7])

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, using the savings algorithm ([4]), and taking all customers in a route to form a cluster. 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 algorithms 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, a 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. (The cross-exchange neighborhood was successfully used in a VNS for the vehicle routing problem with time windows ([2])). 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 many of 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 1 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.

κ	operator	min. segment length	max. 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)
		min. customers	max. 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

Table 1: Set of Neighborhood Structures with $\kappa_{max} = 15$

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 re-optimized). 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. In order to avoid getting trapped in bad local optima it may be beneficial in some situations to also accept non-improving solutions. Hansen and Mladenovic [6] 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 ([8]),

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 given temperature T and the difference between the new solution and the incumbent solution. The temperature T is decreased during search process.

As mentioned at the start 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 constraint. This penalty function is added to the objective function before the solution is evaluated for acceptance. The weight of each time unit is fixed and equal to 1000. Because of the high weight there is a strong bias towards feasible solutions.

3 Computational Experiments

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 2). Eight of the data sets are derived from a month 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 worth of data and 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 5 different classes according to the deviation of the average demand.

We determine delivery routes for a period of 14 days. Considering a two-week period is necessary to ensure driver availability. Drivers are not dedicated to blood deliveries, but are also used for other services, e.g., blood collection from mobile blood collecting units, emergency deliveries of blood, etc. Some drivers are free lancers and need to be informed

Table 2: Real World Instances

instance number	**month**	**deviation**	**class**
1	jan	-44.00%	lowest
2	feb	5.14%	high
	mrz	2.10%	medium
	apr	8.00%	high
	mai	3.24%	medium
3	jun	12.57%	highest
4	jul	8.76%	high
5	aug	-1.71%	medium
6	sept	5.71%	medium
7	oct	7.62%	high
	nov	-3.81%	medium
8	dec	-4.76%	low
9	average	0.00%	medium

a certain period in advance. Furthermore, many medium-sized enterprises with a private fleet for the delivery of products, like the ARC, do not want to recompute delivery routes on a daily basis. The ARC has to fulfill the demands of 55 hospitals in eastern Austria.

Due to 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 some key problem parameters, i.e., 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).

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 two hours and the best solution found is reported. 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 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 Table 3 and Table 4, 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 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 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 planing 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 on 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

Table 3: 75% of capacity, 11 days of spoilage, given initial inventory

Instance	BA	MIP	VNS		MIP- BA %	MIP- VNS %	VNS-BA%
			avg.	min.			
1	3182.3	**2572.8**	2614.3	2611.3	23.7%	1.6%	21.73%
2	4041.5	**2929.4**	3006.2	2997.5	38.0%	2.6%	34.44%
3	4267.9	**2973.4**	3499.2	3473.8	43.5%	17.7%	21.97%
4	4012.4	**2824.5**	3193.9	3159.1	42.1%	13.1%	25.63%
5	3905.3	**2843.8**	3085.6	3010.2	37.3%	8.5%	26.56%
6	3968.1	**2931.2**	3190.1	3126.1	35.4%	8.8%	24.39%
7	4197.5	**2992.2**	3301.4	3223.8	40.3%	10.3%	27.14%
8	3808.7	**2867.6**	3159.8	3116.5	32.8%	10.2%	20.54%
9	3945.4	**2754.9**	2994.8	2958.0	43.2%	8.7%	31.74%
avg.	3925.4	2854.4	3116.1	3075.1	37.4%	9.1%	26.0%
cpu time	1 sec	11hr 34min	13 min				

Table 4: 75% of capacity, 41 days of spoilage, given initial inventory

Instance	BA	MIP	VNS		MIP- BA %	MIP- VNS %	VNS-BA%
			avg.	min.			
1	2453.9	**1509.0**	1496.1	1491.9	62.6%	-0.9%	64.02%
2	3435.6	**2305.5**	2334.4	2273.0	49.0%	1.3%	47.17%
3	3807.6	**2589.8**	2729.1	2711.5	47.0%	5.4%	39.52%
4	3390.3	**2338.6**	2416.8	2376.7	45.0%	3.3%	40.28%
5	3109.0	2192.6	**2057.5**	2037.3	41.8%	-6.2%	51.11%
6	3390.4	2242.3	**2154.6**	2141.6	51.2%	-3.9%	57.36%
7	3468.3	2491.8	**2339.3**	2325.9	39.2%	-6.1%	48.26%
8	2888.6	1946.2	**1931.1**	1920.6	48.4%	-0.8%	49.58%
9	3090.4	2294.0	**2037.9**	2029.1	34.7%	-11.2%	51.65%
avg.	3226.0	2212.2	2166.3	2145.3	46.6%	-2.1%	49.9%
cpu time	1 sec	15hr13min	10 min				

decision variable. In the variable neighborhood search approach, we first calculate a set of reasonable initial inventory levels (e.g., the usage in one day, the usage in two 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 Table 5 and Table 6. 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 periodic visit combinations is not as 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 7, a summary of the deviations between the basic approach, the integer programming approach, and the variable neighborhood search approach are presented. It demonstrates that the basic approach delivers poor results in all cases, the integer programming approach is effective in more tightly constraint environments, and that variable neighborhood search approach performs well in relatively unconstraint environments.

Table 8 shows the average number of hospital visits. The VNS results in slightly fewer visits. Table 9 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 a VMI concepts in the blood delivery strategies of the Austrian Red Cross. We have designed and implemented two alternative solutions approaches. Both approaches still 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 reduction; 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 considering due to the politics involved.

Independent of whether the ARC will move forward with the implementation of a VMI strategy, we will continue to study the problem focusing on the uncertainty associated

Table 5: 75% of capacity, 11 days of spoilage, free initial inventory

Instance	MIP	VNS		MIP-VNS %
	cost	average	min	
1	2655.2	**2591.1**	2589.9	-2.41%
2	2676.9	**2637.5**	2630.5	-1.47%
3	**2810.8**	2893.6	2881.9	2.95%
4	**2780.1**	2892.2	2861.1	4.03%
5	2774.6	**2628.5**	2599.9	-5.27%
6	**2779.3**	2885.9	2863.7	3.83%
7	**2962.8**	3242.9	3221.4	9.45%
8	2695.0	**2591.1**	2588.3	-3.86%
9	2658.0	**2637.3**	2619.6	-0.78%
avg.	2754.8	2777.8	2761.8	0.7%
cpu time	17hr 20min	13 min		

Table 6: 75% of capacity, 41 days of spoilage, free initial inventory

Instance	MIP	VNS		MIP-VNS %
	cost	average	min	
1	1330.1	**1323.5**	1312.7	-0.50%
2	2107	**1980.8**	1907.5	-5.99%
3	**2455.2**	2515.3	2491.3	2.45%
4	2211.3	**2170.9**	2112.7	-1.83%
5	2069.6	**1900.2**	1883.3	-8.19%
6	2066.2	**1924.2**	1907.4	-6.87%
7	2449.8	**2301.6**	2251.8	-6.05%
8	1830.5	**1765.7**	1740.4	-3.54%
9	2084.4	**1778.3**	1756.1	-14.68%
avg.	2067.1	1962.3	1929.2	-5.0%
cpu time	17hr 11min	10 min		

Table 7: Summary

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 8: Average number of visits

Capacity-Spoilage	Given Initial Inventory		Initial Inventory as Variable	
	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
avg.	98	94	95	93

Table 9: Average number of routes

Capacity-Spoilage	Given Initial Inventory		Initial Inventory as Variable	
	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
avg.	12	11	11	10

with the use of blood products, i.e., the investigation of stochastic or robust optimization techniques.

Acknowledgments

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.

References

[1] Bianchi, L. and A. Campbell. "Extension of the 2-p-opt and 1-shift algorithms to the Heterogeneous Probabilistic Traveling Salesman Problem," European Journal of Operational Research, to appear.

[2] Braysy, O. "A Reactive Variable Neighborhood Search for the Vehicle-Routing Problem with Time Windows." *INFORMS Journal on Computing* **15**, 347–368 (2002).

[3] Campbell, A., L.W. Clarke, and M.W.P. Savelsbergh. Inventory Routing in Practice. In "The Vehicle Routing Problem," P. Toth and D. Vigo (eds.), SIAM Monographs on Discrete Mathematics and Applications, 309–330 (2002).

[4] Clarke, G. and J.W. Wright. "Scheduling of vehicles from a central depot to a number of delivery points." Operations Research **12**, 568-81 (1964).

[5] Mladenović, N. and P. Hansen. "Variable Neighborhood Search." *Computers and Operations Research* **24**, 1097–1100 (1997).

[6] Hansen, P. and Mladenović, N. (2000). "Variable Neighborhood Search" In: Pardalos, P. M. and Resende, M. G. C. (eds.): *Handbook of Applied Optimization*, Oxford University Press, New York, pp. 221–234.

[7] Hansen, P. and N. Mladenović. "Variable Neighborhood Search: Principles and Applications." *European Journal of Operational Research* **130**, 449–467 (2001).

[8] Kirkpatrick, S., C.D. Gelatt Jr., and M.P. Vecchi. "Optimization by Simulated Annealing," *Science* 220, 4598, 671–680 (1983).

[9] Polacek, M., R.F. Hartl, K. Doerner, and M. Reimann. "A Variable Neighborhood Search for the Multi Depot Vehicle Routing Problem with Time Windows." *Journal of Heuristics* **10**, 613–627 (2004).

[10] Polacek, M., K. Doerner, R.F. Hartl, G. Kiechle, and M. Reimann. "Scheduling periodic customer visits for a Travelling Salesperson." *European Journal of Operational Research*, to appear.

[11] Gendreau, M., G. Laporte, and J.-Y. Potvin (1997). Vehicle routing: modern heuristics. In "Local Search in Combinatorial Optimization" E. Aarts and J.K. Lenstra (eds.), John Wiley & Sons Ltd., Chichester.

[12] Kindervater, G. A. P. and M. W. P. Savelsbergh (1997). Vehicle routing: handling edges exchanges windows. In "Local Search in Combinatorial Optimization" E. Aarts and J.K. Lenstra (eds.), John Wiley & Sons Ltd., Chichester.

Appendix

In this appendix, we report the results for the experiments in which we the storage capacity at the hospitals is set at 50% (Table 10 and Table 11) and at 100% (Table 12 and Table 13). The conclusions from these experiments reinforce our earlier observations. The integer programming approach performs better in tightly constraint environments and the variable neighborhood search approach performs better in relatively unconstraint environments.

Table 10: 50% of capacity, 11 days of spoilage, given initial inventory

Instance	BA	MIP	VNS		MIP- BA %	MIP- VNS %	VNS-BA%
			avg.	min.			
1	3935.4	**2709.8**	2997.9	2962.5	45.2%	10.6%	31.27%
2	4514.0	**3310.9**	3802.5	3743.2	36.3%	14.8%	18.71%
3	4980.8	**3569.1**	3674.9	3628.9	39.6%	3.0%	35.54%
4	4725.2	**3352.4**	3830.0	3801.5	41.0%	14.2%	23.37%
5	4591.5	**3178.8**	3237.4	3193.6	44.4%	1.8%	41.83%
6	4595.7	**3180.2**	3867.2	3839.2	44.5%	21.6%	18.84%
7	4802.7	**3553.7**	4300.7	4260.3	35.1%	21.0%	11.67%
8	4581.0	**3110.8**	3622.3	3601.7	47.3%	16.4%	26.47%
9	4417.7	**3088.0**	3704.2	3695.0	43.1%	20.0%	19.26%
avg.	4571.6	3228.2	3670.8	3636.2	41.8%	13.7%	25.2%
cpu time	1 sec	15hr 19min	13 min				

Table 11: 50% of capacity, 41 days of spoilage, given initial inventory

Instance	BA	MIP	VNS		MIP- BA %	MIP- VNS %	VNS-BA%
			avg.	min.			
1	3244.2	**2071.1**	2222.5	2202.4	56.6%	7.3%	45.97%
2	4196.0	**3043.2**	3312.9	3230.9	37.9%	8.9%	26.66%
3	4793.5	**3442.7**	3510.8	3486.2	39.2%	2.0%	36.53%
4	4418.5	**3051.89**	3366.7	3329.5	44.8%	10.3%	31.24%
5	4326.6	3116.5	**2749.7**	2711.7	38.8%	-11.8%	57.35%
6	4353.7	**2981.9**	3343.4	3291.3	46.0%	12.1%	30.22%
7	4432.0	**3213.3**	3666.8	3646.1	37.9%	14.1%	20.87%
8	4230.0	**2792.5**	2863.6	2843.2	51.5%	2.5%	47.72%
9	4282.7	**2942.2**	3199.0	3146.2	45.6%	8.7%	33.88%
avg.	4253.0	2961.7	3137.3	3098.6	44.3%	6.0%	36.7%
cpu time	1 sec	16hr 42min	12 min				

Table 12: 100% of capacity, 11 days of spoilage, given initial inventory

Instance	BA	MIP	VNS		MIP- BA %	MIP- VNS %	VNS-BA%
			avg.	min.			
1	3283.8	**2572.8**	2585.6	2583.6	27.6%	0.5%	27.01%
2	3890.9	2776.4	**2687.6**	2627.8	40.1%	-3.2%	44.77%
3	3784.6	**2826.4**	3205.5	3143.4	33.9%	13.4%	18.07%
4	3774.3	2911.0	**2904.7**	2852.0	29.7%	-0.2%	29.93%
5	3734.2	**2847.7**	2854.9	2851.3	31.1%	0.3%	30.80%
6	3833.7	**2815.1**	2877.2	2875.8	36.2%	2.2%	33.24%
7	3973.2	**3021.0**	3237.9	3216.8	31.5%	7.2%	22.71%
8	3498.8	**2711.9**	2855.4	2846.0	29.0%	5.3%	22.53%
9	3777.9	**2778.3**	2845.3	2768.8	36.0%	2.4%	32.78%
avg.	3727.9	2806.7	2894.9	2862.8	32.8%	3.1%	29.1%
cpu time	1 sec	8hr 14min	14 min				

Table 13: 100% of capacity, 41 days of spoilage, given initial inventory

Instance	BA	MIP	VNS		MIP- BA %	MIP- VNS %	VNS-BA%
			avg.	min.			
1	2011.5	1534.8	**1520.7**	1517.5	31.1%	-0.9%	32.27%
2	2918.1	1960.6	**1933.2**	1909.3	48.8%	-1.4%	50.95%
3	3289.9	2090.0	**1829.5**	1815.8	57.4%	-12.5%	79.83%
4	3003.1	**1812.7**	1839.1	1749.0	65.7%	1.5%	63.29%
5	2897.9	1776.4	**1640.2**	1617.0	63.1%	-7.7%	76.68%
6	2998.0	1820.7	**1813.3**	1793.9	64.7%	-0.4%	65.34%
7	3258.7	1931.0	**1920.0**	1898.5	68.8%	-0.6%	69.73%
8	2605.3	1704.0	**1653.4**	1641.8	52.9%	-3.0%	57.57%
9	2840.9	1723.5	**1687.6**	1671.6	64.8%	-2.1%	68.34%
avg.	2869.3	1817.1	1759.7	1734.9	57.5%	-3.0%	62.7%
cpu time	1 sec	14hr 10min	10 min				