International Journal of Mathematics and Statistics Invention (IJMSI)
E-ISSN: 2321 – 4767 P-ISSN: 2321 - 4759
www.ijmsi.org Volume 2 Issue 1 || January, 2014 || PP 62-67

Dynamic Vehicle Routing Problem in Distributed Service Networks

Dr. Shailendra Kumar
Assistant Professor in Mathematics,
Govt. Raza P.G. College, Rampur

Abstract
The Dynamic Vehicle Routing Problem (DVRP) is one of the most mathematically demanding problems in combinatorial optimization. It asks a deceptively simple question: how do you route a fleet of vehicles efficiently when the problem you are solving keeps changing beneath your feet? In distributed service networks, where demand surfaces simultaneously at many spread-out locations rather than arriving at one central point, this challenge deepens considerably. The mathematics behind it spans graph theory, integer programming, stochastic modeling, Markov decision processes, and computational complexity theory. This article works through those mathematical layers carefully, explaining not just what the formulas say but why they behave the way they do. Topics covered include formal problem structure, degree of dynamism, two-stage stochastic programming, online algorithm theory, MDP-based policy formulations, and the coordination gap between local and global optima in distributed settings.

Keywords: *Integer programming, Stochastic Approach, Combinatorial optimization, Markov decision process, NP-hardness*

I. Introduction

There is a particular kind of mathematical problem that looks straightforward until you start actually solving it. The Vehicle Routing Problem is like that. On the surface, it is just a question about delivery routes. Underneath, it conceals layers of graph structure, combinatorial explosion, and computational hardness that have occupied operations researchers for more than half a century.

The static version, where every customer, demand, and road cost is known in advance, is already NP-hard (Lenstra & Rinnooy Kan, 1981). That means, as far as anyone currently knows, no algorithm can solve every instance efficiently. The time needed to find a guaranteed optimal solution grows faster than any polynomial in the problem size. Add dynamics, requests arriving in real time, travel times shifting, customers canceling, and the problem does not just get harder. It changes fundamental character. It stops being a single optimization problem and becomes a sequential decision problem: a policy that must map every possible state of the world to a good action, over and over, as the state keeps evolving.

That transformation is the mathematical heart of this article. We will trace it carefully, from the clean graph-theoretic foundations of the VRP, through the integer programming machinery that formalizes it, into the stochastic and sequential frameworks that the dynamic version demands. Along the way, we will pay particular attention to distributed service networks, where decentralization breaks one of the most convenient assumptions classical models make: that there is one decision-maker with one coherent view of the entire system.

II. The VRP as a Graph Problem: Starting with Structure

Every formulation of the VRP begins with a graph. Let G = (V, A) be a complete directed graph. The vertex set is V = {0, 1, 2, ..., n}, where vertex 0 is the depot and vertices 1 through n are customers. The arc set A contains every ordered pair (i, j) with $i \neq j$, and each arc carries a non-negative cost $c_{\{ij\}}$, typically distance or travel time.

The number of possible distinct routes through n customers is (n−1)! divided by a symmetry factor if the graph is undirected. For n = 20, that is roughly $6 \times 10^{\{17\}}$ routes to consider. For n = 50, the count exceeds the number of atoms in the observable universe. This is why exact enumeration is instantly hopeless and why the mathematical structure of the problem matters so much, only by exploiting structure can any algorithm make progress.

The graph-theoretic object we actually want is a set of vertex-disjoint cycles, each passing through the depot exactly once, such that every non-depot vertex appears in exactly one cycle. These cycles are the routes. The union of all cycles must span all of V. This "cycle cover" perspective connects the VRP to well-studied graph problems like the Traveling Salesman Problem (TSP), where a single Hamiltonian cycle must visit every vertex exactly once. The VRP is, loosely speaking, a partitioned TSP with capacity constraints on each partition.

Laporte (1992) gives the cleanest survey of how this graph structure is exploited in exact methods, particularly through branch-and-bound search trees where the branching decisions correspond to including or excluding specific arcs from the solution.

III. Integer Programming: Mathematical Formulation

Graph theory describes the solution structure. Integer programming provides the computational process.

The standard three-index VRP formulation uses binary decision variables $x_{\{ijk\}} \in \{0, 1\}$, where $x_{\{ijk\}} = 1$ means vehicle k traverses arc (i, j). The objective is:

$$\boldsymbol{min} \sum_{\boldsymbol{k=1}}^{\boldsymbol{K}} \sum_{\boldsymbol{(i,j)\in A}} \boldsymbol{c_{ij}\, x_{ijk}}$$

This is subject to several constraint families. First, degree constraints ensure every customer is visited exactly once:

$$\sum_{k} \sum_{j \neq i} x_{ijk} = 1, \forall i \in \{1, \dots, n\}$$

Second, flow conservation ensures that if a vehicle enters a node, it also exits:

$$\sum_{j} x_{ihk} - \sum_{j} x_{hjk} = 0, \forall h \in \{1, \dots, n\}, \forall k$$

Third, capacity constraints ensure that the total demand on each vehicle's route does not exceed vehicle capacity Q:

$$\sum_{i} \sum_{j} d_i \cdot x_{ijk} \leq Q, \forall k$$

Fourth, and this is mathematically subtle, subtour elimination constraints prevent the solution from containing small disconnected loops that never return to the depot. The standard formulation uses:

$$\sum_{i \in S} \sum_{j \notin S} x_{ijk} \geq 1, \forall S \subseteq \{1, \dots, n\}, S \neq \emptyset, \forall k$$

There are exponentially many of these constraints, one for each non-empty subset S, which is why they cannot all be added upfront. Instead, solvers generate them on demand, adding only the constraints that are violated by the current solution. This technique, called cutting planes or row generation, is central to modern branch-and-cut solvers (Toth & Vigo, 2002).

Adding time windows $[a_i, b_i]$ introduces auxiliary continuous variables $\tau_{\{ik\}}$ representing the time vehicle k begins service at customer i. The constraints $\tau_{\{ik\}} \geq a_i$ and $\tau_{\{ik\}} \leq b_i$ must both hold. This coupling between binary routing variables and continuous timing variables makes the VRPTW a mixed-integer program, genuinely harder than the pure integer case (Cordeau et al., 2002).

IV. Mathematical Formulation of DVR Problem

4.1 Events, States, and the Information Set

In the static VRP, everything is known at time t = 0. The mathematical object is fixed: a graph, a set of demands, a fleet, a cost function. Solve once, implement.

In the DVRP, the problem is parameterized by time. At each moment t, the decision-maker has access to an information set I(t), the collection of everything known up to and including time t. This set grows as time advances. New requests arrive. Vehicle positions update. Road conditions change.

Formally, denote the set of customers known at time t as C(t) ⊆ {1, ..., n}, where n is the eventual total number of customers, unknown in advance. Decisions must be made based only on I(t), even though the true eventual problem involves all n customers. This is the key mathematical distinction between static and dynamic optimization: the constraint set is not fully visible when decisions are being made.

Psaraftis (1988) was the first to formalize this carefully, distinguishing between decisions made at dispatch time and decisions made reactively during execution. His framework established that a dynamic routing algorithm must be understood as a function from information states to actions, not as a one-time solver.

4.2 Degree of Dynamism: Measuring How Dynamic a Problem Really Is

Not all DVRP instances are equally dynamic. The degree of dynamism, introduced by Larsen (2001) and formalized in Larsen et al. (2002), gives a precise scalar measure:

$$\boldsymbol{DoD} = \frac{\boldsymbol{n_d}}{\boldsymbol{n}}$$

Here, n_d counts requests that arrive after the initial routing plan is committed, and n is the total eventual number of requests. DoD = 0 is fully static. DoD = 1 means every single request is unknown at dispatch time.
The mathematically interesting behavior happens in between. When DoD is small, the initial plan handles most of the demand well and re-optimization is a local correction. When DoD is large, the plan is mostly placeholder and reactive dispatch dominates. The worst situation, from an optimization standpoint, is intermediate DoD, roughly 0.3 to 0.6 (Pillac et al., 2013). Here, a planner commits resources based on partial information, and those commitments constrain future flexibility in ways that turn out to be costly.
As illustrated in Figure, this produces a characteristic non-linear relationship between DoD and solution quality degradation.

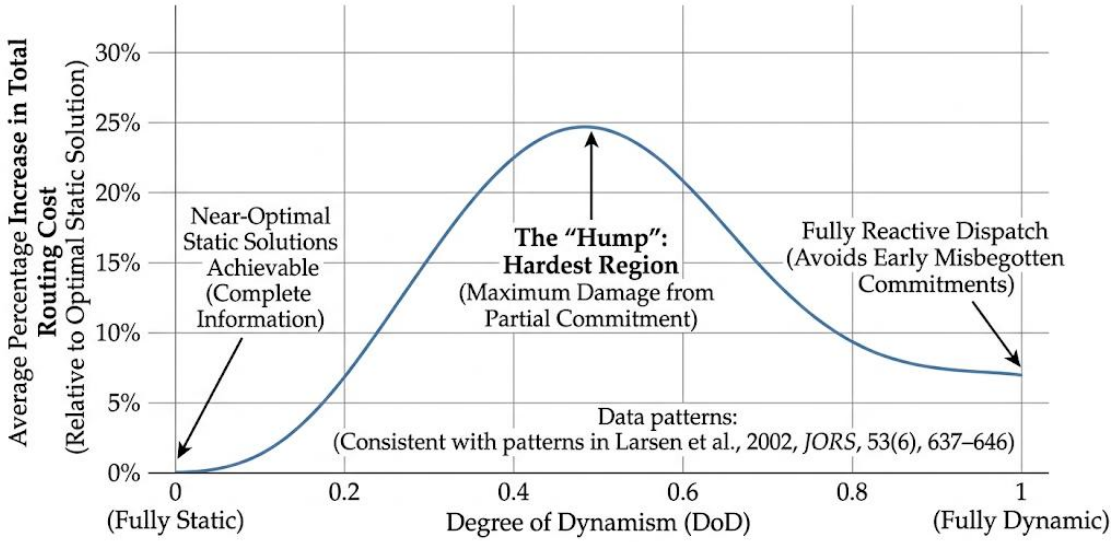

Figure: Degree of Dynamism vs. Percentage Increase in Total Routing Cost Relative to Optimal Static Solution

This graph places degree of dynamism (DoD) on the horizontal axis, ranging from 0 (fully static) to 1 (fully dynamic), and plots the average percentage increase in total routing cost on the vertical axis. The resulting curve is non-linear and peaks in the DoD ≈ 0.4–0.6 range, reflecting maximum damage from partial commitment under incomplete information. At DoD ≈ 0, near-optimal static solutions are achievable. At DoD ≈ 1, fully reactive dispatch avoids the cost of early misbegotten commitments. The hump in between is the hardest region. Data patterns are consistent with empirical and analytical findings in Larsen et al. (2002), *Journal of the Operational Research Society*, 53(6), 637–646.

5. Stochastic Programming: When You Know the Odds but Not the Outcomes

One mathematically powerful response to uncertainty is stochastic programming. Instead of treating unknown future demands as arbitrary surprises, this approach models them as random variables with known probability distributions.
In the two-stage stochastic VRP, the framework goes like this. In the first stage, a base routing plan x is built using expected demand information, specifically, using $E[d_i]$ for each customer i. This is the "here and now" decision. In the second stage, after actual demands d_i are revealed, a corrective recourse action y(x, d) is taken. The total objective minimizes expected cost across both stages:

$$\boldsymbol{min}\ f(x) + E_{d[g(y(x,d))]}$$

where f(x) is the first-stage routing cost, g(·) is the recourse cost (such as a vehicle returning to depot for restocking), and the expectation is taken over the demand distribution.
Gendreau et al. (1996) showed that this recourse structure makes the problem mathematically tractable when demand distributions have standard forms like uniform or Poisson. When a vehicle arrives at customer i and finds that d_i exceeds its remaining capacity, the recourse action, returning to depot, has a computable expected cost that can be incorporated into the first-stage optimization. This elegantly turns an uncertainty problem into a deterministic expected-value problem.
The Poisson demand model is particularly natural. If each customer i independently generates demand with rate λ_i, the probability that total route demand exceeds Q follows from the convolution of Poisson distributions. For routes with many customers, the normal approximation applies by the Central Limit Theorem, giving closed-form expressions for failure probability. Gendreau et al. (1996) used this to derive analytically tractable bounds on expected recourse cost.

The core limitation is distributional. If demands are non-stationary, if Monday's demand profile looks nothing like Friday's, a single distribution fits neither well. In distributed networks with heterogeneous demand nodes, this problem compounds across locations.

6. Markov Decision Processes: The Full Theoretical Framework

The deepest and most general mathematical formulation of the DVRP treats it as a Markov Decision Process (MDP). This framework makes explicit what the others leave implicit: the sequential, state-dependent nature of dynamic decision-making.

An MDP has four components: a state space S, an action space A, transition probabilities P(s'|s, a), and a reward function R(s, a). At each time step, the system is in some state s ∈ S. The decision-maker chooses action a ∈ A(s). The system transitions to state s' with probability P(s'|s, a) and returns reward R(s, a).

In the DVRP, the state s at time t encodes: the positions of all vehicles, the set of customers already served, pending confirmed requests, and a belief distribution over future arrivals. The action a is a dispatch decision: which vehicle moves to which customer (or waits, or repositions). Transition probabilities capture the stochasticity of travel times and future demand arrivals.

The goal is a policy π: S → A that maps states to actions and maximizes expected total reward (or minimizes total cost) over the planning horizon. The optimal policy satisfies the Bellman equation:

$$V*(s) = \max_{a \in A(s)} \left[R(s,a) + \gamma \sum_{s'} P(s'|s,a)\, V*(s') \right]$$

where V*(s) is the optimal value function and γ ∈ [0,1] is a discount factor.

This is mathematically complete and theoretically beautiful. Computationally, it is devastating. The state space of the DVRP is exponentially large, it includes every possible combination of vehicle positions, pending requests, and belief states. Berbeglia et al. (2010) establish formally that solving the general DVRP exactly within this MDP framework is PSPACE-hard, a complexity class strictly above NP. This is not a gap waiting to be closed by a smarter algorithm. It is a fundamental mathematical barrier that justifies, rigorously, why heuristics are the dominant practical approach.

7. Distributed Networks: Decentralization Breaks Classical Assumptions

7.1 Multi-Depot VRP and Its Integer Program

Classical VRP assumes one depot. Real distributed service networks often have many. In the multi-depot VRP, let D = {0_1, 0_2, ..., 0_m} be the set of depot nodes. Each vehicle k is assigned to home depot d(k) and must begin and end its route there. The variable set now includes y_{ik} ∈ {0,1}, indicating whether customer i is assigned to a vehicle based at depot k.

The depot assignment constraints add:

$$\sum_{k:\, d(k)=r} \sum_{j} x_{0_r,j,k} \leq |fleet\ at\ depot\ r|, \forall r$$

And symmetrically for route returns. In dynamic settings, these depot assignments can shift mid-operation, a vehicle completing a task near depot r_2 may be reassigned from depot r_1, invalidating part of the original plan. Li et al. (2012) showed that iterated local search with adaptive neighborhood selection handles multi-depot DVRP with simultaneous pickups and deliveries effectively, producing solutions within a few percent of best-known benchmarks on standard test instances.

7.2 The Coordination Gap: A Mathematical Statement

When multiple autonomous agents make decisions locally, each optimizing its own objective with only local information, the aggregate behavior diverges from global optimality. This divergence has a name: the coordination gap.

Formally, let C* denote the cost of the globally optimal centralized solution with full information sharing. Let C_L denote the cost achieved by a distributed system where agents make locally optimal decisions without communicating. The coordination gap is simply $C_L - C*$, and it is always non-negative.

What makes this mathematically interesting is how the gap behaves. During low-demand periods, local and global optima tend to coincide: each agent can independently handle its local demand, and coordination would buy little. During high-demand surges, one agent may be overwhelmed while a nearby agent sits underloaded. Without information sharing, this imbalance persists; with it, re-assignment corrects it. The gap therefore widens exactly when it is most costly, at peak demand (Zeimpekis et al., 2007).

8. Online Competitive Analysis: Measuring How Good Any Dynamic Algorithm Can Be

One important strand of DVRP theory asks a prior question: how good can any online algorithm possibly be, given that it must act without knowing the future?

The online algorithm framework formalizes this through competitive ratio. Let ALG(σ) be the cost incurred by an online algorithm on request sequence σ. Let OPT(σ) be the cost of the globally optimal offline solution with full hindsight. The competitive ratio of ALG is:

$$\boldsymbol{CR(ALG)} = \max_{\boldsymbol{\sigma}}[\boldsymbol{ALG(\sigma)} / \boldsymbol{OPT(\sigma)}]$$

A competitive ratio of 1 would mean the online algorithm always matches hindsight, obviously impossible in general. Larger ratios mean worse worst-case performance relative to the oracle.

For the Online Traveling Salesman Problem on the real line, the competitive ratio of the best known algorithm is 3/2, achieved by a strategy called SMARTSTART (Afrati et al., 1997, as cited in Pillac et al., 2013). For the 2D plane and general metric spaces, tight bounds are harder to obtain. Savelsbergh and Sol (1995) provide foundational analysis of the general pickup and delivery problem structure that underpins many of these competitive ratio arguments.

The competitive ratio is not always the right metric for practical DVRP, since worst-case sequences may be unrealistic for specific applications. But it is valuable as a theoretical ceiling, if the competitive ratio of an algorithm is very large, it warns that some input sequence exists where performance will be badly degraded, even if that sequence rarely occurs in practice.

9. Conclusion

The Dynamic Vehicle Routing Problem, especially in distributed service networks, is genuinely one of the hardest problems in applied mathematics. That statement is not rhetorical. It has a precise meaning: the general formulation is PSPACE-hard, its distributed multi-agent extension is NEXP-complete, and even polynomial approximations with bounded error ratios are not known for all problem variants.

What the mathematics gives us, beyond those hardness results, is structure. The graph-theoretic formulation tells us what a solution looks like. The integer program tells us what constraints it must satisfy. Stochastic programming tells us how to reason about uncertainty in a principled way. The MDP framework tells us that routing policy, not routing solution, is the right unit of analysis when dynamics are present. Competitive ratio analysis tells us how to evaluate online algorithms against a theoretical baseline. Tabu search and scenario-based optimization tell us how to make near-optimal decisions fast enough to be actionable.

None of these individually solves the problem. Together, they define the problem so clearly that meaningful progress, both theoretical and practical, becomes possible.

Dantzig and Ramser (1959) wrote down a clean combinatorial problem about trucks and depots. Sixty years of mathematics has revealed just how deep that problem goes. The dynamic distributed version they could not have anticipated is deeper still, and the mathematics needed to properly understand it is still, in several important ways, being written.

References

[1]. Azi, N., Gendreau, M., & Potvin, J.-Y. (2010). An exact algorithm for a vehicle routing problem with time windows and multiple use of vehicles. *European Journal of Operational Research*, *202*(3), 756–763. https://doi.org/10.1016/j.ejor.2009.06.034

[2]. Bent, R., & Van Hentenryck, P. (2004). Scenario-based planning for partially dynamic vehicle routing with stochastic customers. *Operations Research*, *52*(6), 977–987. https://doi.org/10.1287/opre.1040.0124

[3]. Berbeglia, G., Cordeau, J.-F., Gama, I., & Laporte, G. (2010). Dynamic pickup and delivery problems. *European Journal of Operational Research*, *202*(1), 8–15. https://doi.org/10.1016/j.ejor.2009.04.024

[4]. Bernstein, D. S., Givan, R., Immerman, N., & Zilberstein, S. (2002). The complexity of decentralized control of Markov decision processes. *Mathematics of Operations Research*, *27*(4), 819–840. https://doi.org/10.1287/moor.27.4.819.297

[5]. Branke, J., Middendorf, M., Noeth, G., & Dessouky, M. (2005). Waiting strategies for dynamic vehicle routing. *Transportation Science*, *39*(3), 298–312. https://doi.org/10.1287/trsc.1050.0114

[6]. Clarke, G., & Wright, J. W. (1964). Scheduling of vehicles from a central depot to a number of delivery points. *Operations Research*, *12*(4), 568–581. https://doi.org/10.1287/opre.12.4.568

[7]. Cordeau, J.-F., Desaulniers, G., Desrosiers, J., Solomon, M. M., & Soumis, F. (2002). VRP with time windows. In P. Toth & D. Vigo (Eds.), *The vehicle routing problem* (pp. 157–193). SIAM.

[8]. Dantzig, G. B., & Ramser, J. H. (1959). The truck dispatching problem. *Management Science*, *6*(1), 80–91. https://doi.org/10.1287/mnsc.6.1.80

[9]. Gendreau, M., Guertin, F., Potvin, J.-Y., & Taillard, É. (1999). Parallel tabu search for real-time vehicle routing and dispatching. *Transportation Science*, *33*(4), 381–390. https://doi.org/10.1287/trsc.33.4.381

[10]. Gendreau, M., Laporte, G., & Séguin, R. (1996). Stochastic vehicle routing. *European Journal of Operational Research*, *88*(1), 3–12. https://doi.org/10.1016/0377-2217(95)00050-X

[11]. Haghani, A., & Jung, S. (2005). A dynamic vehicle routing problem with time-dependent travel times. *Computers & Operations Research*, *32*(11), 2959–2986. https://doi.org/10.1016/j.cor.2004.04.013

[12]. Laporte, G. (1992). The vehicle routing problem: An overview of exact and approximate algorithms. *European Journal of Operational Research*, *59*(3), 345–358. https://doi.org/10.1016/0377-2217(92)90192-C

[13]. Larsen, A. (2001). *The dynamic vehicle routing problem* [Doctoral dissertation, Technical University of Denmark]. DTU Orbit.

[14]. Larsen, A., Madsen, O. B. G., & Solomon, M. M. (2002). Partially dynamic vehicle routing — Models and algorithms. *Journal of the Operational Research Society*, *53*(6), 637–646. https://doi.org/10.1057/palgrave.jors.2601350
[15]. Lenstra, J. K., & Rinnooy Kan, A. H. G. (1981). Complexity of vehicle routing and scheduling problems. *Networks*, *11*(2), 221–227. https://doi.org/10.1002/net.3230110211
[16]. Li, J., Pardalos, P. M., Sun, H., Pei, J., & Zhang, Y. (2012). Iterated local search embedded adaptive neighborhood selection approach for the multi-depot vehicle routing problem with simultaneous deliveries and pickups. *Expert Systems with Applications*, *39*(3), 2715–2723. https://doi.org/10.1016/j.eswa.2011.08.158
[17]. Pillac, V., Gendreau, M., Guéret, C., & Medaglia, A. L. (2013). A review of dynamic vehicle routing problems. *European Journal of Operational Research*, *225*(1), 1–11. https://doi.org/10.1016/j.ejor.2012.08.004
[18]. Psaraftis, H. N. (1988). Dynamic vehicle routing problems. In B. L. Golden & A. A. Assad (Eds.), *Vehicle routing: Methods and studies* (pp. 223–248). North-Holland.
[19]. Savelsbergh, M. W. P., & Sol, M. (1995). The general pickup and delivery problem. *Transportation Science*, *29*(1), 17–29. https://doi.org/10.1287/trsc.29.1.17
[20]. Solomon, M. M. (1987). Algorithms for the vehicle routing and scheduling problems with time window constraints. *Operations Research*, *35*(2), 254–265. https://doi.org/10.1287/opre.35.2.254
[21]. Toth, P., & Vigo, D. (Eds.). (2002). *The vehicle routing problem*. SIAM.
[22]. Zeimpekis, V., Tarantilis, C. D., Giaglis, G. M., & Minis, I. (Eds.). (2007). *Dynamic fleet management: Concepts, systems, algorithms and case studies*. Springer.