ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

TRANSP-OR

Semester Project

Evaluating the effect of better forecasts in a rich inventory routing problem

Author:	Prisca Aeby [1]	prisca.aeby@epfl.ch
Supervisors:	Iliya Markov [2]	iliya.markov@epfl.ch
	Yousef Maknoon [2]	yousef.maknoon@epfl.ch
	Michel Bierlaire [2]	michel.bierlaire@epfl.ch

December 21, 2016

[1] Section of Computer Science, Master Semester 3, École Polytechnique Fédérale de Lausanne (EPFL), Switzerland

[2] Transport and Mobility Laboratory (TRANSP-OR), School of Architecture, Civil and Environmental Engineering (ENAC), École Polytechnique Fédérale de Lausanne (EPFL), Switzerland, `transp-or.epfl.ch`

Abstract

We start from an existing problem formulation and implementation of a rich stochastic inventory routing problem solved at the Transport and Mobility Laboratory at the Ecole Polytechnique Fédérale de Lausanne. The problem inspired from practice consists of a heterogeneous fleet of vehicles which are used for collecting recycable waste from containers over a finite planning horizon. A forecasting model generated from a history of observations coming from the canton of Geneva is used to estimate the probability of container overflows and route failures. The problem is solved using an adaptative large neighborhood search (ALNS) algorithm integrating the forecasting model. We analyze the algorithmic performance of the current solution implemented in Java and R and we improve it through different methods modifying and extending the code. We evaluate how better demand forecasting methodologies impact the solution cost by experimenting artifical ways to reduce the uncertainty in the forecasting of future demands. The stability of the objective function is ameliorated by improving the strategy of the search space, developing new operators for the ALNS, in order to measure accurately the impact of different uncertainty levels.

1 Introduction

The first inventory routing problem (IRP) formulations were introduced as variation of vehicle routing problem models (VRP), taking inventory costs into consideration. The research in this area has been motivated by the introduction of the Vendor Managed Inventory (VMI) technique in supply chain management 30 years ago, which aims at reducing logistic costs and adding business value (Coelho et al., 2014). The general idea consists in coordinating inventory management, vehicle routing and delivery to the customers decisions over the planning horizon. There however exists several formulations of the problem as the tendency is to include specific company functions in the integration process (Bertazzi and Speranza, 2012). In our case, we are interested in analyzing the existing solution proposed for a rich recyclable waste collection problem as it is an important logistic activity within any city which can lead to financial savings as well as pollution reduction (Buhrkal et al., 2012). The solution implementing an adaptive large neighborhood search (ALNS) has already obtained good performance on IRP benchmarks from the literature as well as IRP instances derived from real data of the canton of Geneva, Switzerland (Markov et al., 2016). The problem consists of a heterogeneous fixed fleet of vehicles with different properties which is used for collecting waste over a finite planning horizon. Each tour of a vehicle starts at a depot and is followed by a sequence of collections and disposals at the available dumps. It terminates by a mandatory visit to a dump and it returns at the same depot it started. The depots, containers and dumps have time windows. The containers are equipped with sensors which communicate the waste level at the start of each day and indicate if a container is full. Containers can still serve demand when they are full as people place the waste beside them, however they must be collected within the day with a penalty cost. The solution implements a statistical model with the help of historical data to estimate for each container the point demand forecasts for each day of the planning horizon, as well as the forecasting error used to compute the risk of container overflows and route failures which represents the level of uncertainty with regard to the true demand. The problem falls under the Stochastic Inventory Routing Problem (SIRP) with no inventory holding costs and unlimited inventory capacity at the dumps. It is solved in two main steps, namely (1) it proposes a forecasting model providing the expected daily container demands and the forecasting error (2) it implements an ALNS algorithm with simulated annealing integrating the demand forecasting model in the solution process to compute costs for container overflows and route failures (likelihood that a vehicle does not have sufficient capacity to serve the demands), which has been tested and validated on real data.

Our work consists of evaluating how different levels of uncertainty in the forecasting of future demands, artificially simulated using the true demand and the forecasting error, impacts the solution cost as the value of the objective function is affected by the probability of container overflows and route failures derived from the forecasting model. In order to measure accurately the impact of different uncertainty levels the stability of the objective function needs to be ameliorated. It is achieved by modifying the breadth of the search space that is explored in the ALNS algorithm by adding several destroy and repair operators inspired from the VRP literature.

The remainder of the article is organized as follows. Section 2 positions our work with respect to the relevant IRP literature. Section 3 formulates mathematically our SIRP. Section 4 describes the current ALNS algorithm implementation and the methodology approaches. Section 5 presents the numerical experiments. Finally, our concluding remarks are given in section 6.

2 Related IRP Literature

Over the past 30 years many variants of the IRP have been presented, it is therefore complicated to refer to one general formulation of the problem. However, Coelho et al. (2014) conduct an analysis of the basic versions and propose a structural classification scheme according to seven criteria: time horizon, structure, routing, inventory policy, inventory decisions, fleet composition and fleet size. There is a second classification related to the availability of information on the demand. We first describe the main objective of the general IRP formulation and then we position our problem according to the variants.

Objective of the Problem The main goal is to minimize the whole inventory distribution cost while meeting the demand of each customer. The solution has to respect constraints, mainly concerning the inventory level and the stockout at each customer and supplier, the capacity of the vehicles and the routing constraints. The solution of the problem determines which customers to serve and when, which vehicles to use, how much to deliver to each visited customer, and the delivery routes.

Structural Variants

- **Time horizon** It can be either *finite* or *infinite*. In our case, the problem is solved in a *finite* rolling horizon framework as it is useful in dealing with uncertainty by making forward-looking decisions. The solution is found for a period of seven to ten days.
- **Structure** It denotes the ordering relation between the number of suppliers and the number of customers. As there are multiple containers that are emptied into multiple dumps, the structure is identified as *many-to-many*.
- **Routing** It is *direct* when there is only one customer per route, *multiple* when there are several customers in the same route, or *continuous* when there is no central depot. Vehicles are taking waste from many containers during one tour so it is classified as *multiple*.
- **Inventory policy** Under the *order-up-to level* (OU) policy, each customer has defined a maximum inventory level and when he is served, the delivered quantity is such that the maximum inventory level is reached. Under the *maximum level* (ML) policy every time a customer is served, the delivered quantity is such that the inventory level at the customer is not greater than the maximum level. Visited containers are always fully emptied so our problem falls under the OU policy.
- **Inventory decisions** If *back-orders* are allowed the demand will be served at a penalty cost. If there are no back orders, then the demand is considered as *lost*

sales. The problem falls under the *back-ordering* decision as container overflow is served at a penalty and there is a limit on the number of back-order days.

- **Fleet composition and size** the fleet can be *homogeneous* or *heterogeneous*, and the number of vehicles may be *fixed at one, fixed at many*, or *unconstrained*. There is a *heterogeneous fixed* fleet of vehicles.

Availability of Information on Demand

- **Deterministic** The information is fully available at the beginning of the planning horizon.
- **Stochastic** The probability distribution of the information is known.
- **Dynamic** The demand is not fully known in advance but gradually revealed over time.

Our problem falls under the *stochastic* category information-wise and under the *dynamic* category with new container information revealed each day in a rolling horizon fashion.

3 Problem Formulation

In what follows, section 3.1 explains the derivation of the overflow probabilities, section 3.2 formulates the objective function and section 3.3 describes the constraints.

3.1 Derivation of the Overflow Probabilities

There are no inventory holding costs at the containers or dumps. However, if a container is overflowing on day t an emergency collection occurs and empties the container in question. Two costs are added to the objective function: χ has a monetary value which the collector bears and ζ is a parameter representing the average actual cost of emergency collection. We are interested in the probability of overflow at container i on day t. In order to compute this probability we need to know the expected demand for a container i on day t. This is given by the forecasting model proposed by Markov et al. (2016), which minimizes the sum of squared errors between the observed demand and the expected demand $E(\rho_{it})$ over the set of containers $\mathcal{P}$ and a period $\mathcal{H}$ of data. The error of the forecasting model is represented by a white noise ε_{it} (independent and identically distributed random variable following a normal distribution) which is added to the expected demand:

$$\rho_{it} = E(\rho_{it}) + \varepsilon_{it}$$

For the first day of the planning horizon, the probability that a container overflows is simply given by $P(I_{i0} + \rho_{i0} \geq \omega_i)$. For a container which starts with a zero inventory (it can be either on the first day or at a state of overflow because the inventory is set to zero by the emergency collection) the probability is given by $P(0 + \rho_{ih} \geq \omega_i)$, $\forall h \in \mathcal{T}$. In the other cases, we need the conditional probability that a container overflows on day h: it is the probability that the initial inventory on day g plus the expected demand between

day g and t exceeds the container capacity, knowing that the inventory of the container for the preceding days is smaller than its capacity. It can be formulated as:

$$P(I_{ig} + \sum_{t=g}^{h} \rho_{it} \geq \omega_i | I_{ig} + \sum_{t=g}^{h-1} \rho_{it} \leq \omega_i)$$

This probability gives only the probability of container overflow at day h given the state of day $h-1$. In order to compute the probability of a container overflow at day h, we need to develop a binary state probability tree for all scenarios leading to a container overflow at day h. Day 0 represents the root of the tree. Each node represents either a state of overflow or not and its two children the next two possible scenarios on the following day. We multiply the probability of each branch of the tree leading to a state of overflow at day h and sum up those probabilities to get the probability of overflow at day h.

3.2 Objective Function

The function z which needs to be minimized includes the following costs: the Expected Overflow and Emergency Collection Cost (EOECC), the Routing Cost (RC) and the Expected Route Failure Cost (ERFC):

$$\min(z) = \text{EOECC} + \text{RC} + \text{ERFC}$$

The EOECC is the sum for each point over the planning horizon of the probability of a container overflow times the overflow cost χ and the emergency cost ζ in case there is no regular collection on that day. The RC is a deterministic expression computing the cost of each vehicle used over the planning horizon, taking into account its daily deployment cost φ_k, the unit-distance running cost θ_k in CHF/km and the unit-time running cost β_k in CHF/hour. The RC for one vehicle is therefore the sum over the planning horizon of φ_k if the vehicle is used during the day, plus the distance traveled by the vehicle times θ_k and the service time of the vehicle times β_k. The ERFC reflects costs due to unscheduled trips to the dumps because of the vehicles' inability due to insufficient capacity to serve the containers on the scheduled depot-to-dump or dump-to-dump trips. This happens when the sum of the container inventories collected by the vehicle has been underestimated, due to the uncertainty in the forecasting model.

3.3 Constraints

Several constraints make sure the variables can only take values which lead to a feasible solution given our formulation of the problem. It checks that only available vehicles are used and that the ones executing a tour start at the origin and end with a visit to a dump directly before the destination. Vehicles should not visit inaccessible points and the sum of the quantities they've picked up should not exceed their capacity. Each container has to be visited by at most one vehicle on a given day. Inventory constraints track the container inventories by linking them to the vehicle visits and the pickup quantities. The intra-day temporal constraints make sure ime windows and maximum tour duration are respected.

4 Adaptive Large Neighborhood Search

The ALNS is a type of large neighborhood search (LNS) heuristic which modifies the current solution by a destroy operator removing a number of customers which are reinserted by a repair operator in order to explore neighbors and find better solutions. In what follows section 4.1 details the ALNS algorithm. Section 4.2 presents the operators that were already used in the implementation of Markov et al. (2016) and section 4.3 describes the newly operators implemented.

4.1 Algorithm

During the ALNS process a neighbor is constructed at each iteration and is accepted based on any metaheuristic framework. Markov et al. (2016) use as an acceptance criterion Simulated Annealing because it appears to be the preferred approach in the ALNS literature. Given an incumbent solution s, a neighbor solution s' is randomly drawn by the ALNS algorithm and is only accepted if $f(s') < f(s)$ or with probability $e^{-(f(s')-f(s))/T}$, $f(s)$ repsensenting the solution cost and $T > 0$ the current temperature. The temperature is initialized with a given value T^{start} and is decreased at each iteration by a cooling rate $r \in (0, 1)$. The search stops when T reaches a given threshold T^{end}. The idea of the ALNS algorithm is to remove at each iteration of the search process a number of containers or dumps from the current solution by a destroy operator and to reinsert them elsewhere by a repair operator in order to find a neighbor of the current solution. In our problem, not all containers or dumps need to be visited every day so they don't need to be all reinserted by the repair operator. The operators to be used in each iteration are chosen from the set of destroy/repair operators $\mathcal{O}$ using a roulette wheel mechanism. An operator i is chosen with probability $W_i / \sum_{j \in \mathcal{O}} W_j$. The weight W_i depends on its past performance and a score. At the beginning the weights and scores are set to 1 and 0 respectively. The entire search is divided into segments of F iterations and the weights are updated at the end of each segment. The scores are increased by e_1 if the operators find a new best global feasible solution, by $e_2 < e_1$ if they improve the incumbent solution or by $e_3 < e_2$ if a new solution is accepted by the simulated annealing acceptance criterion. If a destroy-repair couple leads to a visited solution no reward is applied. Let C_i^F denote the score and N_i^F the number of times operator i was applied during the segment of length F. The new weight W_i doesn't change if $N_i^F = 0$ and otherwise is replaced by $(1-b)W_i + bC_i^F/(m_i N_i^F)$, $b \in (0, 1)$ denoting the reaction factor and m_i a normalization factor. The reaction factor b controls the relative effect of past performance and the scores on the new weights. The normalization factor m_i reduces the weights of computationally expensive operators. This factor is set experimentally based on its effect on computation time.

The performance and robustness of the overall heuristic is very dependent on the choice of destroy and repair operators. Local search heuristics are often built on neighborhood moves that make small changes to the current solution, leading to difficulties in moving from one local optimum of the solution space to another when faced with tightly constrained problems like ours (Ropke and Pisinger, 2006a). The way to avoid those local optima is to allow the search to visit infeasible solutions by relaxing some constraints. A dynamically adjusted penalty for each type of violation is introduced in the cost of the complete solution $f(s)$ during the search.

4.2 Heuristics used in the solution

In what follows we list the operators already implemented by Markov et al. (2016) in the solution we start from.

Destroy operators Remove ν containers from a random tour, remove ν containers that would lead to the largest savings, empty a random day, empty a random vehicle, remove a random dump, remove the worst dump, remove consecutive visits of the same container and shaw removal which selects a random container i and removes the containers close to it in term of distance.

Repair operators Insert ν containers randomly, insert ν random containers in the best way, swap ν random containers, insert a random dump in a random position, insert a random dump in the best way, swap random dumps, replace a random dump by a random dump, reorder the dumps in the best way and shaw insertions which inserts containers closed to each other not visited during a random day.

4.3 Heuristics introduced

In what follows we describe the new operators implemented in our work inspired from the IRP and VRP literature. We pay attention to use efficient algorithms as well as appropriate data structures and caching in our Java implementation to obtain reasonable computational times.

4.3.1 Insert Best Containers

The operator already implemented selects ν random containers from $\mathcal{P}$ and insert them in the tour and position that would lead to the smallest increase in the objective function. We modify this insert heuristic by inserting the ν best containers. The best position of each container from $\mathcal{P}$ and the corresponding cost increase are calculated and then the ν containers with the smallest cost increase are inserted in their best position.

4.3.2 Container insertion with regret

Regret heuristics have been used by Potvin and Rousseau (1993) for the VRPTW. They have been implemented as well by Buhrkal et al. (2012) in the context of waste collection vehicle routing problem with time windows in a city logistics context. The principle is to introduce a look-ahead information when selecting the container to insert. Let $R_{ik} \in \{1, ..., m\}$ be a variable that indicates the route for which inserting the container i has the k^{th} lowest insertions cost: $\Delta f_{i,R_{ik}} \leq \Delta f_{i,R_{ik'}}$ for $k \leq k'$. We define a *regret value* $c_i^* = \Delta f_{i,R_{i1}} - \Delta f_{i,R_{ik}}$, it is the difference in inserting the container in its best route and its kth best route. We choose to insert the container i which maximizes c_i^*, in other words the one we will regret the most if we don't insert it now. We insert the request i at its minimum cost position. We can run it with different values of k. Ties are broken by inserting the request i in the route with the lowest insertion cost. If we cannot compute the regret k for a container because it can't be inserted in k different positions, we compute the regret for the k^{max} we find. In order to choose the next container to insert, we first

consider the ones with the smallest k^{max}. The ν first containers are inserted in their best position.

4.3.3 Shaw removal of related containers

This removal has been inspired by the heuristic presented by Ropke and Pisinger (2006a) in which they solve the pickup and delivery problem with time windows. Instead of computing only the distance between two containers, we introduce a new measure: the relatedness $R_{i,j}$. The lower $R_{i,j}$ is, the more related the two containers are. The relatedness chosen for our problem consists of three terms: a distance term, a time term and an overflow probability term. These terms are weighted using the weights a, b and c respectively. The relatedness measure is given by $R_{i,j} = a(\pi_{i,j}) + b(|\lambda_i - \lambda_j| + |\mu_i - \mu_j|) + c(|o_{i,d} - o_{j,d}|)$. The term $\pi_{i,j}$ denotes the distance between the container i and container j, μ_i and λ_i the upper and lower time window bound of container i and $o_{i,d}$ the overflow probability of container i on day d. The three terms are normalized between 0 and 1 by feature scaling $(x - min)/(max - min)$. The algorithm proceeds as follows: it picks a random tour, a random container in this tour and it computes the relatedness of all other containers from the tours scheduled the same day. The relatedness is then normalized again by feature scaling. The container is removed if the relatedness is under a given threshold.

4.3.4 Shaw insertion of related containers

The relatedness measure $R_{i,j}$ between two containers is the same as the one described in section 4.3.3. The idea is to insert containers which are related to each other on the same day of the planning horizon. The algorithm proceeds as follows: it selects a random day and a random container from the ones not visited on this day, it finds all containers not visited which are below the relatedness measure threashold and inserts those containers in their best position considering all tours scheduled on the random day.

4.3.5 Cluster removal of containers using Kruskal algorithm

This clustering technique removal has been inspired by the solution implemented by Ropke and Pisinger (2006b) for vehicle routing problems with backhauls. The motivation for the heuristic is to remove large chunks of related requests instead of removing a few requests from each route. The idea is to select a random day of the planning horizon and to divide the containers visited during this day into k clusters. We choose k to be the number of routes scheduled this day. If there is only one route, we divide the route into 2 clusters. We then choose a random cluster to remove. The size of the cluster has to be smaller than half the number of containers scheduled during the selected random day in order to avoid removing too many containers from one day.
Kruskal's algorithm works as follows: initially, each container is in its own cluster. Then, it considers each edge (there is an edge between all containers) in turn, order by increasing distance. If an edge (i, j) connects two different clusters, then (i, j) is added to the set of edges of the minimum spanning tree, and two clusters connected by an edge (i, j) are merged into a single cluster. On the other hand, if an edge (i, j) connects two containers in the same cluster, then edge (i, j) is discarded. More formally, for a graph $\mathcal{G}$ containing $\mathcal{V}$ vertices (containers) and $\mathcal{E}$ edges (between all containers), the algorithm is:

Data: A graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, number of clusters k
Result: k clusters generated by Kruskal's algorithm
$\mathcal{T} = \emptyset$;
for *each vertex* $v \in \mathcal{V}$ **do**
    MAKE-SET(v);
end
Sort edges of $\mathcal{E}$ in inscresing order by distance;
for *each edge* $e = (u, v) \in \mathcal{E}$ *in order* **do**
    if NUMBER-SETS $==$ k **then**
        break
    end
    if FIND-SET(u) $\neq$ FIND-SET(v) **then**
        $\mathcal{T} = \mathcal{T} \cup \{e\}$;
        UNION-SET(u, v);
    end
end

Algorithm 1: Kruskal's MST algorithm for clustering

We need $\mathtt{Union} - \mathtt{Find}$ data structure that supports:

- MAKE-SET(v): Create set consisting of v
- UNION-SET(u, v): Unite set containing u and set containing v
- FIND-SET(u): Return unique representative for set containing u

5 Numerical Experiments

The ALNS is implemented in Java as a single-threaded application. Numerical experiments are seperated in two different directions which correspond to the distinct parts of the project: the tuning of the new heuristics added to obtain a more stable solution and the analysis of the impact of the forecasting model's uncertainty level on the objective function. First, we assess the performance of the new operators implemented in the ALNS on the past results obtained by Markov et al. (2016) on real data instances. Secondly we evaluate how the objective function reacts to different levels of artificially simulated uncertainty. Section 5.1 describes the test instances. Section 5.2 defines the algorithmic parameters used and explains how the parameters of the new operators were tuned. Section 5.3 analyzes how the different components of the objective function react to changes in the level of uncertainty represented by the forecasting error.

5.1 Test Instances

There are two dinstinct sets of instances used for the numerical experiments: one used for the analysis of the solution quality and stability and the other one to analyze the impact of uncertainty. We describe those sets in section 5.1.1 and 5.1.2 respectively.

5.1.1 Instances for tuning

There are 63 different instances derived from data coming from canton of Geneva, Switzerland. Each of them corresponds to a week (Monday to Sunday) of white glass collections from 2014 to 2016. The time windows are set from 8 a.m. to 12 a.m and each tour has a maximum duration of 4 hours. The heterogeneous fleet of vehicle(s) is composed of up to 2 vehicles of volume capacity around $30,000$ liters and weight capacity of $10,000$ to $15,000$ kg. Their daily deployment cost φ_k is 100 CHF, with an additional 40 CHF/hour running cost θ_k and 2.95 CHF/km running cost β_k. Vehicles are not available over the weekend. There are on average 41 containers per instance and their volumes range from 1000 to 3000 liters. Their overflow cost is set to 100 CHF, which is normally determined by the municipality and the emergency collection cost (ECC) is set to 100. There are 2 dumps located on the outside of the city centre. The forecast demand for each instance is computed using the information of the past 90 days of observations for each container. Each instance has a distinct forecasting error υ.

5.1.2 Instances for uncertainty levels

In order to analyze how the uncertainty level impacts the objective value, there are 5 instances generated from true demand instances. In fact, for each instance described in section 5.1.1 we have access to the true demand. The forecasting error υ is therefore the difference between the true demand and the forecast demand computed using the information of the past 90 days of observations. The 5 instances are generated in the following way: we get the forecasting error υ from the corresponding forecast demand instance and we multiply it by $\eta \in \{0, 0.25, 0.5, 0.75, 1\}$. For simulating the different uncertainty levels we perturb the true demand by adding the resulting forecasting error. There are instances for $ECC \in \{25, 50, 100\}$.

5.2 Tuning

The same algorithmic parameters that were tuned by Markov et al. (2016) are used. For the simulated annealing it uses an initial temperature $T^{start} = 10,000$, a cooling rate $r = 0.99998$ and a final temperature $T^{end} = 0.01$. For the ALNS-related parameters the segment length F is of 2000, the reaction factor $b = 0.5$, the rewards $e_1 = 30, e_2 = 20, e_3 = 5$. The tuning of the operators-related parameters are described in section 5.2.1. Section 5.2.2 presents the value of the objective function and its stability using the new operators.

5.2.1 Tuning the New Operators

Five instances out of the 63 available ones were randomly chosen in order to select the best parameters of the new operators. The parameters have been tuned one by one, in the order we present them. First, we tested the *container insertion with regret k* parameter. Recall that we can compute the distance between inserting a container in its best position and its k^{th} best position. Best values of the objective function are found with $k = 2$. For the *Shaw remvoal of related containers* operator, the weights a, b and c of the distance, time and overflow probability terms respectively as well as the threshold t can be tuned. It turns out that best solutions are obtained for $a = 0.7, b = 0, c = 0.3$ and $t = 0.2$. It

is not surprising that the time term doesn't have any influence as the time windows are always the same. The *Shaw insertion of related containers* operator doesn't improve the solution, we decide to keep the *Shaw insertion* operator which considers only containers close to each other in term of distance.

If we set a normalization factor of $m_i = 2$ for the new operators, we obtain really good results but it takes approximately two times longer than previously to solve an instance (28 minutes). It is necessary to have a more stable value of the objective function for the analysis of the forecasting model error's influence as we will see in the second part of the numerical experiments, however we need to find a compromise between an acceptable running time and a stable result. We first set the normalization factor of the new operators to the same number and increase it progressively to see the limit at which the computational time becomes reasonable (15 minutes). It turns out that it can be reached with normalization factors set to 6. We then give lower importance to destroy operators (which means a higher normalization factor than 6) and a higher importance to the repair operators. The same method is applied the other way around. Good results are obtained setting a normalization factor of 8 to the destroy operators and of 4.5 to the repair operators.

5.2.2 Performance of the New ALNS

We now present the results obtained by the ALNS using the implemented operators solving each instance described in 5.1.1, i.e. the instances with forecast demands. The comparisons with the past results obtained by Markov et al. (2016) are made for the same running environment. Each instance is solved 10 times. The 63 instances have been solved and we compare the past results with the new results in order to check the two types of improvement: reducing the cost of the objective function and improving the stability of the solution. The stability is represented as the difference between the maximum and the minimum costs found, as well as the difference between the average cost and the minimum cost found. Table 1 shows the average of those values over the 63 instances. You can observe that the improvement is mainly in the stability values. The computational time is a little bit higher but remains reasonable.

Table 1: Average values over all instances of the past solution implemented by Markov et al. (2016) and the solution using the new operators

	Past	New	Improvement
Minimum objective value	664.76	662.64	0.32%
Average objective value	679.54	666.67	1.9%
Maximum objective value	699.50	673.62	3.7%
Max - Min	14.78	4.03	72.73%
Avg - Min	34.74	10.99	68.37%
Solution Time	814"	932"	-14%

Table 2: Percentage of instances for which a better value has been found

Better minimum	77%
Better average	97%
Better maximum	95%
Max stability improvement	92%
Avg stability improvement	94%

In addition to the average values, we count how many times those values have been improved. We check if a better value has been found for each instance. For example, a better minimum has been found for 49 out of the 63 instances, which represents 77% of the instances. The results are shown in Table 2 for each value.

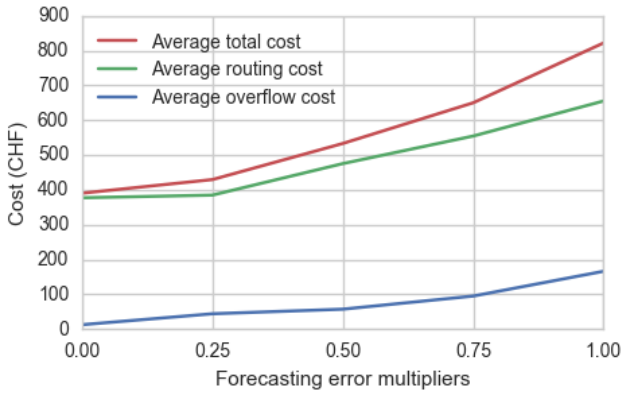
5.3 Impact of Uncertainty Level

We are interested now in analyzing the impact of the different uncertainty levels on the costs composing the objective function. In fact, it is useful to know if good forecasting models are necessary to obtain solutions with low costs or if we can tolerate models with high forecasting uncertainty. All the instances described in 5.1.2 have been solved for the different forecasting error multipliers η, using the parameters described in 5.2.1. We've shown that the total cost computed is now stable so we can accurately derive the impact of the uncertainty level on the value of the objective function. We analyze the impact on the total cost, as well as on the Expected Overflow and Emergency Collection Cost (EOECC) and the Routing Cost (RC) composing it. We don't include the Expected Route Failure Cost (ERFC) in our analysis because we observed that it is negligeable as it is close to 0. In what follows, we present the results obtained setting the ECC to 100 CHF. We ran the same analysis for ECC $\in \{0.25, 0.50\}$ and observed almost the same results. Section 5.3.1 presents a general analysis on the average values of the costs over all instances as well as the average number of container overflows. The costs' disaggregation is presented in section 5.3.2 in order to show the cost of each instance. In section 5.3.3 we fit linear and non-linear functions to the cost in function of the forecasting error to understand how much the uncertainty explains the variability in the value of the objective function. We finally analyze the impact of the different uncertainty levels on a rolling horizon solution approach in section 5.3.4.

5.3.1 Impact on Average Costs and Container Overflows

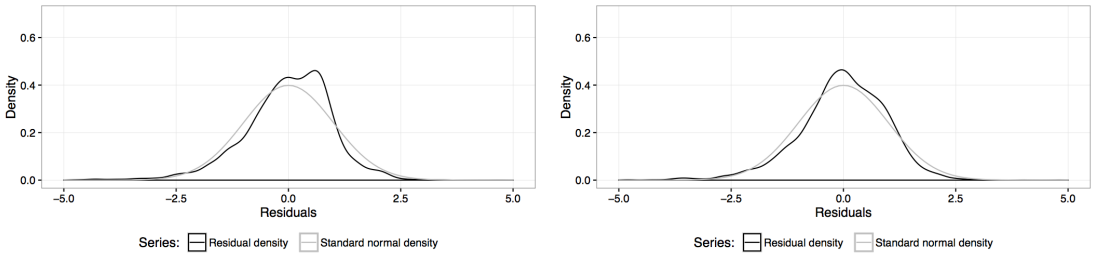
Average Costs Figure 1 shows the average over the 63 instances of the costs (total cost of the objective function, RC and EOECC) at every forecasting error multiplier $\eta \in \{0, 0.25, 0.5, 0.75, 1\}$. We can observe that the total cost grows from 390 CHF with the true, $\eta = 0$, to 822 CHF when the demand is at the point where it is the most uncertain, $\eta = 1$. The total cost is 2.1 times higher at the maximum level of uncertainty. The RC grows from 377 CHF to 655 CHF, increasing 1.74 times. It is interesting to notice that the total cost is mainly composed by the RC but that the RC increases slower than the total cost. This is explained by the EOECC which has a lower value but grows from 13 CHF to 166 CHF, increasing 13.21 times.

Figure 1: Average costs at growing forecasting error



Average Container Overflows After solving each instance, 10,000 simulations are performed in order to evaluate the occurrence of extreme events such as the number of container overflows. The forecasting error is sampled indepedently for each container and each day using the estimate υ. We can see in Figure 2 that the normal distribution is a good approximation of the distribution of the residuals so the conclusions are valid. Figure 3 shows the average number of container overflows at the 75th, 90th, 95th and 99th percentiles of the 10,000 simulation runs on the 63 instances for $\eta \in \{0, 0.25, 0.5, 0.75, 1\}$. As we can expect it, when the demand is known there are no container overflows and we observe a growing number of container overflows according to the growth of the forecasting error multiplier.

Figure 2: Examples of distribution of the residuals



5.3.2 Costs' Disaggregation Analysis

We show in Figure 4 the total cost, RC and EOECC at the different forecasting error multipliers for each instance solved. The total cost increases monotonically for almost every instance. However, even if we saw that the average of the RC increases monotonically (see Figure 4 (a)), there are many instances that have a lower RC for $\eta = 0.25$ and then the RC increases again (see Figure 4 (b)). This is compensated by the EOECC which tends to grow at $\eta = 0.25$ and then drops again at $\eta = 0.5$ to grow from $\eta = 0.75$ (see Figure 4 (c)). In fact, 10 out of the 11 instances having a lower RC at $\eta = 0.25$ have a higher EOECC at $\eta = 0.25$ than $\eta = 0.5$. The same symmetry is applied to the 4 instances having a lower RC at $\eta = 0.5$ which have a higher EOECC at $\eta = 0.5$ than at

Figure 3: Average container overflows at growing forecasting error

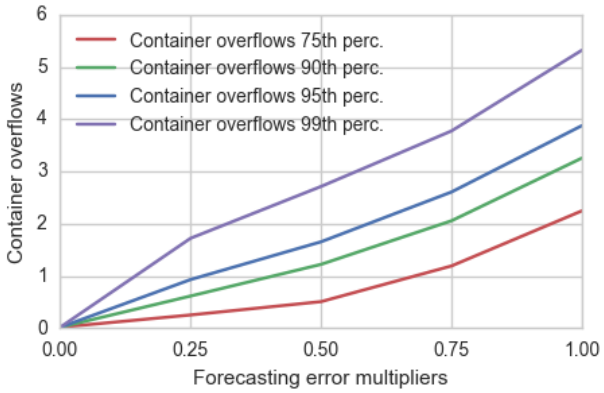
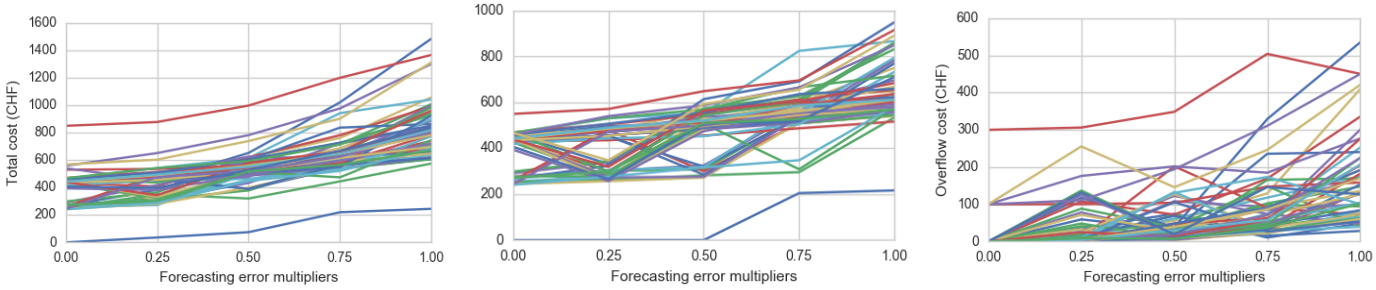


Figure 4: Costs over all instances



(a) Total costs (b) Routing costs (c) Overflow costs

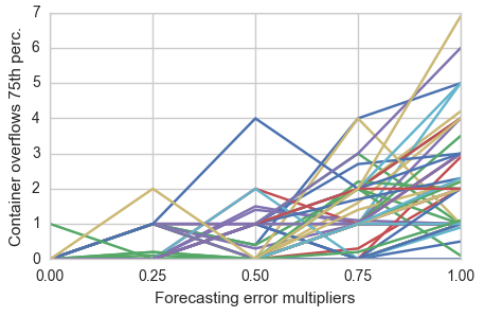
$\eta = 0.75$. If we look at Figure 4 (a) we can observe some instances having a nonmonotonic total cost growth. The average $avg - min$ stability over those 17 instances is the same as the average $avg - min$ stability over the 63 instances. However, the average of the $max - min$ stability of those 17 instances is 12.39 and the average of the $max - min$ stability over the 63 instances is 10.99. There are 32 instances having a nonmonotonic RC or EOECC growth and their average $max - min$ stability is 10.9, which is slightly lower than the average $max - min$ stability over all instances. We can not conclude any relevant relationship between the stability and the nonmonotonic growth of the costs according the forecasting error multipliers. It could be due to the randomness of the applied perturbations, especially for similar forecasting error multipliers.

Figure 5 shows the the average number of container overflows for each instance at the 75th percentile on the 10,000 simulation runs. There is a strong linear relationship between the EOECC and the number of container overflows as the correlation coefficient is equal to 0.84.

5.3.3 Function's Fits

We saw in section 5.3.1 that the total cost grows with the growth of the forecasting error multipliers. We perform now a linear regression with the explanatory factor being

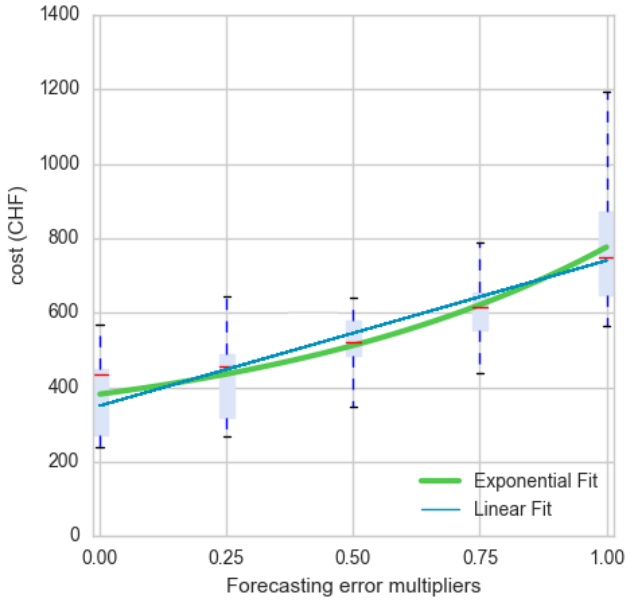
Figure 5: Container overflows at growing forecasting error multiplier over all instances



the forecasting error multiplier η and the dependent variable being the total cost of each instance. We obtain a coefficient of determination $R^2 = 0.495$ which means that it explains approximately half of the variance.

We suppose however from Figure 1 that an exponential function could fit better the observed cost. We run a non-linear regression, fitting our observed cost to the exponential function $a * e^{-b*\eta} + c$. We use η again as predictor. The result of the fits are displayed on Figure 6. We compare the residual sum of squares RSS of both models on our datapoints. For the linear model we obtain a RSS $= 6,337,818$ and for the exponential model RSS $= 6,050,379$.

Figure 6: Linear and Exponential fits



We can see in Figure 4 (a) that the cost for each instance varies from many different ranges, affecting a lot the regression which will fit a line on the average of those costs. To overcome this problem, we can add a dummy variable for each but one instance group of 5 instances, a group being the instance with forecast error multipliers

$\eta \in \{0, 0.25, 0.5, 0.75, 1\}$. We run a linear regression with the explanatory factors being the forecasting error multipliers η and the dummy variables. We obtain $R^2 = 0.857$, our model explains therefore 86% of the variance.

We run a as well the non-linear regression with the explanatory factors being the forecasting error multiplier η and the dummy variables in order to compare the two models. We obtain for the exponential model $RSS = 1,135,308$ and for the linear model $RSS = 1,475,987$. The RSS of the linear model is one third higher than the one of the exponential model. The growth of the cost tends therefore to be exponential according to the level of uncertainty.

5.3.4 Impact on Rolling Horizon Approach

The SIRP solution that has been presented uses a daily rolling horizon approach and uses the latest available container level information. We now present the analysis of a Dynamic and Stochastic Inventory Routing Problem (DSIRP) for which the problem is first solved for a planning horizon of one week, then the tours scheduled on $t = 0$ are executed, the horizon is rolled over by a day, the problem is solved again (with the new container levels updated by true demands and dependent on the solution of the preceding rollover), and it is repeated until the last day of the first week. At each start of a rollover the true demand of the starting day is known and the demands over the planning horizon are stochastic. In order to analyze the impact of uncertainty in a rolling horizon approach, we solve five instances with the same format as the one described in section 5.1.2 integrating different uncertainty levels. The true demand used in the rolling horizon to update the starting conditions for each rollover comes from the corresponding instance with forecasting error multiplier $\eta = 0$. The five instances are solved for $\eta \in \{0.25, 0.5, 0.75, 1\}$ 10 times. We hypothesize that instabilities may accumulate for each rollover so we focus on analyzing the impact on the minimum cost over the 10 runs. We observe that it was infeasible to solve one instance for $\eta = 0.75$. We decide to set this instance's rollover solution cost value for $\eta = 0.75$ to the linear interpolation between the cost for $\eta = 0.5$ and $\eta = 1$.

Figure 7: Average cost of rolling horizon approach at growing forecasting error multiplier

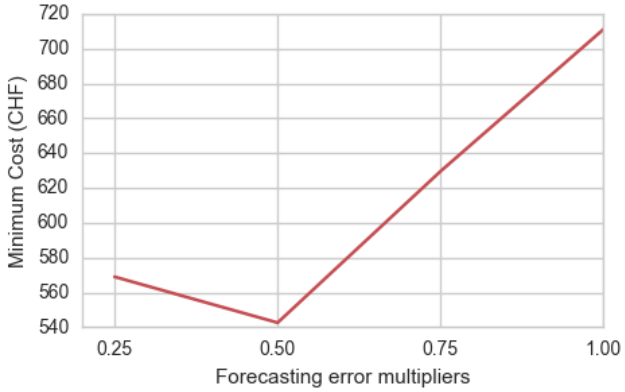
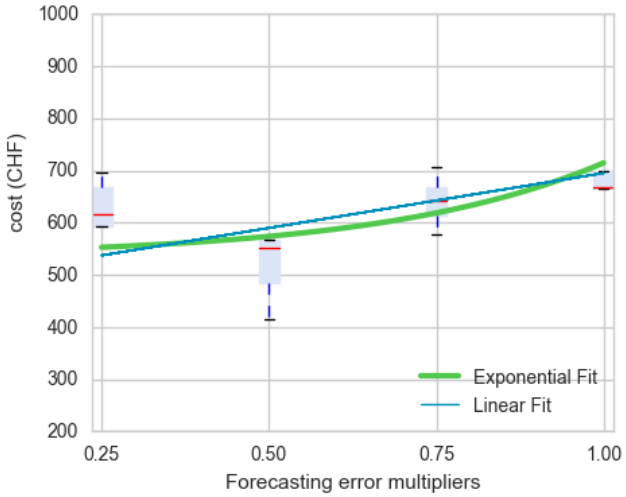


Figure 7 shows the averages over the five instances of the minimum rolling horizon costs computed. The cost grows from 598.97 CHF to 711.11, which corresponds to an increase of 25%. We can observe that the cost at $\eta = 0.5$ is lower than at $\eta = 0.25$. If we look at the cost of each instance, there are three out of the five instances which have a lower cost at $\eta = 0.5$. This might come, as we hypothesized in the costs' disaggregation analysis section, from the randomness of the applied perturbations.

We perform a linear regression with the explanatory factor being the forecasting error multiplier η and the dependent variable the cost of the rollover solution of each instance. We discard from the analysis the cost of the infeasible instance for $\eta = 0.75$. We obtain $R^2 = 0.21$, the uncertainty in the forecast demands explain therefore 21% of the variance of the rolling horizon solution cost. We run a non-linear regression fitting the cost to an exponential function, using η as predictor and we obtain RSS $= 242928$. Figure 8 shows the linear and exponential fits. The linear model has a RSS $= 251987$ and the exponential model a RSS $= 242928$. The exponential function fits better the observed cost.

Figure 8: Linear and Exponential fits on cost of rolling horizon approach



The forecasting error doesn't explain much the variance, but the cost for each instance varies form different ranges so we add dummy variables (see section 5.3.3) and run a linear regression. We obtain $R^2 = 0.64$. Moreover, if we run a non-linear regression with the explanatory factors being the forecasting error multipliers η and the dummy variables, we obtain RSS $= 116023$. The corresponding RSS values for the linear model is 116021. We cannot conclude which model explains better the observed cost. Thorough experiments would be needed, solving more than 5 instances. We weren't able to solve more instances using the rolling horizon approach due to time constraints. In fact, the rolling horizon approach implies the complete SIRP solution computation at each rollover and is therefore computationally very expensive.

6 Conclusion

We analyze an existing problem formulation of a rich recyclable waste collection problem and its solution implemented by Markov et al. (2016) using an ALNS integrating a forecasting demand model. We improve the solution by introducing new destroy and repair operators inspired from the VRP and IRP literature into the search algorithm. We slightly reduce the cost of the objective function but the major improvement resides in the stability of the solution which is considerabily increased. We are therefore able to measure accurately how different uncertainty levels in the forecasting of future demands impacts the solution cost. We observe that the value of the object function grows exponentially according to the level of uncertainty in the forecasting model. We can conclude that forecasting models with low uncertainty lead to considerably better solution costs. The problem lends therefore to potential future work directions. Better demand forecasting methodologies can be explored and could potentially reduce the the cost of the objective function. Better costs could also be found if the algorithm's computational time is reduced, for example by using more efficient algorithms, limiting identical or similar repetitive calculations or not re-evaluating the complete objective function at each neighborhood move by the operators. More experiments could be performed to analyze the effect of uncertainty in the forecasting model on a rollover solution cost.

Acknowledgement

I would first like to thank my Master Semester Project's supervisor Iliya Markov who guided me through the project and the learning process, answered my specific questions in detail and gave me useful comments on the report. I would thank Yousef Maknoon as well who gave valuable remarks on the direction of the project and Michel Bierlaire, director of the Transport and Mobility Laboratory at Ecole Polytechnique Fédérale de Lausanne, for this interesting research project.

References

Bertazzi, L. and Speranza, M. G. (2012). Inventory routing problems: an introduction. *EURO Journal on Transportation and Logistics*, 1(4):307–326.

Buhrkal, K., Larsena, A., and Ropke, S. (2012). The waste collection vehicle routing problem with time windows in a city logistics context. *Procedia - Social and Behavioral Sciences*, 39:241–254.

Coelho, L. C., Cordeau, J. F., and Laporte, G. (2014). Thirty years of inventory routing. *Transportation Science*, 48(1):1–19.

Markov, I., Bierlaire, M., Cordeau, J. F., Maknoon, Y., and Varone, S. (2016). Inventory routing with non-stationary stochastic demands. Technical report, Transport and Mobility Laboratory, Ecole Polytechnique Fédérale de Lausanne.

Potvin, J. Y. and Rousseau, J. M. (1993). A parallel route building algorithm for the vehicle routing and scheduling problem with time windows. *European Journal of Operational Research*, 66(3):331 – 340.

Ropke, S. and Pisinger, D. (2006a). An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. *Transportation Science*, 40(4):455–472.

Ropke, S. and Pisinger, D. (2006b). A unified heuristic for a large class of vehicle routing problems with backhauls. *European Journal of Operational Research*, 171(3):750 – 775.