ARTICLE IN PRESS

European Journal of Operational Research xxx (2008) xxx–xxx

Contents lists available at ScienceDirect

European Journal of Operational Research

journal homepage: www.elsevier.com/locate/ejor

Production, Manufacturing and Logistics

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

Clara Novoa [a,*], Robert Storer [b]

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

ARTICLE INFO

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

Keywords:
Transportation
Stochastic vehicle routing
Approximate dynamic programming

ABSTRACT

This paper examines approximate dynamic programming algorithms for the single-vehicle routing problem with stochastic demands from a *dynamic* or reoptimization perspective. The methods extend the rollout algorithm by implementing different base sequences (i.e. *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%).

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. Gendreau et al. (1996) summarize the literature on various SVRP's. Dror et al. (1989) indicate that optimal solution properties for the VRP do not hold for the SVRP. Further, Gendreau et al. (1995, 1999), Laporte et al. (2002), Ichoua et al. (2006) show that VRP's combining stochastic, integer, and in some cases dynamic elements (i.e. elements varying over time) call for complex solution methodologies.

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 (Marković et al., 2005), sludge, and recycled materials from banks, homes, and industrial plants.

Most VRPSD research assumes an *a priori* solution approach (Bertsimas, 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 *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 *a priori* routes that may prescribe returns to the depot before vehicle capacity is depleted (i.e. proactive returns).

This paper assumes a *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

* Corresponding author. Tel.: +1 512 245 4343; fax: +1 512 245 7771.
E-mail addresses: cn17@txstate.edu (C. Novoa), rhs2@lehigh.edu (R. Storer).

0377-2217/$ - see front matter © 2008 Elsevier B.V. All rights reserved.
doi:10.1016/j.ejor.2008.03.023

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 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, Psaraftis (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 Rinnooy Kan (1992) and Psaraftis (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 fix 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.

In a general context, efficient approximate DP methods for solving dynamic problems are discussed in Bertsekas (2001), Bertsekas and Tsitsiklis (1996), Sutton and Barto (1998), and Powell (2007). Examples of simulation-based methods for adaptively computing value function (i.e. cost-to-go) approximations are Kleywegt et al. (2002) for solving stochastic inventory routing problems, Topaloglu and Powell (2006) and Topaloglu and Powell (2007) for solving vehicle fleet management problems. Other contributions to neuro DP using least square approximations or a rollout approach are Van-Roy et al. (1998) for solving two-echelon inventory problems, Secomandi (2003) for solving a traveling salesman problem with stochastic travel times, and Bertsekas and Castanon (1999) for solving stochastic scheduling problems. Furthermore, de Farias and Van-Roy (2003) use linear programming (LP) to compute coefficients on the cost-to-go approximating function and Adelman (2004) uses dual-prices to approximate the cost-to-go on instances of the stochastic inventory routing problem.

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 $I = 1, 2, \ldots, n$ represents the customers' set. Distances between customers i and j, denoted by $d(i, j)$, 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 $D_i, i = 1, 2, \ldots, n$ be random variables that describe the demand for each customer and p_i be its probability distribution, $p_i(r) = Pr\{D_i = r\}, r = 0, 1, \ldots, R \leqslant Q$. 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

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

policy such that demand at each customer is satisfied and the travel cost is minimized. To satisfy this objective, the routing policy also can prescribe returns to the depot before vehicle capacity is depleted (i.e. proactive returns).

4. Problem formulation

The dynamic VRPSD can be formulated as a stochastic shortest path (SSP) problem. SSP's 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 $x_k = (l, q_l, r_1, \ldots, r_n)$ 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 $q_l \in 0, 1, \ldots, Q$, 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 ? 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 $Pr\{D_i = r'\} = 1$. 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 $O(nQR^n)$. The initial system state is $(0, Q, ?, ?, \ldots, ?)$ and the final system state is $(0, Q, 0, 0, \ldots, 0)$. 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 $\pi = \{\mu_0, \mu_1, \ldots, \mu_{N-1}\}$ is the policy or sequence of functions to optimize where μ_k is a function that associates a control or decision $u_k = \mu_k(x_k)$ to each state. 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 $U_k(x_k) = \{\{m \in \{1, \ldots, n\} | r_m \neq 0\} \cup 0\} \times \{a : a \in \{0, 1\}\}$. Controls u_k are represented as ordered pairs (m,a). The first element, $m \in \{1, \ldots, n\} | r_m \neq 0 \ \cup 0$, is 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 $x_k = (l, q_l, r_1, \ldots, r_m, \ldots, r_n) \in S$ and a control u_k occurs, the system state updates to $x_{k+1} = (m, q_m, r_1, \ldots, r'_m, \ldots, r_n)$ 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 customers demand probability distributions 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 π that minimizes the N-stage cost-to-go J^π_N or expected cost to termination from any initial state. The optimal N-stage cost-to-go from state x is $J^*_N(x) = \min_{\pi \in \Pi} J^\pi_N(x)$ where Π is the admissible-policies set. If J^*_N is known for all states, the optimal control u^*_k at each state results from finding the minimum in the following equation

$$u^*_k = \mu^*_k(x) = \arg \min_{u_k \in U_k(x_k)} g(x_k, u_k) + \sum_{x_{k+1} \in S} p_{x_k x_{k+1}}(u_k) J^*_N(x_{k+1}) | x_k = x \quad \forall x \in S. \tag{1}$$

Eq. (1) shows that at every 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 section 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) π for the problem is available and that its expected cost-to-go $\bar{J}(x_{k+1})$ can be computed efficiently. Under a one-stage RA (ORA), an approximate control $\tilde{u}_k$ for a particular state x_k reached in stage k results from approximating the true cost-to-go $J^*(x_{k+1})$ in Eq. (1) by $\bar{J}(x_{k+1})$. This approximation is done for all states $x_{k+1} \in S$ 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 π is done over the system's states. Given a properly selected policy π, rollout produces an improved policy $\bar{\pi}$. If applying RA to the dynamic VRPSD, the improvement is done over the states found in real-time execution and the improved policy $\bar{\pi}$ 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 $\widetilde{J}(x_{k+1})$ 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 ($m \geqslant 2$) and that they may achieve better performance at the expense of more computation

$$\tilde{\mu}_k(x_k) = \arg \min_{u_k \in U_k(x_k)} g(x_k, u_k) + \sum_{x_{k+1} \in S} p_{x_k x_{k+1}}(u_k) \widetilde{J}(x_{k+1}), \tag{2}$$

$$\widetilde{J}(x_{k+1}) = \min_{u_{k+1} \in U_{k+1}(x_{k+1})} g(x_{k+1}, u_{k+1}) + \sum_{x_{k+2} \in S} p_{x_{k+1} x_{k+2}}(u_{k+1}) \bar{J}(x_{k+2}). \tag{3}$$

One-step and two-step RA's require a base policy previously denoted as π to compute the approximate expected costs-to-go $\bar{J}(x_{k+1})$ and $\bar{J}(x_{k+2})$, respectively. For the dynamic VRPSD, we denote this sub-optimal initial policy as τ 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, τ 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 $\tau = (0, 1, \ldots, t-1, t, t+1, \ldots, n, 0)$ be the initial BS and assume that the vehicle is at customer l 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 l. The cyclic heuristic ζ produces an updated base sequence (UBS) $\tau_t^\zeta = (t, t+1, \ldots, n, 1, \ldots, t-1, 0)$ by following τ cyclically from t, skipping m and any fully served customers such as l, 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 l.

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 τ_t^ζ at state x_{k+2} when the vehicle is at customer t and has capacity q_t is denoted by $\mathbb{E}[L_{\tau_t^\zeta}|x_{k+2}, q_t]$. In a DP context this expected length is the expected cost-to-go of following policy τ_t^ζ from state x_{k+2} to the terminal state. In other words, $\mathbb{E}[L_{\tau_t^\zeta}|x_{k+2}, q_t]$ replaces $\bar{J}(x_{k+2})$ in Eq. (3). Our computational study use the equations derived by Secomandi (1998, 2001) to exactly compute $\mathbb{E}[L_{\tau_t^\zeta}|x_{k+2}, q_t]$. These equations are based on a DP backward recursion that takes $O(n\mathrm{RQ})$ 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, $\widetilde{J}_{k+2}(x) - \widetilde{J}_{k+2}(x') \approx J^*_{k+2}(x) - J^*_{k+2}(x')$ for any two states x and x'. The approximate expected cost-to-go of an UBS is denoted as $\widetilde{\mathbb{E}}[L_{\tau_t^\zeta} \mid x_{k+2}, q_t]$.

The computation of $\tilde{\mathbb{E}}[L_{\tau_t^\zeta} \mid x_{k+2}, q_t]$ under MCS considers all controls and states originated from state x_{k+2} to termination if following the updated base policy τ_t^ζ, 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 $J(x_{k+2}, s')$ 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 $\tilde{\mathbb{E}}[L_{\tau_t^\zeta} \mid x_{k+2}, q_t]$ is the average distance over the scenarios in S'. Thus, $\tilde{\mathbb{E}}[L_{\tau_t^\zeta} \mid x_{k+2}, q_t] = \sum_{s'=1}^{S'} \frac{J(x_{k+2}, s')}{S'}$. Therefore, to compute approximate controls for TRA under MCS, $\tilde{\mathbb{E}}[L_{\tau_t^\zeta}|x_{k+2}, q_t] = \sum_{s'=1}^{S'} \frac{J(x_{k+2}, s')}{S'}$ replaces $\bar{J}(x_{k+2})$ in Eq. (3). Similarly, to compute approximate controls for ORA under MCS, $\tilde{\mathbb{E}}[L_{\tau_t^\zeta}|x_{k+1}, q_t] = \sum_{s'=1}^{S'} \frac{J(x_{k+1}, s')}{S'}$ replaces $J^*(x_{k+1})$ 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 (a 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_k 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) τ 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 sub-section describes other procedures we used to find stochastic and deterministic BS's τ 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. (n2reopt) 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 $m = \lceil \frac{\sum_{i=1}^{n} \mathbb{E}[D_i]}{Q} \rceil$. 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 advantages of including two known parameters: distances and average customer demands.

Combining several base sequences. (combi) Bertsekas et al. (1997) and Bertsekas, 2000 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 J^* can be approximated as $\bar{J} = \min_{p=1,2,\ldots,P} \bar{J}_p$.

The three single BS's we combined are: (1) vrp, that results from solving the VRP exactly as explained above, (2)ORAapr, 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)TRAapr, 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 $\overline{D}$, 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) $\bar{f} = \frac{\sum_{i=1}^{n} \mathbb{E}[\overline{D}_i]}{mQ}$ 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, $\bar{f}' = \max(0, \bar{f} - 1)$ is an approximation for the expected number of route failures in a problem. The levels selected for $\bar{f}$ 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 $\bar{f}$ values, replacing $\mathbb{E}[\overline{D}_i]$ by $\overline{D}$, 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 UBS'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 1
Demand distributions for instances studied

n	Instances in Set1			Instances in Set2		
	Demand distribution	$\overline{D}$	Variance	Demand distribution	$\overline{D}$	Variance
5, 8	U[1, 3] U[2, 4] U[3, 5]	3	0.33	U[1, 7] U[2, 8] U[3, 9]	5	3.00
20, 30, 40, 60	U[1, 5] U[6, 10] U[11, 15]	8	1.33	U[1, 7] U[6, 12] U[11, 17]	9	3.00

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

Table 2
Relative effects for each rollout method

Rollout method	Instances Set1	Instances Set2	Rollout method	Instances Set1	Instances Set2
Perfinf	−13.80%	–	1s_ORAapr_r	−8.09%	−4.22%
2s_stostat_r	−10.42%	−8.04%	1s_n2_mc30	−5.07%	−3.80%
1s_stostat_r	−9.99%	−7.46%	1s_n2_r	−5.14%	−3.76%
2s_combi_r	−9.21%	−6.25%	1s_n2reopt_r	−4.44%	−3.71%
1s_combi_r	−9.37%	−5.89%	1s_n2_mc15	−4.96%	−3.55%
2s_n2_r	−6.84%	−5.31%	1s_n2_mc30pr2	−4.88%	−2.85%
1s_tsp_r	−6.01%	−4.62%	1s_n2_mc30pr1	−4.88%	−1.97%
1s_TRAapr_r	−8.68%	−4.58%	NN2int	0.00%	0.00%
1s_vrp_r	−8.48%	−4.49%			

accurate selection of the next customer to visit at each stage of the algorithm. Comparing the "original RA" to its two-step version (2s_n2_r), the improvement is about 1.6%. On the other hand, there is only a small improvement in solution quality if starting RA from an exact TSP solution instead of a heuristic one (0.87% and 0.86%) and there is no practical change if recomputing the BS at each RA stage. The most encouraging result is that there is no deterioration in solution quality if the length of the UBS's is computed using MCS instead exact computation. The percentages of increase in routing cost are less than 0.07% if comparing 1s_n2_MC30 to 1s_n2_r. If pruning rules to reduce the control set at each stage and state are included, the deterioration in solution quality for 1s_n2_mc30pr1 compared to the "original RA" is only 0.26% in Set1 and 1.80% in Set2.

The resulting percentages of reduction in routing costs are lower for Set2. This result suggests that the benefits of rollout methods may diminish as variability in demands increases. Table 2 also shows that in Set1 our best RA is only 3.38% away of the estimated perfect information lower bound.

http://www.txstate.edu/~cn17/results.htm and permanent repository http://ecommons.txstate.edu/engtfacp/1 contain routing costs tabulated by rollout method, number of customers, and vehicle capacity for both sets of instances. The site also contains tables displaying the percentage of improvement (+) or deterioration (−) in routing costs for each RA if compared to the "original RA" (1s_n2_r). These detailed results suggest that the advantage for TRA's over ORA's is present in instances with large number of customers ($n > 30$) where there are more iterations (stages) to apply the refined two-step minimization procedure. Further, detailed results show the superior quality of rollout methods using stochastic base policies for almost all combinations of number of customers and vehicle capacities.

The web links above also have detailed results for CPU seconds for generating complete trajectories with the RA's. All computational time results discussed in this paper are for 60-customer problems. CPU time is less than 2.352 (5.132) seconds for ORA's with single (combined) BS's. CPU time for TRA's that use single BS's is less that 175.31 seconds. These low CPU times suggest that ORA's and TRA's are implementable in real-time. Besides, ORA's that use MCS such as 1s_n2_mc30 have CPU times of about 0.5 seconds that correspond to a reduction in CPU times of about 60% if compared to the "original RA" (CPU times 1.31 seconds). If implementing pruning rules, there is an additional reduction in computational time of about 40%.

Table 3 presents mean, standard deviation (SD), and 95% lower and upper confidence intervals (CI) for the relative differences between the best rollout procedure from the ANOVA (i.e. 2s_stostat_r) and the perfect information solutions from SYMPHONY for instances in Set1. For conciseness, only confidence intervals for problems with 30, 40, and 60 customers are displayed. The overall mean for the relative difference considering instances ranging from 5 to 60 customers is 3.84% and the corresponding CI is (3.56%, 4.11%). These small differences suggest that the best RA found is close to the optimal solutions. We recall that optimal solution computation to dynamic VRPSD's is an intractable task for instances with more than 20 customers.

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

	Number of customers and vehicle capacity					
	30		40		60	
	137	87	183	116	274	175
Mean	3.05%	3.32%	2.79%	5.03%	2.69%	7.94%
SD	2.14%	2.18%	1.39%	3.14%	1.96%	1.94%
95% LCI	2.57%	2.83%	2.48%	4.33%	2.26%	7.47%
95% UCI	3.53%	3.80%	3.10%	5.73%	3.13%	8.42%

To further compare the RA that uses MCS with the "original RA", the two methods are contrasted on 20 instances with 100 and 150 customers. The filling coefficient is the same as in instances in Set1 and RA is performed for each instance 100 times. Once again, there is no deterioration in routing costs if using MCS with small samples of 30 demand scenarios instead of exact computation of the expected length of UBS's. Fig. 1 shows the computational times. In the "original RA", the time for computing the cost of an UBS is $O(nRQ)$ while in the RA that uses MCS this time is $O(nS')$. Thus, the use of MCS is advantageous if S' can take a value less than RQ. Furthermore, MCS is easy to implement in problems where some or all customers have continuous demand distributions since it only needs the demand realizations to approximate the cost-to-go values.

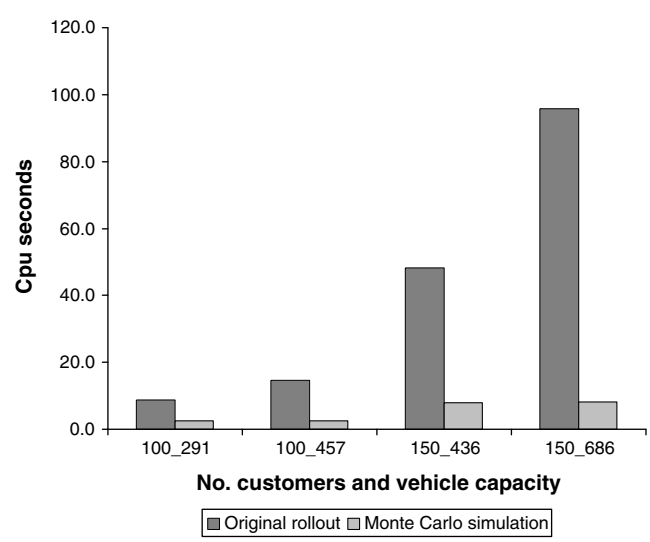

Fig. 1. Average CPU times "original rollout" vs. MCS.

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

Adelman, D., 2004. A price-directed approach to stochastic inventory routing. Operations Research 52 (4), 499–504.

Bastian, C., Rinnooy Kan, A.H.G., 1992. The stochastic vehicle routing problem revisited. European Journal of Operational Research 56, 407–412.

Bertsekas, D.P., 2000. Dynamic Programming and Optimal Control, vol. 1. Athena Scientific, Belmont, MA.

Bertsekas, D.P., 2001. Dynamic Programming and Optimal Control, vol. 2. Athena Scientific, Belmont, MA.

Bertsekas, D.P., Castanon, D.A., 1999. Rollout algorithms for stochastic scheduling problems. Journal of Heuristics, 89–108.

Bertsekas, D.P., Tsitsiklis, J.N., 1996. Neuro-Dynamic Programming. Athena Scientific, Belmont, MA.

Bertsekas, D.P., Tsitsiklis, J.N., Wu, C., 1997. Rollout algorithms for combinatorial optimization. Journal of Heuristics 3, 245–262.

Bertsimas, D.J., 1992. A vehicle routing problem with stochastic demand. Operations Research 40 (3), 574–585.

Bertsimas, D.J., Chervi, P., Peterson, M., 1995. Computational approaches to stochastic vehicle routing problems. Transportation Science 29 (4), 342–352.

Bertsimas, D.J., Simchi-Levi, D., 1996. A new generation of vehicle routing research: Robust algorithms, addressing uncertainty. Operations Research 44 (2), 286–304.

Bianchi, L., Birattari, M., Chiarandini, M., Manfrin, M., Mastrolilli, M., Paquete, L., Rossi-Doria, O., Schiavinotto, T., 2004. Metaheuristics for the vehicle routing problem with stochastic demands. Parallel problem solving from nature PPSN VIII. Lecture Notes in Computer Science. Springer, Berlin, Germany. pp. 1–10.

Chepuri, K., Homem-De-Mello, T., 2005. Solving the vehicle routing problem with stochastic demands using the cross entropy method. Annals of Operations Research 134, 153–181.

de Farias, D.P., Van-Roy, B., 2003. The linear programming approach to approximate dynamic programming. Operations Research 51 (6), 850–865.

Dessouky, M.M., Ordoñez, F., Jia, H., Shen, Z., 2005. Rapid Distribution of Medical Supplies. In: Hall, R.W. (Ed.), Patient Flow: Reducing Delay Management in Health Care Systems. Springer, USA, pp. 309–339.

Dror, M., 1993. Modeling vehicle routing with uncertain demands as a stochastic program: Properties of the corresponding solution. European Journal of Operational Research 64, 432–441.

Dror, M., Ball, M., Golden, B., 1985. A computational comparison of algorithms for the inventory routing problem. Annals of Operations Research 4, 3–23.

Dror, M., Laporte, G., Trudeau, P., 1989. Vehicle routing with stochastic demands: Properties and solution frameworks. Transportation Science 23 (3), 166–176.

Erera, A.L., Savelsbergh, M., Uyar, E., 2007. Fixed routes with backup vehicles for stochastic vehicle routing problems with time constraints. Technical Report, The Supply Chain and Logistics Institute, School of Industrial and Systems Engineering, Georgia Institute of Technology.

Gendreau, M., Guertin, F., Potvin, J., Taillard, É., 1999. Parallel tabu search for real-time vehicle routing and dispatching. Transportation Science 33 (4), 381–390.

Gendreau, M., Laporte, G., Séguin, R., 1995. An exact algorithm for the vehicle routing problem with stochastic demands and customers. Transportation Science 29 (2), 143–155.

Gendreau, M., Laporte, G., Séguin, R., 1996. Invited review: Stochastic vehicle routing. European Journal of Operational Research 88, 3–12.

Hjorring, C., Holt, J., 1999. New optimality cuts for a single-vehicle stochastic routing problem. Annals of Operations Research 86, 569–584.

Ichoua, S., Gendreau, M., Potvin, J., 2006. Exploiting knowledge about future demands for real-time vehicle dispatching. Transportation Science 40 (2), 211–225.

Kleywegt, A.J., Nori, V.S., Savelsbergh, M.W.P., 2002. The stochastic inventory routing problem with direct deliveries. Transportation Science 36 (1), 94–118.

Ladányi, L., Ralphs, T.K., Trotter, L.E.J., 2001. Branch cut and price: Sequential and parallel. In: Naddef, D., Juenger, M. (Eds.), Computational Combinatorial Optimization. Springer, Berlin, pp. 223–260.

Laporte, G., Louveaux, F.V., Mercure, H., 1989. Models and exact solutions for a class of stochastic location-routing problems. European Journal of Operational Research 39, 71–78.

Laporte, G., Louveaux, F.V., Van-Hamme, L., 2002. An integer L-Shaped algorithm for the capacitated vehicle routing problem with stochastic demands. Operations Research 50 (3), 415–423.

Marković, H., Ćavar, I., Carić, T., 2005. Using data mining to forecast uncertain demands in stochastic vehicle routing problem. In: 13th International Symposium on Elecronics in Transport (ISEP), Slovenia, pp. 1–6.

Nemhauser, G.L., Wolsey, L.A., 1999. Integer and Combinatorial Optimization. Wiley-Interscience, New York, NY.

Novoa, C., 2005. Static and dynamic approaches for solving the vehicle routing problem with stochastic demands. Ph.D. Thesis, Department of Industrial and Systems Engineering, Lehigh University, Bethlehem, PA.

Novoa, C., Berger, R., Linderoth, J., Storer, R., 2006. A set-partitioning-based model for the stochastic vehicle routing problem. Technical Report, 06T-008, Industrial and Systems Engineering, Lehigh University.

Powell, W.B., 2007. Approximate dynamic programming: solving the curses of dimensionality. Wiley Series in Probability and Statistics. John Wiley & Sons, New York, NY.

Powell, W.B., Jaillet, P., Odoni, A., 1995. Stochastic and dynamic networks and routing. In: Ball, M.O., Magnanti, T.L., Monma, C.L., Nemhauser, G.L. (Eds.), Network routing, Handbooks in Operations Research and Management Science, 8. Elsevier, Amsterdam, The Netherlands, pp. 141–295.

Psaraftis, H., 1995. Dynamic vehicle routing: Status and prospects. Annals of Operations Research 61, 143–164.

Ralphs, T.K., 2006. SYMPHONY User's Manual, Available at <http://branchandcut.org/SYMPHONY> and at <https://projects.coin-or.org/SYMPHONY>.

Ralphs, T.K., Kopman, L., Pulleyblank, W.R., Trotter, L.E.J., 2003. On the capacitated vehicle routing problem. Mathematical Programming 94 (2–3), 343–359.

Savelsbergh, M., Goetschalckx, M., 1995. A comparison of the efficiency of fixed versus variable vehicle routes. Journal of Business Logistics 16, 163–188.

Secomandi, N., 1998. Exact and heuristic dynamic programming algorithms for the vehicle routing problem with stochastic demands. Ph.D. Thesis, Department of Decision and Information Sciences, University of Houston, Houston, TX.

Secomandi, N., 2000. Comparing neuro-dynamic programming algorithms for the vehicle routing problem with stochastic demands. Computers and Operations Research 27, 1201–1225.

Secomandi, N., 2001. A rollout policy for the vehicle routing problem with stochastic demands. Operations Research 49 (5), 796–802.

Secomandi, N., 2003. Analysis of a rollout approach to sequencing problems with stochastic routing applications. Journal of Heuristics 9, 321–352.

Secomandi, N., Margot, F., in review. Reoptimization approaches for the vehicle problem with stochastic demands, Operations Research

Sutton, R., Barto, A.S., 1998. Reinforcement Learning. MIT Press, Cambridge, MA.

Teodorovic, D., Pavkovic, G., 1992. A simulated annealing technique approach to the vehicle routing problem in the case of stochastic demand. Transportation Planning and Technology 16, 261–273.

Topaloglu, H., Powell, W., 2006. Dynamic programming approximations for stochastic, time-staged integer multicommodity flow problems. INFORMS Journal on Computing 18 (1), 31–42.

Topaloglu, H., Powell, W., 2007. Sensitivity analysis of a dynamic fleet management model using approximate dynamic programming. Operations Research 55 (2), 319–350.

Toth, P., Vigo, D., 2002. Models, relaxations and exact approaches for the capacitated vehicle routing problem. Discrete Applied Mathematics 123, 487–512.

Van-Roy, B., Bertsekas, D.P., Lee, Y., Tsitsiklis, J.N., 1998. A neuro-dynamic approach to retailer inventory management. Technical Report, R-96, Laboratory for Information and Decision Systems, Massachusetts Institute of Technology.

Yang, W., Mathur, K., Ballou, R.H., 2000. Stochastic vehicle routing problem with restocking. Transportation Science 34 (1), 99–112.

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