TRANSPORTATION SCIENCE
Vol. 43, No. 1, February 2009, pp. 27–42
ISSN 0041-1655 | EISSN 1526-5447 | 09 | 4301 | 0027

informs

DOI 10.1287/trsc.1080.0251

Hybrid Adaptive Predictive Control for a Dynamic Pickup and Delivery Problem

Cristián E. Cortés
Civil Engineering Department, Universidad de Chile, Avenue Blanco Encalada 2002, Santiago, Chile, ccortes@ing.uchile.cl

Doris Sáez, Alfredo Núñez, Diego Muñoz-Carpintero
Electrical Engineering Department, Universidad de Chile, Avenue Tupper 2007, Santiago, Chile
{dsaez@ing.uchile.cl, alfnunez@ing.uchile.cl, dimunoz@ing.uchile.cl}

This paper presents a hybrid adaptive predictive control approach that includes future information in real-time routing decisions in the context of a dynamic pickup and delivery problem (DPDP). We recognize in this research that when the problem is dynamic, an additional stochastic effect has to be considered within the analytical expression of the objective function for vehicle scheduling and routing, which is the extra cost associated with potential rerouting arising from unknown requests in the future. The major contributions of this paper are: first, the development of a formal adaptive predictive control framework to model the DPDP, and second, the development and coding of an ad hoc particle swarm optimization (PSO) algorithm to efficiently solve it. Predictive state-space formulations are written on the relevant variables (vehicle load and departure time at stops) for the DPDP. Next, an objective function is stated to solve the real-time system when predicting one and two steps ahead in time. A problem-specific PSO algorithm is proposed and coded according to the dynamic formulation. Then, the PSO method is used to validate this approach through a simulated numerical example.

Key words: pickup-and-delivery system; dynamic vehicle routing problem; hybrid predictive control; particle swarm optimization
History: Received: March 2005; revisions received: April 2006, July 2007, October 2007; accepted: October 2008.

1. Introduction

One of the most studied problems in the literature on logistics is the well-known pickup and delivery problem (with or without time windows), which involves the satisfaction of a set of transportation requests by a vehicle fleet initially located at several depots (Desrosiers, Soumis, and Dumas 1986; Savelsbergh and Sol 1995). A transportation request consists of picking up a certain number of customers at a predetermined pickup location during a departure time interval and taking them to a predetermined delivery location within an arrival time interval. Loading and unloading times are incurred at each vehicle stop. The problem can be generalized to the dynamic case, in which a subset of the requests is not known in advance and dispatch decisions have to be made in real time. The dynamic pickup and delivery problem (DPDP) has become of great interest in the last decade, mainly due to the fast growth in communication and information technologies, as well as the current interest in real-time dispatching and routing. The problem can be characterized as a real-time routed transit and has been treated mostly heuristically by many authors under different policy schemes in the past (as representative references see Psaraftis 1988; Madsen, Raven, and Rygaard 1995; Bertsimas and Van Ryzin 1991, 1993a, b; Malandraki and Daskin 1992; Dial 1995; Gendreau et al. 1999).

In this scenario, if the objective were to transport passengers, inefficient routing decisions could greatly affect the performance of the system as perceived by the users, resulting in a poor level of service, low demand, and insufficient productivity. One of the major issues for improving efficiency is the correct definition of a decision objective function for dispatching, including total travel and waiting times for users as well as a performance measure for vehicles. However, when the problem is dynamic, an additional stochastic effect has to be considered when computing an analytical expression for any decision objective function (whether it affects the user or the operator). In other words, we recognize that current dispatch actions taken in real time can be affected by potential rerouting decisions decided in the future, affecting most customers already in the system, those waiting, as well as those traveling.

The importance of this issue has been underestimated in the dynamic vehicle-routing literature. One assumption behind most of the proposed scheduling-routing rules is that travel and waiting times experienced by customers are considered fixed in the objective function expressions, independent of future

reroutings. In other words, as stated by Spivey and Powell (2004), the complexity of real-time routing schemes have generally restricted research to myopic models (for example, see Wilson and Weissberg 1976; Wilson and Colvin 1977; Psaraftis 1980, 1988; Madsen, Raven, and Rygaard 1995; Gendreau et al. 1999; Swihart and Papastavrou 1999).

However, some recent studies in the field of vehicle routing and dispatching have tried to exploit information about future events to improve decision making (Ichoua, Gendreau, and Potvin 2006; Spivey and Powell 2004). Solution approaches found in this line of research are diverse, with formulations based upon dynamic network models (Powell 1988), dynamic and stochastic programming schemes (Godfrey and Powell 2002, Topaloglu and Powell 2005), etc. Cortés and Jayakrishnan (2004) propose a scheme for making better dynamic decisions by estimating the effective cost of a real-time request insertion based upon future information. The authors realized that the problem conceptually fits within a stochastic predictive control framework, although they did not enter into the control formulation details.

In this paper, we formalize the approach suggested by Cortés and Jayakrishnan (2004) by developing a consistent framework based upon predictive control theory for optimizing the performance of a DPDP that is mainly oriented to passenger movements. The formulation turned out to be highly nonlinear, with a combination of integer/discrete and continuous variables to properly describe the future behavior of the routing process. Hence, an efficient ad hoc algorithm from the computational intelligence literature (particle swarm optimization, PSO) is developed to solve the proposed formulation and to test the benefits of incorporating demand patterns' prediction in current routing decisions under different scenarios.

Unlike others' nonmyopic dynamic vehicle-routing approaches, this formulation is based on state-space variables. The system state is defined in terms of departure time and vehicle loads (stochastic state-space variables), the system inputs (control actions) are routing decisions, the system outputs are effective departure time to stops, and the demand requests are modeled as disturbances. We use a discrete model with variable step size equal to the time between successive calls (events). In order to include future and unknown demand in the current decision, we solve an objective function incorporating the predictive effect via probabilities computed from historical data regarding typical demand patterns.

In summary, we highlight two major contributions of this paper: the development of a hybrid predictive control framework to model the DPDP, and the development of an ad hoc PSO algorithm to efficiently solve the proposed formulation for real-size problems. This line of research represents an innovative attempt to develop control-based algorithms for modeling and solving dynamic transportation problems in a realistic context. Specifically, in this application we have developed a new version of the PSO algorithm (originally conceived for solving continuous problems) in order to add integer variables into the solution and solve the DPDP efficiently. It is important to mention that the proposed algorithm was conceived from the hybrid predictive control scheme (HPC) to deal with the DPDP developed here, and depends exclusively on the structure of the HPC formulation, as shown in §3.4.

The structure of the paper is as follows. In the next section, the relevant background on dynamic vehicle routing is presented. In §3, the dynamic pickup and delivery problem is described in context, and is formulated under an adaptive-predictive control scheme. Thus, the specific state-space formulation for the problem is developed, the associated dispatch objective function is shown, and the solution algorithms are developed to solve the proposed hybrid predictive control scheme. In §4, a numerical example is presented to show the benefits of applying predictive control at least two steps ahead in time. Finally, in §5, analysis, comments, and further research lines are presented.

2. The Dynamic Vehicle-Routing Problem: Approaches and Solution Methods

In this section, the objective is to provide a review on the most relevant dynamic vehicle-routing problem (DVRP) variants, intensely studied by different authors with different applications over the past 15 to 20 years. DVRPs are characterized by routes that are constructed as unknown requests enter the system in real time. Thus, DVRPs are formulated by assuming that inputs may change or have to be updated during the execution of the solution algorithm. Larsen (2000) develops a nice characterization of the different dynamic problems, starting again from the TSP (traveling salesman problem), which yields the dynamic TSP (DTSP) introduced by Psaraftis (1988). This work motivates the development of the dynamic traveling repairman problem (DTRP), introduced by Bertsimas and Van Ryzin (1991) and next extended by Bertsimas and Van Ryzin (1993a, b). Lately, Swihart and Papastavrou (1999), and Thomas and White (2004) formulate and solve two variants of the DTRP.

The dynamic pickup and delivery problem (DPDP) that is designed to solve the dynamic dial-a-ride Problem (DDRP) has been intensely studied in the last 20 years (Psaraftis 1980, 1988; Gendreau et al. 1999; Savelsbergh and Sol 1995). The final output of

such a problem is a set of routes for all vehicles, which dynamically change over time. With regard to real applications, Madsen, Raven, and Rygaard (1995) adapt the insertion heuristics by Jaw et al. (1986) and solve a real-life problem for moving elderly and handicapped people in Copenhagen, whereas Dial (1995) proposes a modern approach to the many-to-few dial-a-ride transit operation ADART (autonomous dial-a-ride transit), currently implemented in Corpus Christi, TX, USA.

With regard to solution methods to handle different DVRPs, Gendreau et al. (1999) modify the tabu search heuristics to solve the DVRP with soft time windows motivated from courier service applications, which is implemented in a parallel platform. Tabu search methods are derived in more sophisticated versions, such as granular tabu search (Toth and Vigo 2003) and adaptive memory-based tabu search (Tarantilis 2005). Tighe, Smith, and Lyons (2004) propose a priority-based solver that considers subproblems of real-time vehicle routing in order to obtain an optimal solution in less time by using fuzzy decisions.

Evolutionary computation techniques have also been proposed to handle such problems. Specifically, genetic algorithms (GA) are applied for various VRP, considering different chromosome representation and genetic operators according to the particular problem (Skrlec, Filipec, and Krajcar 1997 for the single vehicle capacity VRP; Haghani and Jung 2005 for the multivehicle DVRP with time-dependent travel time and soft time windows). Zhu et al. (2006) propose an adapted partial swarm optimization (PSO) algorithm to solve a static VRP with time windows.

Jih and Yung-Jen (1999) and Osman, Abo-Sinna, and Mousa (2005) present a successful comparison of the GA against dynamic programming (DP) in terms of computation time. The former solve the DVRP with time windows and capacity constraints, while the latter solve a multiobjective VRP. Additionally, ant colony methods, as new metaheuristics inspired by the behavior of real ant colonies, have been applied to DVRP (Montemanni et al. 2005, Dréo et al. 2006).

In dynamic as well as stochastic problems, two approaches (myopic and nonmyopic) are found in the literature; these differ based on how the future information is considered in the generation of real-time decisions. The myopic research line does not explicitly consider the expected future information of the system to improve the current solution (as shown the aforementioned papers), whereas the nonmyopic option considers a mechanism to update information regarding the future to make better decisions at present. Such future data may be imprecise or unknown, and therefore developing consistent information update tools are essential for getting good predictions and making better real-time dispatch decisions.

Powell and his team have worked for many years in a nonmyopic line of research that incorporates explicit stochastic and dynamic algorithms with the current information and probabilities of future events to produce more efficient solutions than those obtained through myopic deterministic strategies. They solve the problem of dynamically assigning drivers to loads that arise randomly over time, a scenario motivated from long-haul truckload trucking applications.

Powell (1988) first considers the potential advantages of relocating vehicles in anticipation of future demands. He writes a two-stage stochastic program including a recourse function representing the future cost. Powell, Jaillet, and Odoni (1995) studies a mixed assignment and fleet management problem, modeled as a dynamic-stochastic network, which they solve with a network simplex algorithm on a rolling horizon basis. Spivey and Powell (2004) propose a very general class of dynamic assignment models, and propose an adaptive, nonmyopic algorithm that iteratively solves sequences of assignment problems. Topaloglu and Powell (2005) propose a distributed solution approach to a certain class of dynamic resource allocation problems. Topolaglu and Powell (2007) show how to coordinate the decisions on pricing and fleet management of a freight carrier. The objective is to find the set of prices that maximize the total expected profit over the time horizon, considering random loads (whose distributions depend on the prices) and the cost associated with repositioning the empty vehicles. The authors present a tractable method to obtain sample path-based directional derivatives of the objective function with respect to the prices to search for a good set of prices. Numerical experiments show that their approach yields high-quality solutions.

In his thesis, Larsen (2000) investigates the use of future information by relocating empty vehicles in anticipation of future demands. Ichoua, Gendreau, and Potvin (2006) develop a strategy based on probabilistic knowledge about future request arrivals to better manage a fleet of vehicles for real-time vehicle dispatching. This problem is solved using a parallel tabu search technique.

Figliozzi, Mahmassani, and Jaillet (2007) introduce the VRP in a competitive environment (VRPCE) as an extension of the traveling salesman problem with profits (TSPP) to a dynamic competitive auction environment. The authors develop a dynamic model to compute optimal price expressions for the VRPCE considering both, the expected change due to altering the current fleet assignment scheme and the opportunity costs on future profits created by servicing a new contract. Analytically, they propose an approximate solution approach, using a finite look-ahead horizon based on backward induction, which is compared against a static approach with no look ahead.

A simulation-based approach to evaluate service costs is proposed, which not only outperforms a static pricing, but it also price discriminates by market arrival rate, time windows, and shipment features.

The analysis of these nonmyopic models that incorporate future information is crucial for our purposes, because this paper formalizes the use of future information in dynamic vehicle-routing problems through a hybrid predictive control scheme. In the next section, this scheme is presented in detail.

3. Hybrid Predictive Control Approach to Solve the Dynamic Pickup and Delivery Problem (DPDP)

In the context of control theory, the notion of hybrid systems arises when the problem conditions are characterized by both continuous and discrete/integer variables. In the last decade, hybrid systems have been studied more intensely by researchers from several study areas, such as computer science and automatic control. A systematic methodology for a general control design of hybrid systems has been developed by Bemporad and Morari (1999) and Bemporad, Borrelli, and Morari (2002). Specifically, a hybrid system can be expressed as a nonlinear state-space model given by

$$\begin{aligned} x(k+1) &= f(x(k), u(k)), \\ y(k) &= g(x(k)), \end{aligned} \tag{1}$$

where $x(k)$ are the continuous and/or discrete (integer) state-space variables, $u(k)$ are the continuous and/or discrete input or manipulated variables, $y(k)$ define the continuous and/or discrete system outputs, and f, g are nonlinear functions. In general, a hybrid predictive control design minimizes the following generic objective function:

$$\min_{u(k)} J\big(u(k), \ldots, u(k+N-1), \hat{x}(k+1), \ldots, \hat{x}(k+N), \hat{y}(k+1), \ldots, \hat{y}(k+N)\big), \tag{2}$$

where J is an objective function; k is the current time; N the prediction horizon; $\hat{x}(k+t)$ and $\hat{y}(k+t)$ are, respectively, the expected state-space vector and the expected system output at instant $k+t$; and $\{u(k), \ldots, u(k+N-1)\}$ represents the control sequence, which corresponds to the vector of optimization variables. Once expression (2) is optimized, only the first element of the control vector $u(k)$ is used to update the system conditions, based upon the receding-horizon methodology.

Next, we characterize the dynamic pickup and delivery problem (DPDP) as a hybrid system to show the advantages of this approach when predicting future conditions under unknown dynamic demand.

3.1. Problem Statement

In this paper, we formulate a generic DPDP as a hybrid predictive control problem, following the theory explained above, recognizing that the dynamic routing process behind the real-time dispatch decisions includes discrete/integer and continuous state-space variables, as well as discrete input variables.

Conceptually, the hybrid predictive control framework used to model the DPDP incorporates stochasticity into the routing dispatch rules by considering the impact of future reassignments on the performance of already-scheduled customers. The stochastic prediction allows the dispatcher to incorporate a more realistic measure of effective travel (waiting) time experienced by the users into the decision objective function expression (see §3.3 for details). The focus here is on passenger routing; however, the scheme could be generalized to freight, too.

Let us assume an influence area A, with a service network of length D in distance units. Suppose we have a set of vehicles V of size F. The fleet of vehicles is currently in operation traveling within the area according to predefined routing rules. The demand for service is unknown and comes up in real-time (assume a rate μ of calls per time unit). Quick routing and scheduling decisions are needed to handle such demand with the available vehicles. At any time k, we assume that each vehicle $j \in V$ has been assigned a control action that includes pickups and deliveries, and can be represented by a function $u_j(k) = S_j(k) = [s_j^1(k) \cdots s_j^i(k) \cdots s_j^{w_j(k)}(k)]^T$, in which the ith element of the sequence represents a specific ith stop along vehicle j's route, and $w_j(k)$ is the number of stops.

The complete control action or manipulated variable $u(k) = S(k)$, as the dispatching decision, can be represented by the set of sequences assigned to every vehicle at instant k. Analytically,

$$u(k) = S(k) = \{S_1(k), \ldots, S_j(k), \ldots, S_F(k)\}. \tag{3}$$

Vehicles will travel according to the predefined sequence vector $S(k-1)$ while no new calls are received. When a new service request (call) comes in, the controller or central dispatcher calculates the control sequence in the next step $S(k)$ for the fleet of vehicles, including the stops requested by the new customer. Then, each sequence $S_j(k)$ remains fixed during the whole time interval $(k, k+1)$, unless a vehicle reaches a predefined pickup or delivery stop during such an interval, in which case its sequence will decrease in size to show that the scheduled task has been accomplished. Thus, in this scheme it is necessary to formulate the problem in terms of a variable time step (triggered by events), which represents the time interval between two consecutive requests, that is to say, the predictive controller makes a routing decision when a new call enters the system.

The state of the system at instant k is associated with the previous sequences $S(k-1)$ (the new call is not considered). In the DPDP problem, the state-space variables include the clock time of departure $T_j^i(k)$ and the vehicle load $L_j^i(k)$, after vehicle j leaves stop i, both computed at instant k. At this point, let us define, for each vehicle $j \in V$, the load and departure-time vectors as follows:

$$L_j(k) = \begin{bmatrix} L_j^0(k) & L_j^1(k) & \cdots & L_j^{w_j(k-1)}(k) \end{bmatrix}^T_{(w_j(k-1)+1)\times 1} \quad (4)$$

$$T_j(k) = \begin{bmatrix} T_j^0(k) & T_j^1(k) & \cdots & T_j^{w_j(k-1)}(k) \end{bmatrix}^T_{(w_j(k-1)+1)\times 1} \quad (5)$$

Thus, the set of state-space variables for the entire system at instant k can be written as $x(k) = \{L(k), T(k)\}$, where $L(k)$ and $T(k)$ represent the set of load and departure-time vectors, respectively; that is, $L(k) = \{L_1(k), \ldots, L_j(k), \ldots, L_F(k)\}$ and $T(k) = \{T_1(k), \ldots, T_j(k), \ldots, T_F(k)\}$. The output set $y(k)$ is represented by the vector of observed departure times of vehicles at stops, $T(k)$.

In summary, under this hybrid predictive control approach, Equation (1) can be written for the DPDP by recognizing the dependence of the routing process on the following associated variables: $x(k) = \{L(k), T(k)\}$, $y(k) = T(k)$, $u(k) = S(k)$. The hybrid predictive control scheme proposed in this paper can be represented by a generic flow chart shown in Figure 1.

In the figure, the predictive controller is represented by the dispatcher and the routing is solved by minimizing an objective function that considers the user cost based upon total travel and waiting time spent by the users, and a component as a proxy of the operational cost, as explained in §3.3. The routing process is defined by the online dispatching decision ($S(k-1)$) under uncertain demand (μ), which results in observed departure times ($y(k)$). An adaptive mechanism is also added in the figure due to the variant parameters of the system and dimension of the departure-time and vehicle load models ($\{L(k), T(k)\}$).

Next, this model is analytically described, highlighting the treatment of both the departure-time and load components.

3.2. Predictive Dynamic Model

This research considers a predictive dynamic model based on state-space representation for both the vehicle load and the departure time at stops (as a function of segment travel times). Both the clock time of departure $T_j^i(k)$ and the vehicle load $L_j^i(k)$ are stochastic variables, because they depend on the evolution of the system affected by uncertain demand. Therefore, and in order to work with deterministic values, reasonable estimations of the load and departure-time vectors have to be obtained. The prediction of when a new request will occur is given by the expected value of the state-space vector for vehicle j, $\hat{x}_j(k+1)$. Analytically,

$$\hat{x}_j(k+1) = \begin{bmatrix} E\{L_j(k+1)/k\} \\ E\{T_j(k+1)/k\} \end{bmatrix} = \begin{bmatrix} \hat{L}_j(k+1) \\ \hat{T}_j(k+1) \end{bmatrix} = \begin{bmatrix} f_L(L_j(k), S_j(k)) \\ f_T(T_j(k), S_j(k)) \end{bmatrix} \quad \forall j = 1, \ldots, F \quad (6)$$

where the functions f_L and f_T are the state-space models to be defined in Equations (8) and (9).

The dynamic system for a specific vehicle j can be graphically represented by its sequence $S_j(k)$ computed at certain instant k, and the associated expected values of the state-space variables in the next instant $k+1$, is shown in Figure 2.

The components of $S_j(k)$ are

$$S_j(k) = \begin{bmatrix} r_j^1(k) & 1-r_j^1(k) & \Gamma_j^1(k) & \text{label}_j^1(k) \\ \vdots & \vdots & \vdots & \vdots \\ r_j^i(k) & 1-r_j^i(k) & \Gamma_j^i(k) & \text{label}_j^i(k) \\ \vdots & \vdots & \vdots & \vdots \\ r_j^{w_j(k)}(k) & 1-r_j^{w_j(k)}(k) & \Gamma_j^{w_j(k)}(k) & \text{label}_j^{w_j(k)}(k) \end{bmatrix}, \quad (7)$$

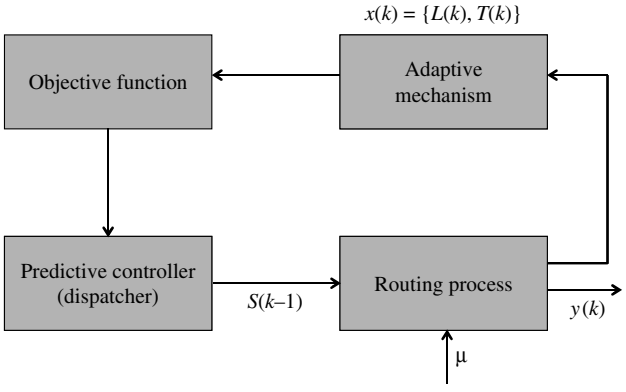


Figure 1 Overall Block Diagram of a Hybrid Predictive Approach for DPDP

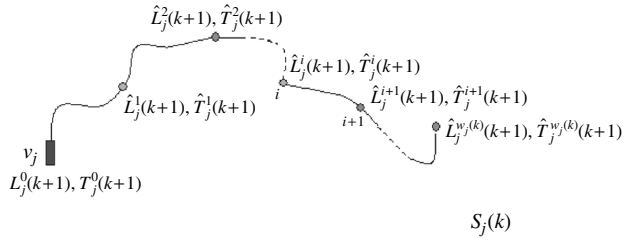


Figure 2 Typical Vehicle Route at Time k and State-Space Variables Estimated at $k+1$

where $r_j^i(k)$ is a binary variable defined as follows:

$$r_j^i(k) = \begin{cases} 1 & \text{if stop } i \text{ belonging to } S_j(k) \text{ is a pickup,} \\ 0 & \text{otherwise.} \end{cases}$$

The first and second columns represent a pair identifying if stop i is either a pickup [1 0] or a delivery [0 1], respectively.

The third column of the $S_j(k)$ matrix represents the external travel time function, where Γ_j^i is the total travel time between points $i-1$ and i plus the transfer operation delay at node i.

For simulation purposes, we assume that vehicles move at constant speed, and therefore their position can be estimated at any moment. The last column, label_j^i, keeps the passenger identifier, which is needed to check the feasibility of the sequence in terms of precedence (the pickup must occur before the delivery of the same client). Finally, the size of the sequence matrix in Equation (3.2) is $w_j(k) \times 4$, comprising $w_j(k-1)$ rows for the previously scheduled stops, and two rows with the information (pickup and delivery locations) of the last call.

Thus, the vehicle load behavior is obtained using the following state-space model:

$$\hat{L}_j(k+1) = f_L(L_j(k), S_j(k)) = A_L\, L_j(k) + B_L(S_j(k)), \quad (8)$$

where the corresponding matrices in (8) are

$$B_L(S_j(k)) = B_L^2 \cdot (S_j(k) \cdot B_L^1),$$

$$A_L = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 1 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 1 & 0 & \cdots & 0 \end{bmatrix}_{(w_j(k)+1)\times(w_j(k-1)+1)},$$

$$B_L^1 = \begin{bmatrix} 1 \\ -1 \\ 0 \\ 0 \end{bmatrix}, \quad B_L^2 = \begin{bmatrix} 0 & 0 & 0 & \cdots & 0 & 0 \\ 1 & 0 & 0 & \cdots & 0 & 0 \\ 1 & 1 & 0 & \cdots & 0 & 0 \\ 1 & 1 & 1 & \ddots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \ddots & 0 & 0 \\ \vdots & \vdots & \vdots & & 1 & 0 \\ 1 & 1 & 1 & \cdots & 1 & 1 \end{bmatrix}_{(w_j(k)+1)\times w_j(k)}.$$

Both the vehicle sequence matrix $S_j(k)$ and the expected load vector $\hat{L}_j(k+1)$, change their dimension dynamically by adding two rows when a new request occurs. Therefore, the matrix dimensions of A_L, B_L^1, B_L^2 are variable. B_L^1 is designed to remove the last two columns of the sequence vector, which are not necessary for representing load changes from step k to step $k+1$. On the other hand, when a request is satisfied, the first row of the sequence is eliminated. In fact, the adaptive behavior is captured by these techniques of expansion and reduction of matrix size.

The vehicle departure-time behavior is obtained by using the same methodology. Analytically,

$$\hat{T}_j(k+1) = f_T(T_j(k), S_j(k)) = A_T \cdot T_j(k) + B_T(S_j(k)), \quad (9)$$

where:

$$B_T(S_j(k)) = B_T^2 \cdot (S_j(k) \cdot B_T^1),$$

$$A_T = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 1 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 1 & 0 & \cdots & 0 \end{bmatrix}_{(w_j(k)+1)\times(w_j(k-1)+1)},$$

$$B_T^1 = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}, \quad B_T^2 = \begin{bmatrix} 0 & 0 & 0 & \cdots & 0 & 0 \\ 1 & 0 & 0 & \cdots & 0 & 0 \\ 1 & 1 & 0 & \cdots & 0 & 0 \\ 1 & 1 & 1 & \ddots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \ddots & 0 & 0 \\ \vdots & \vdots & \vdots & & 1 & 0 \\ 1 & 1 & 1 & \cdots & 1 & 1 \end{bmatrix}_{(w_j(k)+1)\times w_j(k)}.$$

As in the load state-space model, the matrices A_T, B_T^1, B_T^2 change their dimensions dynamically.

3.3. Objective Function

Here, the concept of an objective function is added in order to have a performance measure for deciding the optimal predicted vehicle routes by the controller, considering users' cost as well as a *proxy* of operational cost, as explained next. The major issue in the definition of the objective function is to define a reasonable horizon for prediction N, which depends on the studied problem, and also on the intensity of the unknown events entering the system in real time. In cases where the decision is made at instant k, but considering a predictive horizon greater than one, the decision maker (controller) adds the predictive feature into the formulation, because decisions made at $k+1$ will depend on possible events (new service requests) occurring at future instants ($k+2, k+3, \ldots$, etc.). Thus, the central dispatcher (controller) computes the control decisions for the entire control horizon N, i.e., $\{S(k), \ldots, S(k+N-1)\}$, and applies just the next step sequence set $S(k)$, based on a receding horizon control. The routing decisions will depend on how well the system predicts the impact of rerouting passengers due to unknown insertions.

The objective function for a generic prediction horizon N can be written as follows:

$$\underset{S(k)}{\text{Min}}\ J=\sum_{t=1}^{N}\sum_{j=1}^{F}\sum_{h=1}^{H(k+t)} p_h^{\Delta T(k+t)}\left((C_j(k+t)-C_j(k+t-1))|_{S_j(k+t-2),h}\right), \tag{10}$$

$$C_j(k+t)|_{S_j(k+t-2),h} = \sum_{i=1}^{w_j(k+t-1)} \left\{ \underbrace{[\hat{L}_j^{i-1}(k+t)+1](\widehat{T}_j^{i}(k+t)-\widehat{T}_j^{i-1}(k+t))}_{J_{\text{travel time}}} + \underbrace{r_j^i(k+t-1)\alpha(\widehat{T}_j^{i}(k+t)-T_j^0(k+t))}_{J_{\text{waiting time}}} \right\}\Bigg|_{S_j(k+t-2),h}, \tag{11}$$

where $k+t$ is the instant at which the tth request enters the system, measured from time interval k. $H(k+t)$ is the number of probable requests at instant $k+t$, $p_h^{\Delta T(k+t)}$ is the probability of occurrence of the hth request type (associated with a specific pair of zones, as discussed later in this section) during time interval $\Delta T(k+t)$, noting that $\Delta T(k+t)$ specifies the time interval to which time step $k+t$ belongs. $C_j(k+t)|_{S_j(k+t-2),h}$ in Equation (11) is a function associated with vehicle j at instant $k+t$, which depends on the decision sequence $S_j(k+t-1)$, given the previous known sequence $S_j(k+t-2)$ associated with a potential request h with probability $p_h^{\Delta T(k+t)}$. $w_j(k+t-1)$ is the number of stops estimated for vehicle j in sequence $S_j(k+t-1)$. $C_j(k+t)$, as shown in Equation (11), can be split into two pieces: a travel time ($J_{\text{travel time}}$) and a waiting time ($J_{\text{waiting time}}$) component. Both components are written as functions of the load and departure time. The former is computed as the difference between the departure time of consecutive stops, multiplied by the vehicle load (represented by the number of passengers plus the vehicle driver), whereas the latter considers the customers' waiting time while each vehicle moves on each segment of its assigned route. For the sake of flexibility and economic consistency, the waiting cost component is weighted by a coefficient, α. Analytically, $\hat{L}_j^{i-1}(k+t)$ denotes the expected load over the segment from stop $i-1$ to i; the difference $\widehat{T}_j^{i}(k+t)-\widehat{T}_j^{i-1}(k+t)$ measures the expected vehicle travel time on segment $(i-1,i)$, including the transfer delay at node i; and the difference $\widehat{T}_j^{i}(k+t)-T_j^0(k+t)$ measures the vehicle expected travel time to reach stop i from its current position plus the expected transfer delay at node i. $r_j^i(k+t-1)$ corresponds to the same binary variable used to identify pickup and delivery points in the sequence expression (3.2), but in this case is associated with the future sequence $S_j(k+t-1)$. In the context of the objective function formulation, this binary variable can be interpreted as a waiting time factor.

Note that in the first component of the objective function expression in Equation (11), the expected travel time is weighted by $\hat{L}_j^{i-1}(k+t)+1$. In such a computation, the expected load captures the user cost associated with travel time, whereas the added *one* roughly incorporates a *proxy* for the operational cost through the total time traveled by vehicles, even though some of them do not carry any passenger on certain segments of their routes.

The probabilities of occurrence of each scenario $p_h^{\Delta T(k+t)}$ are parameters in the objective function, and they are computed based on either real-time data, historical data, or a combination of both. In this particular application, we use a simple way to compute these probabilities from historical data (offline implementation). To do that, let us define the call mass center as the geographical location of the most likely call to occur during a specific time period, and within a specific area. As described in the problem formulation, what we really need is the probability that the expected new request will occur between two specific zones (pickup and delivery) within a certain time interval.

In order to apply this methodology, the zone of study has to be split into smaller subareas (clusters). How to choose the zoning will depend upon the demand intensity associated with each specific problem. The probability that a new call will appear for a specific pair of clusters h: $\{1,\ldots,H(k+t)\}$ within a time interval $\Delta T(k+t)$ is computed using the following expression:

$$p_h^{\Delta T(k+t)}=\frac{N_h^{\Delta T(k+t)}}{\sum_{g=1}^{H(k+t)} N_g^{\Delta T(k+t)}}, \tag{12}$$

where $N_h^{\Delta T(k+t)}$ is the total number of travel requests belonging to a specific origin-destination pair of clusters h over a set of pairs $\{1,\ldots,H(k+t)\}$, within a specific time interval $\Delta T(k+t)$. Note that $\sum_{h=1}^{H(k+t)} p_h^{\Delta T(k+t)}=1$, as expected.

With regard to the step size to be used in the prediction, George and Powell (2005) develop and discuss many interesting methods to incorporate a good estimate of optimal step size (such as a Kalman filter). We realize that none of these methods properly replicate the DPDP conditions, considering that in addition to representing a good estimate of the time between calls, what we really want to calibrate is a parameter for optimizing the system performance function over time, which can lead to the optimal routing strategy including future information. To do that, a sensitivity analysis was conducted from simulated data to find the step-size value that minimizes the objective function for more than one step ahead. It is very important to highlight the fact that these variables are continuous; nonoptimal behavior could occur if

they are not properly adjusted by sensitivity analysis. For the two-steps-ahead application (see §4), this parameter is denoted by τ; as discussed above, physically it represents the expected time for a predicted request to happen. However, what τ really represents is the best instant for inserting the future expected call in order to optimize the routing scheme. In general, these parameters are tunable for each step ahead of prediction.

In the context of this paper, we compare a myopic strategy (one-step-ahead) with the two-steps-ahead predictive approach that includes future information from the system to show the improvements in routing when considering a predictive component in the routing decisions under a DPDP system.

The one-step-ahead strategy means that the prediction horizon is $N = 1$, and $H(k+1) = 1$ because the new requirement is one and known, and therefore its probability is equal to 1. This results in the following expression for the objective function using Equation (10):

$$\begin{aligned}\min_{S(k)} J &= \sum_{t=1}^{1}\sum_{j=1}^{F}\sum_{h=1}^{H(k+t)=1} p_h^{\Delta T(k+t)}(k+t)\\ &\quad\cdot (C_j(k+t) - C_j(k+t-1))|_{S_j(k+t-2),\,h}\\ &= \sum_{j=1}^{F} \overbrace{p_1^{\Delta T(k+1)}(k+1)}^{=1}\cdot (C_j(k+1) - C_j(k))|_{S_j(k-1),\,1}\\ &= \sum_{j=1}^{F}\Big(C_j(k+1) - \overbrace{C_j(k)}^{\text{known constant}}\Big)\Big|_{S_j(k-1),\,1} \end{aligned} \tag{13}$$

where

$$\begin{aligned} &C_j(k+1)|_{S_j(k-1),1}\\ &= \sum_{i=1}^{w_j(k)} \Big\{ \underbrace{[\hat{L}_j^{i-1}(k+1)+1](\widehat{T}_j^i(k+1) - \widehat{T}_j^{i-1}(k+1))}_{J_{\text{travel time}}}\\ &\qquad + \underbrace{r_j^i(k)\alpha(\widehat{T}_j^i(k+1) - T_j^0(k+1))}_{J_{\text{waiting time}}}\Big\}\Big|_{S_j(k-1),1} \end{aligned} \tag{14}$$

Note that the difference $(C_j(k+1) - C_j(k))|_{S_j(k-1),\,1}$ is evaluated considering the control action in the previous instant, represented by $S_j(k-1)$. Conceptually, J represents the insertion cost when the system accepts a new call, computed in real time and considering the entire vehicle fleet.

The two-steps-ahead prediction's objective function is different from the previous one, because it includes a prediction of where the following call is going to fall, and with what probability. The controller selects the vehicle's sequence that minimizes the general two-steps-ahead objective function, which is as follows,

$$\begin{aligned} &\min_{S(k)} J\\ &= \sum_{t=1}^{2}\sum_{j=1}^{F}\sum_{h=1}^{H(k+t)} p_h^{\Delta T(k+t)}(k+t)\\ &\quad\cdot (C_j(k+t) - C_j(k+t-1))\big|_{S_j(k+t-2),\,h}\\ &= \sum_{j=1}^{F}\Bigg[\cancel{C_j(k+1)\big|_{S_j(k-1),1}} - C_j(k)\\ &\qquad + \sum_{h=1}^{H(k+2)} p_h^{\Delta T(k+2)}(k+2)\cdot C_j(k+2)\big|_{S_j(k),\,h}\\ &\qquad - \cancel{\overbrace{\sum_{h=1}^{H(k+2)} p_h^{\Delta T(k+2)}(k+2)}^{=1}\cdot \overbrace{C_j(k+1)\big|_{S_j(k-1),1}}^{\text{independent of } h}}\Bigg]\\ &= \sum_{j=1}^{F}\Bigg[\sum_{h=1}^{H(k+2)} p_h^{\Delta T(k+2)}(k+2)\cdot C_j(k+2)\big|_{S_j(k),\,h} - \overbrace{C_j(k)}^{\text{known constant}}\Bigg], \end{aligned} \tag{15}$$

where

$$\begin{aligned} &C_j(k+2)|_{S_j(k),\,h}\\ &= \sum_{i=1}^{w_j(k+1)} \Big\{ \underbrace{[\hat{L}_j^{i-1}(k+2)+1](\widehat{T}_j^i(k+2) - \widehat{T}_j^{i-1}(k+2))}_{J_{\text{travel time}}}\\ &\qquad + \underbrace{r_j^i(k+1)\alpha(\widehat{T}_j^i(k+2) - T_j^0(k+2))}_{J_{\text{waiting time}}}\Big\}\Big|_{S_j(k),\,h} \end{aligned} \tag{16}$$

3.4. Solution Method

Traditional optimization methods are not very efficient and, in most cases, useless for solving problems like the one-step- and two-steps-ahead formulations presented above. This is mostly due to the high nonlinearity of the objective function expressions, in addition to the hybrid (discrete-continuous) nature of the variables. The optimization problem for both one-step- and two-steps-ahead strategies could be solved by using explicit enumeration (EE) that considers all feasible insertion solutions whenever a call request enters the system. However, this inefficient method is neither appropriate for a large vehicle fleet nor for long prediction horizons, due to a computational capacity constraint. For those scenarios, the application of such control algorithms solved with EE is not feasible in real-time routing. Instead, we propose a new ad hoc algorithm to solve the mixed-integer problem behind the DPDP formulation proposed here.

The scheme is based on the particle swarm optimization (PSO) algorithm, which is a new type of evolutionary computation method that has performed quite well in previous applications, not only in terms of solution accuracy, but also in computation time savings (Kennedy and Eberhart 2001). PSO has been inspired by the social behavior of animals and insects, specifically on the behavior of a swarm of particles over a multidimensional search space.

After reviewing the literature, we highlight Coelho, de Moura Oliveira, and Cunha (2005), who present a predictive controller based on recursive linear models where the optimization problem is solved by PSO. A good performance of PSO is shown in comparison with genetic algorithms (GA) and classical quasi-Newton methods. Wang and Xiao (2005) describe a PSO-based predictive controller based on a radial basis function (RBF) neural network model, obtaining slightly better results than those from GA and a quasi-Newton method. In the context of vehicle routing, Zhu et al. (2006) propose a PSO scheme to solve a static vehicle-routing problem with time windows. The effectiveness, in terms of precision and computational time, is shown by experimental results. Next, a description of the basic PSO algorithm is presented, to close the section with a detailed description of the proposed PSO-based algorithm for the DPDP.

PSO Algorithm. The PSO algorithm, used to solve complex nonlinear optimization problems, consists of a particle swarm, which represents a population of candidate solutions. The particles are initialized randomly, and then move iteratively within the search space in order to find new solutions. The particles have a *fitness* associated with the solution quality, usually given by the objective function to be optimized. Each particle is characterized by a position x_i (i is the index of the particle) and a velocity v_i (both are d-dimensional, where d is the dimension of the solution vector). Each particle records its best previous position $x_i^{\#}(t)$ and the best position among all the particles belonging to the swarm, namely $x^*(t)$, with t representing the current iteration. The particles are updated according to their cognitive and social behavior from the following equations:

$$\begin{aligned} v_i(t+1) &= \omega \cdot v_i(t) + c_1 \cdot \varphi_1 \cdot (x_i^{\#}(t) - x_i(t)) \\ &\quad + c_2 \cdot \varphi_2 \cdot (x^*(t) - x_i(t)), \\ x_i(t+1) &= x_i(t) + v_i(t+1), \end{aligned} \tag{17}$$

where ω is the inertia factor, c_1 is the self-cognitive constant, and c_2 represents the social component factor ($\omega, c_1, c_2 > 0$ are tuning parameters). Furthermore, φ_1 and φ_2 are uniformly distributed random numbers in the range $(0, 1)$, which help us preserve the diversity of the swarm. From Equations (17), particles move according to their inertia, their experience, and the experience of the most successful particle of the swarm. The search is conducted over a subset of the entire space (depending on the problem) to effectively guide the particles in the search space towards the optimum by keeping the velocity clamped inside a predefined range.

The above description of PSO was originally conceived for solving continuous problems. In our application, we completely adapted the PSO code in order to add integer variables in the solution. Next, the specific PSO we developed to solve the DPDP problem is described.

Proposed PSO Algorithm for Solving the DPDP. The proposed algorithm based on PSO utilizes particles that belong to R^2. For a new call requesting service, the method finds the best insertion positions for the new pickup and delivery points along a certain vehicle sequence. The algorithm is run for each vehicle, to finally apply the new sequence to the vehicle showing the lowest insertion cost as based on the objective function. In this problem, a particle is associated with an insertion within a sequence for a specific vehicle.

Let us consider a vehicle j with an associated sequence $S_j(k-1)$. When a new call comes up at k, the PSO algorithm generates possible sequences $S_j^{\vartheta}(k)$, with each one associated with a particle that finally determines the insertion position of the incoming call $\vartheta = (pu, de)$ within the sequence, where pu corresponds to the pickup, and de to the delivery. Thus, a sequence generated by PSO is given by

$$S_j^{\vartheta}(k) = \begin{bmatrix} r_j^1(k) & 1-r_j^1(k) & \Gamma_j^1(k) & \text{label}_j^1(k) \\ \vdots & \vdots & \vdots & \vdots \\ r_j^{pu}(k) & 1-r_j^{pu}(k) & \Gamma_j^{pu}(k) & \text{label}_j^{pu}(k) \\ \vdots & \vdots & \vdots & \vdots \\ r_j^{de}(k) & 1-r_j^{de}(k) & \Gamma_j^{de}(k) & \text{label}_j^{de}(k) \\ \vdots & \vdots & \vdots & \vdots \\ r_j^{w_j(k)}(k) & 1-r_j^{w_j(k)}(k) & \Gamma_j^{w_j(k)}(k) & \text{label}_j^{w_j(k)}(k) \end{bmatrix}_{w_j(k)\times 4}, \tag{18}$$

where the puth row and deth row are the positions of the new pickup and delivery, respectively. Note that each potential sequence in Equation (18) must be feasible in terms of precedence. Every particle provided by the PSO is two dimensional (both components are continuous) and has an associated insertion pair $\vartheta = (pu, de)$. Because the particles are real pairs, each component is approximated to the next integer. When particles are either initialized or updated, they might

not be feasible in terms of precedence, or because they could fall outside the allowable range (pickup out of $[1, w_j(k)-1]$, and delivery out of $[2, w_j(k)]$). In these cases, the particles are repaired to get a feasible sequence.

The PSO algorithm specifies a function (detailed in the description of the PSO-based algorithm to follow) to translate each particle into a feasible insertion position, defining a possible sequence to be followed by the vehicle. An example is shown in (19).

$$
\begin{aligned}
&\text{Swarm}\\
&\Leftrightarrow \begin{pmatrix}\text{Particle 1}\\ \text{Particle 2}\\ \text{Particle 3}\\ \text{Particle 4}\\ \text{Particle 5}\\ \text{Particle 6}\\ \text{Particle 7}\end{pmatrix} = \begin{pmatrix} x_1=(0.7,3.9)\\ x_2=(0.2,6.2)\\ x_3=(4.1,4.9)\\ x_4=(2.6,4.9)\\ x_5=(-0.8,7.4)\\ x_6=(3.1,1.1)\\ x_7=(1.9,3.2)\end{pmatrix}\\
&\Leftrightarrow \begin{pmatrix}\vartheta_1=(pu_1,de_1)\\ \vartheta_2=(pu_2,de_2)\\ \vartheta_3=(pu_3,de_3)\\ \vartheta_4=(pu_4,de_4)\\ \vartheta_5=(pu_5,de_5)\\ \vartheta_6=(pu_6,de_6)\\ \vartheta_7=(pu_7,de_7)\end{pmatrix} \Leftrightarrow \begin{pmatrix}(1,4)\\ (1,6)\\ (4,5)\\ (3,5)\\ (1,6)\\ (2,3)\\ (2,4)\end{pmatrix}\\
&\Leftrightarrow \begin{pmatrix}\boxed{3^+}\to 1^+\to 2^+\to \boxed{3^-}\to 1^-\to 2^-\\ \boxed{3^+}\to 1^+\to 2^+\to 1^-\to 2^-\to \boxed{3^-}\\ 1^+\to 2^+\to 1^-\to \boxed{3^+}\to \boxed{3^-}\to 2^-\\ 1^+\to 2^+\to \boxed{3^+}\to 1^-\to \boxed{3^-}\to 2^-\\ \boxed{3^+}\to 1^+\to 2^+\to 1^-\to 2^-\to \boxed{3^-}\\ 1^+\to \boxed{3^+}\to \boxed{3^-}\to 2^+\to 1^-\to 2^-\\ 1^+\to \boxed{3^+}\to 2^+\to \boxed{3^-}\to 1^-\to 2^-\end{pmatrix}. \qquad (19)
\end{aligned}
$$

In this example, for Particles 1, 4, and 7, a simple rounding up generates a feasible sequence in both precedence and allowable range. In the cases of Particles 2 and 5, a simple rounding up generates a sequence feasible in terms of precedence but outside the allowable range; thus, the sequence is repaired. For Particles 3 and 6, a simple rounding up generates a sequence within the allowable range, but unfeasible in terms of precedence. Once again, the sequence is repaired.

For all feasible particles, a PSO fitness value is evaluated in terms of the corresponding objective function as defined in Equation (10). Note that when a sequence is unfeasible in terms of capacity, it is not repaired; rather, a penalized fitness is used instead.

Without loss of generality, the proposed algorithm based on PSO is described for a two-steps-ahead horizon. The algorithm is written as a function of three major procedures: GEN_PAR, REP_PAR, and MOD_PAR. With GEN_PAR the particles associated with potential sequences are randomly generated. REP_PAR takes the particles and generates their associated feasible sequences. Finally, MOD_PAR corresponds to the core of the evolutionary PSO algorithm, updating the particles for the next iteration according to the best previous solutions. The algorithm iterates first at one-step-ahead and then at two-steps-ahead, as detailed below:

PSO-Based Algorithm

Step 0. Initialize parameters PSO, like $n =$ number of particles, $\omega =$ inertia weight, $c_1 =$ cognitive weight, $c_2 =$ social weight.

Step 1. Assume that the predefined sequence set $S(k-1)$ is known. A new service request (call) enters the system. Then, the functions GEN_PAR and REP_GEN based on PSO are utilized to generate n potential sets of sequences $S^{\vartheta_l}(k)$, with l: $1, 2, \ldots, n$ (particles). Note that $[n/F]$ particles are associated with each vehicle, which means that the insertion of the new call falls in the specific vehicle sequence (F is the fleet size).

Step 2. For each particle $S^{\vartheta_l}(k)$, consider $H(k+1)$ probable requests. Then, GEN_PAR and REP_GEN based on PSO are applied to generate n potential sequences $S^{\vartheta_m}(k+1)|_h$, m: $1, 2, \ldots, n$, for each probable request pattern h: $1, 2, \ldots, H(k+1)$.

Step 3. Provided that $S^{\vartheta_l}(k)$ is known, evaluate the fitness function $C(k+2)|_{S^{\vartheta_l}(k),\,h} - C(k+1)|_{S(k-1)}$, defined in Equations (14) and (16), for all potential sequences $S^{\vartheta_m}(k+1)|_h$. If $S^{\vartheta_m}(k+1)|_h$ is unfeasible for capacity, penalize its fitness. Then, the best set of particles $S^{\vartheta^*_m}(k+1)|_h$ for h: $1, 2, \ldots, H(k+1)$ associated with the minimum fitness function is selected.

Step 4. If a tolerance criterion (maximum number of iterations of Steps 3 and 4) is satisfied, then proceed to Step 5. Otherwise, update the position and the velocity of all particles by using the function MOD_PAR (evolutionary stage). With the function REP_GEN, generate the sequences $S^{\vartheta_m}(k+1)|_h$, m: $1, 2, \ldots, n$ for h: $1, 2, \ldots, H(k+1)$ and go back to Step 3.

Step 5. Given that $S^{\vartheta_l}(k)$ is known, and by using $S^{\vartheta^*_m}(k+1)|_h$ for h: $1, \ldots, H(k+1)$ obtained in Step 3, evaluate the two-steps-ahead objective function (fitness) in Equation (15). If $S^{\vartheta_l}(k)$ is unfeasible for capacity, penalize its fitness.

Step 6. From the fitness computed for each particle $S^{\vartheta_l}(k)$, l: $1, 2, \ldots, n$, record the best sequence $S^{\vartheta_l^*}(k)$.

Step 7. If a tolerance criterion (maximum number of iterations Steps 2 to 7) is satisfied, then STOP and $S^{\vartheta_l^*}(k)$ is the optimum. Otherwise, by using the function MOD_PAR (evolutionary stage), update the position and the velocity of all particles for l: $1, 2, \ldots, n$. By using REP_GEN, generate $S^{\vartheta_l}(k)$, l: $1, 2, \ldots, n$, and go back to Step 2.

The detailed procedures involved in the algorithm are explained next:

GEN_PAR
This procedure is performed for the first iteration, $t = 1$.

For every vehicle j, $[n/F]$ particles are assigned. Then, for every particle l associated with a given vehicle j, the following features are set:

Initialize random positions and velocities for the particles.

$$x_l(1) = (x_{1l}(1), x_{2l}(1)),$$
$$(x_{1l}(1), x_{2l}(1)) \in [0, w_j(k) - 1] \times [1, w_j(k)],$$
$$v_l(1) = (v_{1l}(1), v_{2l}(1)),$$
$$(v_{1l}(1), v_{2l}(1)) \in [-w_j(k)/2, w_j(k)/2] \times [-w_j(k)/2, w_j(k)/2].$$

REP_PAR
This function is required to convert a particle l into a feasible sequence. Thus, for every particle l associated with a vehicle j, the repair procedure is as follows:

Round and repair particle out-of-range.

$$pu_l = \max(\min(\lceil x_{1l}(t) \rceil, w_j(k) - 1), 1),$$
$$de_l = \min(\max(\lceil x_{2l}(t) \rceil, 2), w_j(k)).$$

If $pu_l < de_l$, then $\vartheta_l = (pu_l, de_l)$.

Repair unfeasible particle in precedence.

If $pu_l > de_l$, then $y = (pu_l - de_l)/2$, $pu_l = \lfloor pu_l - y \rfloor$, $de_l = \lceil de_l + y \rceil$, $\vartheta_l = (pu_l, de_l)$.

If $pu_l = de_l$, then $de_l = \min(w_j(k), de_l + 1)$, $pu_l = \max(de_l - 1, 1)$, $\vartheta_l = (pu_l, de_l)$.

Finally, depending on the case (one-step- or two-steps-ahead iteration), the sequence $S(k)$ (or $S(k+1)$) associated with the particle l is $S^{\vartheta_l}(k)$ (or $S^{\vartheta_l}(k+1)|_h$).

MOD_PAR
This function corresponds to the evolutionary stage of the PSO algorithm. Thus, for every particle l associated with a vehicle j, the procedure is as follows (for iteration $t+1$):

Update velocity:

$$v_l(t+1) = \omega \cdot v_l(t) + c_1 \cdot \varphi_1 \cdot (x_l^{\#}(t) - x_l(t)) + c_2 \cdot \varphi_2 \cdot (x^*(t) - x_l(t)).$$

If velocity is saturated, then set

$$(v_{1l}, v_{2l}) \in [-w_j(k)/2, w_j(k)/2] \times [-w_j(k)/2, w_j(k)/2].$$

Update position:

$$x_l(t+1) = x_l(t) + v_l(t),$$

where $x_l(t)$, $v_l(t)$ are the position and the velocity of particle l at iteration t, φ_1 and φ_2 are uniformly distributed random numbers in the range $(0, 1)$, ω is the inertia weight, c_1 is the cognitive weight, c_2 is the social weight, $x_l^{\#}(t)$ is the best previous position reached for particle l, and $x^*(t)$ is the best position among all particles belonging to the swarm.

4. Simulation Tests

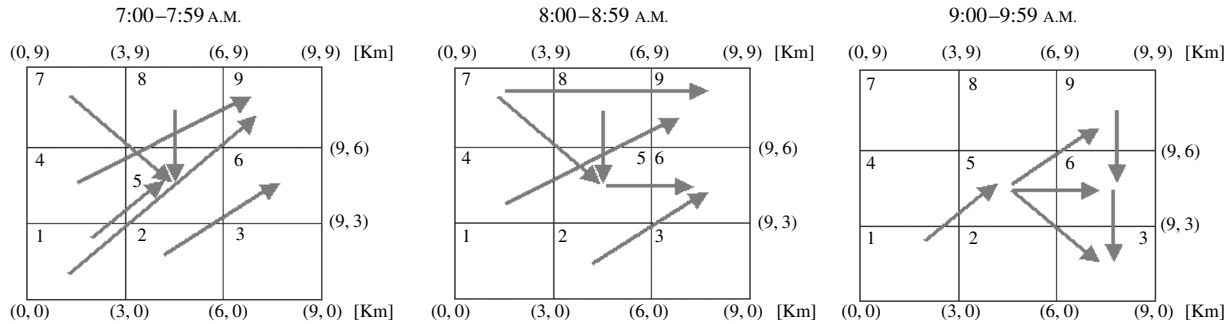
4.1. Experiment Description

A discrete-event system simulation is conducted for a three-hour period to evaluate the performance of the proposed dispatch control algorithm for a dynamic vehicle-routing system. The scheme considers a fleet of nine transit vehicles, each with a capacity for four passengers. Dispatch decisions are made in real-time by the controller. Service requests are unknown; however, the average system pattern is supposed to be known from historical data, obtained from the average demand measured over the preceding week.

The simulation scenario is not real. However, the demand patterns follow a heterogeneous distribution inspired by real data from the Origin-Destination Survey in Santiago, Chile, 2001. We consider an urban service area of approximately 81 km^2. Vehicles are assumed to travel straight between stops at an average speed of 20 km/hr throughout the region. The simulation was performed over three time intervals in a representative work day during the morning peak hour, i.e., T_p = (7:00–7:59, 8:00–8:59, 9:00–9:59), and the demand distribution was assumed to follow various patterns over the studied period, as discussed below.

The objective of the experiments was to test the performance of the predictive algorithm under different conditions and modeling assumptions. One major factor in the definition of the expected occurrence probability of future service requests is the spatial (and temporal) disaggregation of the total area (and time period).

Following the methodology, we generate historical data assuming that 90% of the intervehicle trips occur in six pairs of sectors ($H = 6$). Therefore, we consider this subset of origin destination pairs for the objective function computation. Next, the different arrival rates per geographic pair-interval were obtained, from which the corresponding occurrence probabilities were generated. For simplicity, we assumed that



P/D	5	6	9
1	0.20		0.20
2		0.15	
4			0.15
7	0.15		
8	0.15		

P/D	5	6	9
2		0.10	
4			0.30
5		0.10	
7	0.20		0.10
8	0.20		

P/D	3	5	6	9
1		0.1		
5	0.3		0.1	0.1
6	0.2			
9			0.2	

Figure 3 Origin-Destination Trip Patterns (Nine-Zone Division)
Note. P\D: Pickup and delivery zones.

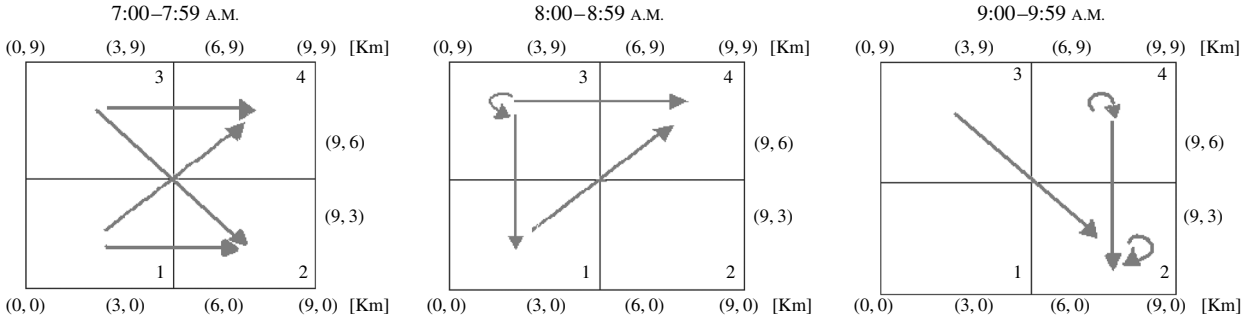
the probabilities do not change within each time interval, but they do from one interval to the next.

First, we tested nine homogeneous zones where historical data show that trips occur in the six most relevant interzonal origin-destination pairs, as shown in Figure 3. The probability associated with each trip pattern is in the table underneath each figure. For example, considering the schedule between 7:00 and 7:59 A.M., the chance that a trip from Zone 1 to Zone 5 occurs is 0.2 among all trips occurring everywhere else within that time interval. The probability of other trips between zones not considered among the six most important pairs is assumed to be negligible. For this zoning desegregation, intrazonal trips were assumed to be negligible too.

A second experiment was to aggregate the spatial area into four zones instead of the nine used in the previous example, assuming the same trip patterns as above. Figure 4 shows the scheme associated with this second case. Note that intrazonal trip probabilities cannot be neglected in this case, due to the more aggregated zoning.

In terms of demand, 120 calls were generated over the whole simulation period of three hours, according to a spatial and temporal distribution with the same behavior as the historical pattern considered.

A distribution for the time interval between successive calls is also assumed in order to compute time-interval probabilities. In this case, we use a negative exponential distribution, with rates of 0.5 [call/minute], 1 [call/minute], and 0.5 [call/minute] for the first, second, and third hours of simulation, respectively. In terms of spatial distribution, pickup and delivery points were generated randomly within each corresponding zone in order to replicate the trip pattern and probabilities shown in Figures 3 and 4, depending on the experiment. Arbitrarily, vehicles were initially located at the mass center of zones. This



P/D	2	4
1	0.15	0.4
3	0.15	0.3

P/D	1	3	4
1			0.23
3	0.14	0.12	0.51

P/D	2	4
2	0.13	
3	0.29	
4	0.29	0.29

Figure 4 Origin-Destination Trip Patterns (Four-Zone Division).
Note. P\D: Pickup and delivery zones.

assumption does not affect the final statistics of the tests, because a reasonable warm-up period was considered. Thirty replications of each experiment were conducted to obtain global statistics, as shown in §4.2. With regard to the objective function, $\alpha = 1$ was used, which means that the travel time is as important as the waiting time in the objective function expression. The one-step- and two-steps-ahead algorithms are then evaluated and compared for both the four- and nine-zone spatial disaggregation cases. The PSO algorithm with no swapping was implemented in Matlab version 7.0 with a Pentium IV processor. The parameters of PSO used in this first approach were $\omega = 1$, $c_1 = 2$, $c_2 = 2$.

As introduced in the previous section, one relevant fine-tuning parameter is the predicted time between call requests, τ, which is relevant when evaluating the performance function of the two-steps-ahead algorithm. We find the optimal value of such a parameter by conducting a sensitivity analysis around the observed interarrival times from the historical data report. Figure 5 shows the mean value of the objective function for different τ values. Ten replications were used in order to obtain the optimum values: $\tau = 4.8$ for nine-zone partitions, and $\tau = 2.4$ for four-zone partitions.

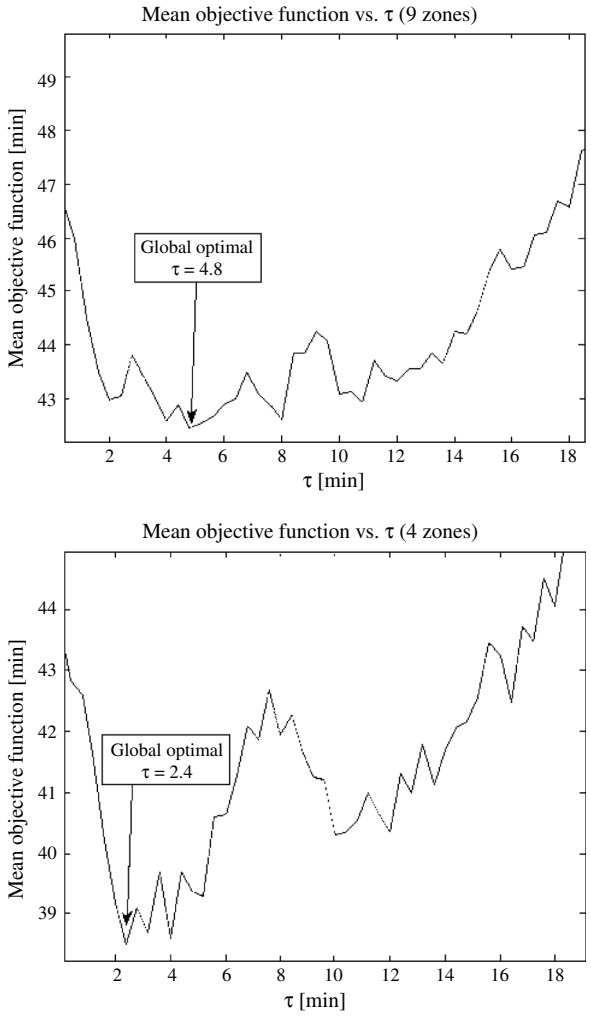


Figure 5 Sensitivity Analysis for τ (Nine- and Four-Zone Partitions)

4.2. Analysis of Results

Figure 6 presents the computational time evolution of both the EE and the proposed PSO algorithm as the number of requests increases, keeping the fleet size fixed at nine vehicles. Both algorithms are run for the two-steps-ahead case. Moreover, PSO is tested for 10 iterations of 10 particles (10–10) and for 20 iterations of 20 particles (20–20). The figure graphically shows our previous guess, confirming the impossibility of using EE in real-size problems for real-time dispatch, because the computational time rapidly explodes as the number of requests surpasses 30.

The only real advantage of using EE is that we can obtain the global optimum. However, the experience in the literature showed that in practice, the gap between the optimal value and the PSO solution is quite small, a conclusion that was supported by some tests we conducted (errors on the order of 2%–3% in 10–10 experiments). In addition, the procedure has to be run repeatedly in real time, which further justifies the use of a procedure such as PSO (or other procedures as discussed in §5) to solve real-time vehicle-routing problems with a predictive scheme. Our example considers 120 calls, and below we present the results obtained from the proposed PSO algorithm.

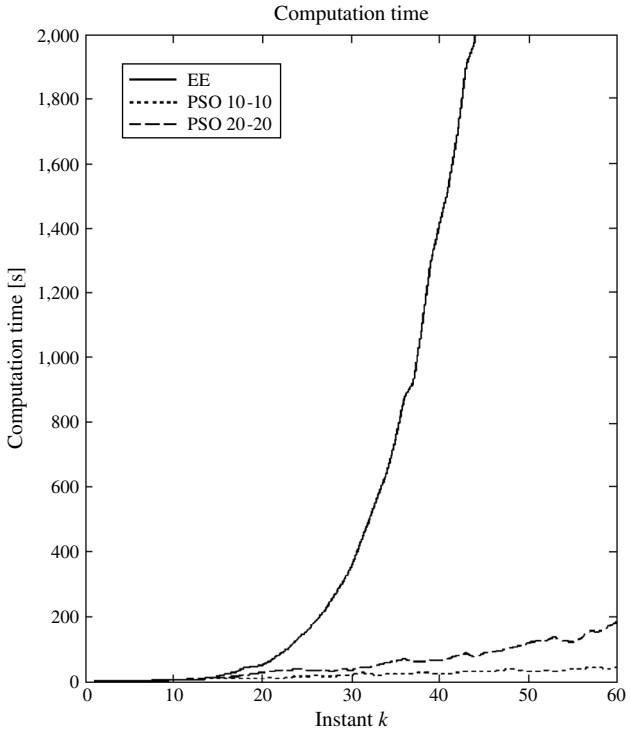


Figure 6 Computational Time vs. the Number of Requests (k)

Table 1 PSO Computational Time and Performance Comparison Using Different Parameters for Four Zones and Four Probabilities

	Total time (waiting + travel) (min)		Operation time (min)		Computational time (sec)	
Two step ahead	Mean	Std	Mean	Std	Mean	Std
5 iterations, 5 particles	46.76	4.41	182.61	6.17	589.6	98.62
10 iterations, 10 particles	41.71	4.97	178.73	8.42	1,737.3	129.44
15 iterations, 15 particles	41.02	4.30	178.35	6.18	2,559.6	576.92

We have chosen the mean and standard deviation values of total travel and waiting times to measure the system performance in terms of user level of service. Additionally, the average total time spent by a vehicle in the system is also reported as a *proxy* of the average operational cost. In Table 1 below, we show the PSO computational time and performance comparison using different parameters (iterations-particles) for four zones and four probabilities. As expected, Table 1 shows a trade-off between the solution accuracy and the PSO computational time for the two-steps-ahead algorithm.

From the table, it seems reasonable to use 10 iterations and 10 particles (10–10) in the following experiments. Note that in terms of solution quality, we do not obtain too much improvement (41.71 versus 41.02 and 178.73 versus 178.35) for adding five more particles and iterations. However, in computation time, the cost of this change is quite considerable. In Figure 6, we can graphically see the low computational effort needed to run the 10–10 PSO algorithm for the two-steps-ahead case.

From such an observation, in Tables 2, 3, and 4, the user and operator performance indicators for the 30 replications of the PSO algorithm with 10 particles and 10 iterations are reported. These values capture the differences of running both algorithms (one-step-ahead versus two-steps-ahead) for both experiments, nine zones (six probabilities), and four zones (four probabilities), respectively. A warm-up period of 20 minutes at both sides (the start and end of the simulation period) was considered to measure system performance under steady-state conditions.

Tables 2 and 3 show the performance of the two algorithms over the whole three-hour period (excluding the warm-up interval). From the results shown in the tables above, we appreciate that the predictive (two-steps-ahead) algorithm systematically performs better than the one-step-ahead algorithm, supporting our initial guess regarding the necessity of adding a predictive component into the real-time routing algorithms applied to this kind of system.

The most important savings from the predictive approach comes from the waiting time component (13.23% in the best case). From this result, we can infer that most prediction benefits are due to avoiding extra waiting at future pickup points on scheduled vehicle sequences, which is quite manageable by the dispatcher once he can make routing decisions based upon future potential requests.

From the example, improvements in travel time from predictive algorithms are lower than those obtained for waiting time (5.38% in the best case). In addition, there is absolutely no observable improvement in operational cost; it remains practically constant, as shown in Table 4. We hypothesize that the magnitude of the travel time benefits as well as savings in operational cost could be improved by considering further adjustments of the objective function formulation. Actually, the current objective function version does not take into account the real weight of the operational cost compared with customer level of service in terms of waiting and travel time as part of the specification (see §5 for discussion of this topic as part of planned further research).

Moreover, it seems that the most aggregated zoning scenario (Table 3) results in a more efficient routing

Table 2 One-Step-Ahead and Two-Steps-Ahead Performance Comparison Nine Zones, Six Probabilities

	Waiting time (min)		Travel time (min)		Total time (min)	
PSO	Mean	Std	Mean	Std	Mean	Std
One step ahead	21.70	2.22	23.88	1.51	45.58	3.47
Two step ahead	20.03	3.33	23.17	1.33	43.20	4.23
Savings	1.67		0.71		2.38	
Improv. %	**7.68**		**2.98**		**5.22**	

Table 3 One-Step-Ahead and Two-Steps-Ahead Performance Comparison Four Zones, Four Probabilities

	Waiting time (min)		Travel time (min)		Total time (min)	
PSO	Mean	Std	Mean	Std	Mean	Std
One step ahead	21.92	4.52	23.98	1.53	45.90	5.33
Two step ahead	19.02	3.36	22.69	1.46	41.71	4.58
Savings	2.74		1.29		4.19	
Improv. %	**13.23**		**5.38**		**10.05**	

Table 4 One-Step-Ahead and Two-Step-Ahead Comparison of Operational Cost Per Vehicle

Vehicle travel time (min/veh)	4 zones, 4 probabilities		9 zones, 6 probabilities	
PSO	Mean	Std	Mean	Std
One step ahead	179.87	11.75	182.39	7.84
Two step ahead	178.73	8.42	183.41	13.06

scheme from the two-steps-ahead algorithm than the solution obtained from the one-step-ahead case, when compared against the nine-zone scheme (Table 2). One of the explanations for this difference is that the more aggregated the modeling area is, the more accurate the prediction of probabilities. Thus, it is easier to find vehicle routes following well-known trip patterns (as those shown in Figure 4), resulting in better prediction of potential rerouting along those paths.

5. Summary, Conclusions, and Further Research

This paper presents an analytical way of modeling the impact of stochastic rerouting delays for dynamic multivehicle pickup and delivery problems based on a hybrid adaptive predictive control scheme to optimize the vehicle dispatch and routing decisions by including future information with regard to unknown demand. Major contributions of this research include the development of an adaptive dynamic state-space model, including two well-used descriptive variables (vehicle load and departure time) and an objective function for dispatching oriented to improve the client level of service provided by the system. By using prediction (two-steps-ahead method) within the optimization scheme, a considerable improvement with respect to myopic rules (one-step-ahead method) was found empirically, showing the benefit of the introduction of such schemes in real-time routing and scheduling of flexible transit systems. We recognize that the problem considered here has a hybrid nature, in that it includes discrete variables (sequences and load) as well as continuous variables (departure time) that need to be adaptive along the system operation time, yet must be predictive for improving the system behavior by considering potential rerouting in the near future. Although we could have used a longer prediction horizon, a two-step-ahead prediction so far seems sufficient to prove the benefits of the approach.

Another significant contribution of this work is the development of a solution algorithm based on PSO specifically designed to solve the DPDP. It finds near-optimal solutions in reasonable computation time, which allows the implementation of real-time systems. This scheme could still be improved after a sensitivitys analysis in order to find the optimal parameters of this algorithm (cognitive, social, and inertia weight).

Experimental results show a considerable reduction in customer waiting times, along with a positive (although small) improvement in travel time savings. Benefits of the predictive approach considerably increase when the urban area is compacted into four zones instead of the original partitioning, corresponding to nine zones. This benefit of the aggregation most probably came from decreasing the uncertainty by involving fewer probabilities in the computations.

In further applications, several elements of the approach must be improved. First, we expect to count on a stricter analytical expression for the state-space model, particularly the A and B matrices as dependent on the vehicle sequence. In more sophisticated applications, additional components could enrich the state-space models as well as the objective function formulation.

Second, and related to the previous issue, a more complete rigorous expression for the objective function must be used. That should explore the inclusion of time windows (hard and soft) and a better consideration of operational costs. A sensitivity analysis with regard to both parameters α and τ is to be investigated, when counting with more sophisticated algorithms to solve the two-steps-ahead problem. We claim that it is possible to improve the estimate of tuning variables, such as the number of probable calls, future step time prediction (τ, which is unknown), prediction horizon (N), service policy, search over different feasible solutions structures, etc.

Third, because the proposed PSO is so much more efficient than EE, we will be able to predict more than two steps ahead. For the same reason, we plan to allow partial swapping of the original vehicle sequences to search a bigger feasible set, and get solutions closer to the optimum. In addition, we are working on a GA in an attempt to find an efficient algorithm besides PSO to solve this problem.

Fourth, the way in which the system information is internalized through probabilities can also be improved through a more realistic scheme. This requires a better understanding of the historical data (fuzzy clustering). We also propose to combine historical data (offline) with online information in a more elaborate model able to capture imminent events that could affect the system performance.

Acknowledgments

This research was partially financed by Fondecyt, Chile, Grants 1040698, 1030700, and 1061261, the Millennium Institute "Complex Engineering Systems," and the ACT-32 Project "Real Time Intelligent Control for Integrated Transit Systems." The authors also acknowledge the helpful comments of Riju Lavanya, as well as those from anonymous referees.

References

Bemporad, A., M. Morari. 1999. Control of systems integrating logic, dynamics and constraints. *Automatica* **35** 407–427.

Bemporad, A., F. Borrelli, M. Morari. 2002. Model predictive control based on linear programing. The explicit solution. *IEEE Trans. Automatic Control* **47**(12) 1974–1985.

Bertsimas, D., G. van Ryzin. 1991. A stochastic and dynamic vehicle routing problem in the Euclidean plane. *Oper. Res.* **39** 601–615.

Bertsimas, D., G. van Ryzin. 1993a. Stochastic and dynamic vehicle routing problem in the Euclidean plane with multiple capacitated vehicles. *Oper. Res.* **41** 60–76.

Bertsimas, D., G. van Ryzin. 1993b. Stochastic and dynamic vehicle routing with general demand and interarrival time distributions. *Appl. Probab.* **25** 947–978.

Coelho, J. P., P. B. de Moura Oliveira, J. B. Cunha. 2005. Greenhouse air temperature predictive control using the particle swarm optimisation algorithm. *Comput. Electronics in Agriculture* **49** 330–344.

Cortés, C. E., R. Jayakrishnan. 2004. Analytical modeling of stochastic rerouting delays for dynamic multi-vehicle pickup and delivery problems. *The Fifth Triennial Symposium on Transportation Analysis, TRISTAN V*. Le Gosier, Guadalupe, 13–18 June.

Desrosiers, J., F. Soumis, Y. Dumas. 1986. A dynamic programming solution of a large-scale single-vehicle dial-a-ride with time windows. *Amer. J. Math. Management Sci.* **6** 301–325.

Dial, R. 1995. Autonomous dial a ride transit—Introductory overview. *Transportation Res.—Part C* **3** 261–275.

Dréo, J., A. Pétrowski, P. Siarry, E. Taillard. 2006. *Metaheuristics for Hard Optimization Methods and Case Studies*. Springer-Verlag, Berlin.

Figliozzi, M., H. Mahmassani, P. Jaillet. 2007. Pricing in dynamic vehicle routing problems. *Transportation Sci.* **41**(3) 302–318.

Gendreau, M., F. Guertin, J. Potvin, E. Taillard. 1999. Parallel tabu search for real-time vehicle routing and dispatching. *Transportation Sci.* **33** 381–390.

George, A., W. Powell. 2005. Adaptive stepsizes for recursive estimation with applications in approximate dynamic programming. http://www.castlelab.princeton.edu/.

Godfrey, G., W. B. Powell. 2002. An adaptive dynamic programming algorithm for stochastic resource allocation problems I: Single period travel times. *Transportation Sci.* **36** 21–39.

Haghani, A., S. Jung. 2005. A dynamic vehicle routing problem with time-dependent travel times. *Comput. Oper. Res.* **32** 2959–2986.

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

Jaw, J., A. Odoni, H. Psaraftis, N. Wilson. 1986. A heuristic algorithm for the multivehicle many-to-many advance-request dial-a-ride problem. *Transportation Res. B: Methodological* **20**(3) 243–257.

Jih, W., J. Yung-Jen. 1999. Dynamic vehicle routing using hybrid genetic algorithms. *Proc. IEEE Internat. Conf. Robotics & Automation, May*, Detroit, MI, 453–458.

Kennedy, J., R. Eberhart. 2001. *Swarm Intelligence*. Morgan Kaufmann Publishers, San Francisco.

Larsen, A. 2000. The dynamic vehicle routing problem, Ph.D. thesis, Technical University of Denmark, Denmark.

Madsen, O., H. Raven, J. Rygaard. 1995. A Heuristics algorithm for a dial-a-ride problem with time windows, multiple capacities, and multiple objectives. *Ann. Oper. Res.* **60** 193–208.

Malandraki, C., M. S. Daskin. 1992. Time dependent vehicle routing problems: Formulations, properties and heuristic algorithms. *Transportation Sci.* **26** 185–200.

Montemanni, R., L. M. Gambardella, A. E. Rizzoli, A. Donati. 2005. Ant colony system for a dynamic vehicle routing problem. *J. Combin. Optim.* **10**(4) 327–343.

Osman, M., M. Abo-Sinna, A. Mousa. 2005. An effective genetic algorithm approach to multiobjective routing problems (MORPs). *Appl. Math. Comput.* **163** 769–781.

Powell, W. B. 1988. A comparative review of alternative algorithms for the dynamic vehicle allocation problem. B. L. Golden, A. A. Assad, eds. *Vehicle Routing Methods and Studies*. North-Holland, Amsterdam.

Powell, W. B., P. Jaillet, A. Odoni. 1995. Stochastic and dynamic networks and routing. M. Ball, T. Magnanti, C. Monma, G. Nemhauser, eds. *Network Routing. A Handbook in Operations Research and Management Science*, Vol. 8. North-Holland, Amsterdam, 141–296.

Psaraftis, H. 1980. A dynamic programming solution to the single many-to-many immediate request dial-a-ride problem. *Transportation Sci.* **14**(2) 130–154.

Psaraftis, H. 1988. Dynamic vehicle routing problems. B. L. Golden, A. A. Assad, eds. *Vehicle Routing Methods and Studies*, North-Holland, Amsterdam, 223–248.

Savelsbergh, M., M. Sol. 1995. The general pickup and delivery problem. *Transportation Sci.* **29**(1) 17–29.

Skrlec, D., M. Filipec, S. Krajcar. 1997. A heuristic modification of genetic algorithm used for solving the single depot capacitated vehicle routing problem. *Proc. Intelligent Inform. Systems, IIS '97*. Washington, D.C. IEEE, 184–188.

Spivey, M., W. B. Powell. 2004. The dynamic assignment problem. *Transportation Sci.* **38**(4) 399–419.

Swihart, M. R., J. D. Papastavrou. 1999. A stochastic and dynamic model for the single-vehicle pick-up and delivery problem. *Eur. J. Oper. Res.* **114** 447–464.

Tarantilis, C. 2005. Solving the vehicle routing problem with adaptive memory programming methodology. *Comput. Oper. Res.* **32** 2309–2327.

Thomas, B., C. White, III. 2004. Anticipatory route selection. *Transportation Sci.* **38**(4) 473–487.

Tighe, A., F. Smith, G. Lyons. 2004. Priority based solver for a real-time dynamic vehicle routing. *IEEE Internat. Conf. Systems, Man and Cybernetics*. IEEE, 6237–6242.

Topaloglu, H., W. Powell. 2005. A distributed decision-making structure for dynamic resource allocation using non linear functional approximations. *Oper. Res.* **53**(2) 281–297.

Topaloglu, H., W. Powell. 2007. Incorporating pricing decisions into the stochastic dynamic fleet management problem. *Transportation Sci.* **41**(3) 281–301.

Toth, P., D. Vigo. 2003. The granular tabu search and its application to the vehicle-routing problem. *INFORMS J. Comput.* **15**(4) 333–346.

Wang, X., J. Xiao. 2005. PSO-based model predictive control for nonlinear processes. *Lecture Notes in Computer Science*, Vol. 3611. Springer-Verlag, Berlin, 196–203.

Wilson, N., N. Colvin. 1977. Computer control of Rochester dial-a-ride system. Report R77-31, Department of Civil Engineering, M.I.T., Cambridge, MA.

Wilson, N., H. Weissberg. 1976. Advanced dial-a-ride algorithms research project: Final report. Report R76-20, Department of Civil Engineering, M.I.T., Cambridge, MA.

Zhu, Q., L. Qian, Y. Li, S. Zhu. 2006. An improved particle swarm optimization algorithm for vehicle routing problem with time windows. IEEE Congress on Evolutionary Computation, Vancouver, July 16–21.