Re-optimization of technician tours in dynamic environments with stochastic service time

Erwann Delage

CIRRELT Montréal
Ecole des Mines de Nantes
erwann.delage@etu.univ-nantes.fr

Abstract. This report presents some works dealing with dynamic and stochastic vehicle routing problems. We introduce two methods coming from different horizons in order to deal with the multi-depot VRPTW with stochastic service times where reoptimization is needed. The first one is an online procedure using tabu search and the stochastic programming with recourse framework. The second one is an offline method that rely on dynamic programming. We propose to compare theoretically these two methods, until we do it empirically.

1 Introduction

The Vehicle Routing Problem (**VRP**) defined by *Dantzig and Ramser* [6] in 1959 is one of the most analyzed problem in the fields of transportation, distribution and logistics. Typically, it consists in determining *efficient* routes for a fleet of vehicles starting from one (or more) depot and visiting a set of customers. Since 50 years, the VRP and all its derivatives have been intensively studied by the logistic community. However, most of these studies deal with the static case, where all the input data (number of customers, demands...) are known in advance. Unfortunately in real life, things generally do not go as planned. Indeed, during a tour, a lot of events can occur (traffic congestion, emergencies...) and perturb initial plans.

With the recent advent of new technologies in telecommunication and information systems such as mobile phones, Global Positioning Systems (GPS), Geographical Information Systems (GIS), and Intelligent Transportation Systems (ITS), it has become realistic to solve dynamic routing problems in real-time as new informations arrive, and the importance of developing workable algorithms for such problems increased.

The problem addressed in this internship belongs to the Dynamic Vehicle Routing Problem with Time Window (**DVRPTW**) family, and also to the VRP with Stochastic Service Times (**VRPSST**) family (defined by *Roberts* in [20]). The goal of this work is to design an efficient re-optimization procedure to handle new events occurring in the day.

This report is organized as follows. Section 2 describes the problem in details and discusses our preliminary thinking. Section 3 presents a literature review dedicated to dynamic and stochastic vehicle routing problems. Section 4 and 5 introduces our models and resolution method, respectively. Finally section 6 reports first experiments and section 7 summarizes our findings and opens new perspectives.

2 Problem description

This problem is inspired from the work of *Fabien Tricoire* for *Veolia Eau* [25] and from its extension realized by *Frédéric Dugardin* [7], we will thus use some of the vocabulary they proposed in their works.

2.1 Presentation

The problem presented in this document is a concrete one that many companies dealing with technician tours (EDF, Veolia...) face every day, it can be summarized as follows. A company with a limited number of technicians must conduct some tours to perform reparations, or maintenance operations. Some of these demands come directly from customers, the other ones are interventions planned by the company. For each customer demand, a day and a time window determine the period where the intervention must be done. However, concerning the interventions planned by the company, the assiciated period of validity is wider. For each work, the company estimates its expected duration.

Every morning, a list of tours (one per technician) is determined depending on the schedule and the expected duration of each task. But sometimes, interventions last longer or are shorter than expected, and a solution that was optimal at time t becomes inadequate or even infeasible at time $t+1$. Therefore, the company must actualize its tours.

In our work, we are not interested in how the appointments are scheduled, and we focused on building the tours for only one day. We could have consider planning tours for one week, but the main goal of this project is to design a re-optimization procedure and not to plan tours on a long horizon. More precisely, the work that we proposed to perform can be split in two parts : construction and re-optimization.

- Construction : Every morning before technicians start their work, we build tours from a schedule of *appointments* and a list of *interventions* planned by the company. In almost all cases, we have to build *selective tours* because there are more feasible interventions than the technicians can handle in the day.
- Re-optimization : Then, during the day, tours are re-optimized in real-time when it is required, *i.e.* when uncertainties perturb the initial solution.

2.2 Definitions

Demands As we said in the presentation part, we consider two kinds of demands:

- **The important appointments** are requests coming from customers and have time-window. These important appointments (that we will simply call *appointments* thereafter) have priority over the postponable interventions.
- **The postponable interventions** are interventions planned by the company, like a *change of meter* for example. This kind of works (that we will call *postponables* thereafter) can be done at any time during the day.

We will employ indifferently the terms customers, requests, demands and interventions when talking about the overall appointments and postponables.

Uncertainties During a tour, some unexpected events can occur and thus modify the problem data. The typical unexpected events a company can cope are : emergency requests (water leak, power outage...), traffic congestion, sick technicians, vehicle failures...

However here, we only focus on three unexpected events, because almost no work has been carried over on the subject :

- **Variable service time** : Each intervention of a technician can be shorter or longer than expected because of a bad problem evaluation.
- **Unavailable customer** : When the technician arrives at the customer location, the customer is not here
- **Impossibility of service** : The technician does not have the adequate equipment and thus can not achieve the intervention

It is worth noting that the *unavailable customer* and *impossibility of service* events can be regarded as a *variation of service time* event where the service time is null.

2.3 Objectives

This problem presents one principal objective to optimize, that is :

- **Maximize the number of performed interventions**

When we are dealing with selective tours, it is clear that we want to do a maximum of work during the day. More especially, we really want to maximize the number of satisfied *appointments*. Indeed, there is a customer behind each appointment, and dissatisfying customers is a problem for a company that wants to keep a good quality of service, and thus, keep customers. The second objective in the lexicographic order is the classical minimization of transportation costs.

The previous objectives will be part of the mathematical models, but we have other goals that are related to the problem :

- **Minimize the response time of the algorithm during re-optimization phases**
- **Build robust solutions**

Clearly, a technician who call to notify a lateness or an advance is not going to wait 10 minutes to get his next destination back. Consequently, it is absolutely mandatory to design an algorithm that gives an efficient solution in a short time. And building robust solutions, which means solutions that will resist pretty well to uncertainties is a key because those solutions need less re-optimization and are more able to be modified to cope with an uncertainty.

2.4 Constraints

We list here the constraints attached to the problem, which are classical routing problem constraints :

- Each technician does only one tour per day
- There are at most K (number of technicians) vehicle routes, each one starting and ending at a depot
- Each technician has one starting and one ending depot (typically it can be their home)
- Each request belongs to *at most* one route.
- Time windows constraints : Every location has an associated period of time during which the service has to take place. For depots and postponables, it is working hours, for appointments it is the period in which the technician must arrive and end its service.

2.5 Thinkings

When dealing with optimization problems, which include uncertain elements, several approaches have to be considered and various choices have to be done. In this section, we evoke these different approaches and describe some possibilities offered by the problem.

Deterministic and stochastic method As the problem we face contains stochastic element (i.e. the service times), we may consider solving this problem in two different ways : stochastic or deterministic way. In a stochastic method, we take in account the stochastic elements of the problem and build the tours according to those uncertainties. On the other side in a deterministic method, tours are built in an *optimal way*, not addressing the stochastic aspect of the problem (i.e. not considering uncertainties). To summarize, with stochastic methods we try to anticipate events whereas with deterministic methods we assume that all will run normally.

A-priori optimization and real-time optimization In *a-priori* optimization methods, the goal is to determine a-priori solutions. By a-priori solution we mean a solution based on probabilistic information on future events. In real-time optimization, routes are not constructed beforehand but in an on-going fashion as new data arrive.

Construction and reparation strategy When a new event occurs during the day, there are two possibilities to handle it. Firstly, we can consider that we face a completely new problem and thus build a new solution with a constructive method. Secondly, we can see the new problem like a slight modification of the previous one, and then repair the previous solution to make it take into account the new event. Both methods have their advantages, the reparation strategy is less expensive in processing time and the construction strategy allows to find more diversified solutions.

Hard and soft time windows The soft time windows, unlike the hard ones, are not restrictive. It means that a technician can arrive out of the time window allowed to an intervention even if that leads to a penalty in the objective function. The great advantage of the soft time windows is that it authorize a kind of " better late than never " policy and improve the productivity by eliminating waiting times. However in real life, a customer is not necessarily happy to see a technician arrives at his home earlier or later than expected. Moreover, the customer is likely to be away if the technician arrives out of the time-window.

Continuity of solutions By continuity of solutions, we mean the difference between two consecutive solutions. When a new event occur, one could prefer a new solution that does not change that much from the previous one. The idea behind this principle is to prevent disruption in the planning which could lead to degradation of the quality of service. However, in our case, only the technician planning can be disrupted because appointments must be respected anyway. As the modification does not affect the QoS of customers, we think that it would be contradictory with new technologies to not disrupt previous plans.

Distribution of the customers The distribution of customers can also be an important factor. We can differentiate the case where interventions are randomly and uniformly distributed on the map and the case where there exists clusters of customers (like in towns).

3 State of art

In this section, we present some works dealing with DVRP and some others talking about SVRP. All of the following works do not correspond exactly to our problematic but are always related to DVRP or SVRP, and it is interesting to see how different problems had been studied and solved. Moreover, even if thereafter we use only a small fraction of what is presented here, these studies have allowed us to choose resolution principles adapted to our problem, making them essential for our work.

3.1 The Dynamic Vehicle Routing Problem

Despite the fact that some similar problems were mentioned before, it is Psaraftis in [12] who the first defined the class of Dynamic Vehicle Routing Problems. He introduced definitions for static and dynamic VRP that Larsen in his thesis [15] summarized as follows :

Definition 1 *Dynamic Vehicle Routing Problem*

1. *Not all information relevant to the planning of the routes is known by the planner when the routing process begins.*
2. *Information can change after the initial routes have been constructed.*

As Psaraftis pointed out, dynamic routing problems differs from static in several ways, we present in the following list the more interesting differences for our case :

1. Time dimension is essential : We need to know the spatial location of all vehicles within our fleet at any important point in time
2. Future information may be imprecise or unknown : As in any real life situation, the future is almost never known with certainty in a DVRP. Like in our case, probabilistic information about the future may be available but in many cases even that type of information may not exist.
3. Near-term events are more important : In dynamic routing, it would be unwise to immediately commit vehicle resources to requirements that will have to be satisfied way into the future, because other intermediate events may make such decisions suboptimal, and because such future information may change anyway.
4. Information update mechanisms are essential : It is imperative that information update mechanisms be an integral part of the algorithm structure and input/output interface in a dynamic situation.
5. Re-sequencing and reassignment decisions may be warranted : In a dynamic vehicle routing situation, the appearance of a new input may render decisions already made before that input's appearance suboptimal. Thus, a re-sequencing of the stops of one (or more) vehicle(s) can be necessary.
6. Faster computation times are necessary : The need to re-optimize routes in real-time necessitates faster computation times than those necessary in a static situation.

7. Objective function may be different : Measures of performance that have more meaning in a dynamic situation are more *throughput* or *productivity* related, whereas in a static situation the objective is generally to minimize the total distance traveled.

3.2 Solving the DVRP

Introduction We can distinguish two main categories of methods to solve the DVRP, the stochastic method that use probabilistic information on future event to construct routes, and the real-time optimization method that constructs route during the day. Within the first method, routes are planned the morning before vehicles leave the depot whereas in the second one routes are constructed while the vehicles are on-route.

Since the VRP is NP-hard, by generalization, our problem is also NP-hard and we thus focused on non-exact resolution methods. A dynamic VRP can be seen as a succession of severals static VRP, and thus it is possible to adapt the algorithms designed for the static case to the dynamic one.

Adaptation of static algorithms There are generally two ways to adapt a static algorithm to a dynamic case. The first one is to rerun the procedure from scratch each time an event occurs. This would involve generating a new set of routes at each input update while guaranteeing that decisions already made are not compromised.

The second one would be to handle dynamic input updates via some reparation strategies. This involve running the static algorithm just to initialize the process and rely on local operations when necessary. This approach presents the advantage to be reasonable in term of processing time and to not deviate that much from the initial solution. However, the choice of these methods is strongly connected to the degree of dynamism of the problem and to the length of the horizon.

Simple policy algorithms and classical heuristics Simple policy algorithms are routing strategies that describe a general behavior . *Bertsimas and Van Ryzin* in [2] define some policies and waiting strategies for *a-priori optimization* concerning the Dynamic Traveling Repairman Problem (DTRP). We illustrate on figure 1 the behavior of the *Traveling Salesman Policy* because it presents the interest to separate the area containing the customers in several parts:

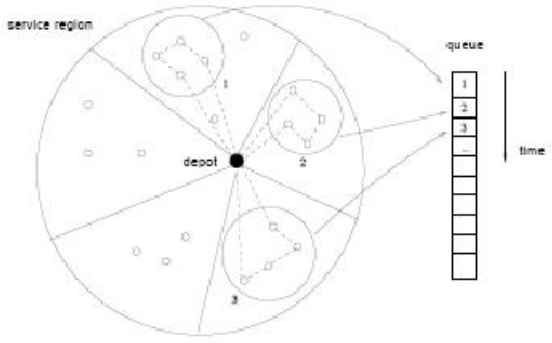

Fig. 1. Illustration of the traveling salesman policy

The area A is divided into K subareas and a TSP is solved for each sector.

Best Insertion : The *Best Insertion* heuristic defined by Solomon in [23] is an efficient insertion method for the VRPTW and is often used to generate excellent initial solutions for DVRPTWs in a reasonable amount of computing time. It works as follows :

1. Initialization : Initialize the route by adding either
   - the farthest unrouted customer
   - or the unrouted customer with the earliest deadline
2. Insert the more interesting customer compared with the insertion costs on both distance and time dimensions.

Of course there exists other simple methods, like adaptations of Clark and Wright algorithm or sweep algorithm. However even if these simple methods can obtain quite good solutions, generally they are not sufficient into themselves and are embedded in a metaheuristic to improve their results.

Tabu Search The tabu search is a metaheuristic belonging to the class of the local search techniques. It uses a neighborhood search procedure to move from a solution x^1 to a solution x^2 in the neighborhood of x^1, until some stopping criterion has been satisfied. It improves the ILS (Iterative Local Search) because it uses a memory storing some solutions as "taboo", avoiding the algorithm to do uninteresting moves.

The tabu search has been intensively used to solve Dynamic Vehicle Routing Problem. The interested reader could find good examples of utilization of the tabu search in [13] or [21]. Here we present the work of Gendreau, Guertin, Potvin and Taillard [8] who adapted a tabu search designed for a static case to a dynamic environment.

The tabu search used by *Gendreau et al.* for the static case is characterized by the exploitation of an adaptive memory. The neighborhood structure used in this procedure is the *ejection chain*. This algorithm can be summarized as follows :

1. Construct I different initial solutions with a stochastic insertion heuristic (i.e. the rule for choosing the next customer to be inserted contains stochastic elements).
2. Apply tabu search to each solution and store the resulting routes in the adaptive memory
3. For W iterations do :
   (a) Construct an initial solution from the routes found in the adaptive memory and set this solution as the current solution
   (b) For C cycles do :
      i. Decompose the current solution into D disjoint subsets of routes
      ii. Apply tabu search to each subset of routes
      iii. Merge the resulting routes to form the new current solutions
   (c) Add the resulting routes to the adaptive memory (if indicated)
4. Apply a post-optimization procedure to each individual route of the best solution found

The idea of this method is that the tabu search works in background between the events, trying to find a better set of planned routes. The search threads are interrupted whenever an input update occurs (a new request is received, or a vehicle has completed its service at a customer location). New solutions are created depending on the new situation generated by the appearance of the new event, and are stored in the memory. After these modifications, the search threads can be restarted with new solutions constructed from the updated adaptive memory. Here follows the successive steps of the algorithm :

1. While there is no new event, optimize the planned routes using the tabu search
2. If an event occurs, then
   (a) Stop each tabu search thread and add the routes of their best solution to the adaptive memory (if indicated)
   (b) If the event is the occurrence of a new request, then
      i. Update the adaptive memory through the insertion of the new request in each solution
      ii. If no feasible insertion place is found, reject the request
      Otherwise (end of service at a customer location)
      i. Identify the driver's next destination, using the best solution stored in the adaptive memory
      ii. Update the other solutions accordingly
   (c) Restart the tabu search processes with the new solutions obtained from the adaptive memory

It has been empirically shown that tabu search was one of the most efficient metaheuristics for solving DVRPs.

Genetic Algorithm (GA) Genetic algorithm are a class of probabilistic optimization algorithms inspired by the biological evolution process that uses the concepts of *natural selection* and *genetic inheritance* defined by Darwin in 1859. The idea behind GAs is to make a population of solution evolves by iteratively applying a set of stochastic operators. The three stochastic operators applied iteratively are :

1. Selection : Select solutions in the initial population, generally according to their quality by the mean of a stochastic process. A good quality solution has more chance to be selected than a bad solution.
2. Recombination : Decomposes two distinct solutions and then randomly mixes their parts to form new solutions.
3. Mutation : Randomly perturbs the new solutions.

This is the typical operation of the stochastic operators, but they can and should be adapted to each specific problem. The figure 2 illustrates the classical behavior of a genetic algorithm

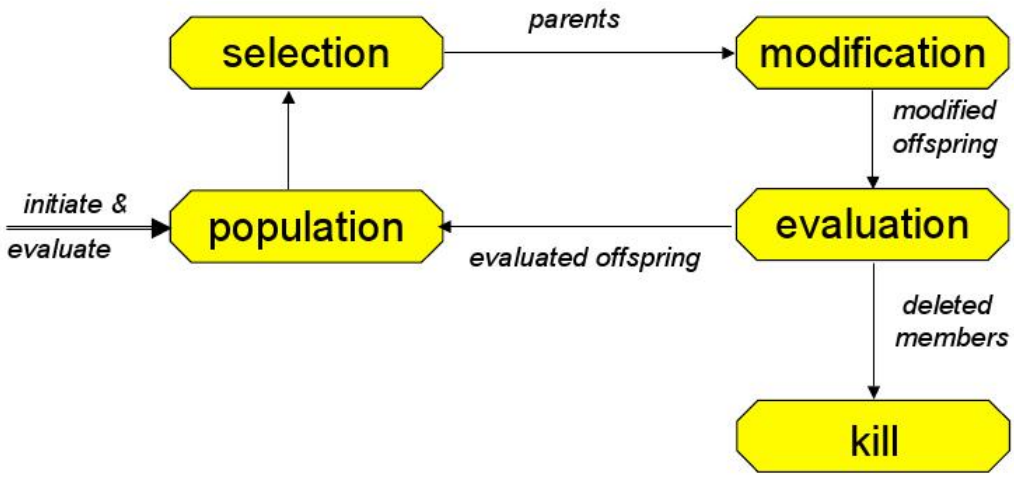

Fig. 2. Behavior of genetic algorithms

In matter of resolution methods for DVRPs, the tabu search has long been preferred to any other metaheuristic. However, since a few years the other metaheuristics and especially the genetic algorithms began to generate interest. Housroum in [11] describes a GA that use previous solutions and adapt them to build new solutions corresponding to the new situation. He developed a simulator to

handle new events and shows that his GA gave better results than a simple heuristic and was competitive with the tabu search. An extension of his work to time-dependent travel times model can be found in [26]

Memetic Algorithm Memetic algorithms represent one of the recent growing areas of research in evolutionary computation. They are defined by Moscato like optimization methods using a population, and lying on the utilization of a local search. Here, we report the memetic algorithm used by Fabien Tricoire [25] in his thesis :

1. Build an initial population
   (a) Adaptation of Solomon *Best Insertion* heuristic [23]
   (b) Improvement with different local search heuristics
2. Crossover
   (a) Initialize a new solution s by building "empty tours" just associating a day and a technician
   (b) Take tours from each parent or an empty tour (mutation) and add it to s
3. Improve s with a local search

Ant Colony Optimization (ACO) Ant colony optimization is inspired from the behavior of real ants, which always find the shortest path between their nest and a food source. Figure 3 summarize the principles of this metaheuristic proposed by Marco Dorigo in 1996.

1. **Generate population of ants P**
2. **Repeat**
    1. For each a in P
        1. Move in a random direction,
           but if there is nearby pheromone, prefer that direction.
        2. Leave pheromone from a trail
        3. If food found, pick it up and return home
    2. Globally reduce the amount of all pheromone
3. **Until stopCondition**

Fig. 3. Principle of ant colony optimization

The utilization of ACOs to solve DVRPs is very recent and is not as frequent as the other metaheuristics presented before. We can still refer to the works of *Montemanni et al.* [17] who developed several ACOs and obtained quite good results on DVRPs. *A.Runka* in [1] improved ACOs designed for the DVRP with a local search but conclude that this improvement was not sufficient to outperform the solution quality obtained by GA and Tabu.

3.3 Stochastic Vehicle Routing Problem

We talk about stochastic vehicle routing problems (**SVRP**) when some elements of the problem are stochastic in nature. We discuss in this section several methods to handle SVRPs, in particular stochastic programming. Stochastic programming is a framework for modeling optimization problems that involve uncertainty. Whereas deterministic optimization problems are formulated with known parameters, real world problems almost invariably include some unknown parameters. When the parameters are known only within certain bounds, one approach to tackling such problems is called *robust optimization*. Here the goal is to find a solution which is feasible for all such data and optimal in some sense. Stochastic programming models are similar in style but take advantage of the fact that probability distributions governing the data are known or can be estimated. The goal here is to find some policy that is feasible for all (or almost all) the possible data instances and maximizes the expectation of some function of the decisions and the random variables. A stochastic program is usually modeled either as a *Chance Constrained Program* (**CCP**) or as a *Stochastic Program with Recourse* (**SPR**) . Other methods like *Markov decision process* and *fuzzy logic* can also be used to deal with SVRPs.

Unfortunately, most of the studies on the SVRP concern stochastic demands. Only a few studies have been published on stochastic travel time and almost no on stochastic service times.

Robust optimization method The idea of the robust optimization approach is to optimize the worst case scenario in order to obtain a solution that is feasible for all possible scenarios.

Routing technicians under Stochastic Service Times : a robust optimization approach We present here the work of *Cortes et al* [5] because their topic is strongly related to ours. The only difference between their problematic and ours is that they consider a static environment whereas we consider a dynamic one. Even if the resolution methods they used are not adapted to our problem, we can find interesting idea in their modeling. Especially, they define a mathematical model that take stochastic service times into account.

In order to deal with their problem, they propose a robust optimization approach that is efficient for all realizations of the problem uncertainty. The robust solution is obtained by solving the robust counterpart problem, whose goal is a solution that optimizes the worst case value over all data uncertainty by using a min-max objective. The resulting solution from the robust counterpart problem is insensitive to the data uncertainty, as it is the one that minimizes the worst case, and therefore is *immunized* against this uncertainty. Clearly, the size of the uncertainty set influences the quality of the robust solution.

The authors define an upper bound U_k which represents the total deviation of service times faced by a technician k :

$$U_k = \alpha \sum_{i \in P^k} \gamma_i$$

where P^k represents the path followed by technician k and γ_i the longest possible service time at customer i. The most important parameter of this equation is α that controls the level of robustness and allows to tune the problem. Technically speaking, if we fix $\alpha = 1$ we then obtain the theoretical worst case scenario $\overline{U_k} = \sum_{i \in P^k} \gamma_i$ that never happens in reality. But if we use $0 \leq \alpha < 1$, then the worst case scenario represented by U_k is less worse than the theoretical worst case scenario. They thus use this upper bound U_k to define their model by adapting the classical constraint to the stochastic context.

Chance-constrained Programming Chance-constrained programming is a methodology proposed by *Charnes and Cooper* in 1959 to specify levels of confidence for stochastic constraints within optimization problems. Chance-constrained programming belongs to the major approaches for dealing with random parameters in optimization problems. In CCPs one seeks a first stage solution for which the probability of failure is constrained to be below a certain threshold. The main difficulty of such models is due to (optimal) decisions that have to be taken prior to the observation of random parameters. The advantage is that chance constrained model can be represented like deterministic model and thus use the same resolution methods.

The minimum unmet demand SVRP

In this paper, *Shen et al.* [18] are interested in routing vehicles to minimize unmet demand under uncertainty both on demand and travel time. Their topic is motivated by the problem of distributing medical supplies in large-scale emergency response (earthquake, terrorist attacks...). They present a chance constrained formulation of the problem that is equivalent to a deterministic problem with modified demand and travel time parameters.

The idea behind the chance constrained model is to assume that t_{ijk} (the uncertain travel time to go from i to j with vehicle k) and d_i (the amount of demand need at node i) are unknown at the time of planning but follow some known probability distribution. They define parameters α_t and α_d representing the confidence level of the chance constraints defining the unmet demand at each node and the arrival time of each vehicle at each node respectively. Thus, the constraints with stochastic parameters must hold with these given probabilities.

$$P\left\{t|(T_{ik}+t_{ijk}-T_{jk}) \leq (1-X_{ijk})M\right\} \geq 1-\alpha_t \ , \quad (\forall i,j \in C,\ k \in K)$$

$$P\left\{d|\sum_{k\in K}\left[\sum_{j\in C} Y_{jik}-\sum_{j\in C} Y_{ijk}\right]+U_i-d_i \geq 0\right\} \geq 1-\alpha_d \ , \qquad (\forall i \in D)$$

They then used a standard tabu search procedure to solve their problem and compare the results between the deterministic model and the chance constrained model. They conclude that their CCP model is effective in producing the preplanned routes under moderate deadlines and supplies and can provide a better coverage of the overall demand with some uncertainty present.

Stochastic programming with recourse In SPRs, the aim is to determine a first stage solution that minimizes the expected cost of the second stage solution : this cost is made up of the cost of the first stage solution, plus the expected net cost of recourse. As *Gendreau et al.* noted in [9] SPRs are typically more difficult to solve than are CCPs, but their objective function is more meaningful.

Here we present the work of *Li et al.* [16] who develop both a CCP and a SPR model for the VRPTW with stochastic travel and service time. Their CCP model consists of planning a set of routes that is subject to the chance constraints that the probability of route failure must be below a certain threshold. They define two kinds of chance constraint, those of driver duration and time windows. First, it is expected that the vehicle arrives at all customers within their time windows, with a given confidence level α. Moreover, it is stipulated that the vehicle must return to the depot not later than the associated deadline l_0.

Their recourse version of the problem is modeled in two stages. In the first stage, the priori set of vehicles is determined depending on transportation costs.

After random variables of travel and service times are realized, the expected correction costs of route failure are then considered in the second stage where the objective is to minimize the gap between the effective arrival time and the time windows. Their objective is, as mentioned before, to design the first-stage solution to minimize the expected transportation costs of the second-stage solution. Their conclusion is that the CCP model is not well suited for their problem and that they obtain better results with the SPR model.

The difference between their work and ours is that they do not consider the selective tours approach. Moreover, they do not consider neither a dynamic environment and thus do not use any re-optimization procedure. Finally, we focus on service times variation while they focused on both service and travel times uncertainties.

Fuzzy optimization approach The fuzzy logic concept was first introduced by *L.A. Zadeh* in 1965. In contrast with "crisp logic", where binary sets have binary logic, the fuzzy logic variables may have a membership value of not only 0 or 1 – that is, the degree of truth of a statement can range between 0 and 1 and is not constrained to the two truth values of classic propositional logic. For each $\alpha \in [0, 1]$, the α-cut of a fuzzy set is the ordinary set of values where the membership is equal or greater than α.

Brito et al. in [3] establish a state of art in *fuzzy optimization in vehicle routing problems*. *Gupta et al.* in [10] defined a fuzzy model for routing problem with uncertainty in service time that use fuzzy numbers in their objective function. Another example is [14] where the authors model travel times as triangular fuzzy numbers.

Markov decision process Markov decision processes (MDPs), named after Andrey Markov, provide a mathematical framework for modeling decision-making in situations where outcomes are partly random and partly under the control of a decision maker. More precisely, a Markov Decision Process is a discrete time stochastic control process. At each time step, the process is in some state s, and the decision maker may choose any action a that is available in state s. The process responds at the next time step by randomly moving into a new state s', and giving the decision maker a corresponding reward $R_a(s, s')$.

We report here the work of Thomas and White [24] who use a MDP to model the anticipatory route problem. In their problem, service requests are anticipated according to the probability of occurrence of the request and to the reward associated. They compared their anticipatory policy to a reactive strategy and show that the first one outperform the second one, especially when customer service requests are likely to occur late.

Dynamic Programming Dynamic programming is based on sequential decision process where the goal is to optimize some objective. The main idea is to use *additional information* when taking a decision at each step of the algorithm (in the case of discrete time). Dynamic Programming is a set of mathematical and algorithmical tools that study these sequential decision process and calculate an optimal *policy*, i.e. a rule that tell at each step what decision to take. Dynamic Programming rely on the *principle of optimality* that states :

Definition 2 *An optimal policy has the property that whatever the initial state and initial decision are, the remaining decisions must constitute an optimal policy with regard to the state resulting from the first decision.*

This principle gives the idea of a resolution method, that is called *backward induction* that determine an optimal policy starting from the end of the problem and proceeding backward.

We report here one of the work of Secomandi and Margot [22] that deal with the VRP with stochastic demands under re-optimization. They propose a MDP that can be solved to optimality on instances with no more than 15 customers, and they also propose *partial reoptimization* based on the idea of restricting attention to a subset of all the possible MDP states and solving for an optimal policy on this subset of states. They develop two heuristics for selecting this subset and show that their method outperform the rollout algorithm.

The interested reader could find a state of art of SVRPs done by *Gendreau et al.* in [9]

4 Modeling

In this section we give the general deterministic model of our problem and we also define some models adapted to its dynamic and stochastic nature.

4.1 Notations

From a graph theoretical perspective, the problem can be stated as follows. Let $G = (V, E)$ be a complete undirected graph with V the vertex set of size n, E the edge set of size m and K be the set of technicians. V can be decomposed in four subsets $V_1, ..., V_4$:

- V_1 is the set of starting depots
- V_2 is the set of arriving depots
- V_3 is the set of the appointments
- V_4 is the set of the *postponables*

All edges (i, j) of E have an associated cost t_{ij} representing the travel time to go from one vertex to another. To simplify we consider that the cost t_{ij} is

symmetric and proportional to the euclidean distance $dist(i, j)$. Each vertex i has a time window $[e_i, l_i]$ where e_i and l_i are the earliest arrival and latest departure time at i, respectively. Moreover, s_i represents the service time at vertex i and the service times at depots are null (*i.e.* $s_i = 0,\ \forall i \in (V_1 \cup V_2)$). Each technician k has a starting depot s^k belonging to V_1 and an ending depot e^k belonging to V_2. Finally, we note $\delta^-(i)$ the set of predecessors of vertex i and $\delta^+(i)$ the set of its successors.

Variables

$$x_{ij}^k = \begin{cases} 1 \text{ if the vehicle } k \text{ takes the arc } ij \\ 0 \text{ either} \end{cases}$$

$$y_i^k = \begin{cases} 1 \text{ if the vertex } i \text{ is visited by the vehicle } k \\ 0 \text{ either} \end{cases}$$

t_i^k the arrival time at vertex i of technician k.

4.2 Deterministic model

We can model this problem as a MILP, we first present here the deterministic version of the models :

$$\max z_1 = \sum_{k \in K} \left(\sum_{i \in V_3} c_i^1 y_i^k + \sum_{i \in V_4} c_i^2 y_i^k \right) \tag{1}$$

$$\min z_2 = \sum_{k \in K} \sum_{(i,j) \in E} t_{ij} x_{ij}^k \tag{2}$$

$$\sum_{k \in K} \sum_{j \in \delta^+(i)} x_{ij}^k \leq 1 \qquad \forall i \in V \tag{3}$$

$$\sum_{i \in \delta^-(j)} x_{ij}^k - \sum_{i \in \delta^+(j)} x_{ji}^k = 0 \qquad \forall k \in K, \forall j \in (V_3 \cup V_4) \tag{4}$$

$$\sum_{k \in K} y_i^k \leq 1 \qquad \forall i \in (V_3 \cup V_4) \tag{5}$$

$$\sum_{j \in \delta^+(s^k)} x_{s^k j}^k = 1 \qquad \forall k \in K \tag{6}$$

$$\sum_{i \in \delta^-(e^k)} x_{i e^k}^k = 1 \qquad \forall k \in K \tag{7}$$

$$y_j^k - \sum_{i \in V} x_{ij}^k = 0 \qquad \forall k \in K, \forall j \in (V_3 \cup V_4) \tag{8}$$

$$x_{ij}^k (t_i^k + s_i + t_{ij} - t_j^k) \leq 0 \qquad \forall k \in K, \forall (i,j) \in E \tag{9}$$

$$t_i^k + M(1 - y_i^k) \geq e_i \qquad \forall\ i \in V, \forall k \in K \tag{10}$$

$$t_i^k + s_i - M(1 - y_i^k) \leq l_i \qquad \forall\ i \in V, \forall k \in K \tag{11}$$

$$s_i = 0 \qquad \forall i \in (V_1 \cup V_2) \tag{12}$$

$$x_{ij}^k \in \{0,1\} \qquad \forall (i,j) \in E, \forall k \in K \tag{13}$$

$$y_i^k \in \{0,1\} \qquad \forall i \in V, \forall k \in K \tag{14}$$

$$t_i^k \in \mathbb{R}^+ \qquad \forall i \in V, \forall k \in K \tag{15}$$

$$M \text{ is a large constant} \tag{16}$$

The principal objective (1) is to maximize the number of satisfied requests, and especially to maximize the number of satisfied appointments. The second objective (2) is less important, it aims to reduce the transportation costs. We thus define $\underset{i\in V_3}{c_i^1} >> \underset{i\in V_4}{c_i^2}$ to prioritize the appointments.. Flow constraints (3-8) define the distribution of vehicles on the graph. (3) ensures that an arc is taken by at most one vehicle, (4) implies that the entering degree equals the exiting degree for each nodes (except for the depots). Constraint (5) ensures that a node is visited by only one technician (except for the depots) and (6-7) obligate the technician to start and end at its respective starting and ending depot. Constraints (8) maintains the coherence of the system : if a customer is serviced then there must be an arc that enter the vertex. On the contrary, if the customer is not serviced, no arcs have been taken to go visit it. Constraint (9) is the subtour elimination constraint, it guarantee that, for any successor j of customer i on the path followed by vehicle k, $t_j^k > t_i^k$. This constraint cannot be verified if there is a cycle in vehicle k route. The linearized version of (9) is :

$$t_i^k + d_i + t_{ij} - t_j^k \leq (1 - x_{ij}^k)\ M, \quad \forall k \in K, \forall (i,j) \in E \tag{17}$$

Finally, constraints (10-11) correspond to the respect of time windows.

4.3 Robust model

A robust model is a model that take in account all the uncertainties so that it can be suitable for all scenarios. The only uncertainty we consider in the problem is the service time. By experience we know that the service time can variate between certain values, precisely $\underline{s_i} \leq s_i \leq \overline{s_i}$. However like *Cortes et al* point out in [5], the worst case scenarios faced by technicians in a day are far smaller than the sum of the worst case service times of every client in the route.

The first robust model we give is intuitive, we just have to play with a coefficient α_i for each vertex to control the level of robustness of the model.

$$\begin{aligned}
&(1-8)\\
t_i^k + d_i + t_{ij} - t_j^k &\leq (1 - x_{ij}^k)\ M && \forall k \in K, \forall (i,j) \in E\\
t_i^k + M(1 - y_i^k) &\geq e_i && \forall\ i \in V, \forall k \in K\\
t_i^k + \alpha_i \overline{s_i} - M(1 - y_i^k) &\leq l_i && \forall\ i \in V, \forall k \in K\\
\underline{s_i} \leq s_i &\leq \overline{s_i} && \forall i \in V\\
\frac{\underline{s_i}}{\overline{s_i}} \leq \alpha_i &\leq 1 && \forall i \in V\\
&(12-16)
\end{aligned}$$

Concretely, if we choose $\alpha = 1$ we have $s_i = \overline{s_i}$ and we optimize the worst possible case.

The second model is directly inspired from the work of Cortes [5] and involve their upper bound $U_k = \alpha \sum_{i \in V} \gamma_i$ presented in the section 2 :

$$
\begin{aligned}
& (1-8) \\
t_i^k + d_i + t_{ij} - t_j^k \leq (1 - x_{ij}^k)\ M & \qquad \forall k \in K, \forall (i,j) \in E \\
t_i^k + M(1 - y_i^k) \geq e_i & \qquad \forall\ i \in V, \forall k \in K \\
t_i^k + s_i + z_i - M(1 - y_i^k) \leq l_i & \qquad \forall\ i \in V, \forall k \in K \\
\sum_{i \in V} z_i y_i^k \geq \alpha \sum_{i \in V} \gamma_i y_i^k & \qquad \forall k \in K \\
\underline{s_i} \leq s_i \leq \underline{s_i} + \gamma_i & \qquad \forall i \in V \\
0 \leq z_i \leq \gamma_i & \qquad \forall i \in V \\
0 \leq \alpha \leq 1 & \\
& (12-16)
\end{aligned}
$$

γ_i and z_i are the maximum and the effective deviation of service time at customer i, respectively.
α is still the parameter that controls the level of robustness. The idea of this method is to define the bound U_k such like the sum of the effective deviation of service time tends to U_k

4.4 SPR model

$$
\begin{aligned}
& \min z_3 = F(x) + E(\omega) \\
& F(x) = \sum_{k \in K} \left(\sum_{i \in V_3} c_i^1 y_i^k + \sum_{i \in V_4} c_i^2 y_i^k \right) \\
& E(\omega) = \sum_{k \in K} E(P_k) \\
& P_k = \sum_{i \in V} \left(\lambda_i \max(t_i^k + s_i - l_i, 0) + \sigma_i \max(e_i - t_i^k, 0) \right) \\
& \quad (3-9) \\
& \quad (12-16)
\end{aligned}
$$

The optimal policy from such a model is a single first-stage decision and a collection of recourse decisions defining which second-stage action should be taken in response to each random outcome. $F(x)$ is the first-stage objective function, it aims to maximize the number of satisfied requests. $E(\omega)$ is the second-stage objective function that represents the recourse costs. $E(P_k)$ is the expected value of P_k which is the total penalties associated to the violation of time windows on route k. λ_i and σ_i are penalties associated to the violation of time window

which penalize if the technician finishes his task later than expected, and if the technician arrives earlier than expected, respectively. Technically, if we raise λ_i and σ_i to high values, we obtain the case of hard time windows.

5 Contribution

In this part, we will focus in explaining the two methods we designed. As the instances of Fabien Tricoire consider one technician per depot, we can see this problem as a multi depot VRPTW with one technician per depot, or as a multi-TSPTW.

5.1 Tabu Search

Initialization The initialization is an important part of a metaheuristic, where a starting solution is built thanks to an appropriate method selected depending on the nature of the problem. In our case, we can decompose this initialization phase into two main sub-phases :

- Construction of routes containing only *appointments*
- Addition of *postponables* to these routes

We first start by assigning each appointment to its nearest depot. As we are considering a m-TSP, it amounts to assign each appointment to a technician future route, we thus obtain $|K|$ groups of appointments. Then, in each group, we sort the appointments depending on the middle of their time window, and we build the resulting routes. This method is used by *Polacek et al* in [19] and by *Cordeau et al* in [4]. At the end of this first sub-phase, we obtain small routes containing only appointments, as it is illustrated in figure 4

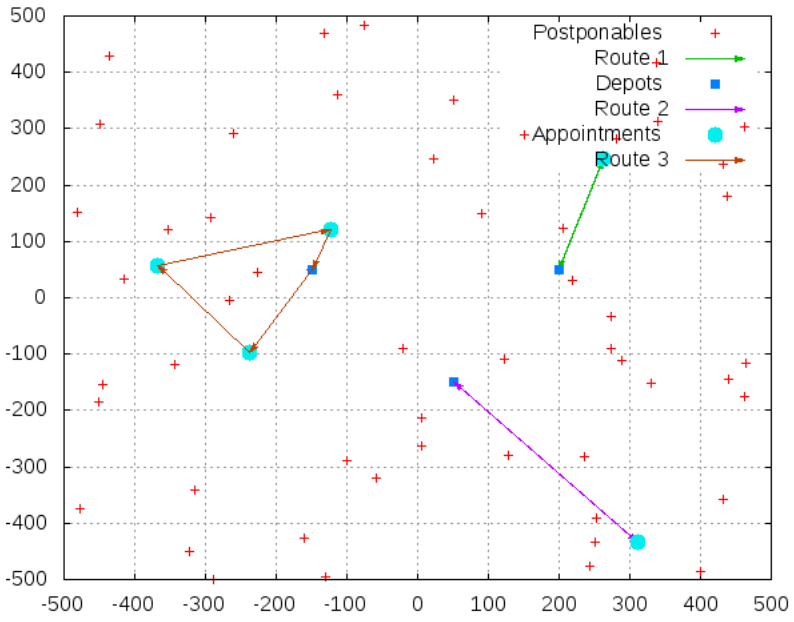

Fig. 4. Routes after the first sub-phase

The second sub-phase is a bit more complicated. Here we use a slightly modified version of the Solomon Best Insertion heuristic described in part 3. This procedure works in parallel on each route and is iterative. At each step, we look for the best postponable to insert in each route, regarding to its *insertion cost*. In case where a same postponable is selected for several routes, we privilegiate the insertion with minimum cost. We repeat the procedure until the *maximum length* of all routes is reached. The *maximum length* is a parameter set by the user and is equivalent to the maximum duration as we assimilate distances and travel times. Its standard value corresponds to the duration of a working day, but it can be modified to obtain more or less robust solutions.

We use two criterias to evaluate the insertion cost :

- a distance cost c_1
- a time cost c c_2

Figure 5 represents the insertion of customer 2 between customer 1 and 3, we will use it to define the costs defined above.

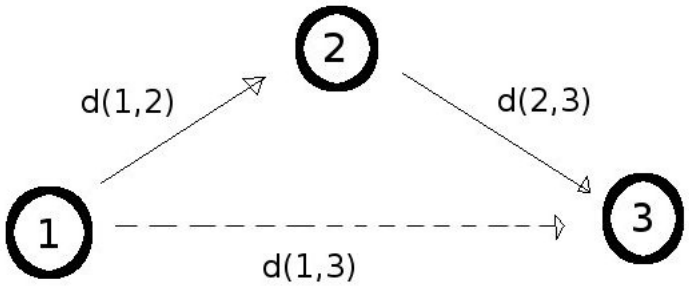

Fig. 5. Illustration of an insertion

The first one simply calculates the distance added by the insertion of the new customer.

$$c_1 = d(1,2) + d(2,3) - d(1,3)$$

The second computes the difference between the new arrival time a' and the previous arrival time a at the customer located just after the one inserted.

$$c_2 = \underbrace{\Big(d_1 + d(1,2) + s_2 + d(2,3)\Big)}_{a'_3} - \underbrace{\Big(d_1 + d(1,3)\Big)}_{a_3}$$

where d_i is the departure time from customer i and s_i the service time at customer i. We use expected service times as value for s_i.

As Solomon observed, we can combine and parametrize those two criterias in order to obtain different solutions.

$$\textit{insertion cost} = \alpha c_1 + (1-\alpha)c_2 \quad , 0 \leq \alpha \leq 1$$

We let the user decides of the number of runs but empirically, 5 runs seems to be enough. We then keep the *best solution* that is chosen acording three lexicographic objectives : the cumulative lateness of the solution, the number of satisfied requests and the distance cost. This solution becomes the starting solution for the tabu search procedure. Figure 6 illustrates the tours after the initialization phase.

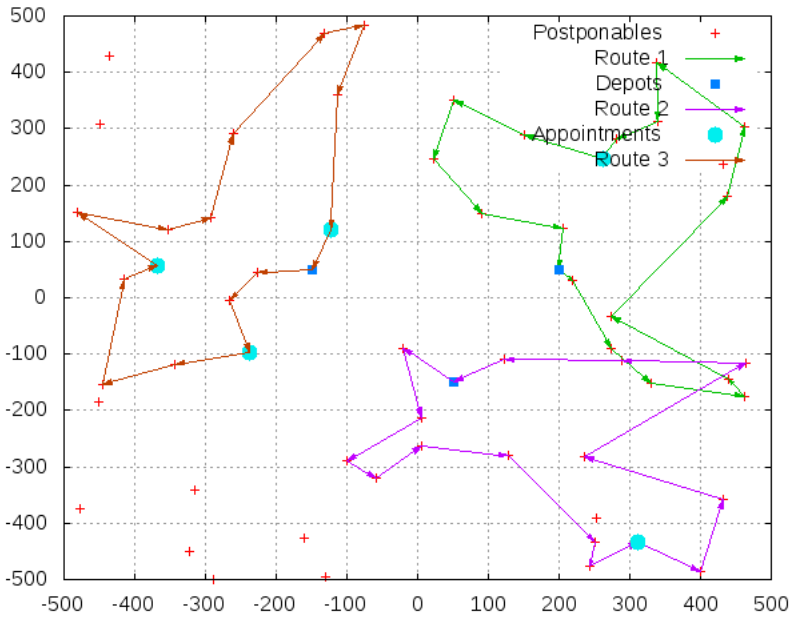

Fig. 6. Routes after the initialization phase

Evaluation The evaluation is made in two times, as we use the stochastic programming with recourse framework. We first look at its weight in terms of satisfied requests. Secondly, we subject this solution to a *Monte-Carlo simulation* to evaluate its robustness.

Evaluate the profit As we have requests that are more important than others (*the appointments*), the goal is not exactly to maximize the number of satisfied requests but rather to maximize the weight of each route, the weight of a route beeing the sum of the weight of all the customers belonging to it.

Evaluate the robustness Evaluating the profit would be enough if we consider deterministic service times. However, as we are dealing with stochastic service times, we want to have a solution that can resist and adapt to uncertainties. In most cases, an optimal solution for the deterministic case would not resist really well to uncertainties, that's the reason why we are not looking for an *optimal solution* in the sense we usually know, but rather for a compromise between robustness and optimality.

Monte-Carlo simulation is a simple method that rely on repeated random sampling to compute results. In our case, it amounts to generate service times (following a probability law)for each customer and observe the behavior of the solution, and more especially the lateness induced by these new service times. This operation is repeated for a given number of iterations, and at the end we look at the mean of all the lateness obtained at each run of the method.

The algorithm 1 shows the behaviour of one run of the Monte-Carlo simulation.

Algorithm 1: One run of MC simulation

```
1.1  input : Solution S ;
1.2  begin
1.3      lateness = 0 ;
1.4      for each route R in S do
1.5          for each C in R do
1.6              Actualize arrival time at customer C ;
1.7              Generate service time of customer C according to a probability
                 law ;
1.8              Actualize departure time at customer C ;
1.9              lateness += max(C.departureTime − C.endTW, 0) ;
1.10         end
1.11     end
1.12     return lateness ;
1.13 end
```

Modifying service times implies a modification of the arrival and departure times, so we need to *actualize* these values in order to compute later the lateness generated by the new service times.

Algorithm 2: Monte-Carlo simulation

```
2.1  input : Solution S, int nbIterations ;
2.2  begin
2.3      lateness = 0 ;
2.4      for i ← 0 to nbIterations do
2.5          lateness+ = Algorithm1(S) ;
2.6      end
2.7      lateness = lateness/nbIterations ;
2.8      return lateness ;
2.9  end
```

Doing this, we obtain a correct approximation of what can be the real expected lateness. The more the mean lateness is small, the more the solution is robust.

Stopping criterion The stopping criterion of this method is a limit of time. When building the initial solution, the limit of time can be of several hours

because generally this procedure can be start the night before. But during the day, the tabu search is more limited because we have to give the technician a quick response and because new events occur frequently.

Neighborhood structures The neighborhood structure we proposed initially are :

- Suppression / Insertion
- Swap

During the day When a technician notifies a lateness, or more generally a plan modification, we launch the tabu search with the new information given by the technician. After a certain amount of time (30 seconds for example), we stop the tabu search and we transmit the results to the technician.

It would also be possible to run the tabu search in *background* as *Gendreau et al* did but we did not implement that feature yet.

5.2 Dynamic Programming

To develop this second approach, we modified the nature of the problem, restricting the possibilities of re-optimization. Indeed now, we consider that each appointment is assigned to only one technician and that even during the re-optimization phase we can't assign an appointment to another technician. This choice was made in order to limit the possible decisions that have to be taken in dynamic programming algorithms.

Initial Solution As in the first method, we need an initial solution (i.e. a *plan*). In order to obtain this plan, we used CPLEX 12.0 with the following model that corresponds to the new definition of the problem. We introduce a new parameter u_i^k that is defined like it follows :

$$u_i^k = \begin{cases} 1 \text{ if appointment } i \text{ is assigned to technician } k \\ 0 \text{ either} \end{cases}$$

$$\max z = \sum_{k \in K} \sum_{i \in (V_3 \cup V_4)} c_i y_i^k \tag{18}$$

$$\sum_{k \in K} \sum_{j \in \delta^+(i)} x_{ij}^k \leq 1 \qquad \forall i \in V \tag{19}$$

$$\sum_{i \in \delta^-(j)} x_{ij}^k - \sum_{i \in \delta^+(j)} x_{ji}^k = 0 \qquad \forall k \in K, \forall j \in (V_3 \cup V_4) \tag{20}$$

$$\sum_{k \in K} y_i^k \leq 1 \qquad \forall i \in (V_3 \cup V_4) \tag{21}$$

$$\sum_{j \in \delta^+(s^k)} x_{s^k j}^k = 1 \qquad \forall k \in K \tag{22}$$

$$\sum_{i \in \delta^-(e^k)} x_{i e^k}^k = 1 \qquad \forall k \in K \tag{23}$$

$$y_j^k - \sum_{i \in V} x_{ij}^k = 0 \qquad \forall k \in K, \forall j \in (V_3 \cup V_4) \tag{24}$$

$$x_{ij}^k (t_i^k + s_i + t_{ij} - t_j^k) \leq 0 \qquad \forall k \in K, \forall (i,j) \in E \tag{25}$$

$$t_i^k + M(1 - y_i^k) \geq e_i \qquad \forall\ i \in V, \forall k \in K \tag{26}$$

$$t_i^k + s_i - M(1 - y_i^k) \leq l_i \qquad \forall\ i \in V, \forall k \in K \tag{27}$$

$$\sum_{i,j \in E} t_{ij} x_{ij}^k + \sum_{i \in V} (\mathbb{E}(s_i) + w_i) y_i^k \leq \alpha l_{e^k} \qquad \forall k \in K \tag{28}$$

$$y_i^k \leq u_i^k \qquad \forall\ i \in V_3, \forall k \in K \tag{29}$$

$$x_{ij}^k \in \{0,1\} \qquad \forall (i,j) \in E, \forall k \in K \tag{30}$$

$$y_i^k \in \{0,1\} \qquad \forall i \in V, \forall k \in K \tag{31}$$

$$u_i^k \in \{0,1\} \qquad \forall i \in V, \forall k \in K \tag{32}$$

$$t_i^k \in \ \mathbb{R}^+ \qquad \forall i \in V, \forall k \in K \tag{33}$$

$$M \text{ is a large constant} \tag{34}$$

As the objective and most constraints are the same than in the deterministic model previously presented, we will only present the new constraints. Constraint (28) states that the total length of each route must be lower or equal than a threshold αe_{l^k} where l_{e^k} is the latest possible arrival time at the ending depot. The parameter α serves to control the *level of robustness* of the solution. Constraint (29) indicates that an appointment i belongs to route k only if i was assigned to k.

Re-optimization A dynamic programming model rely mainly on a state space, and on transition and cost functions. As we said in the bibliographic part, the goal of dynamic programming is to find an optimal policy for taking decisions at each step of the algorithm. We now describe our dynamic programming model that is defined for a route and not for the whole solution :

State At each step k we know :

- p_k the current position of the technician
- time t_k that corresponds to the end of service at p_k
- the set of appointments not yet serviced R_k
- the optimal route of postponables L_k to go to the next appointment. L_k is function of p_k et t_k and is made of postponables that are near the route k

We can represent the system at step k by :

$$x_k = \Big(p_k, t_k, R_k, L_k(p_k, t_k) \Big) \tag{35}$$

Notations

- l_1 first postponable on route L_k
- p is a probability value ranged between 0 and 1
- α_i are penalties associated with lateness at the ending deopt d , $\alpha_i < 0$
- w_i define the intervals for the piecewise linear penalty.

Cost function

$$J_k\Big(p_k, t_k, R_k, L_k(p_k, t_k) \Big) = \max \Big\{ J_k^D(...), J_k^U(...), J_k^R(...) \Big\} \tag{36}$$

- $J_k^D \rightarrow$ Cost if we go directly to the next appointment
- $J_k^U \rightarrow$ Cost if we go to the next postponable of list L_k
- $J_k^R \rightarrow$ Cost if we go back to the depot

$$J_k^D = \max_{r \in R_k^{'}} \left\{ c_r - \omega(p_k, r) + \mathbb{E}_{s_r} \left[J_{k+1} \left(r, \max(t_k + t_{p_k r}, e_r) + s_r, L_{k+1}(r, \max(t_k + t_{p_k r}, e_r) + s_r), R_k \setminus \{r\} \right) \right] \right\} \tag{37}$$

$$J_k^U = c_{l_1} + \left(\mathbb{E}_{s_{l_1}} \left[J_{k+1}\big(l_1, t_k + t_{p_k l_1} + s_{l_1}, R_k, L_k(p_k, t_k) \setminus \{l_1\}\big) \right] \right) \tag{38}$$

$$J_k^R = \alpha_1(\max(\min(t_k + t_{p_k d}, w_1), 0)) + \alpha_2(\max(\min(t_k + t_{p_k d}, w_2) - w_1), 0)) + \alpha_3(\max(t_k + t_{p_k d} - w_2, 0)) \tag{39}$$

With $\omega(p_k, r) = \alpha_1(\max(\min(t_k + t_{p_k r}, w_1), 0)) + \alpha_2(\max(\min(t_k + t_{p_k r}, w_2) - w_1), 0)) + \alpha_3(\max(t_k + t_{p_k r} - w_2, 0))$ the penalty associated to the lateness at appointment r.

And with $R_k^{'}$ the set of possible decisions defined like follows :

$$R_k^{'} = \big\{ r | r \in R_k, t_{p_k} + t_{p_k r} \leq e_r + \tau_r, \mathbb{P}\Big[s_r | \max(t_{p_k} + t_{p_k r}, e_r) + s_r \leq l_r \Big] \geq p \big\} \tag{40}$$

This set uses several constraints in order to limit the possible choices. The first one states that we can't go service an appointment if we know that the arrival time will be greater than a certain threshold. The second constraint states that we can't go service a customer if the probability that the service ends after the deadline is greater than a certain probability p. Playing with parameters p and τ lead to authorize or not a certain transition.

Behavior The most difficult part in dynamic programming is the construction of the state space. Because we need to build a state space with a limited number of states, so that the calculation can be possible, and we have to build it such like it correspond to what we want. The state space is nothing more than a graph where nodes correspond to states and arcs correspond to transitions between states. Then, solving our problem amounts to solve a stochastic shortest path problem. And to solve it, we simply use the *backward induction* mentionned before and that is based on the cost functions seen above. Figure 7 illustrates the representation of the state space.

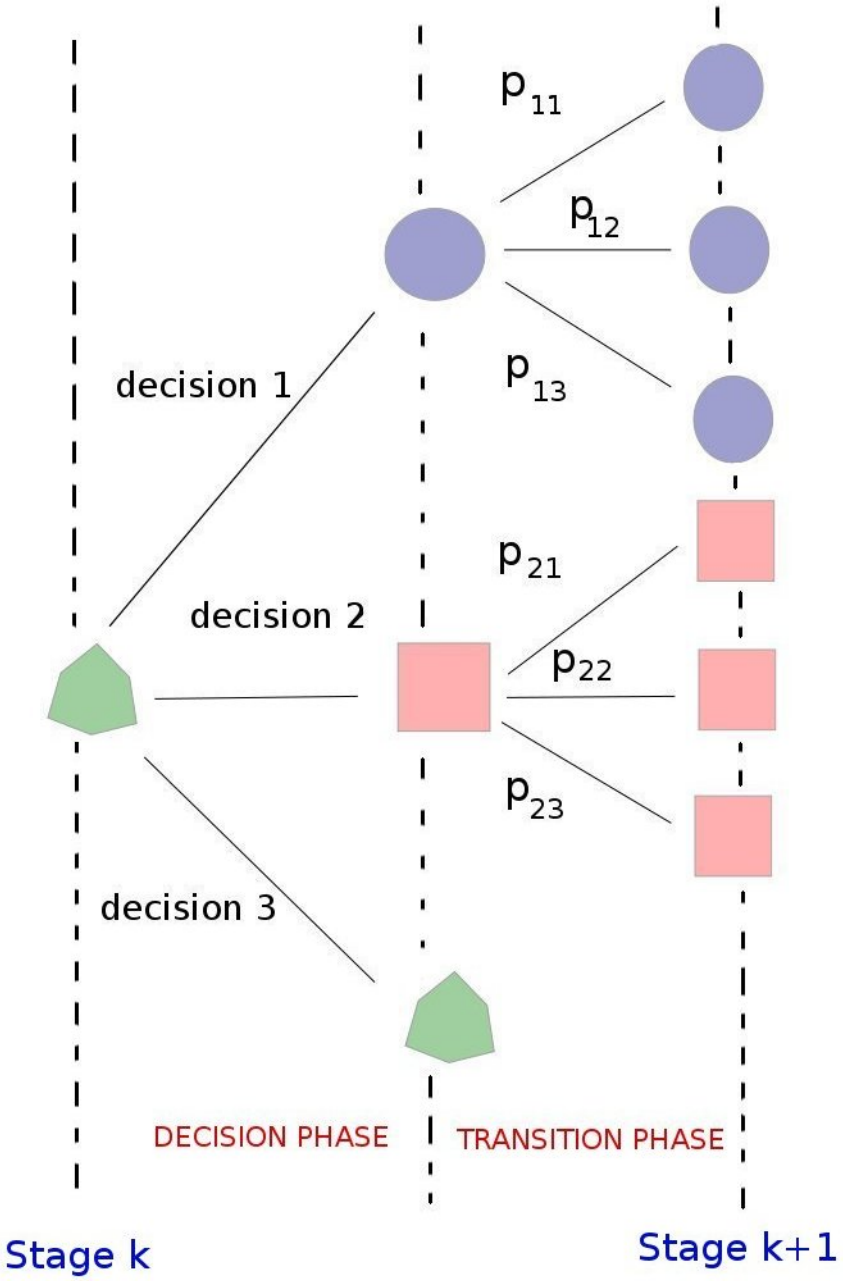

Fig. 7. Illustration of dynamic programming

At stage k, we ended our service at a customer (here in the example we are at the depot). We have three possible future destinations that are a postponable, an appointment and the ending depot. If for example, the DP algorihm and backward induction told us that going to the appointment is the best possible choice, we move to the appointment state. But as service times are stochastic, we have some probabilities to end service at the appointment 20 minutes late, or 20 minutes sooner, etc... That's why there is a transition phase that obey to a probability law which in our case is the normal law.

Now during the day when a technican report a modification in the plan, we simply calculate the *shortest path* from the current position of the technician to the goal target that is the depot. It is not exactly a shortest path calculation but more a *maximum weigh path* because we still want to maximize the number of satisfied requests, however it is the same principle, we just have to apply the dynamic programming algorithm (backward induction).

6 Experiments

We only have a few results for the moment, all tests are made on the instances of the Fabien Tricoire thesis.

6.1 Tabu Search with Monte-Carlo simulation

As we put aside this method when I arrived in Montreal, we only have results after the initialisation phase and the first evaluation by Monte-Carlo simulation. It appears to work well, as you have seen on the graphs presented in the method explanation part. But we don't now yet if the tabu search is efficient or not because it is not implemented.

6.2 Dynamic programming

Initial Solution We obtain optimal results on small instances where technicians can handle all the requests (5 over 20 instances) . However, on large instances the resolution takes too much time for a result far from the optimum. We launch CPLEX for a day on instances with 60 customers without finding good solutions, even when tuning branching rules and using different methods (simplex, interior point,...).

Re-optimization We spent a lot of time defining the dynamic programming model, and choosing the range of re-optimization. We developed several models where the set of possible decisions was too wide, so the state space was too wide too and the calculation were impossible. We now found a proper formulation and the results will follow soon.

7 Conclusion

In conclusion, we proposed two approaches coming from differents horizons to treat a problem which has been very little studied for now. The first one allows a large re-optimization but with no guaranty about optimality while the second one limits the range of re-optimization but try to focus on optimal solution. Now it remains to obtain more results in order to compare empirically these two methods and see if they are well adapted to our problem. For future work, it could be interesting to focus on transportation times which can be subject to uncertainties too. It also could be interesting to introduce a different priority for each postponable in order select in priority the postponables that are known for a long time. Finally, we could test these methods on instances where there are several technicians per depot.

References

1. Runka Andrew. Ant colony optimization algorithms with local search for the dynamic vehicle routing problem. Master's thesis, Brock University, 2008.
2. Dimitris Bertsimas and Garrett. Van Ryzin. A stochastic and dynamic vehicle routing problem in the euclidean plane. Technical report, 1991.
3. J. Brito, J.A. Moreno, and J.L. Verdegay. Fuzzy optimization in vehicle routing problems. *IFSA-EUSFLAT*, 2009.
4. J.F. Cordeau, G. Laporte, and A. Mercier. A unified tabu search heuristic for vehicle routing problems with time windows. *Journal of the Operational Research Society*, 52(8):928–936, 2001.
5. Cristian E. Cortes, Fernando Ordonez, Sebastian Souyris, and Andres Weintraub, editors. *Routing technicians under Stochastic Service Times: A Robust Optimization Approach*. TRISTAN VI, June 2007.
6. G. B. Dantzig and J. H. Ramser. The truck dispatching problem. *Management Science*, 6:80–91, 10 1959.
7. Frédéric Dugardin. Optimisation réactive de tournées de service en environnement dynamique. 09 2006.
8. Michel Gendreau, François Guertin, Jean-Yves Potvin, and Eric D. Taillard. Parallel tabu search for real-time vehicle routing and dispatching. *Transportation Science*, 33(4):381–390, 1999.
9. Michel Gendreau, Gilbert Laporte, and René Séguin. Stochastic vehicle routing. *European Journal of Operational Research*, 88(1):3–12, 1996.
10. Radha Gupta, Bijendra Singh, and Dhaneshwar Pandey. Fuzzy vehicle routing problems with uncertainty in service time. *Int.J.Contemp.Math.Sciences*, 2009.
11. Housroum Hayan. *Une approche genetique pour la resolution du probleme VRPTW dynamique*. PhD thesis, Universite d'Artois, 2005.
12. H.N.Psaraftis. Dynamic vehicle routing problems. *Vehicle Routing : Methods and Studies*, 1988.
13. Soumia Ichoua, Michel Gendreau, and Jean-Yves Potvin. Diversion issues in real-time vehicle dispatching. *Transportation Science*, 34(4):426–438, 2000.
14. Brito J., Campos C., J.P. Castro, F.J. Martinez, B. Melian, J.A. Moreno, and J.M Moreno. Fuzzy vehicle routing problems with time windows. *IPMU*, 2008.

15. Allan Larsen. *The Dynamic Vehicle Routing Problem*. PhD thesis, Technical University of Denmark, 2001.
16. Xiangyong Li, Peng Tian, and Stephen C.H. Leung. Vehicle routing problems with time windows and stochastic travel and service times: models and algorithm. *International Journal of Production Economics*, In Press, Accepted Manuscript:–, 2010.
17. Roberto Montemanni, Luca M. Gambardella, Andrea E. Rizzoli, and Alberto V. Donati. A new algorithm for a dynamic vehicle routing problem based on ant colony system. Technical report, 2002.
18. Fernando Ordonez, Zhihong Shen, and Maged Dessouky. The minimum unmet demand stochastic vehicle routing problem. November 2006.
19. Michael Polacek, Richard F. Hartl, Karl Doerner, and Marc Reimann. A variable neighborhood search for the multi depot vehicle routing problem with time windows. *J. Heuristics*, 10(6):613–627, 2004.
20. Daron Roberts. Algorithms for stochastic vehicle routing problems, 1998.
21. Y. Rochat and E. Taillard. Probabilistic diversification and intensification in local search for vehicle routing. *Journal of Heuristics*, 1:147–167, 1995.
22. N. Secomandi and F. Margot. Reoptimization approaches for the vehicle-routing problem with stochastic demands. *Operations research*, 57(1):214–230, 2009.
23. M. M. Solomon. Algorithms for the vehicle routing and scheduling problems with time window constraints. *Oper. Res.*, 35(2):254–265, 1987.
24. Barrett W. Thomas and Chelsea C. White III. Anticipatory route selection. *Transportation Science*, 38(4):473–487, 2004.
25. Fabien Tricoire. Vehicle and personnel routing optimization in the service sector: application to water distribution and treatment. *4OR*, 5(2):165–168, 2007.
26. Xin Zhao, Gilles Goncalves, and Remy Dupas. A genetic approach to solving the vehicle routing problem with time-dependent travel times. *16th Mediterranean Conference on Control and Automation*, 2008.