An approximate dynamic programming approach for the vehicle routing problem with stochastic demands

Clara Novoa\textsuperscript{a,}\textsuperscript{*}, Robert Storer\textsuperscript{b}

\textsuperscript{a}Texas State University, Ingram School of Engineering, Industrial Engineering Program, San Marcos, TX 78666, USA
\textsuperscript{b}Industrial and Systems Engineering Department, Lehigh University, Bethlehem, PA 18015, USA

\textbf{A R T I C L E   I N F O}

Article history:
Received 19 September 2007
Accepted 18 March 2008
Available online xxxx

Keywords:
Transportation
Stochastic vehicle routing
Approximate dynamic programming

\textbf{A B S T R A C T}

This paper examines approximate dynamic programming algorithms for the single-vehicle routing problem with stochastic demands from a \textit{dynamic} or reoptimization perspective. The methods extend the roll-out algorithm by implementing different base sequences (i.e. \textit{a priori} solutions), look-ahead policies, and pruning schemes. The paper also considers computing the cost-to-go with Monte Carlo simulation in addition to direct approaches. The best new method found is a two-step lookahead rollout started with a stochastic base sequence. The routing cost is about 4.8\% less than the one-step rollout algorithm started with a deterministic sequence. Results also show that Monte Carlo cost-to-go estimation reduces computation time 65\% in large instances with little or no loss in solution quality. Moreover, the paper compares results to the perfect information case from solving exact a posteriori solutions for sampled vehicle routing problems. The confidence interval for the overall mean difference is (3.56\%, 4.11\%).

\textcopyright 2008 Elsevier B.V. All rights reserved.

1. Introduction

The classical, deterministic, vehicle routing problem (VRP) seeks minimum cost routes from a depot to a geographically dispersed customers’ set having known demands. In this problem, all vehicles start and end their route at the depot, all customer demands are satisfied by exactly one vehicle, and vehicle capacities are not exceeded. The VRP has received extensive attention in the literature. Bertsimas and Simchi-Levi (1996) and Toth and Vigo (2002) review exact approaches, algorithms, and relaxations for the VRP.

Stochastic vehicle routing problems (SVRP’s) result when one or more VRP elements are random variables. Random elements might be the customers’ set, the travel times, or the customers’ demands. Bertsimas et al. (1995) and Yang et al. (2000) design \textit{a priori} routes that may precede returns to the depot before vehicle capacity is depleted. The objective is to find a route sequence that minimizes total expected cost from original distance traveled and from extra trips to and from the depot. To minimize costs further, Bertsimas et al. (1995) and Yang et al. (2000) design \textit{a priori} routes that may prescribe returns to the depot before vehicle capacity is depleted (i.e. proactive returns).

This paper studies the Vehicle Routing Problem with Stochastic Demands (VRPSD). In this problem, customers’ demands follow known probability distributions and a customer’s actual demand is only revealed when the vehicle arrives at the customer location. The VRPSD’s goal is to minimize total expected route cost. VRPSD’s occur in practice when delivering petroleum products, industrial gases (Chepuri and Homem-De-Mello, 2005), and home heating oil (Dror et al., 1985). Other VRPSD’s arise delivering products to cities under emergency (Dessouky et al., 2005), hospitals, restaurants, vending machines (Yang et al., 2000), and bank branches. Random demands are also present when collecting money (Laporte et al., 1989), packages (Markovic et al., 2005), sludge, and recycled materials from banks, homes, and industrial plants.

Most VRPSD research assumes an \textit{a priori} solution approach (Bertimas, 1992; Teodorovic and Pavkovic, 1992; Gendreau et al., 1995; Savelsbergh and Goetschalckx, 1995; Hjorring and Holt, 1999; Laporte et al., 2002; Bianchi et al., 2004; Novoa et al., 2006). In the first stage, complete \textit{a priori} routes are designed before any actual demands become known. In the second stage, routes are followed, demands are revealed, and extra trips to the depot for replenishment are performed if a customer’s demand exceeds current vehicle capacity. The routes order is not changed. The objective is to find a route sequence that minimizes total expected cost from original distance traveled and from extra trips to and from the depot. To minimize costs further, Bertsimas et al. (1995) and Yang et al. (2000) design \textit{a priori} routes that may prescribe returns to the depot before vehicle capacity is depleted (i.e. proactive returns).

This paper assumes a \textit{dynamic} solution approach that models the problem in multiple stages. Other authors (Dror et al., 1989; Bertsimas, 1992; Dror, 1993; Secomandi, 2001; Laporte et al., 2002; Secomandi and Margot, in review) call this approach “reoptimization”. In this approach, routing decisions occur concurrently...
with service and are based on the most current system state. The system state updates every time the vehicle arrives at a location and observes demand. There is no planned route and decisions at each stage are which customer to visit next and whether or not to send the vehicle to the depot for replenishment to minimize expected routing costs.

Powell et al. (1995) mention that stochastic and dynamic models are key for designing decision support systems that respond to changing conditions often observed in practical applications. Further, Psarafitis (1995) indicates that dynamic approaches respond to the need for efficient real-time logistic and that they are implementable due to advances in communications technologies such as wireless phones and global positioning systems (GPS) which facilitate interaction between drivers and dispatchers. Bastian and Rinooy Kan (1992) and Psarafitis (1995) mention that the dynamic or reoptimization approach is computationally challenging but results in flexible routes that may reduce total routing costs. Erera et al. (2007) favor fixed routes from a priori approaches since they may decrease management costs, increase drivers performance and achieve service regularity. Nevertheless, Savelsbergh and Goetschalckx (1995) present instances with a 10% increase in transportation cost from using fixed routes instead of reoptimization.

This paper examines the rollout algorithm as an efficient heuristic method for solving the dynamic single VRPSD in real-time. The rollout algorithm, originally proposed by Bertsekas and Tsitsiklis (1996), Bertsekas et al. (1997), and Bertsekas (2000, 2001), overcomes the curse of dimensionality in dynamic programming (DP). The only previous computational approaches applying rollout to the VRPSD are Secomandi (1998, 2000, 2001), Novoa (2005), and Secomandi and Margot (in review). Secomandi’s contribution in regard to rollout methods is the development of a one-step rollout algorithm.

Our paper has three main contributions. The first is the development of a two-step rollout algorithm that provides 1.6% cheaper solutions than the one-step rollout algorithm. The second is the use of Monte Carlo simulation (MCS) for computing the updated base sequences expected cost as an alternative to the exact computation in Secomandi (1998, 2000, 2001). We demonstrate that MCS may reduce the total computational time by about 65% for large instances. The third contribution is the development of improved base sequences and pruning schemes leading to a cost reduction of about 4% over previous methods. The best rollout method evidences the benefits of linking a priori and dynamic approaches for VRPSD.

Paper organization is as follows. Section 2 reviews literature on dynamic approaches. Section 3 describes the problem. Section 4 presents the dynamic programming formulation. Section 5 describes the proposed rollout algorithms and the Monte Carlo simulation approach. Section 6 contains numerical results and Section 7 concludes the paper.

2. Literature review

There are few papers on dynamic approaches to the VRPSD relative to those studying the a priori approach. The earliest contributions are theoretical models in Dror et al. (1989) and Dror (1993) that model the dynamic VRPSD as a Markov Decision process. However, these papers do not provide any computational results. Secomandi (1998, 2000, 2001, 2003) is the first author that provides a computationally tractable heuristic.

Secomandi (2000) solves the VRPSD using two approximate DP algorithms: optimistic approximate policy iteration (OAPI) and one-step rollout algorithm (ORA). OAPI is a neuro-dynamic programming (NDP) algorithm (Bertsekas and Tsitsiklis 1996) that approximates the optimal cost-to-go functions for all system states as linear (or nonlinear) functions of pre-selected problem features (i.e., vehicle location, vehicle capacity, etc.) with coefficients estimated by least squares. ORA is a simplified NDP algorithm that resembles the one-step policy iteration method. Given a properly selected a priori base sequence, ORA sequentially improves it. The ORA in Secomandi (2000) performs better than OAPI. For small instances, (less than 10 customers), Secomandi (1998, 2001) finds that ORA is less than 3% away from the optimum computed in Secomandi (1998). For problems with up to 60 customers, Secomandi (2001) finds that ORA produces average improvements from 1% to 4% over a priori solutions.

More recently, Secomandi and Margot (in review) solve VRPSD’s with up to 100 customers using a partial-reoptimization approach. Under this framework, optimal policies are computed on selected subsets of the state set using backward DP. To be able to compute these optimal policies, it is assumed that if a customer demand exceeds current vehicle capacity (i.e., a route fails) and the customer has not been completely served, the vehicle travels to the depot and return to this customer. To find the desired subsets, the authors use two methods for partitioning the initial customer sequence: disjoint and overlapping. The partial-reoptimization approach is less than 1.5% away from the optimum in problems with 10 and 15 customers. Disjoint and overlapping methods produce average improvements of 1.71% and 2.24% over the ORA in Secomandi (2001) for instances with 5–100 customers.


3. Problem description

A single-vehicle with fixed capacity Q departs from a depot to perform only deliveries (or only pick-ups) at different customer locations. Node 0 represents the depot and l = 1, 2, . . . , n represents the customers’ set. Distances between customers i and j, denoted by dij, are known, symmetric, and satisfy the triangle inequality. Travel costs are proportional to distances.

In the computational study, customers’ demands follow known discrete distributions. They are assumed statistically independent but may come from different families and may have different parameters. Let Di, i = 1, 2, . . . , n be random variables that describe the demand for each customer and pi be its probability distribution, pij = Pr(Di = r), r = 0, 1, . . . , R ≤ Q. Exact demand for a customer is revealed only when the vehicle arrives at the customer for the first time. Because of random demands, vehicle routes will fail when a customer’s demand exceeds remaining vehicle capacity. This research assumes split deliveries only when failures occur. In such a case, the vehicle may partially satisfy a customer demand, then complete it in another non-necessarily immediate visit. Upon route failure, the vehicle must return to the depot to restore the vehicle capacity to Q. The problem objective is to find a routing
4. Problem formulation

The dynamic VRPSD can be formulated as a stochastic shortest path (SSP) problem. SSPs are Markov decision models that reach an absorbing cost-free termination state in a random number of stages and have a discount factor of one. The formulation in this section follows closely the ones in Novoa (2005) and Secomandi (1998, 2000, 2001).

Let $x_k = (l, q, r_1, \ldots, r_n)$ be an array with $n + 2$ components that represents the system state at decision stage $k$. $l \in 0, 1, \ldots, n$, is the current vehicle location and $q, r_1, \ldots, r_n$ is the vehicle capacity. The remaining $n$ components, $r_i$, represent the demand yet to be delivered to customer $i$. An unknown demand value is notated as $\gamma$ and it can take any value in $1, \ldots, R$. If a customer has been visited and fully served, its demand is zero. If a customer has been visited and partially served, its demand updates to a known value, $r'_i = 1, \ldots, R - 1$, and $Pr(D = r'_i) = 1$. Therefore, each $r_i$ can take values in the set $\{0, 1, \ldots, R\}$. Let $S$ be the state space. The number of states in the system is $O(nR^2)$. The initial system state is $(0, 0, 0, 0, \ldots, 0)$ and the final system state is $(0, 0, 0, 0, \ldots, 0)$. The final state occurs when the vehicle returns to the depot after fully serving all customers and recovers its initial capacity. Let $N$ be a random variable that represents the number of stages or transitions from the initial to the terminal state regulated by the customer demands probability distribution and by the route followed. A new stage occurs when the vehicle arrives to a customer location. The array $x = (l_0, q_1, \ldots, q_n, \pi)$ is the policy or sequence of functions to optimize where $q_k$ is a function that associates a control or decision $u_k = \mu_k(x_k)$ to each stage. A control $u_k$ is a member of the feasible control set for state $x_k$, that is, $u_k \in U_k(x_k)$ and $U_k(x_k) = \{m \in \{1, \ldots, n\} | r_m \neq 0 \} \cup \{a \in \{0, 1\} \}$. Controls $u_k$ are represented as ordered pairs $(m, a)$. The first element, $m \in \{1, \ldots, n\}$, is never 0. Any non-fully served customer; $m$ is zero when all demands are satisfied and the system enters its termination state. The second element, $a \in \{0, 1\}$, is zero if the vehicle visits the next customer directly and one if the vehicle stops first at the depot for replenishment and then at the customer. The case $a = 1$ permits proactive depot trips to avoid route failures.

If the system is at stage $k$ and in state $x_k = (l_k, q_k, r_1, \ldots, r_m, \ldots, r_n) \in S$ and a control $u_k$ occurs, the system state updates to $x_{k+1} = (m, q_{k+1}, r_1, \ldots, r_{m'}, \ldots, r_n)$ incurring on a transition cost $g(x_k, u_k)$ and following transition probabilities, $p(x_k, x_{k+1}|u_k)$. Transition costs, $g(x_k, u_k)$, are direct or indirect distance between two customers (i.e. $d(l,m)$ or $d(l,0) + d(0,m)$, respectively). Transition probabilities are given by the customer demands probability distribution and the control selected. The vehicle capacity after serving customer $m$, $q_m$, and the demand at customer $m$, $r_m$, are updated according to the customer’s demand realization. The problem’s objective is to find a policy $\pi$ that minimizes the $N$-stage cost-to-go $J_N(x)$ or expected cost to termination from any initial state. The optimal $N$-stage cost-to-go from state $x$ is $J_N(x) = \min_{\pi \in \Pi} J_N(x)$ where $\Pi$ is the admissible-policies set. If $J_N(x)$ is known for all states, the optimal control $u_k^*(x)$ at each state results from finding the minimum in the following equation

$$u_k^* = \arg \min_{u_k \in U_k(x_k)} \left[ g(x_k, u_k) + \sum_{x_{k+1} \in S} p(x_k, x_{k+1}|u_k) J_N(x_{k+1}) \right] x_k \forall x \in S.$$  \hspace{1cm} (1)

Eq. (1) shows that at each stage, decisions are ranked based on present cost and expected future cost sum, assuming optimal decision making for subsequent stages. However, the large size of the state space makes it computationally prohibitive to find all the optimal cost-to-go values. In the remaining sections, the subscript $N$ on the cost-to-go is eliminated to simplify the notation.

5. Rollout algorithms

We introduce this section with a brief description of the rollout algorithm (RA) proposed by Bertsekas (2001, 2000) and Bertsekas et al. (1997) and implemented by Secomandi (1998, 2000, 2001) to solve the VRPSD dynamically. Then, we divide the sections in four subsections which describe the RA’s elements that we further study to extend the work in Secomandi (1998, 2000, 2001).

RA is an approximate method to solve large DP problems. RA assumes that a sub-optimal initial policy (i.e. base a priori policy) $\pi$ for the problem is available and that its expected cost-to-go $J(x_k)$ can be computed efficiently. Under a one-stage RA (ORA), an approximate control $\tilde{u}_k$ for a particular state $x_k$ reached stage $k$ results from approximating the true cost-to-go $J(x_k)$ in Eq. (1) by $\tilde{J}(x_k)$. This approximation is done for all states $x_k \in S$ that can be reached one-stage ahead of current stage $k$.

Rollout is a type of policy iteration since a single policy improvement step for the policy $\pi$ is done over the system’s states. Given a properly selected policy $\pi$, rollout produces an improved policy $\tilde{\pi}$. If applying RA to the dynamic VRPSD, the improvement is done over the states found in real-time execution and the improved policy $\tilde{\pi}$ is implemented at the same time it is computed. We suspected that in applying RA for VRPSD, the number of lookahead stages, the method to compute the cost-to-go, the number of feasible controls explored in a particular state and the adopted base policies might affect both the solution quality and the computational time.

5.1. Number of lookahead stages in the rollout algorithm

The ORA for VRPSD is described in Secomandi (2001) and Secomandi (1998). In this subsection, we describe the two-stage (i.e. two-step) rollout algorithm (TRA) that we implemented for the VRPSD. Eqs. (2) and (3) formally define a TRA as initially stated by Bertsekas (2001, 2000) and Bertsekas et al. (1997). These equations show that TRA involves more computation than ORA since the approximate cost-to-go values $\tilde{J}(x_{k+1})$ are obtained after performing a one-step RA. To determine the approximate control for a particular stage $k$ and state $x_k$, TRA finds not only a minimum at stage $k$ but also a minimum for all the possible states $x_{k+1}$ that can be generated from the current state $x_k$. The motivation for using TRA comes from Bertsekas (2001, 2000) and Bertsekas et al. (1997) who mention that it is possible to define rollout policies that use $m$-step lookahead ($m \geq 2$) and that may achieve better performance at the expense of more computation

$$\tilde{u}_k(x_k) \left[ = \arg \min_{u_k \in U_k(x_k)} \left[ g(x_k, u_k) + \sum_{x_{k+1} \in S} p_{x_k, x_{k+1}|u_k} J_N(x_{k+1}) \right] \right],$$  \hspace{1cm} (2)

$$\tilde{J}(x_{k+1}) \left[ = \arg \min_{u_k \in U_k(x_{k+1})} \left[ g(x_{k+1}, u_{k+1}) + \sum_{x_{k+2} \in S} p_{x_{k+1}, x_{k+2}|u_{k+1}} \tilde{J}(x_{k+2}) \right] \right],$$  \hspace{1cm} (3)

One-step and two-step RAs require a base policy previously denoted as $\pi$ to compute the approximate expected costs-to-go $J(x_{k+1})$ and $\tilde{J}(x_{k+2})$, respectively. For the dynamic VRPSD, we denote this sub-optimal initial policy as $\tau$ and left the details on how to find it for subsection 5.4. $\tau = (0, 1, \ldots, t - 1, t, t + 1, \ldots, n, 0)$ is an almost surely suboptimal visitation order, where 1 represents the first customer on the tour and so on. By fixing the visitation order, the high number of branches to explore when computing the true-cost-to-go value is greatly reduced. In this paper, $\tau$ is also named the base sequence (BS) or base tour.
At any state, RA’s require multiple BS’s updates to recompute the approximate cost-to-go considering only unsatisfied customers. Bertsimas (1992) proposes the cyclic heuristic for performing these updates. We use this simple heuristic in the TRA implementation. The cyclic heuristic slightly changes the visitation order in the BS and it always terminates from any customer where the system can be. Let \( r = (0, 1, \ldots, t-1, t, t+1, \ldots, n, 0) \) be the initial BS and assume that the vehicle is at customer \( I \) after fully serving this customer. If performing a TRA, also assume that there are two or more customers remaining to visit. To select the next customer \( m \) to visit (directly or stopping first at the depot), it is necessary to update the BS from each candidate customer \( t \) that is two-step neighbor of customer \( I \). The cyclic heuristic \( \zeta \) produces an updated base sequence (UBS) \( \hat{r}_t^I = (t, t+1, \ldots, 1, \ldots, t-1, 0) \) by following \( r \) cyclically from \( t \), skipping \( m \) and any fully served customers such as \( I \), and at the end, returning to the depot. In TRA’s last step, there is only one customer remaining to visit and therefore ORA is applied updating the BS from \( m \), the one-step neighbor of \( I \).

Bertsekas and Tsitsiklis (1996) and Bertsekas et al. (1997) mention that if the BS for RA is computed with a sequentially consistent heuristic, then the resulting rollout sequence is an improved one. Secomandi (1998) provides a proof that the cyclic heuristic is sequentially consistent. Bertsekas et al. (1997) and Bertsekas (2000) describe in detail the sequentially consistent property, and Secomandi (2001) discusses other important RA’s properties.

5.2. Computation of the expected cost-to-go of an updated base sequence (UBS)

Exact computation. The expected length of \( \hat{r}_t^I \) at state \( x_{k,2} \) when the vehicle is at customer \( t \) and has capacity \( q_t \), is denoted by \( E[L_{J^I_t}|x_{k,2}, q_t] \). In a DP context this expected length is the expected cost-to-go of following policy \( \hat{r}_t^I \) from state \( x_{k,2} \) to the terminal state. In other words, \( E[L_{J^I_t}|x_{k,2}, q_t] \) replaces \( J_k(x_{k,2}) \) in Eq. (3). Our computational study use the equations derived by Secomandi (1998, 2001) to exactly compute \( E[L_{J^I_t}|x_{k,2}, q_t] \). These equations are based on a DP backward recursion that takes \( O(nRQ) \) time and \( O(nQ) \) space. The formulas in Secomandi (1998, 2001) are used because they consider that the vehicle may take proactive returns to the depot, they have low computational complexity, and they permit direct two-step and one-step RA comparisons.

Approximate computation. To extend the work in Secomandi (2001) we also study the computation of the expected cost-to-go of an UBS using on-line Monte Carlo simulation (MCS). Bertsekas (2001) introduces MCS to approximate the cost-to-go functions in DP problems. We implement Monte Carlo cost-to-go approximations to reduce computational time for RA. Our goal is also assess MCS performance compared to using the exact formulas in Secomandi (2001). Further, the use of approximations is motivated by Bertsekas (2000) and Bertsekas et al. (1997) who cite that more than approximating the true cost-to-go over a range of relevant states, it is important to closely approximate the cost-to-go differentials \( J_k(x_{k+1}^I) - J_k(x_{k}^I) \approx J_k(x_{k+1}^I) - J_k(x_{k}^I) \) for any two states \( x_k \) and \( x' \). The approximate expected cost-to-go of an UBS is denoted as \( E[L_{J^I_t}|x_{k,2}, q_t] \).

The computation of \( E[L_{J^I_t}|x_{k,2}, q_t] \) under MCS considers all controls and states originated from state \( x_{k,2} \) to termination if following the updated basic policy \( r_t^I \), and generating a finite number scenarios \( S' \in S' \) for the customers’ demands distributions. Given a joint demand scenario \( s' \in S' \), the cost-to-go \( J(x_{k}, s') \) is the distance traveled by the vehicle considering that as demands realize vehicle capacity updates and that some returns to the depot are required. The value for \( E[L_{J^I_t}|x_{k,2}, q_t] \) is the average distance over the scenarios in \( S' \). Thus, \( E[L_{J^I_t}|x_{k,2}, q_t] = \sum_{s' \in S'} P(s' | x_{k,2}) \). Therefore, to compute approximate controls for TRA under MCS, \( E[L_{J^I_t}|x_{k,2}, q_t] = \sum_{s=1}^{S} P(s'|x_{k,2}) \) replaces \( J_k(x_{k,2}) \) in Eq. (3). Similarly, to compute approximate controls for ORA under MCS, \( E[L_{J^I_t}|x_{k+1,2}, q_t] = \sum_{s=1}^{S} P(s'|x_{k+1,2}) \) replaces \( J_k(x_{k,1}) \) in Eq. (1).

Note that we continue using \( S \) to denote the state space as in Secomandi (2001), Dror et al. (1989) and Dror (1993). Since in many stochastic programming literature \( S \) is traditionally used to represent the set of demand scenarios, we slightly changed the notation for the entire set of \( (S' \) subset of demand scenarios to \( S' (S) \).

5.3. Number of feasible controls

The multiple computations of the expected cost-to-go from different states represent a significant portion of RA’s computational time. In addition to studying MCS to compute the cost-to-go, we study a pruning rule to explore only a subset of all the feasible controls to explore at any state. We recall that controls \( u_t \) are ordered pairs \( (m,a) \) where \( m \) is the next customer to visit and \( a = 0 \) if the vehicle visits customer \( m \) directly or \( a = 1 \) if the vehicle stops at the depot before visiting the customer.

The pruning rule is based on the probability of route failure at the next customer \( m \). If this probability is lower than a selected low threshold, the only controls considered are the ones visiting the customer directly. Similarly, if the probability of route failure exceeds a high threshold, the only controls considered are the ones visiting the depot before visiting the customer.

5.4. Base sequences (BS’s)

Secomandi (2001) finds a BS (base tour) \( r \) for his ORA implementation by ignoring customers’ demands and applying the Nearest Neighbor plus 2-interchange (NN+2-int) heuristic to the traveling salesman problem (TSP). Nemhauser and Wolsey (1999) provides a detailed description of this TSP heuristic. This subsection describes other procedures we used to find stochastic and deterministic BS’s \( r \) to start ORA and TRA. Abbreviated names for each procedure are given in parenthesis. Nearest Neighbor plus 2-interchange procedure is abbreviated as n2.

Base sequence from an exact solution to the TSP. (tsp) The parallel branch, cut, and price framework SYMPHONY in Ladányi et al. (2001) and Ralphs et al. (2006) exactly solves the TSP.

Recomputing the base sequence. (n2e2opt) The solution to NN+2-int at each decision stage considering only the non served customers generates a recomputed BS. The aim of recomputing the BS is to assess the increase in computational time. If times remain small, RA’s can solve VRPSD extensions, where customers and locations reveal dynamically over time.

Including customers’ demands to find a base sequence. (vrp) SYMPHONY solves the VRP exactly as described in Ralphs et al. (2003) and Ralphs et al. (2006). Customers’ demands are replaced by expected demands and total routes \( m \) are estimated as \( m = \sum_{k=1}^{n_{m}} |\hat{m}_{k}| \). Then, the resulting routes’ set is transformed into a single BS by selecting arbitrarily the starting route, the connection order, and the orientation. The purpose of studying this method to compute a BS is to assess any advantage of including two known parameters: distances and average customer demands.

Combining several base sequences. (combi) Bertsekas et al. (1997) and Bertsekas and Tsitsiklis (1996) mention that some problems may have several good base policies available and that it is possible to incorporate all of them in a RA. If there are \( P \) base policies, the optimal cost-to-go values \( f \) can be approximated as \( f = \min_{p=1}^{P} f_{p} \).

The three single BS’s we combined are: (1) vrp, that results from solving the VRP exactly as explained above, (2) ORAap, that results after performing an “a priori” ORA with customers’ demand realizations equal to their expected values and vrp as the procedure to find the BS, and (3) TRAap, that results after performing an “a
priori” TRA with customers’ demand realizations equal to their expected values and vrp as the procedure to find the BS. The sequences generated in (2) and (3) have several proactive and/or reactive returns to the depot and some customers visited more than once. To transform these sequences into a BS we skip repeated visits to the depot or the customers.

Base sequence from an a priori heuristic solution to the VRPSD. (stostat) Novoa (2005), Novoa et al. (2006) develop a stochastic set-partitioning based model for solving heuristically a multiple VRPSD under an a priori or static approach. We adapt this model for the single-vehicle situation. The adapted model generates a large set of visitation sequences that cover all customers. The exact expected cost is computed for each sequence and the two possible route orientations. The model selects the lowest cost sequence.

6. Numerical results

This section compares the different RA’s studied using an analysis of variance procedure (ANOVA) and assesses the gap between RA’s and a posteriori or perfect information solutions. All the methods compared were coded in C++ and were run in a Pentium-IV, 2.4-Ghz, 512 MB. The section starts with a brief description of the procedures to generate the instances and to label the different algorithms compared.

6.1. Instance generation

The procedure used to generate instances is similar to the one in Gendreau et al. (1995) and Secomandi (1998, 2000, 2001). The RA’s are tested on the 120 instances studied in Novoa (2005). These instances are grouped in two sets, labeled as Set1 and Set2, according to the customers’ demand distributions. Each set contains 60 different instances resulting from six sizes for the customers’ set n = {5, 8, 20, 30, 40, 60}, two vehicle capacities, and five different assignments for customer locations and demand distributions. The different assignments result from changing the random seeds for providing to each customer a geographic location and a demand distribution from the ones in Table 1. This table also shows the a priori expected demand for each customer Di and the demand distribution variance. Customer locations are assigned randomly in a bi-dimensional grid [0,1]2 and the depot location is fixed at (0,0).

Each set studies two levels for the expected filling coefficient (i.e. filling rate) \( f = \frac{Q}{mQ} \) where \( m = 1 \) for the single VRPSD. This coefficient measures the total expected demand relative to vehicle capacity and represents approximately the expected number of trips originated from the depot to serve all customers. Thus, \( f = \max(0, f - 1) \) is an approximation for the expected number of route failures in a problem. The levels selected for \( f \) are 1.75 and 2.75 for instances in Set1, and 3.00 and 4.00 for instances in Set2. Then, the two values for Q in each set can be computed from the filling rate formula assuming \( m = 1 \) and the given f values, replacing \( E[D_i] \) by \( D_i \) and rounding the resulting value to the nearest integer.

6.2. Algorithms compared

We baptized the rollout methods studied with abbreviated names. The first two symbols in the abbreviated name describe the type of lookahead procedure, that is, one-step (i.e. 1s) or two-step (i.e. 2s). The symbols after the first underscore describe the procedure for obtaining the BS and correspond to the names given in Section 5.4. The symbols after the second underscore describe the procedure for computing the expected cost-to-go of the UB’s and the use of any pruning rule to reduce the control set. Abbreviated names ending in “r” use the exact recursive formula in Secomandi (2001). Abbreviated names ending in mc30 (mc15) use MCS with demand scenarios of size 30 (15). If the method is using the pruning rule described in Subsection 5.3, the suffix pr1 (pr2) is used to indicate that the low and high thresholds for the pruning rule are 0.3 (0.2) and 0.7 (0.8), respectively. The RA method equivalent to the one implemented by Secomandi (2001), is 1s_n2_r. We refer to this method as the “original RA” in the remainder of the paper.

The RA’s listed above are compared to follow a priori deterministic sequences coming from solving TSP with NN+2int. This method is labeled as NN2int. Further, for instances in Set1, this work computes a lower bound for the RA’s. This bound is the a posteriori or perfect information value for the expected routing cost and it is labeled as Perfinf. It results from solving exactly the deterministic VRP problem for a sample of joint demand scenarios S with SYMPHONY (Ladányi et al., 2001; Ralphs et al., 2006).

6.3. Analysis of results

The performance measure to compare the RA’s is the estimated expected routing cost. It results from executing each rollout method 100 times on each instance (i.e. 100 Monte Carlo replicates for the on-line customers’ demand realizations). Thus, each RA is executed 6000 times in each experimental set. The reason for computing the expected routing cost approximately is that exact computation becomes intractable for most of the instances due to the large number of customers and the granularity in their demand distributions.

We use one-way analysis of variance (ANOVA) to determine if there exist difference in routing costs among the RA’s. In the ANOVA procedure, rollout method is the factor and instance characteristics (number of customers, vehicle capacity, customer locations and demands distribution assignment) are treated as a block. Table 2 displays the relative effects (% of reduction in routing cost) for each method when compared to NN2int. The relative effects result from dividing the ANOVA’s absolute effects over the average routing cost computed over all the observations.

Table 2 shows that several proposed RA’s perform better than the “original RA” (1s_n2_r). The best rollout method is the one that uses a two-step lookahead procedure started with an a priori solution to the VRPSD. The improvement if comparing to the “original RA” is 5.28% (4.28%) for instances in Set1 (Set2). Results suggest that in the instances studied is productive to incorporate stochastic factors on the BS’s. Table 2 also shows that one and two-step RA’s that combine deterministic base policies have excellent performance being about 4% better in Set1 (2.5% better in Set2) than the “original RA”.

The comparison between one-step and two-step methods in Table 2 suggest that even at expense of more computations two-step RA’s delay the use of the sub-optimal base policy favoring a more
Table 2
Relative effects for each rollout method

<table>
<thead>
<tr>
<th>Rollout method</th>
<th>Instances Set1</th>
<th>Instances Set2</th>
<th>Rollout method</th>
<th>Instances Set1</th>
<th>Instances Set2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Perfinf</td>
<td>-13.80%</td>
<td>-</td>
<td>1s_ORAappr</td>
<td>-8.09%</td>
<td>-4.22%</td>
</tr>
<tr>
<td>2s_stostat_r</td>
<td>-10.42%</td>
<td>-8.04%</td>
<td>1s_n2_mc30</td>
<td>-5.07%</td>
<td>-3.80%</td>
</tr>
<tr>
<td>1s_stostat_r</td>
<td>-9.99%</td>
<td>-7.46%</td>
<td>1s_n2_r</td>
<td>-5.14%</td>
<td>-3.76%</td>
</tr>
<tr>
<td>2s_combi_r</td>
<td>-9.21%</td>
<td>-6.25%</td>
<td>1s_n2ropt_r</td>
<td>-4.44%</td>
<td>-3.71%</td>
</tr>
<tr>
<td>1s_combi_r</td>
<td>-9.37%</td>
<td>-5.89%</td>
<td>1s_n2_mc15</td>
<td>-4.96%</td>
<td>-3.53%</td>
</tr>
<tr>
<td>2s_n2_r</td>
<td>-6.84%</td>
<td>-5.31%</td>
<td>1s_n2_mc30p2</td>
<td>-4.88%</td>
<td>-2.85%</td>
</tr>
<tr>
<td>1s_TRAappr_r</td>
<td>-6.01%</td>
<td>-4.62%</td>
<td>1s_n2mc30pr1</td>
<td>-4.88%</td>
<td>-1.97%</td>
</tr>
<tr>
<td>1s_vrp_r</td>
<td>-8.68%</td>
<td>-4.58%</td>
<td>NN2int</td>
<td>0.00%</td>
<td>0.00%</td>
</tr>
<tr>
<td></td>
<td>-8.48%</td>
<td>-4.49%</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 3
Confidence intervals for the differences in instances with 30–60 customers

<table>
<thead>
<tr>
<th>Number of customers and vehicle capacity</th>
<th>30</th>
<th>40</th>
<th>60</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean</td>
<td>3.05%</td>
<td>3.32%</td>
<td>2.79%</td>
</tr>
<tr>
<td>95% SD</td>
<td>2.14%</td>
<td>2.18%</td>
<td>1.39%</td>
</tr>
<tr>
<td>95% UCI</td>
<td>2.57%</td>
<td>2.83%</td>
<td>2.48%</td>
</tr>
<tr>
<td>95% UCI</td>
<td>3.53%</td>
<td>3.80%</td>
<td>3.10%</td>
</tr>
</tbody>
</table>

Fig. 1. Average CPU times “original rollout” vs. MCS.

Please cite this article in press as: Novoa, C., Storer, R., An approximate dynamic programming approach for the vehicle routing ..., European Journal of Operational Research (2008), doi:10.1016/j.ejor.2008.03.023
7. Conclusions

The paper contribution is the development of efficient rollout algorithms (RA’s) for solving the single VRPSD under a dynamic approach. RA is a type of policy iteration where single or multiple initial suboptimal base policies are sequentially improved. Studied RA’s extend the computational work on single-stage RA in Secomandi (2001) and the theoretical work on RA’s in Bertsekas et al. (1997), Bertsekas and Tsitsiklis (1996) and Bertsekas (2000, 2001). Developed RA’s consider only states found in real-time system’s execution and permit the actual implementation of improved policies. Results are promising since routing cost for ORA’s that compute expected lengths of UBS’s with MCS is as good as the one for ORA’s computing expected lengths of UBS’s exactly. Further, ORA’s using MCS reduce in about 60% (92%) the computational time if compared to the “original RA” for studied problems with 60 (150) customers.

Pruning rules may reduce the computational time with an increase in routing cost of less than 1% on average. The best rollout, a TRA based on a stochastic BS, improves routing cost in about 4.8% if compared to the “original RA” and it is on average only 3.8% away from the perfect information solution. Besides, the best RA improves about 8–10% over the results from following visitation sequences that ignore stochastic and dynamic factors such as following the solution from NN+2-int. The low computational times for all RA’s studied encourages the real-time solution of problem’s extensions such as the multiple vehicle with stochastic customer arrivals. Further research should integrate best levels of the rollout factors explored, that is, develop a TRA that incorporates pruning rules, and starts with a stochastic BS to be updated with MCS.

References


