Uma abordagem baseada em clusterização hierárquica para resolver o Last-Mile Incremental Capacitated Vehicle Routing Problem (ICVRP)

Ruan Heleno Correa da Silva
rhcs@ic.ufal.br
Instituto de Computação, Universidade Federal de Alagoas
Maceió, Alagoas, BRA

Ascânio Sávio de Araujo Neves
asan2@ic.ufal.br
Instituto de Computação, Universidade Federal de Alagoas
Maceió, Alagoas, BRA

Rian Gabriel Santos Pinheiro
rian@ic.ufal.br
Instituto de Computação, Universidade Federal de Alagoas
Maceió, Alagoas, BRA

Bruno Costa e Silva Nogueira
bruno@ic.ufal.br
Instituto de Computação, Universidade Federal de Alagoas
Maceió, Alagoas, BRA

RESUMO

O objetivo desse projeto é resolver o *Last-Mile Incremental Capacitated Vehicle Routing Problem* (Last-Mile ICVRP) a partir da predição de modelos clusterizados organizados de maneira hierárquica. A hierarquia é constituída por dois níveis. No primeiro nível, o modelo inicial contém uma quantidade pré-definida k de clusters. No segundo nível, cada cluster $i \in \{1, \ldots, k\}$ tem M_i subclusters, tal que M_i é definido pelo algoritmo de distribuição das unidades de carregamento. A partir disso, cada subcluster do segundo nível será responsável por uma unidade de carregamento. Dessa forma, as entregas são encaminhadas para as unidades de carregamento a partir da predição de tais modelos. Nos resultados parciais obtidos, o algoritmo desenvolvido a partir dessa abordagem conseguiu bons resultados ao ser comparado com os resultados de outros algoritmos.

KEYWORDS

Clusterização, Last-Mile ICVRP

ACM Reference Format:
Ruan Heleno Correa da Silva, Ascânio Sávio de Araujo Neves, Rian Gabriel Santos Pinheiro, and Bruno Costa e Silva Nogueira. 2021. Uma abordagem baseada em clusterização hierárquica para resolver o Last-Mile Incremental Capacitated Vehicle Routing Problem (ICVRP). In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUÇÃO

O *Last-Mile Incremental Capacitated Vehicle Routing Problem* (Last-Mile ICVRP) pode ser descrito como se segue. Dado um conjunto (histórico) de clientes atendidos anteriormente $H = \{v_1, \ldots, v_\ell\}$, um conjunto $\mathcal{U}$ de unidades de carregamento e um conjunto com n novos clientes $D = \{v_{\ell+1}, \ldots, v_{\ell+n}\}$, o Last-Mile ICVRP tem como objetivo alocar cada uma das entregas do conjunto D a uma unidade de carregamento. Cada demanda é alocada, uma por vez, em uma única unidade de carregamento de acordo com a ordem de chegada pré-definida. As unidades de carregamento tem sua capacidade máxima que, quando atingida, precisa ser despachada para entrega. O problema busca achar uma alocação das entregas às unidades de carregamento, de maneira que a distância percorrida pelos veículos de entrega seja minimizada.

O *Last-Mile Incremental Capacitated Vehicle Routing Problem* (Last-Mile ICVRP) é um caso particular do problema SD-CVRP (Stochastic and Dynamic Capacitated Vehicle Routing Problems) [6]. O problema é denominado Last-Mile por caracterizar as entregas entre o centro de expedição e o destinatário final.

De maneira geral, a resolução dos problemas relacionados ao SD-CVRP pode ser dividida em duas etapas. A primeira etapa, chamada de *planejamento*, ocorre de maneira *offline* e utiliza dados históricos de entrega para computar um conjunto de decisões pré-definidas. A segunda etapa, chamada de *execução*, ocorre em tempo real e utiliza as ações pré-definidas na etapa anterior assim como as informações sobre os eventos em tempo real (ex.: chegada de uma nova demanda) para gerar as rotas finais.

O objetivo desse projeto é resolver o *Last-Mile Incremental Capacitated Vehicle Routing Problem* (Last-Mile ICVRP) usando uma estratégia baseada em clusterização hierárquica, organizada de maneira hierárquica em dois níveis.

2 TRABALHOS RELACIONADOS

Tradicionalmente, os VRP dinâmicos e estocásticos são formulados como processos de Markov [5, 8, 9] ou como programação estocástica [1]. Ambas as metodologias modelam os dados do problema como variáveis aleatórias que seguem uma distribuição previamente conhecida. O objetivo é otimizar uma medida de risco (como o valor esperado, a variância ou o valor em risco condicional de alguma função de custo), sujeito à satisfação de algumas restrições [3]. Note que ambas as abordagens não estão alinhadas com o problema encontrado na Loggi.

A partir de apresentações institucionais da empresa Loggi, é possível identificar duas estratégias recentes para resolver o Last-Mile ICVRP. A primeira estratégia foi apresentada em [2] e usa

Conference'17, July 2017, Washington, DC, USA

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

uma combinação de agentes autônomos com mineração de dados. Nesta abordagem, os agentes autônomos modelam as rotas a serem construídas. Além disso, técnicas de mineração de dados são usadas para extrair informações estocásticas sobre a distribuição de pacotes e assim otimizar a alocação de cada pacote em uma das rotas. Para determinar em qual rota r novo pacote p será alocado, a seguinte função é maximizada: $bet(p,r) = \alpha * bet_{DataMining}(p,r) + (1-\alpha) * bet_{Distance}(p,r)$, onde $bet_{DataMining}(p,r)$ é um valor extraído da mineração de dados, $bet_{Distance}(p,r)$ é uma função que mede a distância entre o pacote p e os pacotes já inseridos na rota r, e α é um valor no intervalo $[0,1]$. Os resultados mostram que o número de rotas que precisam ser geradas pela abordagem proposta é em média superior em 17% quando comparado com o número de rotas gerado por um solver para a versão estática do *CVRP*.

A segunda estratégia da empresa para resolver o Last-Mile IC-VRP foi apresentada em video [7]. A partir do vídeo, é possível depreender que a estratégia consiste em duas etapas: uma etapa de clusterização rotas boas geradas a partir do histórico de entregas (*offline*), seguida de uma etapa alocação dos pacotes em um dos clusters de rotas (*online*). Três métodos de clusterização (*bag of cells*, distância euclidiana e distância entre ruas) e dois métodos de alocação de rotas (guloso e clusters artificiais) foram experimentados. A combinação do método de clusterização baseado em distância entre ruas com o de alocação usando clusters artificiais apresentou os melhores resultados. Estes resultados mostram que a solução apresentada é apenas 16% pior que as encontradas por meio de um solver para a versão estática do *CVRP*.

3 MÉTODO PROPOSTO

Para explicar o funcionamento do algoritmo desenvolvido, as principais funcionalidades foram divididas em cinco tópicos.

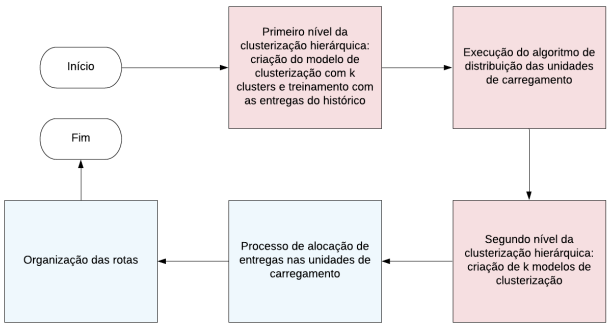

Figura 1: Fluxograma do método proposto.

A Figura 1 mostra o fluxograma do algoritmo desenvolvido utilizando uma abordagem baseada em Clusterização Hierárquica para resolver o *Last-Mile Incremental Capacitated Vehicle Routing Problem(ICVRP)*. Em vermelho, estão relacionados os processos referente à primeira etapa, chamada de *planejamento*, que ocorre de maneira *offline*. Em azul, estão relacionados os processos referente à segunda etapa, chamada de *execução*, que ocorre em tempo real.

3.1 Primeiro Nível da Clusterização Hierárquica

O primeiro nível é constituído pelo modelo de clusterização inicial. No algoritmo desenvolvido, é executado o algoritmo *K-Means* sobre as entregas com o valor pré-definido de k. O modelo é treinado a partir dos pontos das entregas presentes no histórico.

3.2 Distribuição das Unidades de Carregamento

A distribuição das Unidades de Carregamento tem como objetivo fornecer a quantidade de unidades de carregamentos para cada cluster. Ao utilizar as entregas do histórico e separar tais entregas em seus determinados clusters, a distribuição é feita de forma proporcional à quantidade de entregas presentes nesses clusters em relação a quantidade total de entregas. Com isso, multiplicamos cada percentual pelo número de unidades de carregamento. Por ser um número inteiro é necessário depois fazer o balanceamento para que a quantidade de unidades de carregamento que foram distribuídas seja igual ao número de unidades de carregamento inicial, além de buscar garantir que cada cluster tenha, no mínimo, uma unidade de carregamento. O balanceamento é feito removendo uma unidade de carregamento, por iteração, do primeiro cluster que tem mais de uma unidade de carregamento.

3.3 Segundo Nível da Clusterização Hierárquica

O segundo nível é constituído por k modelos de clusterização. A quantidade de clusters de cada modelo será o valor dado pelo algoritmo de distribuição de unidades de carregamento, de modo que cada modelo a ser criado será referente a um cluster do modelo inicial. A clusterização hierárquica busca tornar cada cluster dos modelos criados do segundo nível responsável por uma unidade de carregamento. Com isso, no algoritmo desenvolvido, cada modelo $i \in \{1,\ldots,k\}$ tem M_i subclusters, tal que M_i é definido pelo algoritmo de distribuição das unidades de carregamento, e com isso, é com as entregas do histórico referente àquele cluster.

3.4 Alocação das Entregas nas Unidades de Carregamento

A partir dos modelos, a análise de cada entrega será feita usando predições. Inicialmente é feita a predição da entrega a ser alocada no modelo do primeiro nível. O valor da predição da entrega no modelo do primeiro nivel será usado para identificar o modelo do segundo nível. Com isso, a predição do modelo do segundo nível será usada para identificar o cluster responsável pela unidade de carregamento.

No Algoritmo 1, é mostrado o pseudocódigo referente à alocação de entregas nas unidades de carregamento. A entrada contém um array com as entregas a serem alocadas, um array referente à distribuição, um array referente as unidades de carregamento, o modelo K-Means referente ao primeiro nível e o array de modelos K-Means referente ao segundo nível da clusterização hierárquica.

No algoritmo desenvolvido, aplicamos no ponto da entrega atual, a função *predict*, presente no algoritmo *K-Means*, que busca predizer o cluster mais próximo à um determinado ponto, no modelo do primeiro nível e no modelo do segundo nível, e assim identificamos o número referente a unidade de carregamento escolhida.

Após as predições, utilizamos um loop, que vai de 0 até o $u-1$, para identificar a posição da unidade de carregamento no array que

foi retornado pelo algoritmo de distribuição de unidades de carregamento. Após identificação, é feita a análise dos pesos já presentes naquela unidade de carregamento para saber se ao somar o peso da entrega, o valor total ultrapassa a capacidade. Caso ultrapasse, antes de adicionar a entrega atual, a unidade de carregamento contendo entregas já alocadas, é despachada. Depois dessa verificação, a entrega atual é alocada na determinada unidade de carregamento.

Algorithm 1 Pseudocódigo referente à alocação das entregas nas unidades de carregamento

Require: $deliveries : [Delivery], D : [int], UCS : [UC], Model1 : KMeans, Models2 : [KMeans]$
Ensure: $routes \leftarrow [], u \leftarrow len(UCS)$
  for *delivery* in *deliveries* **do**
    $k \leftarrow Model1.predict(delivery.point)$
    $M \leftarrow Models2[k].predict(delivery.point)$
    for i in $range(0, u)$ **do**
      if $k = D[i]$ **then**
        $M \leftarrow M - 1$
        $uc \leftarrow i$
      end if
      if $M < 0$ **then**
        break
      end if
    end for
    if $(UCS[uc].size + delivery.size) > capacity\_vehicles$ **then**
      $routes.push(UCS[uc].deliveries)$
      $UCS[uc].pop(UCS[uc].deliveries)$
    end if
    $UCS[uc].push(delivery)$
  end for

3.5 Organização das rotas

O despacho de uma unidade de carregamento acontece quando a capacidade máxima da mesma é atingido. Com isso, é executado o algoritmo que busca uma solução para o *TSP(Traveling Salesman Problem)* em cada conjunto de entregas. No fim, o mesmo processo de despacho acontece nas unidades de carregamento que não ultrapassaram a capacidade, executando o determinado algoritmo para organizar as rotas.

4 RESULTADOS PARCIAIS/PRELIMINARES

O algoritmo do projeto foi desenvolvido na linguagem de programação *Python*. Foi utilizado o método *K-Means*, da biblioteca *Sklearn*, como método de clusterização, além da utilização da biblioteca *Numpy*. Houve a instalação do servidor *OSRM* para utilizar as distâncias do *OpenStreetMap* e a aplicação do algoritmo implementado com a ferramenta *OR-Tools*, do *Google*, que busca resolver o *TSP* e *CVRP* disponibilizado pelo repositório *loggibud* [4], da *Loggi*, presente na plataforma *GitHub*.

Como forma de avaliação do algoritmo desenvolvido foi utilizado as instâncias cvrp, */train* e */dev*, dos diretórios *df-0*, *pa-0* e *rj-0* presentes no banco de dados disponibilizado pela *Loggi*. A pasta */train*, contém os dados que utilizamos como o histórico, distribuídos em 90 instâncias. A pasta */dev* é utilizada para gerar os resultados, de modo que ela contém 30 instâncias de testes.

O seguinte formato foi usado para encontrar um resultado a ser considerado geral de cada diretório:

(1) Inicialmente, separamos as entregas das instâncias de treino em batches (ou lotes) para que o algoritmo referente ao CVRP consiga encontrar uma solução. O tamanho do batch escolhido para os resultados foi 300, pois foi o tamanho que o algoritmo do arquivo */loggibud/v1/baselines/shared/ortools.py*, do repositório *loggibud*, consegue resolver em tempos razoáveis.
(2) Buscamos a distância percorrida das soluções encontradas para cada instância. As distâncias percorridas encontradas denominamos em resultados *off-line*.
(3) Executamos os algoritmos, de modo que a pegamos a distância percorrida das soluções geradas, utilizando a função *evaluate_solution*, presente no repositório *loggibud*, no arquivo */loggibud/v1/eval/task1.py*, e fazemos o cálculo da diferença em relação ao resultado obtido de cada instância do passo 1.
(4) Fazemos o cálculo da média dos percentuais, somando o cálculo da diferença encontrada entre o determinado algoritmo e o algoritmo que busca resolver o CVRP de cada instância (off-line) e dividimos pela quantidade de instâncias presente no diretório. O resultado do cálculo da média será considerado o resultado geral do diretório.

Como forma de comparação, utilizamos os resultados encontrados após a execução de dois algoritmos presentes no repositório *loggibud*, na pasta */loggibud/v1/baselines/task2*: *kmeans_greedy.py* e *qrp_sweep.py* com os valores de k no intervalo de 4 a 10.

4.1 Diretório Pará (pa-0)

O diretório pa-0 contém mais de 25000 entregas em seu histórico. Por ter uma média próxima de 300 entregas por instância, consideramos esse diretório como um diretório com pequenas instâncias, em comparação com os outros dois diretórios utilizados. Para os testes, foram 14 unidades de carregamento no algoritmo referente à Clusterização Hierárquica.

Tabela 1: Resultados encontrados do diretório pa-0

N_clusters	C_Hierárquica	KMeans_greedy	QRP_sweep
4	18.50%	**16.34%**	41.30%
5	**14.96%**	17.22%	24.61%
6	21.99%	23.89%	33.68%
7	21.55%	23.17%	34.86%
8	22.22%	22.44%	30.91%
9	24.93%	25.34%	25.99%
10	26.29%	26.72%	**20.79%**

A Tabela 1 apresenta os resultados referentes a instância pa-0. Nesta tabela, as colunas 2, 3, 4. representam os resultados, para cada quantidade de cluster, dos métodos: *clusterização hierárquica*, *kmeans_greedy.py* e *qrp_sweep.py*. Em negrito, destacamos os melhores resultados de cada método.

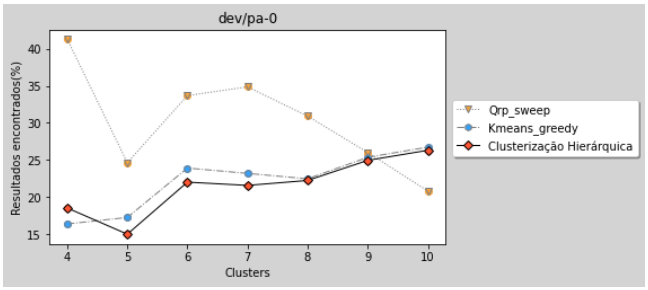

Figura 2: Comportamento dos resultados encontrados do diretório pa-0.

A Figura 2 mostra o comportamento dos resultados encontrados de pa-0 dos três métodos no intervalo de número de clusters utilizado.

4.2 Diretório Distrito Federal (df-0)

O diretório df-0 contém mais de 88000 entregas em seu histórico. Por ter uma média próxima de 1000 entregas por instância, consideramos esse diretório como um diretório com instâncias médias, em comparação com os outros dois diretórios utilizados. Para os testes, foram 14 unidades de carregamento no algoritmo referente à Clusterização Hierárquica.

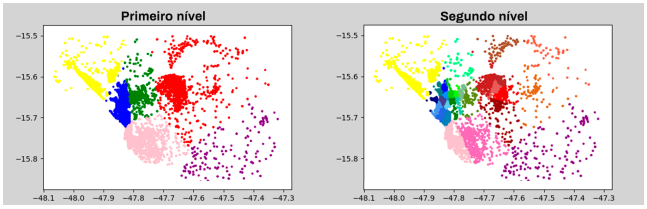

Figura 3: Aplicação da Clusterização Hierárquica com dois níveis de clusterização.

A Figura 3 apresenta um exemplo de aplicação utilizando as entregas do histórico de df-0. Clusterização com 6 clusters no primeiro nível, e 28 clusters no segundo nível distribuído entre os 6 modelos.

Tabela 2: Resultados encontrados do diretório df-0

clusters	C_Hierárquica	KMeans_greedy	QRP_sweep
4	16.94%	18.21%	25.56%
5	14.19%	17.22%	20.53%
6	**10.55%**	**12.04%**	23.30%
7	12.95%	15.41%	19.92%
8	18.60%	16.88%	18.19%
9	18.40%	16.92%	**18.09%**
10	17.09%	17.68%	21.56%

De forma similar à Tabela 1 e à Figura 2, temos a Tabela 2 e a Figura 4, respectivamente. Eles apresentam os resultados encontrados em relação ao algoritmo off-line para cada número de cluster nas instâncias e seu comportamento no intervalo de 4 a 10 clusters.

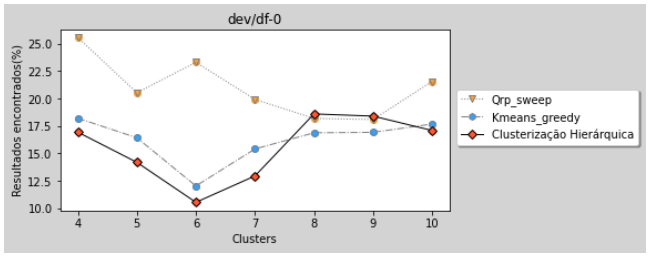

Figura 4: Comportamento dos resultados encontrados do diretório df-0.

4.3 Diretório Rio de Janeiro (rj-0)

O diretório rj-0 contém mais de 320000 entregas em seu histórico. Por ter uma média próxima de 3500 entregas por instância, consideramos esse diretório como um diretório com instâncias grandes, em comparação com os outros dois diretórios utilizados. Para os testes, foram 28 unidades de carregamento no algoritmo referente à Clusterização Hierárquica.

Tabela 3: Resultados encontrados do diretório rj-0

N_clusters	C_Hierárquica	KMeans_greedy	QRP_sweep
4	17.43%	25.67%	23.56%
5	16.82%	25.17%	23.87%
6	**16.72%**	24.93%	21.46%
7	17.54%	23.87%	21.44%
8	17.33%	**23.00%**	20.47%
9	18.34%	23.86%	**19.31%**
10	18.09%	23.56%	19.80%

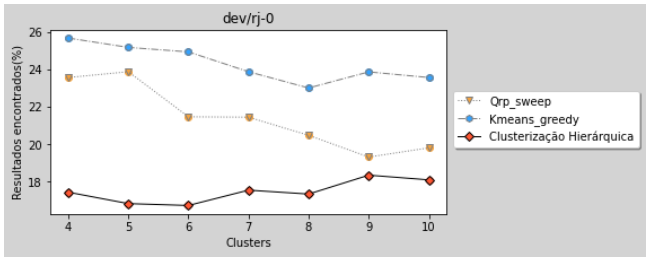

Figura 5: Comportamento dos resultados encontrados do diretório rj-0.

De forma similar à Tabela 1 e à Figura 2, temos a Tabela 3 e a Figura 5, respectivamente. Eles apresentam os resultados encontrados em relação ao algoritmo off-line para cada número de cluster nas instâncias e seu comportamento no intervalo de 4 a 10 clusters.

5 CONCLUSÃO

Neste projeto trouxemos uma abordagem baseada em clusterização hierárquica com o objetivo de ser uma alternativa para resolver o *Last-Mile Incremental Capacitated Vehicle Routing (Last-Mile ICVRP).*

Tal abordagem se fez presente a partir da aplicação do algoritmo de distribuição das unidades de carregamento que tornou possível a criação dos modelos que constituem o segundo nível da hierarquia. Assim como há a dependência de tal distribuição para o andamento do algoritmo, a quantidade de unidades de carregamento também interfere nos resultados. Ao analisar os resultados parciais obtidos, percebemos que os melhores resultados do algoritmo desenvolvido encontrados em cada diretório são mais satisfatórios que os melhores resultados dos algoritmos *KMeans_greedy.py* e *QRP_Sweep.py*. Podendo, assim, concluir que tal abordagem é válida para aplicação.

6 TRABALHOS EM ANDAMENTO E FUTUROS

Na análise do desenvolvimento do algoritmo, entre os possíveis próximos trabalhos, pode-se relacionar:

(1) Adição de novos níveis na clusterização hierárquica.
(2) Desenvolvimento de uma nova forma de distribuição das unidades de carregamento.

Em relação aos trabalhos em andamento, está sendo estudado uma maneira de simplificar a criação e o treinamento dos modelos do segundo nível da clusterização hierárquica.

REFERÊNCIAS

[1] Dimitris Bertsimas, Philippe Chervi, and Michael Peterson. 1995. Computational approaches to stochastic vehicle routing problems. *Transportation science* 29, 4 (1995), 342–352.

[2] Juan Camilo Fonseca-Galindo, Gabriela de Castro Surita, José Maia Neto, Cristiano Leite de Castro, and André Paim Lemos. 2020. A Multi-Agent System for Solving the Dynamic Capacitated Vehicle Routing Problem with Stochastic Customers using Trajectory Data Mining. *arXiv preprint arXiv:2009.12691* (2020).

[3] Chrysanthos E Gounaris, Panagiotis P Repoussis, Christos D Tarantilis, Wolfram Wiesemann, and Christodoulos A Floudas. 2016. An adaptive memory programming framework for the robust capacitated vehicle routing problem. *Transportation Science* 50, 4 (2016), 1239–1260.

[4] Loggi. 2021. loggiBUD: Loggi Benchmark for Urban Deliveries. https://github.com/loggi/loggibud.

[5] Warren B Powell. 1988. A Comparative review of alternative algorithms for the dynamic vehicle allocation problem. In *Vehicle routing: methods and studies. (Studies in Management Science and Systems, Volume 16)*, B. Golden and A. Assad (Eds.). Amsterdam ; New York : North-Holland, 249–291.

[6] Ulrike Ritzinger, Jakob Puchinger, and Richard F Hartl. 2016. A survey on dynamic and stochastic vehicle routing problems. *International Journal of Production Research* 54, 1 (2016), 215–231.

[7] Gabriela Surita and José C. Maldonado. 2020. Evolução das Soluções e Experiência de Roteirização na Loggi. https://www.youtube.com/watch?v=IDmaN7ima7k. [Online; acessado em 28 Março 2021].

[8] Barrett W Thomas and Chelsea C White Iii. 2004. Anticipatory route selection. *Transportation Science* 38, 4 (2004), 473–487.

[9] Marlin W Ulmer, Justin C Goodson, Dirk C Mattfeld, and Marco Hennig. 2019. Offline–online approximate dynamic programming for dynamic vehicle routing with stochastic requests. *Transportation Science* 53, 1 (2019), 185–202.