A HYBRID EVOLUTIONARY ALGORITHM FOR VEHICLE ROUTING PROBLEM WITH STOCHASTIC DEMANDS

ROBINSON A. JAQUE P.

ID: 299800

Thesis presented as a partial requirement for the degree of

MASTER OF SCIENCES

COMPUTER SCIENCE

Advisor:
GERMÁN HERNÁNDEZ, PH.D.
Associated Professor

NATIONAL UNIVERSITY OF COLOMBIA
SCHOOL OF ENGINEERING
DEPARTAMENT OF COMPUTER AND INDUSTRIAL ENGINEERING
BOGOTA D.C.
2011

Approved by the School of Engineering in fulfillment of the requirements to grant the title of **Master of Sciences — Computer Science**

Germán Hernández, Ph.D.
Advisor of the Thesis

1, Ph.D.
Member of the Jury

2, Ph.D.
Member of the Jury

National University of Colombia
Bogota D.C., Junio of 2011

ABSTRACT

A HYBRID EVOLUTIONARY ALGORITHM FOR VEHICLE ROUTING PROBLEM WITH STOCHASTIC DEMANDS

by

ROBINSON A. JAQUE P.
Master of Sciences in Computer Science

NATIONAL UNIVERSITY OF COLOMBIA

Advisor: Germán Hernández, Ph.D.

In this work we propose a hybrid dynamic programming-evolutionary algorithm to solve the vehicle routing problem with stochastic demands, it is a well known NP-hard problem where uncertainty enhances the computational efforts required to obtain a feasible and near-optimal solution. We develop an evolutionary technique where a rollout dynamic programming algorithm is applied as local search method to improve the quality of solutions. Motivated by computational considerations, the rollout algorithm can be applied partially, so, this finds competitive solutions in large instances for which the global rollout dynamic programming strategy is time unfeasible.

ACKNOWLEDGMENTS

Last thing to do :-)

DEDICATION

To Hari

Contents

List of Figures

List of Tables

Notation

Symbol	Definition
n	Number of customers
d_{ij}	Distance between a pair of customers i and j
$E[L_\tau]$	Expected distance of an apriori solution (base sequence) τ
l	Location of vehicle
q_l	Vehicle capacity when customer l has already been served.
$p_i(j)$	Probability of customer i take the demand value j
K	Maximal demand
$\bar{D}_i$	Maximal demand of customer i
Q	Maximal capacity of vehicle
P_k	Evolutionary algorithm generation in the generation k.
$I_i^{P_k}$	Individual i of the population P_k.
$\circledast$	Crossover operator in evolutionary algorithm.
$Prob_m$	Probability of mutation.
$Delta_{E'}^{P_k}$	Rate of fitness improve in the generation P_k
$\bar{\Gamma}_{P_k}$	Best expected distance obtained by an individual in the generation k
κ	Number of iterations of evolutionary algorithm

CHAPTER 1

Introduction

1.1. Introduction

The classic issue about Vehicle Routing Problem (VRP) is a well known *NP-hard* [**?**] optimization problem of high importance in different logistics; it consists in delivery goods to a set of customers geographically dispersed, using a fleet of vehicles that begin its route in the central depot. The problem consists in assigning to each vehicle a route with the objective of minimizing the transportation cost.

The cost generated in the transport of goods, such as the size of the fleet of vehicles maintenance, combustible, and so on, are significant due to the fact that to the transport processes are involved at all stages of production processes, accounting for 10% to 20% of the final product cost [**33**].

One of the first investigation that studied the vehicle routing problem took place in the year 1959. In that work, Dantzing and Ramser [**11**] analyzed an oil dispatching problem with trucks; that problem arise as a generalization of the classic traveling salesman problem (TSP), where the salesman has to visit a set of customers for only one time and then come back to the origin point, building a Hamiltonian road on the graph consisting of customers (vertices) and the possible paths between clients (edges).

Different variations of the VRP have been proposed with the aim of approaching the problem real contexts; these problems include the addition of variables and constraints. The figure below shows a diagram with the most popular variants of VRP.

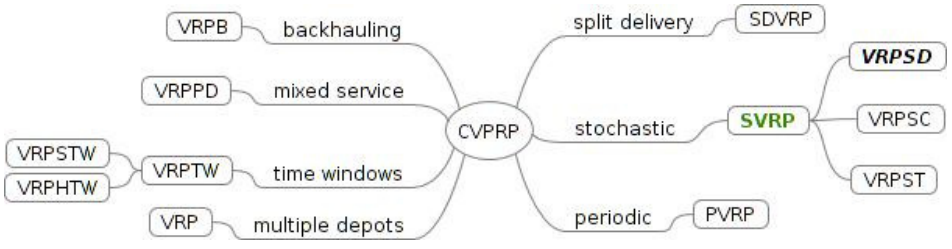


FIGURE 1.1.1. Basic variants of the Vehicle Routing Problem

When vehicle capacity is fixed, the Capacited Vehicle Routing Problem (CVRP) originates; if there are many depots, then we have Multiple Deposit Vehicle Routing

Problem (MDVRP). The Split Delivery Vehicle Routing Problem (SDVRP) is a relaxation of the problem that allows a custumer to be served by several vehicles, being important in cases where the customer demand exceeds the vehicle capacity. Generally the VRP provide a planning for a fixed period and the Periodic Vehicle Routing Problem (PVRP) provides that planning for m periods.

One of the most studied variants of the problem is caused by including time windows for deliveries, Vehicle Routing Problem with Time Windows (VRPTW), for this problem can be considered hard time windows (VRTHTW) were deliveries outside the established periods can not be done, and soft time windows (VRPSTW) in which deliveries can be made outside these periods, but with a penalty.

The problems that include pick-up and delivery can be divided in Vehicle Routing Problem with Backhauls (VRPB) and Vehicle Routing Problem with Pick-Up and Delivery (VRPPD) in the VRPB the group of custumers is divided in two subgroups, for the first group, all pick-up's are made of some product, then the vehicle returns to the depot and the deliveries are realized to the second group of custumers. In the VRPPD, the pick-up's and deliveries to customers are made simultaneously.

The Stochastic Vehicle Routing Problem (SVRP) arise when there is uncertainty about some of the components of the VRP, i.e., one or more variables are random. Usually these problems include Vehicle Routing Problem with Stochastic Customers (VRPSC) where customers appears randomly, the Vehicle Routing Problem with Stochastic Times (VRPST) where travel times are random and Vehicle Routing Problem with Stochastic Demands (VRPSD), where the customers demands are known only with a probability distribution.

The SVRP's differ from the deterministic VRP in many important aspects. The solution concept is different, since many properties of the deterministic problem are not manageable in the stochastic case and solution methodologies are considerably more complex; hence, SVRP is often considered a computationally intractable problem and only small instances can be solved optimally and algorithms are difficult to design and evaluate [**19**].

The SVRP is either often modeled in the framework of stochastic programming (optimization) integer or mixed or as a Markov decision process. In stochastic programming, problems are usually modeled as two-stage, as chance constrained program (CCP) or as a stochastic program with recourse (SPR).

The VRPSD is an open problem of great importance in logistics owing to the diversity of real situations it represents. This problem occurs in the delivery of home heating oil [**13**] in which each customer maintains a local inventory of the product and

consumes an amount of oil each day; therefore, each day a fleet of trucks is dispatched to resupply a subset of customers. Stochastic demands are also evident in the collection of money by the vehicles of values, e.g. when collect money by a central bank [**16**] from several but not all of its branches every day; here the capacity of the vehicle used may be constrained for an upper bound on the amount of money that a vehicle might carry for safety reasons. The distribution of demand at each certain branch may be different, associated with the amount of money it handles. Other VRPSDs arise in delivering the post to large customers [**21**], vending machines [**34**] or delivering medical supplies in response to large-scale emergency [**12**], or in recycling and waste management, among others.

1.2. Proposal

To design a hybrid evolutionary algorithm which combine stochastic dynamic optimization operators to find solutions for the vehicle routing problem with stochastic demands. Moreover, to evaluate the algorithm performance.

1.3. Objective

To design and test a hybrid evolutionary algorithm - Stochastic Dynamic Optimization (SDO) operator for the vehicle routing problem with stochastic demands.

1.3.1. Specifics objectives

(1) To analyze the alternatives for modeling and representing the problem, and select the most computationally convenient one.
(2) To design evolutionary and stochastic dynamic optimization operator algorithm to solve VRPSD.
(3) To implement the designed algorithm.
(4) To select benchmarking problems instances and alternative algorithms for testing.
(5) To develop experimental analyses and comparisons.

1.4. Contributions

The evolutionary algorithms have not been broadly used to solve the vehicle routing problem with stochastic demands; thus, a hybrid evolutionary algorithm which combine stochastic dynamic optimization operators is proposed. This contribute with a new methodology to deal with the problem.

1.5. Outline

In this first chapter a summary of the issues to be addressed was carried out, in chapter 2 presents the background and the state of the art of the VRPSD; chapter 3 presents the methodology used and the proposed algorithm. Chapter 4 presents the experiments and the numerical results and compares and discuss the results. Finally, chapter 5 presents the conclusions of the study.

CHAPTER 2

Background

In this chapter, we carry out a specialized literature review of VRPSD, and point out applications and methodologies used to deal with it. Moreover, we examine modeling approaches to this problem in order to exploit the structure and solution properties following the methodology proposed, focusing on the stochastic dynamic programming (SDP) approach, where we first show a dynamic programming background before presenting the SDP formulation for VRPSD.

2.1. A review of Vehicle Routing Problem with Stochastic Demands

In recent years the literature related to the stochastic vehicle routing problem has grown du to its application to real life problems, as well as to the academic interest in studying the problem theoretically. Consequently, a number of solutions have been proposed to deal with these problems.

2.1.1. Application cases

There are many applications of the VRPSD. In a huge number of real situations, the customer demand is unknown and its probability distribution can only be estimated. Eventhough, in spite of the fact that the demand can be known in many cases, when the vehicle arrives to the customer, the demand value changes, e.g., delivering petroleum products or industrial gases (Chepuri and Homem-De-Mello [**8**]). To ilustrate these cases, we focus on the problem where gas stations are placed geographically dispersed and a fuel transport vehicle is entrusted to deliver a fuel quantity determined when the vehicle arrives at the customer location; the demand is unknown since despite the remaining fuel was known at the time to make the order, when the vehicle arrives after this time, the fuel stock has decreased.

In the case of Automatic Teller Machines (ATM), not only the daily demand for cash is uncertain, but the maximal amount of cash that may be picked up by trucks for money transportation is also limited for security and insurance reasons.

The Traveling Repairman Problem (TRP)

This problem was introduced by Bertsimas and Van Ryzin [**6**] in 1991; they analyzed the mathematical model for dynamic and stochastic VRP with the dynamic TRP. In that model, the objective is to minimize the average time demands in the system.

That problem is presented when a machine breaks down and must be repaired, in order to achieve such objective, it is important to know the distance between the starting point and the arrival point, the urgency of the situation (hight or low), the availability of the repairman, etc. A real example of that problem is on electric power company, which has to travel point by point to fix the problems that can suddenly arise; the problem has the characteristics of increased costumer, and the time spent in fixing the problems variates according to the particular environment.

Many authors have studied the problem; Jothi and Raghavachari [**22**] analyze two algorithms to solve the problem; in 1993 Das and Wortman [**31**] studied the TRP with a single repairman, and they proposed a probability model that is useful for evaluating the system performance measures, such as the inactivity time, the availability of a machine and of the repairman.

Currier mail services

Nowadays, there are many companies that offer the service of pick-up packages or mail and delivery of goods to the point where the costumer indicates; this problem, becomes dynamic because the driver unknow where will the next pick-up or quantity or volume of goods take place.

In 2004 Timon *et.al.* [**32**], investigated the DVRP (Dynamic VRP) applied to electronic commerce (e-commerce), they say that the environment of the e-commerce has voluminous, unpredictable, and dynamically changing customer orders. They studied the problem Business to Costumer (B2C) in an e-commerce environment, and the purpose was a solution in three phases: initial-routes formation, inter-routes improvement, and intra-routes improvement.

Alan Slater [**30**] proposes a routing and scheduling method for the e-commerce environment, which allows the costumers to select their own delivery time windows. The methodology that he used is based in parallel tour-building and parallel insertion algorithms, and the confirmation to the costumers is realized using GPS tracking and tracing.

Emergency services

Some services that arise into our society are "emergency services". These occur when, for instance, a person is sick and needs and ambulance, a person is robed and needs the police, in the case when there is a fire and a person has to call the firefighters. These are examples of services that can arise suddenly in any moment; in such case it is important to minimize the distance between the origin and arrival point, the availability of the resources (cars, tracks etc.), and the level of the urgency.

Taxi cab services

The taxi service is another application of the DVRP. It is difficult to plan the taxi route before the taxi leave the central, since the costumers can suddenly need the service; such problem of assigning a taxi route is more complex depending on the number of costumers, the positions of the taxi, the time and the arrival point.

Other VRPSD arise in delivering the post to large customers [**21**], vending machines [**34**] or delivering medical supplies in response to a large-scale emergency [**12**]. It is also possible to view the features of the problem in recycling and waste management, among others.

2.1.2. Solution methods

There are two different ways for dealing with VRPSD, either with fixed routes or a dynamic approach often called reoptimization. The methodology used to solve it depends on the type of model used to represent the problem; in section 2.2 we point out some main formulations used to address the problem.

Some solution methods find the optimal solution, but however, the use of these techniques is limited to small size problems. Therefore, other methods have been proposed to aproximate optimal solutions.

2.1.2.1. *Exact methods*

The relevant exact methods are *branch-and-bound* algorithms where the problem can be formulated as a lineal model. Laporte *et.al.* [**23**] proposed an *L-shape method* for solve VRPSD, a *branch-and-cut* algorithm.

Bernard *et.al.* [**9**], present a Dantzing and Wolf decomposition to resolve a dynamic model analyzing multi-periodic vehicles fleet size and modeling the fleet size of each one. At the end, the authors obtained optimal solutions for different demand distribution.

In 2007 Christian H *et.al.* [**10**], presented a Branch and price algorithm for the capacitated vehicle routing problem with stochastic demands; they used dynamic programming to solve a subproblem.

2.1.2.2. *Aproximate methods*

The cyclic heuristic was proposed by Bertsimas [**5**], it is a simple and inexpensive algorithm that guarantee to reach the final state. We explain it in the section 3.2.1.1 where in our methodology is used. Gans and Ryzin [**17**] have studied the dynamic vehicle dispatching systems where the congestion is the main measure of performance. They used a lower and upper bound; based in a simple batching heuristic, they found stability conditions for the optimal work in heavy traffic.

Yang *et.al.* [**34**] tested two heuristic algorithms which solve the problem in two stages: the *route-first-cluster-next* and *cluster-first-route-next*; they cluster the customers first and find the best route for each cluster after that. Furthermore, the cross-entropy heuristic method is proposed by Chepuri and Homem-de-Mello [**8**], to estimate the expected distance they used monte carlo sampling.

Local search heuristics commonly used for TSP have been used to solve VRPSD as OrOpt ([**34**], [**7**]). Yang et. al. used the 3-opt local search algorithm as well.

Moretti *et.al.* [**25**] implemented an algorithm to solve the DVRP. They consider the demand and the location as stochastic variables, and the time windows as soft. Their objective is to define a set of routes that are dynamically updated, taking into account the new costumers. To solve they used a constructive algorithm with an adaptive tabu search framework.

A genetic algorithm is another option used to solve this kind of problems, Haghani and Jung [**20**] presented a formulation for the DVRP with pick up and delivery and soft time windows, multiple vehicles with different capacities, variant real time service and and travel times. They proposed a genetic algorithm and their results were compared with exact methods.

2.1.2.3. *Dynamic programming*

The dynamic programming is one of the techniques most used to solve this kind of problems. Bertsekas [**2**] proposed efficient methods such as Markov decision analysis, linear control model with quadratic cost, policies in the stochastic inventory problem, and so on. Furthermore, Secomandi [**29**] compared neuro-dynamic programming algorithms for VRPSD.

Neuro Dynamic programming has been designed to deal with dynamic programming problems where the number of states is too large or is completely unknowed [**?**], Many authors have followed this approach applying approximate policy iteration methods to deal with VRPSD. Bertsekas [**4**] presents the *rollout algorithm* (RA) applied to combinatorial optimization problems; later, Secomandi ([**28**], [**27**]) showed how to apply it to VRPSD. Novoa and Storer [**26**] proposed a solution for the VRPSD using dynamic programing algorithms; they also considered the cost-of-go with the help of Monte Carlo simulation, which showed that in that kind of problems the best method found is the one-step roll-out that started with a stochastic base sequence. In addition, (jC Goodson, 2013) proposed rollout policies for dynamic solutions to the multivehicle routing problem with stochastic demands and constrained times.

2.1.2.4. *Hybrid methods*

Bianchi *et.al.* [**7**] implemented a simulated annealing, tabu search, iterated local search, ant colony optimization and evolutionary algorithms, and combined these with 2 local search tecniques: OrOpt and 3-opt. The second local search has been used with good results in TSP.

Mendoza *et.al.* [**24**] proposed a memetic algorithm for the multi-comparment vehicle routing problem with stochastic demands (MC-VRPSD). It is modeled as stochastic programming with resource and under each iteration of the genetic algorithm, a 2-opt local search is performed. The results are compared with the deterministic version of the problem.

2.2. Formulation of VRPSD

Given a graph $G(V, E)$, where $V = \{0, 1, 2, \ldots, n\}$ and the node 0 denotes the starting point for vehicles (depot), and the remaining nodes represent individual customers. The set E of edges in the graph represents the roads or paths between a pair of customers (i, j) and d_{ij} is the distance between them, which is assumed to be known, symmetric and satisfies the triangle inequality.

A vehicle (only one) with fixed capacity $Q < \infty$ starts from the depot and executes deliveries (or pickups only) of a product to different customers, D_i denotes the random variable representing the demand of customer i, and the probability distribution D_i is discrete and known and is denoted by $p_i(k) = Pr\{D_i = k\}, k = 0, 1, \ldots, K \leq R$. The customer demands are assumed to be independent and their exact value is known only when the vehicle arrives to the customer location. If a customer demand exceeds the available capacity of the vehicle, i.e. a *route failure*, the vehicle must return to the

depot to restore its original capacity. Hence, the depot must have a capacity at least equal to nR.

Yang, *et.al.* [**34**] propose a simple resource action for early replenishment, where the vehicle come back to the depot even when it has not depleted its stock, in order to restore the capacity to Q, allowing proactive depot trips to avoid route failures. Hence, considering proactive restocking of the vehicle is not necesary to consider multiple routes, in fact, Yang, *et.al.* [**34**] point out that a single route is more efficient than multiple vehicle route system, assuming that only distance constrain the route, ommitting for example time duration.

The objective is to minimize the expected distance by finding out a routing solution, probably in the form of routing rules, so that demand of each customer is satisfied. Thus, VRPSDs are usually modeled as mixed or pure integer stochastic programs, or as Markov decision processes.

2.2.1. Stochastic programming

The stochastic programming goal is to find an optimal decision in problems that involve uncertainty in the data. Stochastic VRPs can be cast within the frame-work of stochastic programming [**19**]. Stochastic programs are modeled in two stages. In a first stage, an *a priori* solution is determined and the realizations of the random variables are then disclosed; in a second stage, a recourse or corrective action is then applied to the first stage solution. The recourse usually generates a cost or a saving that may have to be considered when designing the first stage solution. A stochastic program is usually modeled either as a Chance Constrained Program (CCP) or as a stochastic program with recourse (SPR).

2.2.1.1. *Chance-constrained programming*

In CCPs, one seeks a first stage solution for which the probability of failure is constrained to be below a certain threshold. A CCP solution does not take into account the cost of corrective actions in case of failure. Mainly, for a given customer demands parameters, e.g., means, variances. One subjectively especifies a control probability looking for avoid that a *route fail*. Following to Dror [**15**], a VRPSD is formulated as:

$$\text{minimize} \sum_{v} \sum_{i,j} d_{ij} x_{ij}^{v} \tag{2.2.1}$$

$$\text{subject to } Pr\{\sum_{i,j} D_i x_{ij}^{v} \leq Q\} \geq 1 - \alpha, \forall v = 1, \ldots, NV, \tag{2.2.2}$$

$$x = [x_{ij}^{v}] \in S_{NV} \tag{2.2.3}$$

where $x_i j^v$ is a binary decision variable that takes the value 1 if vehicle v travels directly from customer i to j and 0 otherwise, and S_{NV} is the set of feasible routes for the traveling salesman problem (TSP) with NV salesmen.

These models are based on the premise that stochastic optimization problems are transformable to deterministic problems controlling the probability of route failure events occurring. Nevertheless, this artificial control might result in bad routing decisions.

2.2.1.2. *Stochastic programming with resources*

In SPRs, the aim is to determine a first stage solution that minimizes the expected cost of the second stage solution: this is made up of the cost of the first stage solution, plus the expected net recourse cost. SPRs are typically more difficult to solve than CCPs, but their objective function is more meaningful [**19**].

Yang [**34**] propose two heuristic methods to solve the problem and Laporte *et.al.* [**23**] and Gendrau *et.al.* [**18**] propose an *L-shape* method to find optimal solutions, i.e. a branch-and-cut algorithm adjusted for the stochastic approach. Below, we present the model formulated by [**15**] based on the Laporte model ,although allowing proactive replenishments of the vehicle in a single route, supported on Yang's affirmation, this is more efficient than multiple routes.

Let $T(\hat{x}, D) = \sum_i^n \sum_j^n d_{ij} x_{ij}$ be the cost of the routing solution where $\hat{x} = \{x_{ij}\}, i \in V, j \in V$ is the vector of routing decisions, $\hat{x}_{ij} = 1$ if the vehicle directly visits the node i from j node and 0 otherwise; D is a vector of the customer demands disclosed one a time when the vehicle arrive at customer location. Both $\hat{x}$ and D are random variables.

$$\min_{\hat{x}} E_D[T(\hat{x}, D)] \tag{2.2.4}$$

$$\text{subject to} \sum_{i=0}^{n} \hat{x}_{ij} \geq 1, \forall i \in V \tag{2.2.5}$$

$$\sum_{j=0}^{n} \hat{x}_{ij} \geq 1, \forall j \in V \tag{2.2.6}$$

$$\sum_{i \in S} \sum_{j \notin S} \hat{x}_{ij} \geq 1, S \subseteq 1, \ldots, n; |S| \geq 2 \tag{2.2.7}$$

$$\hat{x}_{ij} \in 0, 1, i, j \forall i, j \in V \tag{2.2.8}$$

$T(\hat{x}, D)$ can be divided in two parts; in the first part we have the term cx denoting the cost (distance) of the *a priori* sequence represented by x, and in the second part $\mathcal{Q}(x, D)$, would be the recourse cost given x and a realization of D. $\mathcal{Q}(x, D)$ represents the cost of return trips incurred by route failures, minus some resulting savings. $T(\hat{x}, D) = cx + \mathcal{Q}(x, D)$ where x represents a TSP route and $\hat{x}$ is the binary routing vector which includes all the resource decisions. In order to keep $\hat{x}$ as binary, it is assumed that the probability of a node demand greater than capacity of an vehicle, is zero, as well as the probability that a vehicle, upon failure, returning to a node to complete its delivery after visiting the depot, is also zero.

Setting the expectation $\mathcal{Q}(x) = E_D[\mathcal{Q}(x, D)]$ the objective function becomes:

$$\min_{x} cx + \mathcal{Q}(x) \min_{x} cx + E_D[\mathcal{Q}(x, D)] \tag{2.2.9}$$

In the standard modeling of the Two-Stage Stochastic Linear Programs, customers deliveries in the secon stage are represented as follows:

$$\mathcal{Q}(x, D) = \min_{y} \{cy | Wy = h(D) - T(D)x, y \in Y\} \tag{2.2.10}$$

Where y is the binary vector representing the recourse initiated trips to the depot, $T(D)$ represents the deliveries made by the x vector given that D and $h(D)$ is the demand realization for D which has to be met (delivered) either by x or the resource y. Below, we show the model used by **[23]** for the L-shape method:

$$\min_x \{cx + \mathcal{Q}(x)\} \tag{2.2.11}$$

$$\text{subject to} \sum_{i=0}^{n} x_{ij} = 1, \forall i \in V \tag{2.2.12}$$

$$\sum_{j=0}^{n} x_{ij} = 1, \forall j \in V \tag{2.2.13}$$

$$\sum_{i \in S} \sum_{j \notin S} x_{ij} \geq 1, S \subseteq 1, \ldots, n; |S| \geq 2 \tag{2.2.14}$$

$$\hat{x}_{ij} \in 0, 1, i, j \forall i, j \in V \tag{2.2.15}$$

An extended review of the models presented above, and others for VRPSD is presented by **[15]** and **[14]**

2.2.2. Stochastic Dynamic Programming

Stochastic Dynamic Programming (SDP) provides a frame-work where decisions are made in an finite number of stages under uncertainty. In these problems there is a set of states S, decisions variables U and the uncertainty is represented by a set of random variables W; the dynamic system is of the form:

$$f : U \times W \times S \to S$$

or

$$x_{k+1} = f_k(x_k, u_k, w_k), k = 0, 1, \ldots, N-1 \tag{2.2.16}$$

where $x_k \in S_k$ is the state of system at $k-th$ time and summarizes past information, u_k is the control or decision variable to be selected in a given nonempty subset $U(x_k) \subset U$ which depends on the current state x_k, i.e. $u_k \in U_k(x_k) \forall x_k \in S_k$, $w_k \in W$ is a random parameter whose value is disclosed at time k and is characterized by a probability distribution $P_k(\cdot|x_k, u_k)$ that may depend on x_k and u_k but not on prior values of the random variable $w_{k-1}, \ldots, w_0$. N is the horizon or number of times that control is applied

The objective is to minimize a cost function $g_k(x_k, u_k, w_k)$ of the form

$$g_k : S \times U \times W \to \mathbb{R}$$

The cost function is assumed additive, i.e. the cost incurred accumulates over time.

$$g_N(x_N) + \sum_{k=0}^{N-1} g_k(x_k, u_k, w_k)$$

Nevertheless, given the cost as a random variable, we formulate the problem as an optimization of the expected cost:

$$E\left\{ g_N(x_N) + \sum_{k=0}^{N-1} g_k(x_k, u_k, w_k) \right\} \tag{2.2.17}$$

We define *admissible policies* $\pi \in \Pi$, where Π is the set of all admissible policies, as a sequence of functions $\mu_k : x_k \to u_k$, in which μ_k maps state x_k to controls $u_k = \mu_k(x_k)$ such that $\mu_k(x_k) \in U_k(x_k) \forall x_k \in S_k$.

$$\pi = \mu_0, \ldots, \mu_{N-1}$$

Given a policy π and a initial state x_0, the equation 2.2.16 is rearranged as:

$$x_{k+1} = f_k(x_k, \mu_k(x_k), w_k), k = 0, 1, \ldots, N-1 \tag{2.2.18}$$

making x_k and w_k random variables with probability distributions defined. Hence, the expected cost function g_k 2.2.17, with $k = 0, 1, \ldots, N$ is well defined:

$$J_\pi(x_0) = E_{w_k}\left\{ g_N(x_N) + \sum_{k=0}^{N-1} g_k(x_k, \mu_k(x_k), w_k) \right\} \tag{2.2.19}$$

An optima policy π^* for a given initial state x_0 is one that minimizes the cost $J(x_0)$, i.e

$$J_{\pi^*}(x_0) = J^*(x_0) = \min_{\pi \in \Pi} J_\pi(x_o)$$

The SDP model applied to VRPSD is formulated below in the section 2.2.3

2.2.2.1. *Finite-Stage Models*

Suppose that the states are the integers, and let A, a finite set, be the set of all possible actions. $R(i, a)$ is the reward in ith-state given that $a \in A$ action was chosen, and the next state is j with probability $P_ij(a)$. Let $V_N(i)$ denote the maximum expected return for an N-stage problem that starts in state i.

When $N = 1$ we have:

$$V_1(i) = \max_{a \in A} R(i, a) \tag{2.2.20}$$

Considering a N-stage problem that starts in i and has $N-1$ time periods to go. We can assess the expected return given initially we choose action a:

$$R(i,a) + \sum_j P_{ij}(a) V_{N-1}(j)$$

then,

$$V_N(i) = \max_a [R(i,a) \sum_j P_{ij}(a) V_{N-1}(j)] \tag{2.2.21}$$

2.2.3. Stochastic Dynamic Programming approach for VRPSD

VRPSD is modeled in the framework of SDP as a stochastic shortest path problem; it is a Markov decision process (MDP) where is necesary to make decisions under situations where outcomes are partly random reaching an absorbing cost-free termination state in a random number of stages.

The problem formulation is presented below, based in the Novoa [**26**] and Secomandi [**27**] notation as a Markov decision model.

The objective of the problem is to find a routing policy so customer's demand is satisfied and the expected transportation costs (distance) minimized, this policy may order returns to the depot before the vehicle capacity runs out.

2.2.3.1. *Types of policies*

Secomandi [**29**] classifies the routing policies in three groups, static, dynamics and mixed

Static: Static policies describe a sequence τ of customers to be visited in that order for the vehicle.

Dynamic: Dynamic policies provide a policy π that given the current state of the system, prescribe which location should be visited next.

Mixed: Mixed policies combine elements of both static and dynamics policies.

Mixed policies not only follow a sequence τ of customers but also prescribe decisions that dependend on the state that allows proactive replenishments. In the figure 2.2.1, we ilustrate static policy (left) where reactive replenishment or resource action are carried out when the customer demand is greater than the vehicle capacity, and is therefore forced to return to the depot for restocking, while on the right image, we have a dynamic policy in which the vehicle can do proactive replenishments, going to the station even when the vehicle capacity is not empty.

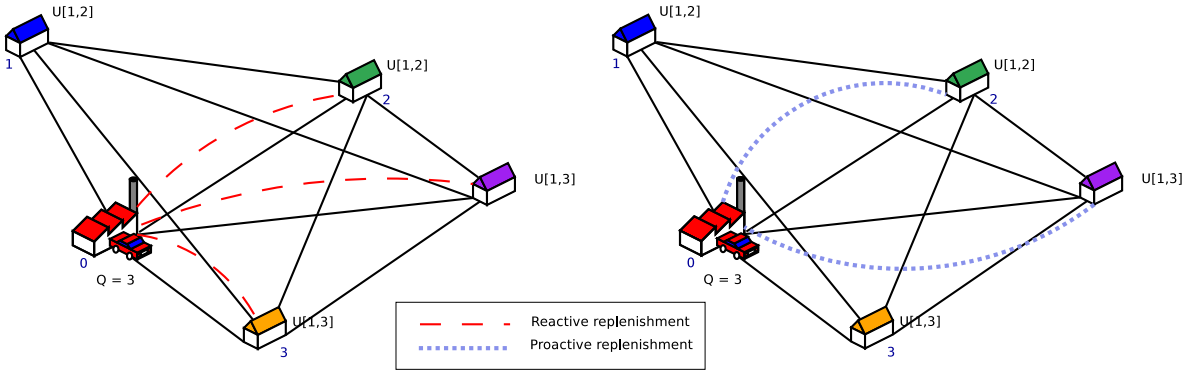


FIGURE 2.2.1. static and mixed routing policies

In order to represent the system state at stage k, the vector x_k is defined as $x_k = (l, q_l, r_1, \ldots, r_n)$ of size $n+2$, where $l \in \{0, 1, \ldots, n\}$, is the current location of the vehicle and $q_l \leq Q$ is its available capacity *after* delivery to customer l; the elements r_i represents the remaining demand to satisfy to the costumer i. An unknown demand is denoted as -, if customer i is visited and its demand has been completely satisfied, r_i will take the value 0; otherwise, it will take any value between 1 and R. The initial state of the system x_0 is $(0, Q, -, -, \ldots, -)$ and the final state x_N occurs when the vehicle returns to the depot after serving the demands of customers, represented as $(0, Q, 0, 0, \ldots, 0)$. Thus, the number of states in the system is $O(nQR^n)$

Let N be a random variable representing the number of stages or transitions from initial state to the end, the vector $\pi = \mu_0, \mu_1, \ldots, \mu_{N-1}$ is the policy or sequence of functions to optimize, where μ_k is a function that associates a decision or control $u_k = \mu_k(x_k)$ for each state, $u_k \in U_k(x_k)$ and $U_k(x_k) = \{\{m \in \{1, \ldots, n\}\} | r_m \neq 0\} \cup 0\} \times \{a : a \in \{0, 1\}\}$. Control u_k is represented as ordered pairs (m, a), m is any costumer not yet served, m is 0 when all demands have been satisfied and the system enters its completion stage, a is 0 if the vehicle directly visits customers and 1 if the vehicle first stops at the depot to resupply.

Given a state $x_k = (l, q_l, r_1, \ldots, r_m, \ldots, r_n)$ and a control u_k in which it is decided to visit the node m at the next stage, the random variable D_m is realized ($r_m = D_m$ if r_m is unknown; otherwise $r_m \neq ?$) and the remaining demand of the customer m changes to r'_m as soon as the capacity of vehicle becomes q_m, where

$$q_m = \begin{cases} max(0, q_l - r_m), & \text{whether } u_k(m, 0) = \mu_k(x_k) \\ q_l + Q - r_m, & \text{whether } u_k(m, 1) = \mu_k(x_k) \end{cases} \tag{2.2.22}$$

and

$$r'_m = \begin{cases} min(0, r_m - q_l), & \text{whether } u_k(m,0) = \mu_k(x_k) \\ 0, & \text{whether } u_k(m,1) = \mu_k(x_k) \end{cases} \tag{2.2.23}$$

so the system goes to state $x_{k+1} = (m, q_m, r_1, \ldots, r'_m, \ldots, r_n)$. The transition between states is graphically represented as:

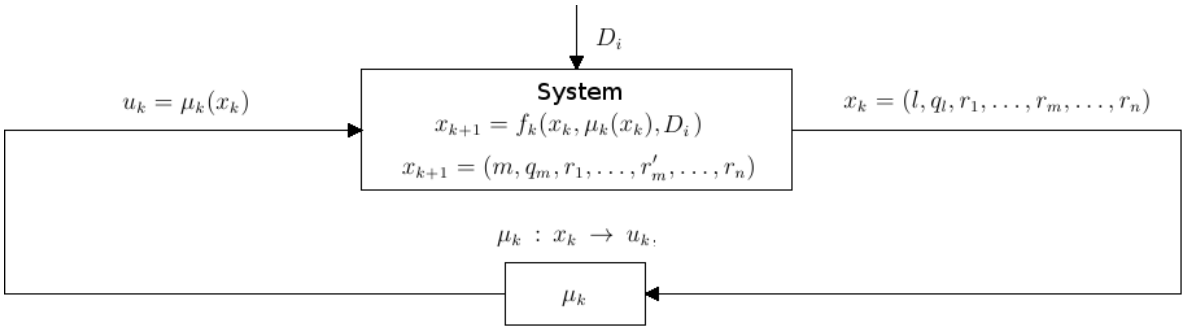


FIGURE 2.2.2. Stochastic Dynamic System for VRPSD

Incurring in a transition cost $g(x_k, u_k, x_{k+1})$

$$g(x_k, \mu_k(x_k), x_{k+1}) = \begin{cases} d(l,m), & \text{whether } u_k(m,0) = \mu_k(x_k) \\ d(l,0) + d(0,m), & \text{whether } u_k(m,1) = \mu_k(x_k) \end{cases} \tag{2.2.24}$$

The objective of the problem is to find a policy π that minimizes the cost of transport J_N^π (2.2.25) in the N-stages or the expected cost to complete a given initial state. The optimal cost of transport in the N-stage x is $J_N^*(x) = min_{\pi \in \Pi} J_N^\pi(x)$, where Π is the set of admissible policies.

$$J_N^\pi(x_0) = E\left\{ \sum_{k=0}^{N-1} g(x_k, \mu_k(x_k), x_{k+1}) \right\} \tag{2.2.25}$$

If $J_N^*(x)$ is known for all stages, the optimal control u_k^* at each stage is to find the minimum of the following equation (2.2.26):

$$u_k^* = \mu_k^*(x) = \arg \min_{u_k \in U_k(x_k)} g(x_k, u_k, x_{k+1}) + \sum_{x_{k+1} \in S} p_{x_k x_{k+1}}(u_k) J_N^*(x_{k+1}) | x_k = x, \forall x \in S \tag{2.2.26}$$

The problem is that $J_N^*(x)$ is unknown and its calculation is a computationally intractable problem given the size of state space. Secomandi [**27**] points out that computing an optimal policy becomes quickly intractable when n grows beyond 10. Chapter 3 deals with issue of approximating this function through a dynamic-programming method efficiently computable.

2.3. Summary

The VRPSD has been studied for more than 20 years, with important progress in 90's and 00's and wide areas of application in logistics, following the conclusions of [**15**] the most promising approach is modeling the problem as a Markov decision process. Hence, a stochastic programming model is selected: a methodology for sequential decisions made under uncertainty, based on dynamic system, where the main idea is to use an approximate a function J in order to make decisions in complex dynamic systems, allowing to deal with instances considered intractable for their size. In the next section, the dynamic programming solution is addressed.

CHAPTER 3

Stochastic Dynamic programming solution

3.1. Dynamic approach for VRPSD

Dynamic programming is based on principle of optimality formuled by Bellman [**1**]

> *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.*

Following to Bertsekas [**3**] the principle of optimality point out that an optimal policy can be constructed backward, first finding an optimal policy for the *tail subproblem* involving the last stage, then extending the optimal policy to the problem regard with the last two stages, and continuing until cover the whole problem in the first stage, hence a optimal policy is constructed for the entire problem.

$$J^*(x_0) = \min_{u_k^* \in U_k(x_k)} E_{w_k}\left\{ g_k(x_k, u_k^*, w_k) + J_{k+1}^*(f(x_k, u_k^*, w_k)) \right\}, \quad \forall k = 0, 1, \ldots, N-1 \quad (3.1.1)$$

However, the exact assesing of J^* is computationally infeasible given the size of the state space. Therefore, is necessary approximate it function for generate good, but not necessarily optimal policies.

Let $\tilde{J}_k$ be an approximation of J_k^*, then a control can $\tilde{u}_k$ be assesed

$$\tilde{u}_k = \tilde{\mu}_k(x_k) = \arg\min_{u \in U_k(x_k)} \left\{ g(x_k, u_k, x_{k+1}) + \sum_{x_{k+1} \in S} p_{x_k x_{k+1}} \right\} \quad (3.1.2)$$

The following section 3.1.1 discuss the computation of $\tilde{J}_k$

3.1.1. Expected distance

The expected distance $\tilde{J}$ or *cost-to-go* is computed based on the algorithm propossed by Secomandi [**27**]. We implemented the algorithm Γ represented below 1 to compute expected distance in $O(nRQ)$ time and $O(nQ)$ space.

input : tour $\tau_{1\times n+2}$, $d_{n+1\times n+1}$ distance, x state
output: E expected distance of an apriori solution τ (base sequence)
$l = x_1$;
$q_l = x_2$;
if $l = n$ **then** is the last customer on τ
  $E = d(\tau(l+1)+1, 1)$;
end
else
  if *l=0* **then** is the first node *depot* in the tour τ
    E = $\Gamma(\tau, l+1, q_l)$;
  end
  else
    $E_0 = d(\tau(l+1), \tau(l)) + \sum_{j=0}^{\min\{q_l, \bar{D}_{\tau(l+1)}\}} \Gamma(\tau, l+1, q_l - j) *$
    $p^{\tau(l+1)}(j) + \sum_{j=q_l+1}^{\bar{D}_{\tau(l+1)}} 2 * d(0, \tau(l+1)) + \Gamma(\tau, l+1, Q + q_l - j)) * p^{\tau(l+1)}(j)$;
    $E_1 = d(0, \tau(l)) + d(0, \tau(l+1)) + \sum_{j=0}^{\bar{D}(\tau(l+1))} \Gamma(\tau, l+1, Q - j) * p^{\tau(l+1)}(j)$;
    $E = \min\{E_0, E_1\}$;
  end
end

Algorithm 1: Expected distance algorithm $E = \Gamma(\tau, l, q_i)$

This algorithm was validated comparing the outcomes with the expected distance computed by an exhaustive algorithm applied to small instances. In addition, larger instances was benchmarked using monte carlo simulation to asses expected distance.

If the vehicle capacity is depleted, the methods used to compute the expected distance take into account that the vehicle can go to the depot for proactive restocking with less cost than to visit the customer first which makes the route fail, as we show in 3.1.1.

THEOREM 3.1.1. *if $q_l = 0$ then the vehicle must go first to the depot for replenishment and later visit the next customer on the route, it is better than it visit the next customer to know its demand and go to the depot for replanishment after.*

PROOF. Given the current state $x_k = (l, 0, r_1, \ldots, r_n)$ where the vehicle capacity is depleted, i.e. $q_l = 0$, assume that the next customer to be visited on the route is l'. If the customer is visited first, then the vehicle must go to depot for replenishment and go back to l' to satisfy its demand, so the distance is $\delta(l, l') + 2\delta(l', 0)$. Otherwise, if a proactive restocking of the vehicle is performed then the total distance is $\delta(l, 0) + \delta(0, l')$. Assume $\delta(l, 0) + \delta(0, l') \leq \delta(l, l') + 2\delta(l', 0)$ then by triangular inequality $\delta(l, 0) \leq \delta(l, l') + \delta(l', 0)$ proving 3.1.1 □

LEMMA 3.1.2. *The theorem 3.1.1 can be generalized for all* $q_l \geq D'_i$ *where* D'_i *is the demand of the customer* i *who is the next on the tour.*

3.2. Policy iteration

The policy iteration algorithm is a dynamic programming technnique, it generates a sequence of stationary policies, each with improved cost over the preceding one. The algorithm describe the following sequence of steps 2:

1. Initialization: Guess an initial stationary policy π_0
2. Policy evaluation: Given the stationary policy π_k compute the corresponding cost function J_{π_k}
3. Policy improvement: Obtain a new stationary policy π_{k+1}
Repeat step 2 to 3

Algorithm 2: Policy iteration algorithm

If the policy evaluation step compute $J_{\pi_k}(x)$ for all states $x \in S$ and the algorithm runs until $J_{\pi_m} = J_{\pi_{m+1}}$, $\forall x \in S$ then the algorithm finds the optimal policy π^*. In contrast, when $|S|$ is too large this algorithm is inpractical; although these steps can be approximated to deal with large-scale systems despite that convergence to optimal solution is not guaranteed.

3.2.1. Rollout algorithm

The rollout algorithm is an approximate policy iteration technnique, used to to increase the effectiveness of a heuristic by iteratively applying it, or rolling it out, at each decision stage (Goodson, 2010). The algorithm require to know a base policy π for the problem. Also assume that the cost-to-go of this base policy from any given state x can be easily computed (Secomandi, 2000). Thus, it build a policy π starting from any given state and following an *a priori* solution to VRPSD.

3.2.1.1. *Defining initial policy*

Cyclic heuristic $\mathcal{C}$ shifts τ to obtain an permutation keeping an cyclic order. It builds the cyclic tour that start at l and follow τ cyclically, so, $\tau_l^{\mathcal{C}} = (l, l{+}1, \ldots, n, 1, \ldots, l{-}1, 0)$ represent an apriori policy $\pi^{\mathcal{C}}$. Cyclic heuristic was propossed by Bertsimas 92 and improved by himsel 95 and later used by a many authors because it is simple, inexpensive and sequentially consistent (Secomandi dissertation, 1998)

Following cyclic heuristic the rollout algorithm 3 describe a fixed number of iterations given an instance of the problem changing τ after the first state. Since there are a transition of x_l state to x_{l+1}, is only shifted the segment of τ that follow to l,

maintaining in this section the customers still no visited or with demand different to 0. Since that already fully served customers are assumed to be skipped in $\tau_l^{\mathcal{C}}$.

In order to compute rollout policy controls, the control which define the first customer l_1 to be visited by $\tilde{\pi}^{\mathcal{C}}$ is found as:

$$\mu_0(x_0) = u_0 = (l_1, 0) = \arg \min_{l \in V - \{0\}} \{\tilde{J}_{\pi_0}(x_0)\}$$

where the expected length $\tilde{J}_{\pi_0}(x_0)$ of π_0 which follow the cylcic tour $\tau_l^{\mathcal{C}} = (0, l, l+1, \ldots, 1, n, \ldots, l-1, 0)$, in the initial state x_0, is computed using the algorithm Γ 1. Hence, the system moves to next state visiting the customer l_1 directrly who is fully served since $q_0 = Q$ and $r_l \leq Q$.

input : π_0, state x_0
output: $\tilde{\pi}^{\mathcal{C}}$ policy
Given a initial policy $\pi_0 = u_1, u_2, \ldots u_{N-1}$ and initial state x_0
$\tilde{\pi}^{\mathcal{C}} = \emptyset$
repeat
  $\tilde{\mu_k} = \arg \min_{u_k \in U_k(x_k)} \{\min\{\tilde{J}^0_{\pi_k}(x_k), \tilde{J}^1_{\pi_k}(x_k)\}\}$
  Add $\tilde{\mu_k}$ to $\tilde{\pi}^{\mathcal{C}}$
  Apply the control $\tilde{\mu_k}$ to the state x_k.i.e. $x_{k+1} = f_k(x_k, \tilde{\mu_k}(x_k), D)$
  Roll out π_k since $\tilde{\mu_k}$ to μ_{N-1} following cyclic heuristic
until *the final state x_N is reached*;

Algorithm 3: Rollout algorithm

For some state $x_k = (k, q_k, r_1, \ldots, r_n)$, $\tilde{J}^0_{\pi_k}(x_k)$ compute the expected distance of move the vehicle directly to the next customer following the policy π_k thereafter.

$$\tilde{J}^0_{\pi_k}(x_k) = d(\tau_l, m) + \sum_{k=0}^{q_l} p_m(k)\Gamma(\tau, l+1, q_l - k) + \sum_{k=q_l+1}^{K_m} 2d(0, m)p_m(k)\Gamma(\tau, l+1, q_l + Q - k) \tag{3.2.1}$$

While $\tilde{J}^1_{\pi_k}(x_k)$ assess the expected distance performing an proactive replanishment before of move the vehicle to the next customer following the policy π_k thereafter.

$$\tilde{J}^1_{\pi_k}(x_k) = d(0, \tau_l) + d(0, m) + \sum_{k=0}^{K_m} p_m(k)\Gamma(\tau, l+1, Q-k) \tag{3.2.2}$$

Thus, the control $\tilde{\mu_k}$ which decides to visit the customer l is computed as follows depending on the smaller cost:

$$\tilde{\mu_k} = \begin{cases} (l,0), & \text{if } \tilde{J}^0_{\pi_k}(x_k) \leq \tilde{J}^1_{\pi_k}(x_k) \\ (l,1), & \text{in other case} \end{cases} \tag{3.2.3}$$

Rollout explore $\frac{1}{2}n(n+1)$ different policies with an computational cost of $O(nRQ)$ in order to evaluate the expected distance for each one, hence, the rollout algorithm runs in $O(n^3RQ)$ time.

3.3. Summary

Rollout algorithm is an approximate atochastic dynamic programming technnique implemented to improve a *a priori* policy. We use cyclic heuristic as base policy since it is simple, inexpensive and sequentially consistent. Finally. we describe the rollout algorithm and point out the computational complexity to perform it.

CHAPTER 4

Hybrid evolutionary approach

An evolutionary algorithm (GA) is a technique bio-inspired by the evolution of the espicies. It was proposed by John Holland early in the 70's (Holland, 1975). This algorithm seeks to evolve a population of individuals that represent solutions to the problem through genetic operators such as crossover, mutations and selection. The population generated in each iteration is evaluated and then the individuals with a better fitness are chosen for the next generation with more chance.

4.1. Hybrid evolutionary algorithm

A hybrid evolutionary algorithms or hybrid genetic algorithms are a very popular techniques that offers practical advantages to deal with complex and hardly optimization problems. Grosan (Grosan, 2007) present a review of hybrid genetic architectures frecuently used.

Hybridization can be performed using prior knowledge, heuristics, local search, other techniques. We use it to carry out local search through rollout algorithm. Some times, a hybrid genetic algorithm which combine other technique to local search is known as memetic algorithm.

Generally, the purpose of hibridization is:

- Improve the performance of the evolutionary algorithms.
- Improve quality of solutions obtained by evolutionary algorithm
- Incorporate evolutionary algorithm as part of a large system

Evolutionary algorithm behavior is determined by the exploitation and exploration; In exploitation local search is performed to improve solutions, and exploration avoid local optimum extending the search space, our implementation of memetic algorithm works to keep these relation throughout the run. So, in this application hybridazation not only improve the quality of the solutions obtained by evolutionary algorithm, also is assembly as an framework for rollout algorithm in order to avoid local optimum.

4.1.1. A basic genetic algorithm for vehicle routing problem with stochastic demands

We show an example of basic GA in general form in figure 4.1.1. First, an initial population is selected and fitness function is assessed for each individual in the population. In order to produce a new population for the next generation, crossover and mutation operators are applied to individuals allowing those who have better fitness function to get more chances to reproduce themselves. In the selection stage, offspring's fitness is evaluated to choose the individuals to integrate the new population; individuals with competitive fitness regarding to the population have more probability to be selected. These steps are repeated until stopping criteriums are resolve.

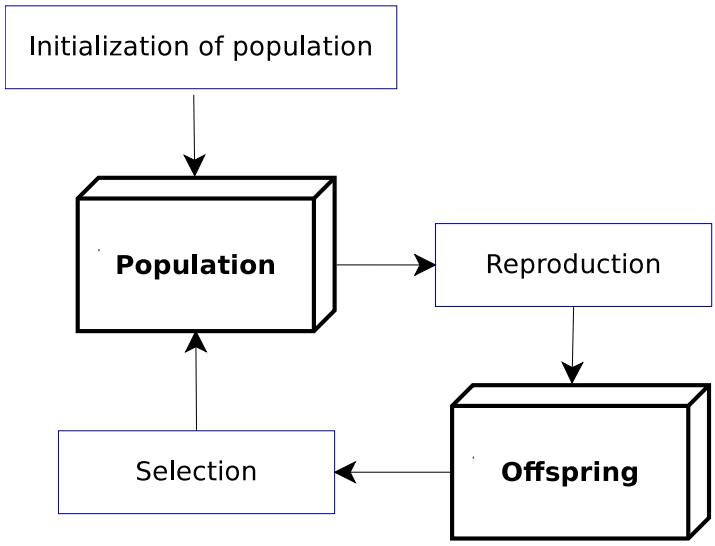


FIGURE 4.1.1. Basic genetic algorithm

Initialization

We represent an individual in the population as a policy tour $\pi^{\mathcal{C}}$ whose fitness is the expected distance $\tilde{J}_{\pi^{\mathcal{C}}}$ computed using the algorithm Γ 1.

The initial population P_0 with an fixed size $|P_0| = n$, is obtained using the cyclic heuristic in $O(n)$ time. It performs in order to reduce the computational cost since we can evaluate fitness to P_0 in $O(n)$ time when rollout is accomplished under some individual in the population.

Crossover

The crossover operator $\circledast$ consist in to select two different individuals $I_i^{P_k}$ and $I_j^{P_k}$ with probability dependently of their fitness value respectively; meanwhile, a random cut point $\rho \in [1, n]$ is selected uniformly. So, a new individual raise to concatenate the subsequence $I_i^{P_k}[1..\rho]$ with the subsequence $I_j^{P_k}[\rho + 1, .., n]$.

A crossover operation can yield an new individual that represents an unfeasible policy. However, we implement this operator to run in time $O(n^2)$ and guaraneeing only feasible solutions as a result.

Mutation

This operator performs three types of mutation with same probability to produce and individual in the offspring: swap two elements of policy selected randomly, flip a random subsequence of policy or shift the policy an random number of times.

The mutation of an individual can happened with probability $Prob_m = 0.04$ in the experiments presented in this document and it compute in $O(1)$ time.

Selection of the new population

New population originates as a result of crossover and mutation operators. However, the evolutionary algorithm also obtain individuals to the offspring performing the cyclic heuristic. Hence, those who have better fitness can be choosen with more chance to integrate the new population.

In order to punish generation which dicrease the fitness and reward to those who increase this value respect to the previos generation the size or number of individuals available to the next generation changes.

Let $\Delta_{E'}^{P_k}$ be the rate of fitness change in the generation P_k, i.e.

$$\Delta_{E'}^{P_k} = \frac{\bar{\Gamma}_{P_{k-1}} - \bar{\Gamma}_{P_k}}{\bar{\Gamma}_{P_{k-1}}} \tag{4.1.1}$$

where $\bar{\Gamma}_{P_k}$ is the best expected distance obtained by an individual in the generation k.

Then, the size of the next population P_{k+1} is computed so:

$$||P_{k+1}|| = \begin{cases} \lfloor \min\{n(1+\alpha), ||P_k||(1+\alpha)\} \rfloor, & \text{if } \Delta_{E'}^{P_k} > 0 \\ \lceil \max\{n\alpha, ||P_k||\alpha\} \rceil, & \text{if } \Delta_{E'}^{P_k} < 0 ||P_k||, \text{in other case.} \end{cases} \tag{4.1.2}$$

where α is a tuneable parameter which should be fixed in the range $(0, 1]$ depending of computational resources. Consequently it reward a offspring that improve the quality of the solutions in comparison to theirs parents, increasing the population size at most an α times the size. Otherwise, when the quality of the solutions decrease respect to the previos generation, the size of population is punished decreasing it at least an α factor of the size of population. The figure **??** shows results of basic genetic algorithm applied to one instance with $\alpha = 0.5$, the last chart below on the right side of figure exhibit the size of population for each generation.

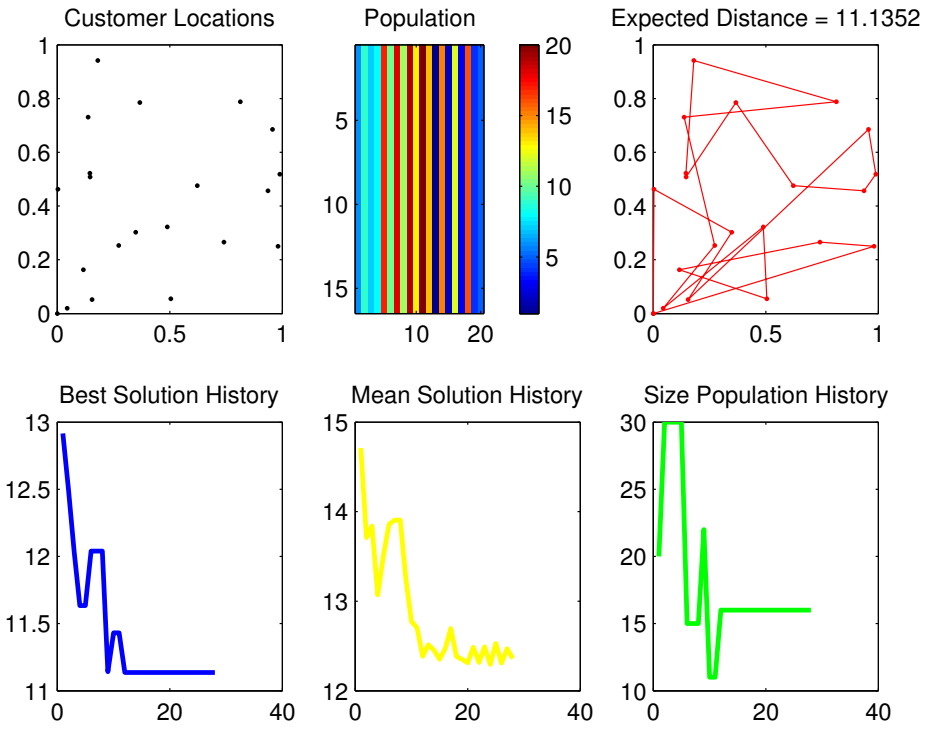


FIGURE 4.1.2. Basic genetic algorithm applied to instance of 20 customers. The second image in the first row ilustrate the last population and the next image shows the best solution finded.

Stopping criterion

Finally, the evolutionary algorithm runs until reach a fixed number or iterations κ. Nevertheless, it can stop once accopmlish $\mathbf{m}$ number of consecutive iterations whitous a significative change, i.e., $|\Delta_{E'}^{P_k}| \leq \varepsilon$. Both $\mathbf{m}$ and ε are tuneable parameters.

local search

A classic genetic algorithm does not yield competitive results itself, due to basic GA does not exploit problem knowing to produce high quality solutions. To be effective, we combine local search methods. Local search can be incorporated in the initial population or among the offspring.

We apply rollout algorithm as local search method. The GA incorporates it in the initialization stage as in each iteration under the best policy obtained. Figure 4.1.3 shows the evolutionary algorithm with local search applied to the same instance used by the basic genetic algorithm above and showed in the figure 4.1.2

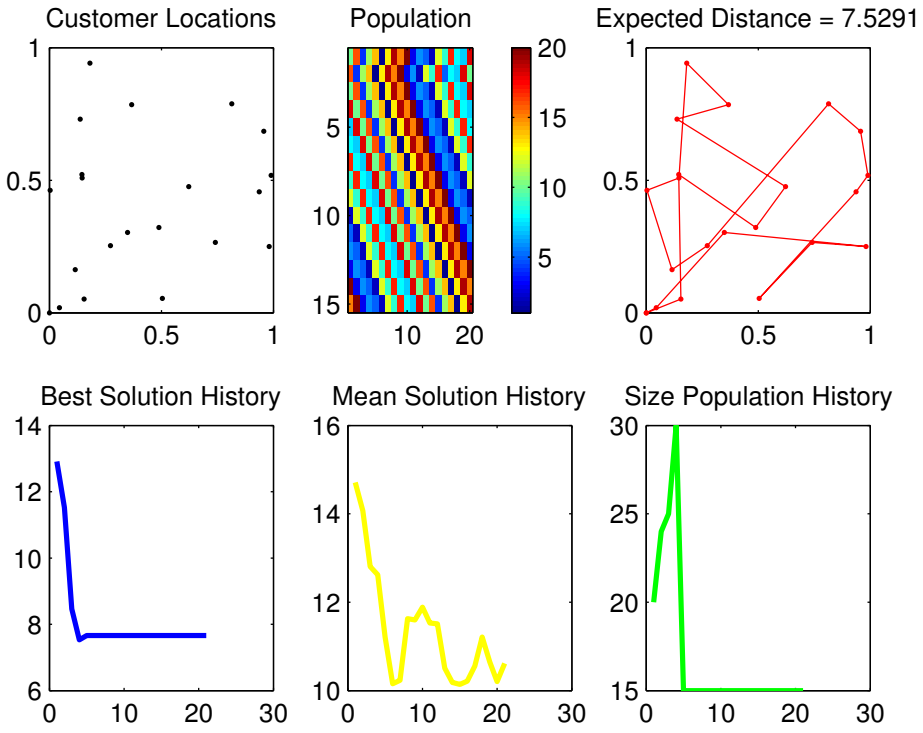


FIGURE 4.1.3. Memetic algorithm applied to instance of 20 customers. The second image in the first row ilustrate the last population and the next image shows the best solution finded.

4.2. Summary

In this chapter we presented a basic genetic algorithm for vehicle routing problem with stochastic demands. In addition, we showed an hybrid approach integrating the rollout algorithm as local search method in the evolutionary algorithm.

CHAPTER 5

Experiments and numerical results

In this chapter we present the computational results obtained for solving the instances selected.

5.1. Instances of VRPSD

In order to evaluate the algorithms proposed, we select instances of two sources generated following a procedure similar to the used in Secomandi [**29**]. The first set of instances is the used by Novoa [**26**], while the second was created by us randomly.

5.1.1. Instance generation

The set of instances contain 45 different instances resulting from combine three number of customers $n \in \{5, 10, 20\}$, three vehicle capacities given for f' factor and five different assignments for customer locations and demand distribution for each one, the assignments result from changing the random seeds.

The customers' demands are both discrete and distributed uniformly in this posible sets $U[1, 5]$, $U[3, 9]$, $U[6, 12]$, in each instance, each customer is assigned to any of the three groups with equal probability. Customers' locations are random points in $[0, 1]^2$, with the depot fixed at $(0, 0)$.

The filling rate f is an index of the total expected demand relative to vehicle capacity.

$$f = \sum_{i=1}^{n} \frac{E[D_i]}{mQ} \tag{5.1.1}$$

where $E[D_i]$ is the expected demand of customer i, m is the number of available vehicles, when $m = 1$, f can represent approximately the expected number of replenishment needed to serve all customers demands. It follows that, a priori, in all instances $E[D_i] = (3 + 6 + 9)/3 = 6$, for any customer i, and $Q = 6n/f$. $f' = f - 1$ is the expected number of route failure in a given instance. Hence, we is defined for this factor $f' \in \{1.0, 1.5, 2.0\}$, following the same factors used by Secomandi [**29**].

TABLE 5.1.1. Vehicle capacity for each factor

f'	n		
	5	10	20
1.0	15	30	45
1.5	12	24	36
2.0	10	20	30

The 160 instances used by Novoa [**26**] was generated following the same method described above. This set is composed by 70 small size instances (5 to 20 vertex), 60 medium size (30 to 60 vertex) and 30 instances of large size whose number of vertex is greter than 100. Table 5.1.2 shows the range of demands for each size instance, where the demand values is the difference between the maximun and minimun demand which some customer can take. In addition, 5.1.2 exhibit the demands mean and variance for each instance and in the figure 5.1.1 we classify each instance according to number of vertices n and demand range which is computed such as demand values in th table 5.1.2.

	demand values									
n	4	5	7	8	9	15	17	29	33	Total
5	2	4	1	1	8	2	3			21
8		4	2		8		5			19
20					5	5	10	5	5	30
30						5	5	5	5	20
40						5	5	5	5	20
60						5	5	5	5	20
100						5	5	5	5	20
150						5	5			10

TABLE 5.1.2. Instances characterization

instances with

5.1.2. Expected distance algorithm

In the figure A.0.3, we show the time consumed by algorithm Γ 1 to compute the expected distance for each instance. As expected, An higher value of n and Q increase the computational cost to compute expected distance. However, the figure A.0.3 shows that the demands distribution affect the algorithm performance.

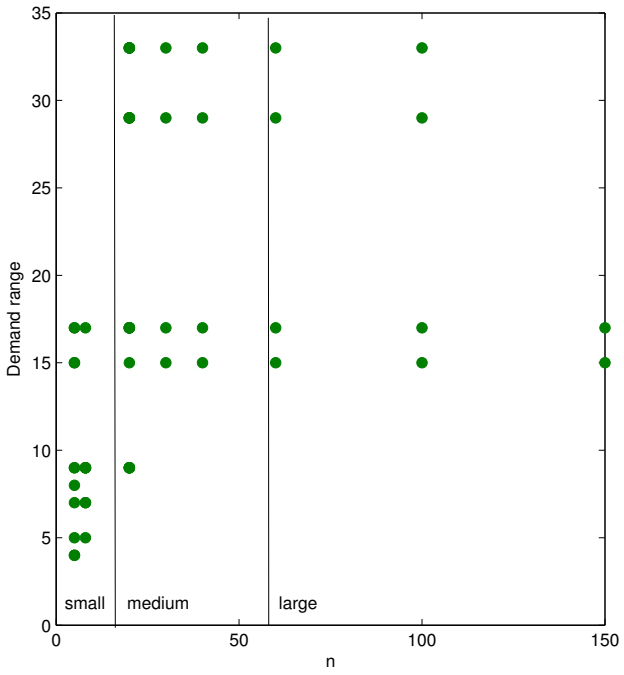


FIGURE 5.1.1. Instance demands.

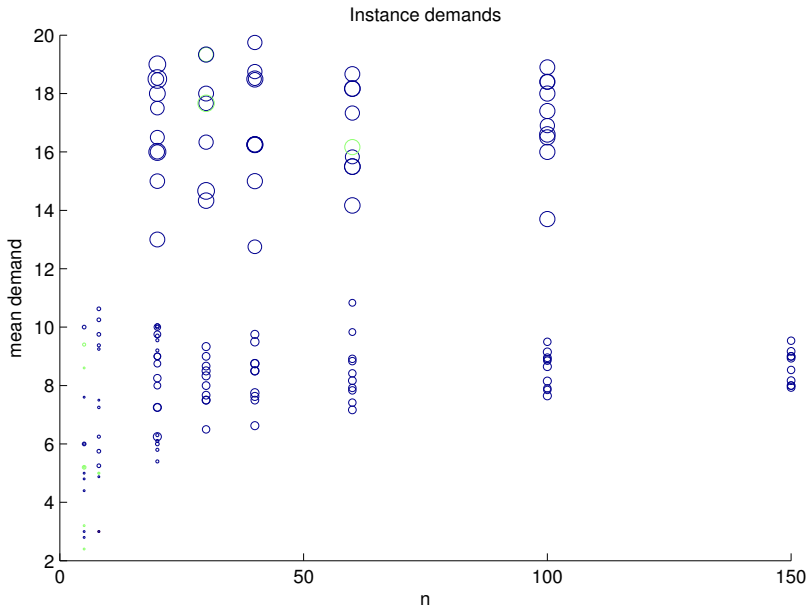
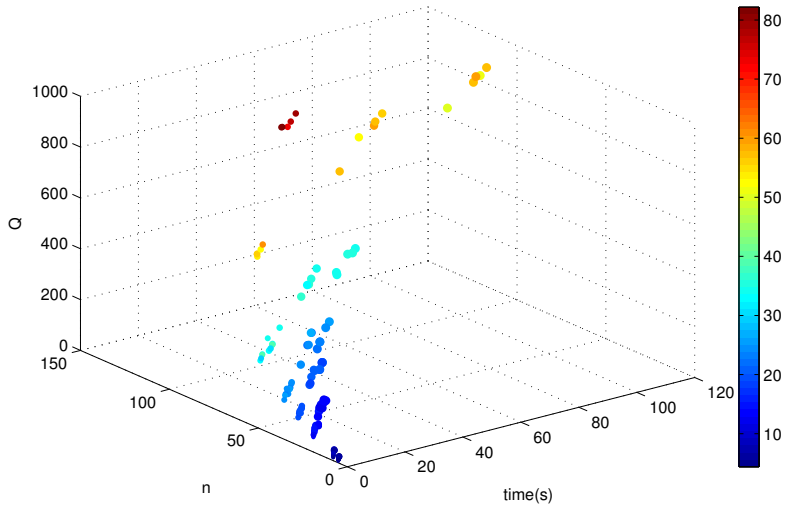
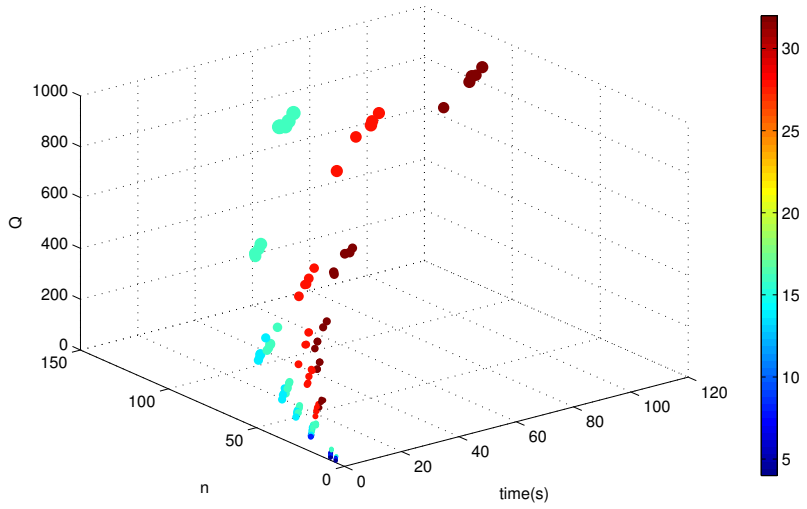


FIGURE 5.1.2. Instance demands. Circles area represents the demand variance



(A) Color shows expected distance to an arbitrary policy



(B) color represents demand disribution

FIGURE 5.1.3. Expected distance algorithm Γ performance

5.1.3. Rollout algorithm

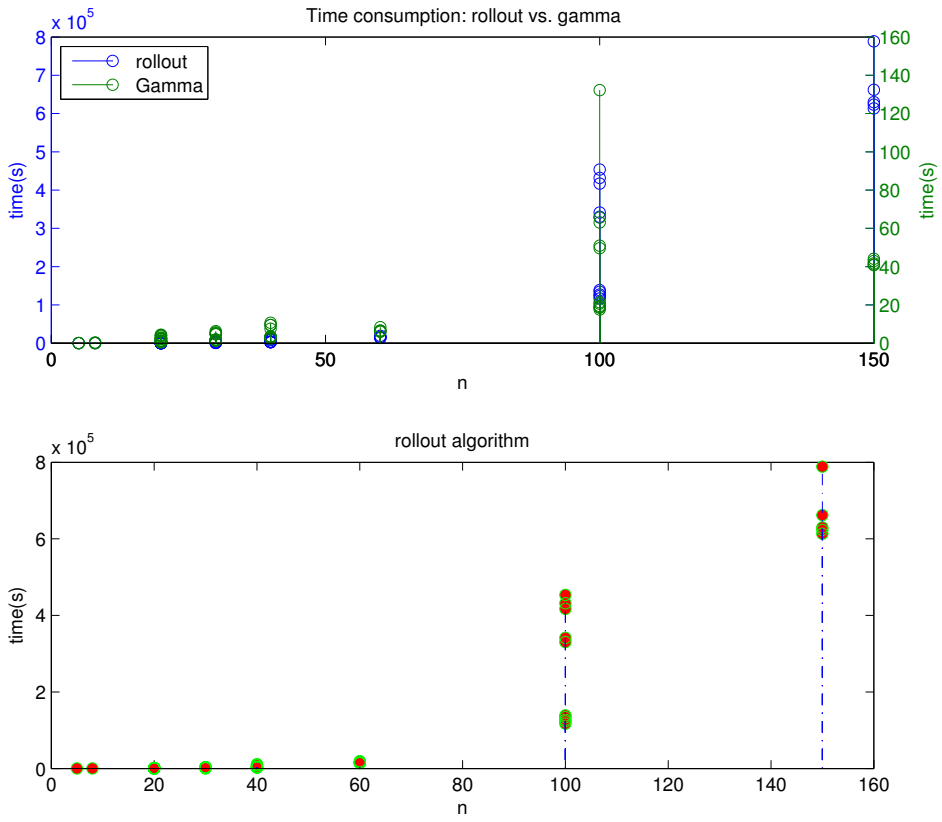


FIGURE 5.1.4. Performance rollout algorithm vs. Γ algorithm

5.1.4. Evoulutionary approach

when local search is applied the evolutionary algorithm stops when the number of iterations is achived.

Both $\mathbf{m}$ 10% of the iterations and ε are $1x10^{-3}$.

Suitable parameters was chosen by experimental results whitout

In the case of number of iterations κ and size of population α the parameters time consumption restrict this since we conclude that a greater value for these increase the solutions explored.

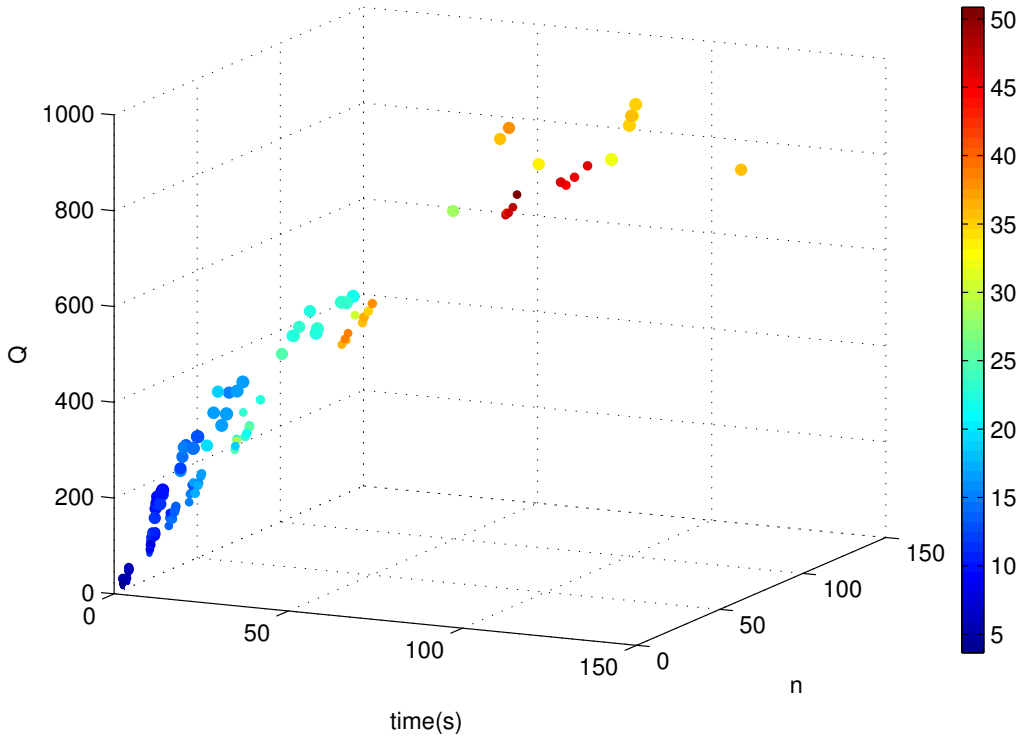


FIGURE 5.1.5. Performance rollout algorithm

Performance of the genetic algorithms

This section describes the algorithm performance for a specific problem instance with 10 customers and a factor $f' = 1$, the customers' location and demand value (known when the vehicle arrive to the customer location) are showed in the figure **??**.

The genetic algorithm outcome for the instance problem described above is presented in the figure **??**, the red line describes the route, the segmented black line represents returns to the depot for replenishment, this tour was selected as the best solution found in 100 iterations of the GA.

The figure **??** shows the improving of the objective function found, the chart shows the lesser total cost history, when a better value is found the line fall, therefore we can see the improving is given among the iteration 25 and 40, after this iterations the algorithm converges and it's not found a better solution.

5.1.5. Comparative results

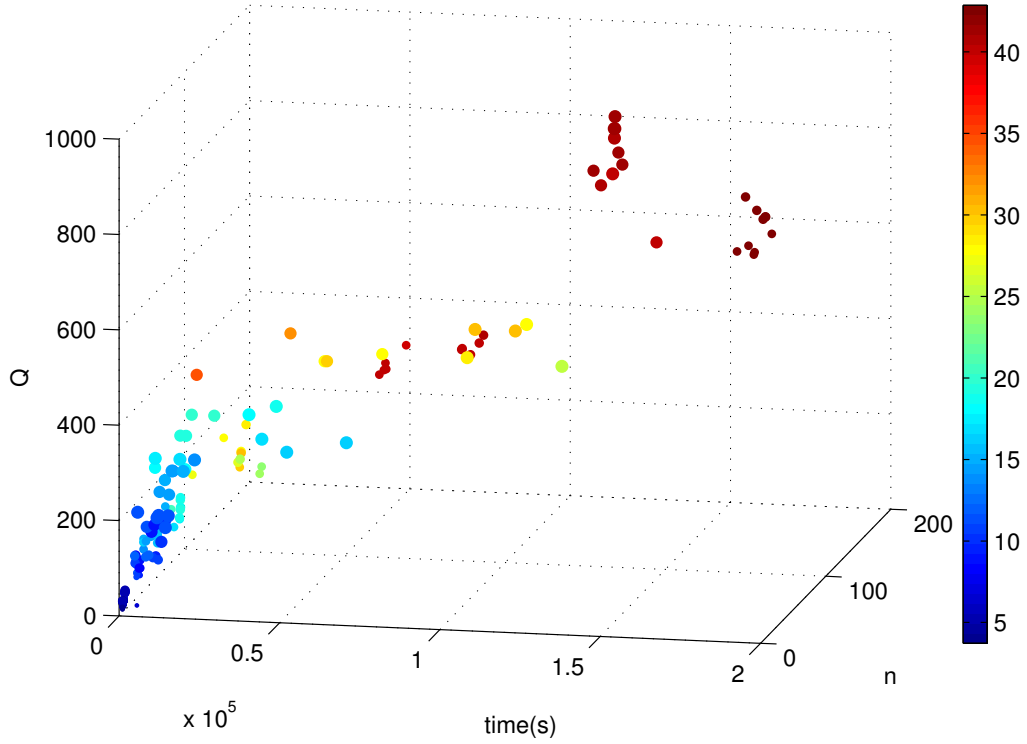


FIGURE 5.1.6. Performance ga

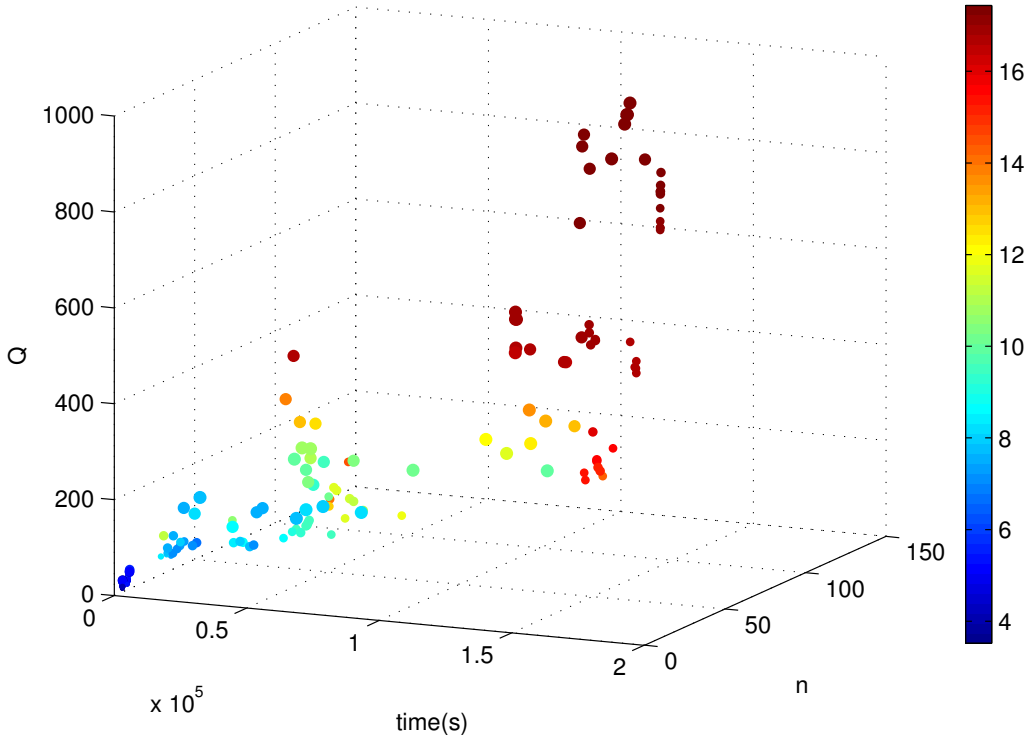


FIGURE 5.1.7. Performance Memetic

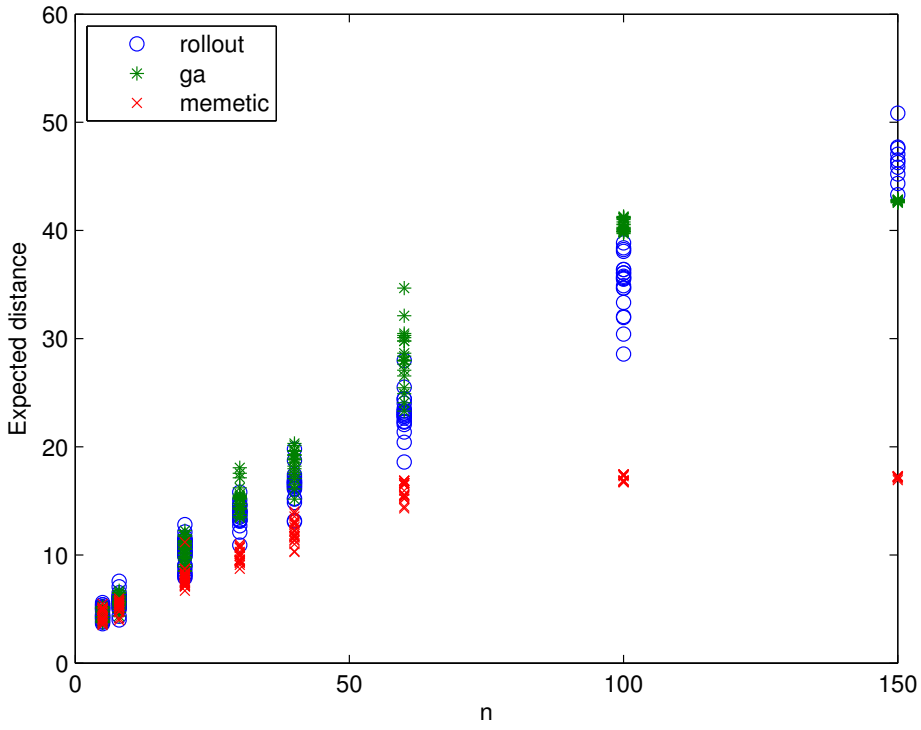


FIGURE 5.1.8. Compared results ra,ga,Memetic

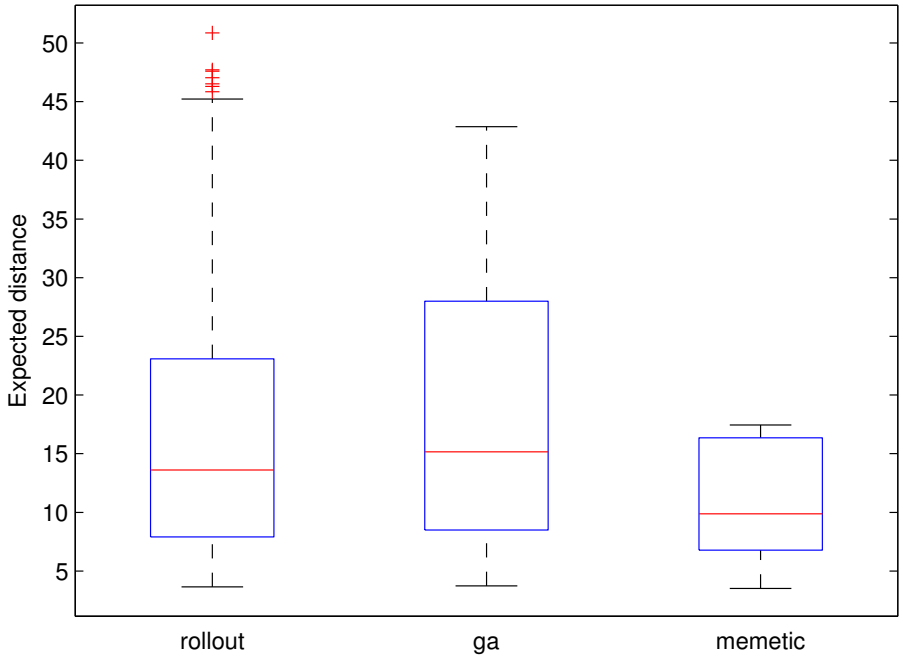


FIGURE 5.1.9. Boxplot for expected distance ra,ga,Memetic

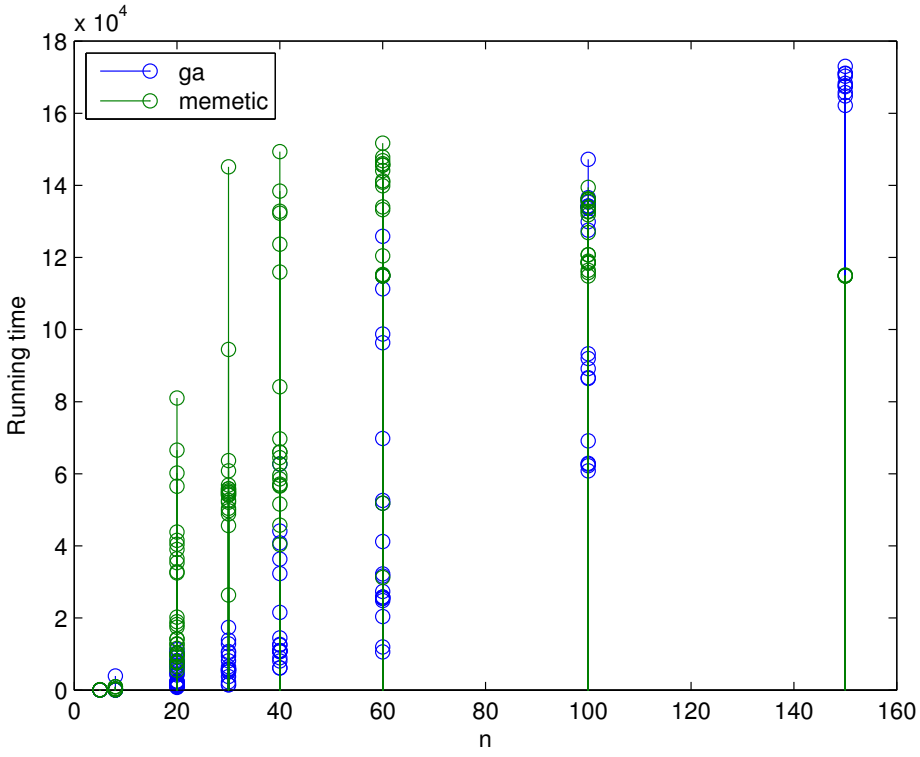


FIGURE 5.1.10. Time performance evolutionary algorithms

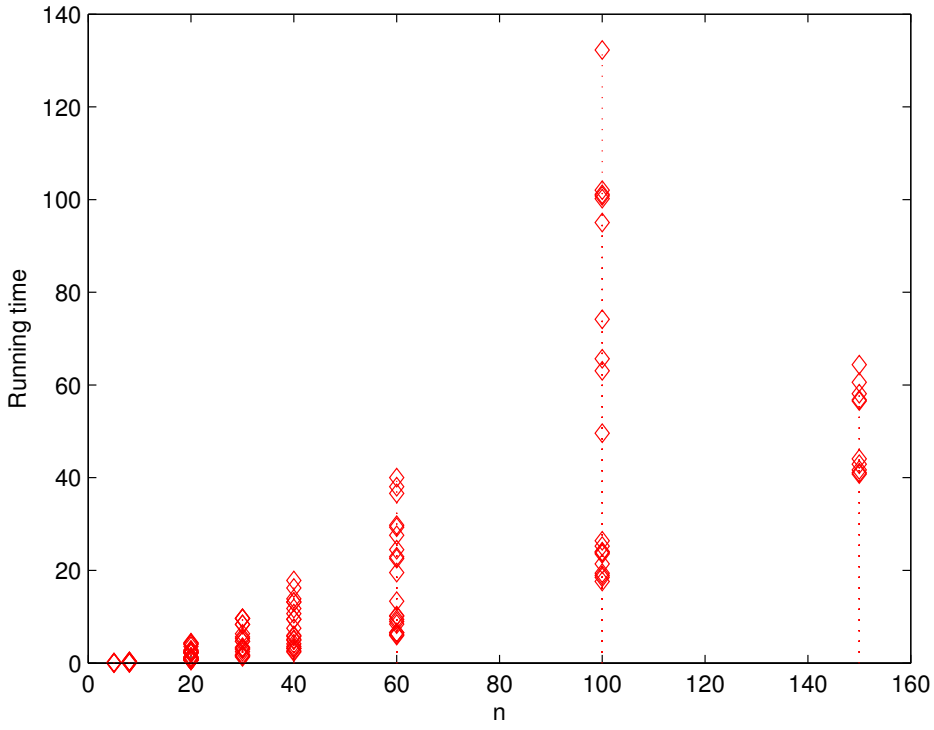


FIGURE 5.1.11. Time performance evolutionary algorithms

CHAPTER 6

Conclusions

6.1. Conclusions

Although the hybridization approach increases the computational time cost, this effort is rewarded in many cases given that the quality of the solutions is improved than if we apply the rollout algorithm alone.

Hence, since the local searchis performed every time a new solution is generated.

Often memetic algorithms employ long runnig times because local search is the most time-consuming component. For this reason, we partialy apply the rollout algorithm in large instances to reduce the time consumption to compute the solution.

Although the genetic operators attempt to explore by extending the search space to avoid local optimum. This operators by themselves do not improve the solutions. So, it is necessary to focus on local search to improve the quality solutions and the convergence.

To evaluate expected distance consume a lot of computational time, so an efficient approximation improves the algorithm performance.

Given that local search is performed every time a new solution si generated often memetic algorithms employ long runnig times.

Local search is the most time-consuming component.

Evaluate expected distance consume a lot of execution time, efficiently approximation improve the algorithm performance.

Focus local search improve the quality solutions and the convergency.

when local search is applied the evolutionary algorithm stops when the number of iterations is achived.

6.2. Perspectives

APPENDIX A

Algorithms

A.1. Algorithms detaileds

where SN is the set of nodes that still need to be visited, i.e. $\forall l \in N, D_l > 0$, x_f is the final state, where the vehicle comeback to depot and each customer was visited and its demand is 0, $x = (0, Q, 0, \ldots, 0)$, $\bar{\tau}$ is the minimum τ selected in each algorithm iteration. $q_l = x_2$, $l = x_1$ and $r_l = x_{l+2}$
When $l = 0$ or $i = 1$ in the algorithm, $q_l = Q$ $sh(\tau, i)$ shift sub τ vector from position i to the final position.

The $g^a(l, m, x)$ function is the expected distance from l to m given the state x, where $a = 1$ if earlier replanishment is specified or $a = 0$ in otherwise. The function is described below.

$$g^0(l,m,x) = d(\tau_l, m) + \sum_{k=0}^{q_l} p_m(k)\Gamma(\tau, l+1, q_l - k) + \sum_{k=q_l+1}^{K_m} 2d(0,m)p_m(k)\Gamma(\tau, l+1, q_l + Q - k) \tag{A.1.1}$$

$$g^1(l,m,x) = d(0,\tau_l) + d(0,m) + \sum_{k=0}^{K_m} p_m(k)\Gamma(\tau, l+1, Q-k) \tag{A.1.2}$$

$x_l = \Upsilon(x, u)$ represent the transition of the state x to state x_l given that the control u is realized.

input : tour $\tau_{1\times n+2}$, state $x_{1\times n+2}$
output: π policy
$\bar{\tau} = \tau$;
$i = 1$;
while $x \neq x_f$ **do**
    $\tau = \bar{\tau}$;
    for $j \in SN$ **do**
        if *i=1* **then** is the first node in the tour τ, i.e. l=0
            $\tilde{J} = \Gamma(\tau, 0, q_l)$;
            $a_{min} = 0$;
        end
        else
            $J^0 = g^0(i, \tau_{i+1}, x)$;
            $J^1 = g^1(i, \tau_{i+1}, x)$;
            $\tilde{J} = \min\{J^0, J^1\}$;
            $a = \arg\min\{J^0, J^1\} - 1$;
        end
        if $\tilde{J}_{min} > \tilde{J}$ **then**
            $\tilde{J}_{min} = \tilde{J}$;
            $a_{min} = a$;
            $\bar{\tau} = \tau$;
            $\tau = sh(\tau, i)$;
        end
    end
    $l = \bar{\tau}_{i+1}$;
    $\pi \leftarrow u = (l, a_{min})$;
    $x = \Upsilon(x, u)$;
    if $r_l = 0$ **then**
        $SN = SN - l$;
    end
    $i = i + 1$;
end

Algorithm 4: Rollout algorithm

APPENDIX A

Results

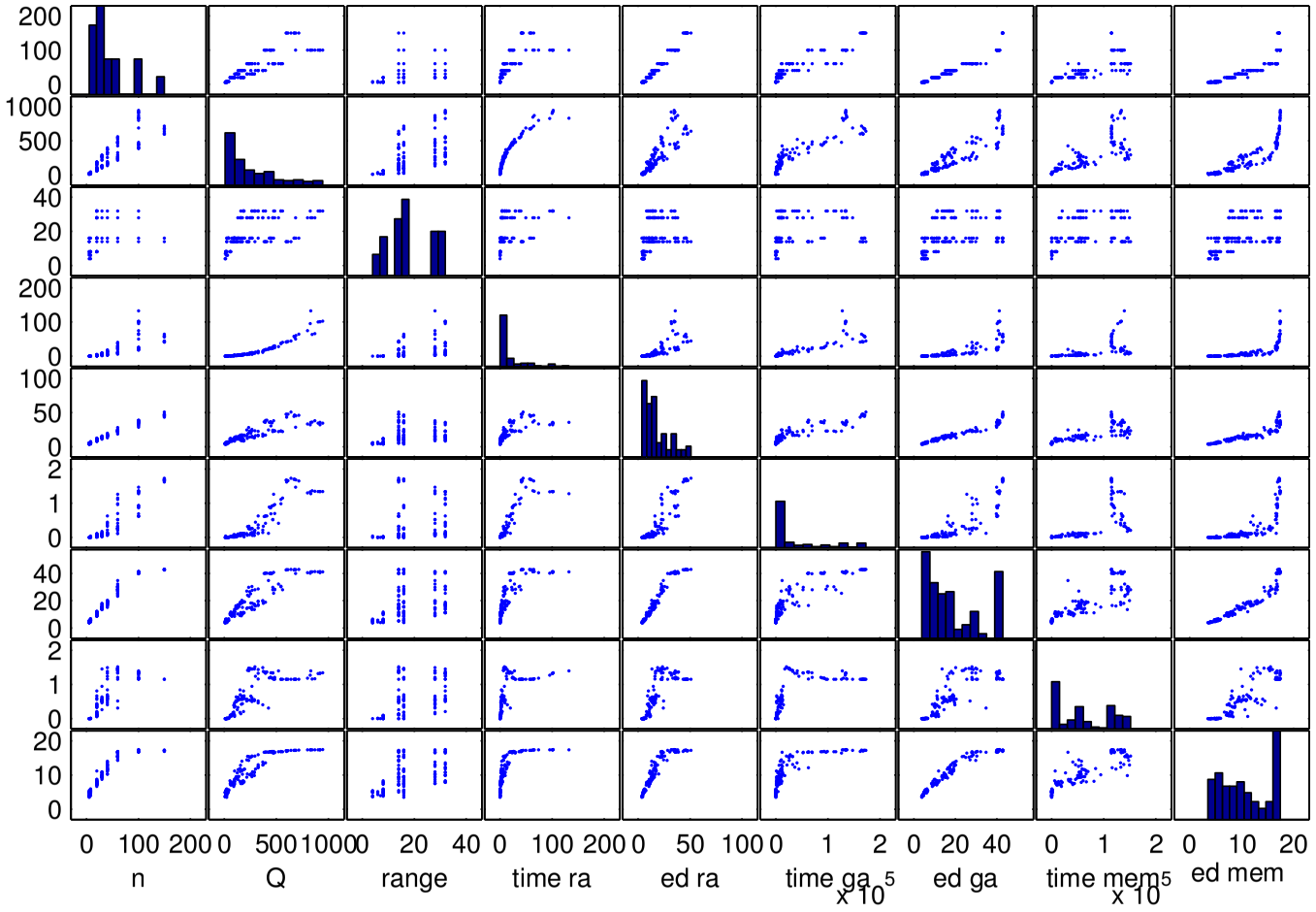


FIGURE A.0.1. Scatter matrix comparing results and times

A.0.1. Peformance rollout algorithm

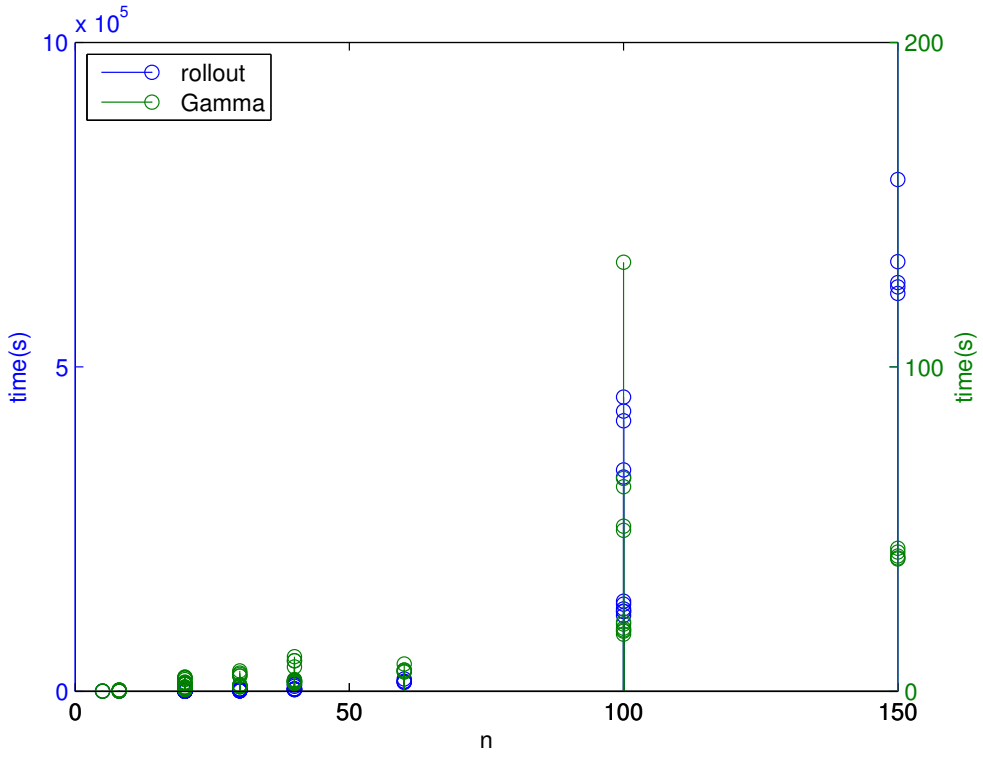


FIGURE A.0.2. Performance rollout algorithm vs. Γ algorithm

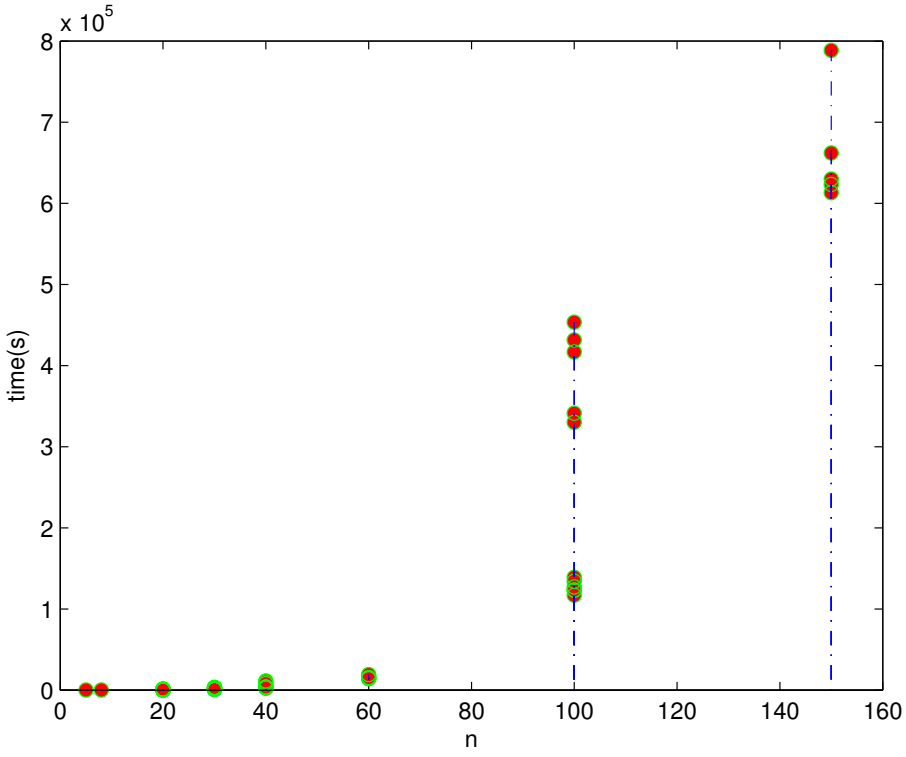


FIGURE A.0.3. execution time to accomplish rollout algorithm

Bibliography

[1] R. Bellman. The theory of dynamic programming. *Bull. Amer. Math. Soc*, 60(6):503–515, 1954.

[2] Dimitri Bertsekas. Dynamic programming and stochastic control. *Journal of the American Statical association*, 74:510–511, 1979.

[3] Dimitri P. Bertsekas. *Dynamic Programming and Optimal Control*. Athena Scientific, 1995.

[4] Dimitri P. Bertsekas. *Differential Training Of Rollout Policies*. 1997.

[5] Dimitris J. Bertsimas. A vehicle routing problem with stochastic demand. *Operations Research*, 40(3):574–585, May 1992.

[6] Dimitris J. Bertsimas and Garrett van Ryzin. A stochastic and dynamic vehicle routing problem in the euclidean plane. *Operations Research*, 39(4):601–615, August 1991. ArticleType: primary_article / Full publication date: Jul. - Aug., 1991 / Copyright Âĺ 1991 INFORMS.

[7] Leonora Bianchi, Mauro Birattari, Marco Chiarandini, Max Manfrin, Monaldo Mastrolilli, Luis Paquete, Olivia Rossi-Doria, and Tommaso Schiavinotto. Hybrid metaheuristics for the vehicle routing problem with stochastic demands. *Journal of Mathematical Modelling and Algorithms*, 5(1):91–110, April 2006.

[8] Homem-De-Mello T Chepuri K. Solving the vehicle routing problem with stochastic demands using the cross entropy method. *Annals of Operations Research*, 55:153–181, 2005.

[9] Bernard K.-S. Cheung, K.L. Choy, Chung-Lun Li, Wenzhong Shi, and Jian Tang. Dynamic routing model and solution methods for fleet management with mobile technologies. *International Journal of Production Economics*, 113(2):694–705, June 2008.

[10] Christian H. Christiansen and Jens Lysgaard. A branch-and-price algorithm for the capacitated vehicle routing problem with stochastic demands. *Operations Research Letters*, 35(6):773–781, November 2007.

[11] G. B. Dantzing and J. H. Ramser. The truck dispatching problem, Oct 1959.

[12] Maged Dessouky, Fernando Ordonez, Hongzhong Jia, and Zhihong Shen. Rapid distribution of medical supplies. In *Patient Flow: Reducing Delay in Healthcare Delivery*, pages 309–338. 2006.

[13] M. Dror, M. Ball, and B. Golden. A computational comparison of algorithms for the inventory routing problem. *Annals of Operations Research*, 4(1):1–23, December 1985.

[14] Moshe Dror. Modeling vehicle routing with uncertain demands as a stochastic program: Properties of the corresponding solution. *European Journal of Operational Research*, 64(3):432–441, 1993.

[15] Moshe Dror. Vehicle routing with stochastic demands: Models & computational methods. In *Modeling Uncertainty, International Series in Operations Research & Management Science*. Springer New York, 2005.

[16] Jianhua Fan, Xiufeng Wang, and Hongyun Ning. A multiple vehicles routing problem algorithm with stochastic demand. In *Intelligent Control and Automation, 2006. WCICA 2006. The Sixth World Congress on*, volume 1, pages 1688–1692, 2006.

[17] Noah Gans and Garrett van Ryzin. Dynamic vehicle dispatching: Optimal heavy traffic performance and practical insights. *Operations Research*, 47(5):pp. 675–692, 1999.

[18] Michel Gendreau, Gilbert Laporte, and Rene Seguin. An exact algorithm for the vehicle routing problem with stochastic demands and customers. *TRANSPORTATION SCIENCE*, 29(2):143–155, May 1995.

[19] Michel Gendreau, Gilbert Laporte, and Ren Sguin. Stochastic vehicle routing. *European Journal of Operational Research*, 88(1):3–12, 1996.

[20] Ali Haghani and Soojung Jung. A dynamic vehicle routing problem with time-dependent travel times. *Computers & Operations Research*, 32(11):2959–2986, November 2005.

[21] Ivana Cavar Ton?i Cari? Hrvoje Markovic. Using data mining to forecast uncertain demands in stochastic vehicle routing problem. In *13th International Symposium on Elecronics in Transport (ISEP), Slovenia*, 2005.

[22] Raja Jothi and Balaji Raghavachari. Approximating the k-traveling repairman problem with repairtimes. *Journal of Discrete Algorithms*, 5(2):293–303, June 2007.

[23] Gilbert Laporte, FranÇois V. Louveaux, and Luc Van Hamme. An integer l-shaped algorithm for the capacitated vehicle routing problem with stochastic demands. *Operations Research*, 50(3):415–423, May 2002.

[24] Jorge E. Mendoza, Bruno Castanier, Christelle Guéret, Andrés L. Medaglia, and Nubia Velasco. A memetic algorithm for the multi-compartment vehicle routing problem with stochastic demands. *Computers & Operations Research*, 37(11):1886–1898, November 2010.

[25] Rodrigo Moretti. Adaptive granular local search heuristic for a dynamic vehicle routing problem. *Computers & Operations Research*, 36:2955–2968, 2009.

[26] Clara Novoa and Robert Storer. An approximate dynamic programming approach for the vehicle routing problem with stochastic demands. *European Journal of Operational Research*, 196(2):509–515, July 2009.

[27] N. Secomandi. A rollout policy for the vehicle routing problem with stochastic demands. *Operations Research*, pages 796–802, 2001.

[28] Nicola Secomandi. *Exact and Heuristic Dynamic Programming Algorithms for the Vehicle Routing Problem with Stochastic Demands*. PhD thesis, Deparment of Decision and Information Sciences, University of Houston, Houston, TX, 1998.

[29] Nicola Secomandi. Comparing neuro-dynamic programming algorithms for the vehicle routing problem with stochastic demands. *Computers & Operations Research*, 27(11-12):1201–1225, September 2000.

[30] Alan Slater. Specification for a dynamic vehicle routing and scheduling system. *International Journal of Transport Management*, 1(1):29–40, February 2002.

[31] Martin A. Wortman Tapas K. Das. Analysis of asymmetric patrolling repairman systems. *European Journal of Operational Research*, 64:45–60, 1993.

[32] Eldon Y. Li Timon C. Du and Defrose Chouc. Dynamic vehicle routing for online b2c delivery. *The international journal of management science*, 33:33–45, 2004.

[33] Paolo Toth and Daniele Vigo. *The vehicle routing problem*. SIAM, 2001.

[34] Wen-Huei Yang, Kamlesh Mathur, and Ronald H. Ballou. Stochastic vehicle routing problem with restocking. *TRANSPORTATION SCIENCE*, 34(1):99–112, February 2000.