Dynamic Dispatch Algorithm Proposal for Last-Mile Delivery Vehicle

D. O. Mota

***Abstract*— The complex urban scenario demands decisions which needs to be taken not only fast, but considering a high level of uncertainty. In such context, this study proposes an algorithm to address the vehicle allocation scheduling with a request list unknown *a priori*, and is revealed along the operation horizon. Based on the OSCO (online stochastic combinatorial optimization) problem, it is presented a framework to operate such problem in a last-mile delivery system where customers' requests from e-commerce (or food delivery) along an operations day. The deterministic calculation was realized using a permutational flowshop scheduling problem with the objective of minimizing makespan. Experiments indicated convergence in the algorithm along the planning trials, and a satisfactory makespan prediction, before the orders are revealed, with accuracy. The probability distribution adheres with a bell-shaped (normal) distribution with its population parameters estimated using classical statistics inference theory. Incorporating uncertain in such type of mathematical modeling could open doors for a new era of systems in current days with cloud computing, big data, smart cities, and artificial intelligence.**

***Keywords*— Last-mile delivery, scheduling, dynamic decision process.**

I. Introdução

As entregas em última milha (ou *last-mile deliveries*) têm atraído a atenção de pesquisadores na área de pesquisa operacional em virtude da complexidade no planejamento destas operações. Dimensionamento de frotas, roteirização, alocação de pessoal e diversas outras decisões são submetidas a fatores complicadores como: trânsito, violência urbana, migração temporária de trabalhadores das regiões residenciais para as regiões industriais, dentre outros. Estes fatores são ainda potencializados quando se trata dos grandes centros conhecidos como megacidades (centros urbanos com população superior a 10 milhões de habitantes). Segundo [1], as megacidades integram uma rede de cidades com grande densidade populacional e movimentação de pessoas, atividades e negócios podendo, portanto, determinar e ditar o rumo da civilização. Esta rede geográfica é formada por "40 megaregiões dos 191 países do mundo que impulsionam a economia mundial – elas representam 1/5 da população, 2/3 do rendimento econômico mundial e mais de 85% da inovação global"

Neste contexto, o emprego de ferramentas matemáticas para otimização das operações logísticas ainda apresenta elevados desafios uma vez que estão submetidos a um elevado grau de incerteza, deixando aspectos determinísticos para efeitos estocásticos e probabilísticos. A preocupação em se utilizar modelos quantitativos para tomada de decisões buscando soluções ótimas (em termos matemáticos) é bem reportada em [2], onde o autor propõe a associação dos recursos produtivos como: (1) máquinas em uma fábrica; (2) pistas de decolagem em aeródromos; (3) equipes de construção em um canteiro de obras; (4) lotes de processamento em um ambiente computacional, todos estes podem ser "recursos finitos" do ponto de vista da alocação ao se atribuir uma alocação durante um determinado período de tempo. Logicamente que cada um destes sistemas pode apresentar objetivos específicos como: minimização do tempo de finalização de todas as atividades, diminuição de membros de uma determinada equipe de trabalhadores, atendimento em uma data específica prometida, dentre outras.

Os métodos tradicionais de otimização na programação da produção (*scheduling*) apresentam fundamentação teórica explorada por [4] e que servem de base para a solução do problema quando são incorporadas questões de incerteza. A referência [5] propõe o uso de algoritmos genéticos para solução do problema e [6] discute a solução determinística do problema proposto utilizando programação inteira mista.

Por outro lado, o uso de técnicas de alocação ainda apresenta aspectos de inovação em virtude da busca por algoritmos genéricos e eficientes para atender as particularidades de um sistema. Mesmo assim, diversos trabalhos foram realizados na área, como: [7]–[10]. Todos eles, relacionam temas relativos à programação de meios para tornar o transporte mais eficientes.

O problema de alocação dinâmica de requisições (que podem ser ordens de produção, solicitação de entregas, dentre outros) vem sido explorado em termos conceituais, desde os anos 90, e [3] propõe o nome OSCO (*online stochastic combinatorial optimization*) para este problema, onde o foco está na exploração das informações disponíveis a cada momento em que uma decisão é tomada. O mesmo autor ainda propõe explorar a incerteza dos modelos por meio de correlações das variáveis aleatórias, e o uso de *Machine Learning* para evidenciar padrões.

Ao se associar as questões relacionadas ao transporte, com complexidades em um centro urbano, alguns pontos de atenção devem ser destacados, como por exemplo a relação bidimensional entre os ambientes urbanos, ou seja, quais as sobreposições entre os distritos, ou ainda se existe diferenciação entre as tarifas que podem levar à mudança nos critérios de tomada de decisão (não somente tempo ou distâncias, por exemplo). Com o avanço das técnicas empregando GIS (*Geographic Information System*, ou Sistema de Informação Geográfica), os modelos matemáticos tem incorporado tal tecnologia como parâmetros de seus modelos, mas ainda, segundo [11], "computadores não possuem a capacidade cognitiva e precisam de uma programação complexa para fazer aquilo que uma pessoa equipada com um mapa detalhado pode responder facilmente".

D. O Mota, Universidad de São Paulo, São Paulo, São Paulo, Brasil, danielmota@usp.br

Portanto, o objetivo deste artigo está em apresentar uma proposta de algoritmo para alocação de requisições de entrega em centros urbanos onde existe incerteza no conteúdo da entrega ao longo da operação, ou seja, as requisições são apresentadas após o processo de entrega já ter sido iniciado. E desta forma, preencher a lacuna da literatura relativa ao uso de técnicas de alocação no contexto de *last-mile deliveries* onde a necessidade de reprogramação e atualização dinâmica de dados ambientais (transito, congestionamento, cancelamento de pedidos e novos pedidos) são frequentes e precisam ser endereçados de maneira eficiente para que se tenha sucesso na operação.

O presente artigo está organizado da seguinte forma: Seção I apresenta a introdução do tema, bem como objetivos e algumas referências bibliográficas para formação do embasamento conceitual; Seção II descreve o método empregado, bem como os aspectos computacionais da implementação utilizada neste estudo; Seção III reporta os resultados e discute os mesmos à luz do problema apresentado; e por fim, Seção IV apresenta a conclusão do estudo, bem como os potenciais futuros passos esperados da pesquisa aqui reportada.

II. MATERIAIS E MÉTODOS

A. Premissas e Descrição do Problema

Dentre as estruturas de alocação conhecidas, este estudo utilizou o *Flowshop* permutacional para representar o processo de entrega. Nesta estrutura os *Jobs* (requisições) são os pedidos de entrega, onde cada cliente segue sua própria agenda de intenções quanto ao tamanho do pedido, instante de realização. Todas as requisições seguem o mesmo roteiro (etapas de preparo e entregas), porém cada uma com seu tempo de processamento (representando a variabilidade do tempo de entrega entre as diferentes ordens). O objetivo da programação será atingir o menor tempo de finalização de todas as entregas previstas (minimização de *makespan*), assumindo que os clientes contam com a eficiência da empresa e são solidários em receber o produto o quanto antes possível (desde que, obviamente, após o momento de realização do pedido).

No presente estudo, não são consideradas incertezas no tempo de processamento e somente na lista de requisições. Desta forma, uma vez disponibilizada uma nova requisição, esta terá suas etapas conhecidas de maneira determinística.

Conhecidas as premissas, deve-se considerar o instante atual t em um horizonte de planejamento H onde ($t \in H$) e pode ser interpretado por um período de operação das entregas, um turno, um dia, ou até mesmo uma semana (com operação ininterrupta de 24 horas por dia), existem dois conjuntos de requisições que podem ser rotulados: (R_t) requisições reveladas até o instante t e (R_H) que representa todas as requisições às quais o sistema será submetido ao longo do horizonte H. Cada requisição (R_i) é composta pelas seguintes informações: $<id_i, r_i, P_{i,1}, P_{i,2},... P_{i,m}>$, onde id_i representa o código de identificação da requisição i , r_i indica o instante a partir do qual a requisição i é revelada ao sistema, estando disponível para ser programada, e $P_{i,1}, P_{i,2},... P_{i,m}$ reflete o tempo de processamento da ordem i em cada uma das etapas que precisa ser percorrida para o processamento completo da requisição.

Segundo [12], o algoritmo genérico para a solução do problema descrito é apresentado no Algoritmo 1, que , embora simples, conduz a discussão do método para a forma como a seleção da requisição (s_t) é feita para incorporar o conjunto de requisições selecionadas (S_H) que é construído ao longo de $t \in H$. A função *escolheRequisição*(S_{t-1}, R_t) depende das requisições alocadas previamente e das disponíveis para alocação.

Algoritmo 1: Solução genérica do problema *Online*

Entrada: lista completa (R_H) de requisições no horizonte H
Saída: lista de alocação (S_H) no horizonte H

1. requisições ordenadas ← vazio
2. **para cada** t ∈ H **faça:**
3. s_t ← escolheRequisição(S_{t-1}, R_t)
4. S_t ← S_{t-1} + s_t

A apresentação do Algoritmo 1, embora simples, conduz a discussão do método para a forma como a seleção da requisição (s_t) é feita para incorporar o conjunto de requisições selecionadas (S_H) que é construído ao longo de $t \in H$. A função *escolheRequisição*(S_{t-1}, R_t) depende das requisições alocadas previamente e das disponíveis para alocação.

Existe uma gama de alternativas a serem exploradas quando se busca a construção de uma programação ótima em meio à incerteza das requisições que chegarão ao sistema no futuro. O uso de métodos gulosos (*Greedy*) leva a construção de um resultado rápido e viável, embora não se possa garantir critérios de otimalidade sobre a programação apresentada.

A proposta de um método guloso para solução do problema descrito visa resolver o problema completo, sendo composto por três partes: (1) requisições já alocadas (S_{t-1}); (2) requisições já reveladas (R_t); e (3) requisições previstas (A_t). Estas partes interagem segundo o Algoritmo 2. Vale ressaltar que as requisições previstas, embora desconhecidas *priori*, elas são descritas por meio de suas distribuições de probabilidade.

Algoritmo 2: escolheRequisição

Entrada:
Lista de alocação (S_{t-1}) anterior ao instante t
Lista completa (R_t) de requisições até o instante t
Saída: índice da requisição (s_t) a ser alocada no instante t

1. F ← R_t – S_{t-1}
2. A ← F + random
3. **para cada** f ∈ F **faça:**
4. C^f_{max} ← resolveProblema(S_t, f, A_t)
5. **retorna** f = max arg (C^f_{max})

A codificação da aplicação foi realizada na linguagem Python 3.7 (Anaconda – IDE Spyder) e executado sob o Sistema Operacional Windows 10 64bits. Um hardware notebook, core i7 (2,59 GHz), 6ªgeração com 16Gb de memória RAM (2133 MHz) e disco rígido SSD. Foram incluídas bibliotecas as *numpy* e *matplotlib* para tratamento dos aspectos de aleatoriedade e geração de gráficos respectivamente. As imagens do funcionamento do algoritmo podem ser observadas nas figuras 1, 2 e 3.

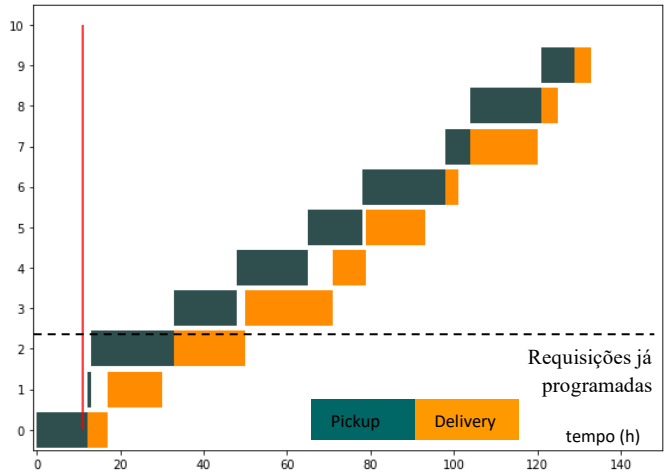


Fig. 1. Execução exemplo do algoritmo proposto no instante t=10.

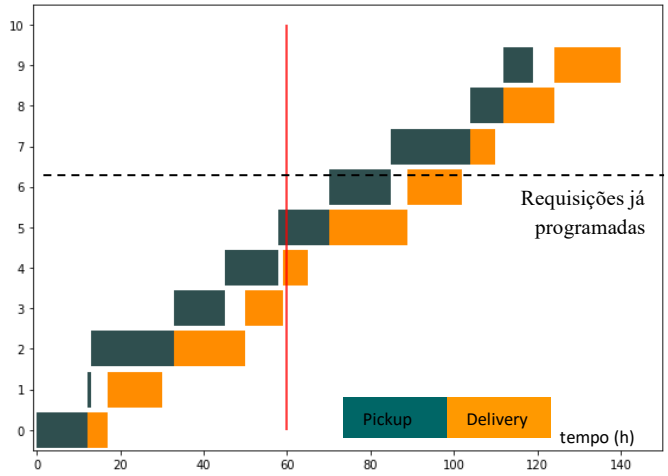


Fig. 2. Execução exemplo do algoritmo proposto no instante t=60.

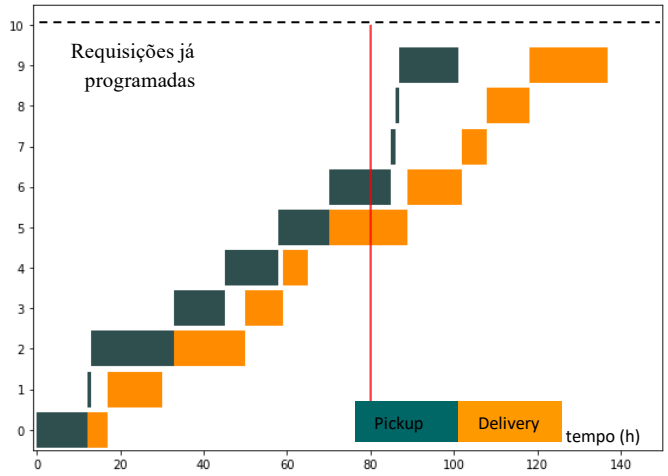


Fig. 3. Execução exemplo do algoritmo proposto no instante t=80.

Sendo o eixo horizontal o tempo de execução do algoritmo marcado pela linha vertical cruzando todo o gráfico de Gantt, observam-se ao longo dos três instantes representados que o avanço da barra horizontal reflete o avançar das requisições já programadas e que não podem sofrer alterações (S_t). Já a barra vertical indica o tempo atual de processamento do algoritmo, indicando o passar do tempo, e a partir do qual a revelação das requisições vai ocorrendo.

Assim, no instante t=10, percebe-se (por meio da comparação com os demais gráficos) que as requisições 0, 1 e 2 já estão programadas (requisições abaixo da linha pontilhada). Adicionalmente, observa-se na figura 2 (t=60) que as requisições 3, 4, 5 e 6 foram complementarmente programadas. Por fim, na figura 3 (t=80) já se pode observar a programação de todas as requisições.

O tempo de execução deste algoritmo é desprezível utilizando o hardware descrito, porém o problema resolvido de maneira heurística requer um baixo esforço computacional. A própria natureza do problema *NP-Hard* [13] impõe um maior tempo de execução associado ao aumento do número de requisições ou etapas de processos (que no caso, foram utilizadas somente duas).

Para exemplificar o funcionamento do algoritmo, será apresentada na figura 4 uma situação intermediária do procedimento de alocação dinâmica proposto.

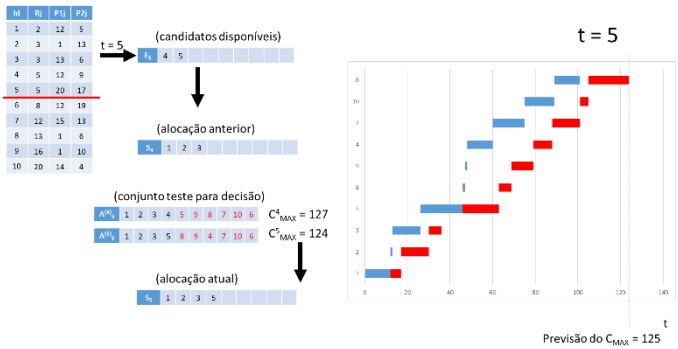


Fig. 4. Aplicação exemplo do funcionamento do algoritmo proposto.

Para uma lista R_H de requisições a ser programada, somente torna-se visível as requisições com tempo de liberação (r_j) anterior (ou igual) ao tempo atual, que no caso, t=5. Existe um conjunto de requisições previamente alocada (S_{t-1}, ou seja, S_4) e uma lista de requisições candidatas à alocação (F_t, ou seja, F_5), que são aquelas que já foram apresentadas, mas ainda não foram transferidas para o conjunto S_t. Neste caso, duas requisições do subconjunto R_t foram definidas como candidatas: F_5 :{4,5}.

Cada uma destas requisições irá compor o conjunto de programação candidato (A_t, ou seja, A_5) que irá incorporar uma das candidatas e completada com as requisições restantes (um conjunto de tamanho previamente estabelecido, refletindo a capacidade do sistema em processar pedidos). Cada requisição restante irá ser criada tendo seus parâmetros (no caso, tempo de processamento) idêntico ao conhecido (ou estimado) das requisições já conhecidas, tanto em termos da sua distribuição de probabilidade quanto em termos de seus parâmetros constituintes: média, desvio padrão, forma, escala, taxa, dependendo da distribuição de probabilidade que gere a melhor aderência aos dados históricos.

Assim, cada candidato testado gera uma programação estimada que pode ser analisada em termos do *makespan* alcançado. Para o exemplo apresentado, foi utilizado o operador SWAP [14] pela sua simplicidade computacional em implementar, mas pode se associar esta etapa do algoritmo à solução ótima gerada por programação inteira mista (MILP) ou algum tipo de heurística ou meta-heurística [15]–[17].

Aquela programação que, após a inclusão da requisição candidata resultar em um menor tempo de conclusão será eleita

e irá incorporar o conjunto de requisições alocadas para as próximas etapas de execução do algoritmo, agora sendo ela fixada.

Este procedimento iterativo se repete até a alocação de todas as ordens seja realizada e também que o tempo do horizonte de programação (H) seja completamente esgotado.

III. RESULTADOS E DISCUSSÃO

A. *Delineamento Experimental*

Foram geradas instâncias com instante de liberação e duração de processamentos aleatórios do problema (F_2/r_j,*permut*/C_{max}), de acordo com a notação clássica de alocação apresentada por [4]. A variável dependente do estudo é o número de requisições alocadas pelo algoritmo proposto. Os parâmetros são apresentados na tabela 1.

TABELA I
PARÂMETROS EMPREGADOS NO EXPERIMENTO REALIZADO

Parâmetro	Variável	Valor	Observação
Horizonte de planejamento	H	150	
Número de etapas no roteiro	M	5	
Tempo proc. real	R_H	UNIF(1,20)	inteiros
Tempo proc. previsto	A_t	UNIF(1,20)	Inteiros
Instante de liberação	r_j	UNIF(1,H)	Inteiros
Passo do algoritmo	t	1	fixo

Para experimentação do conceito proposto foram criadas instâncias do problema proposto para analisar a variabilidade da predição do *makespan* tendo em vista a antecipação da decisão tomada. Para tal avaliação foi criado o indicador apresentado na equação 1.

$$\Delta C_{max} = C_{max}(t) - C_{max}(H) \quad (1)$$

O instante de disponibilização de uma requisição é sorteado aleatoriamente por meio de uma distribuição de probabilidade uniforme variando dentro do horizonte de programação disponibilizado para operação do algoritmo. Assim como os tempos de processamento reais, porém, desconhecidos no início do processamento do algoritmo, foram gerados de maneira independentes para cada uma das duas etapas do roteiro de execução de cada uma das requisições experimentadas. Seguindo a mesma lógica, os tempos de processamentos previstos durante a execução do algoritmo como forma de prever as requisições ainda por vir também foram geradas de forma independentes. Desta forma, todas as variáveis aleatórias do estudo *i.i.d* (independentes e identicamente distribuídas) [18]. Impedindo que qualquer tipo de correlação seja incorporada aos dados no momento de sua geração, e garantindo que qualquer associação entre os dados observada nos resultados seja exclusivamente consequência do processamento ao qual os mesmos estiveram submetidos. O histograma dos resultados do experimento realizado (N=500 requisições) pode ser observado na figura 5.

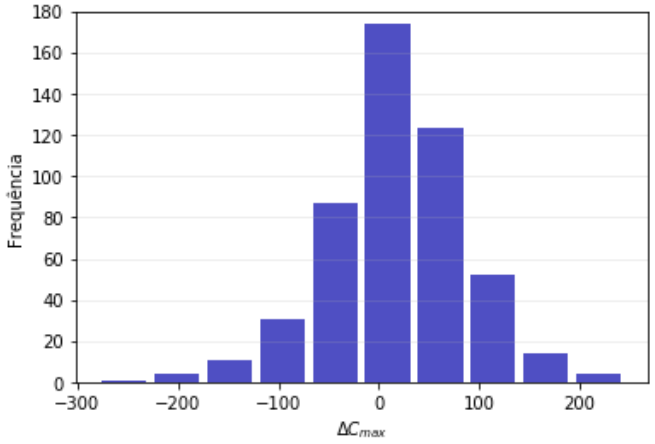


Fig 5. Distribuição de frequência dos ΔC_{max} no experimento realizado.

Neste histograma observa-se que a variação do C_{max} durante o processo de previsão esteve, na média, próximo ao valor zero, seguindo o teorema do limite central e portanto sendo $E[C_{max}(t)] \approx C_{max}(H)$, ou seja, próximo ao C_{max} da última iteração do algoritmo, quando toda a informação já está revelada para tomada de decisão. Observa-se ainda que existe uma nítida convergência a ser explorada na figura 6.

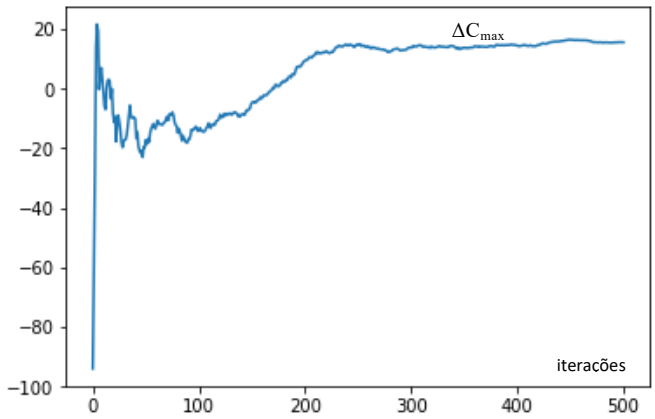


Fig. 6. Convergência de C_{max} ao longo das interações.

Uma vez verificada a convergência do método de previsão das ordens durante a realização de um único experimento, ou seja, verificado que durante a execução da programação das requisições, a falta de informação relativa à incerteza do instante de inicialização da ordem foi satisfatoriamente suprida pelo processo de geração de ordens partindo da mesma função geradora, neste caso, distribuição de probabilidade uniforme.

Em seguida, foram realizados experimentos onde diversas instâncias do mesmo problema foram construídas para verificar a convergência do método na previsão do *makespan* com incerteza no instante de despacho das requisições. Para realização destes experimentos foram empregadas as mesmas configurações descritas na tabela 1, porém um conjunto de 50 requisições (N=50) foram replicadas no total de 1000 vezes, e os resultados são apresentados na tabela 2.

TABELA II
RESULTADOS DOS EXPERIMENTOS REALIZADOS

Rep. #	Temp.	Máq.	Req.	Cmax	Méd. Prevista Cmax	Desv. Previsto Cmax
1	101	5	50	754	715,8	31,5
2	101	5	50	647	679,5	32,9
3	99	5	50	683	694,1	22,7
4	103	5	50	716	704,6	28,1
5	99	5	50	703	698,4	25,9
6	101	5	50	780	748,3	30,2
...	...	...	...	...	...	...
996	100	5	50	771	729,8	31,3
997	101	5	50	714	691,5	26,6
998	101	5	50	751	714,3	32,7
999	101	5	50	693	697,6	29,3
1000	99	5	50	711	711,3	37,1

Como consequência, observou-se a convergência do C_{max} previsto para o C_{max} real, conforme apresentado na figura 7.

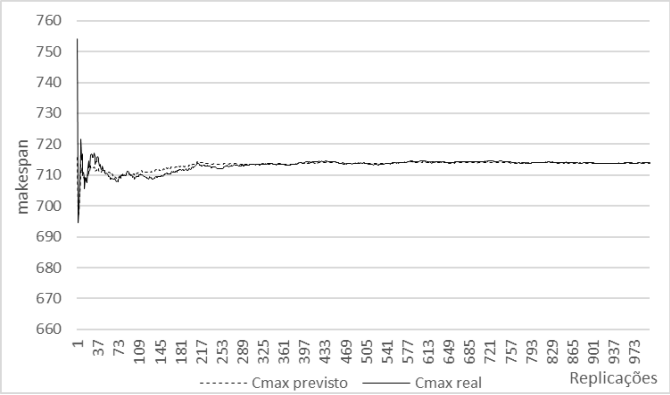


Fig. 7. Convergência de C_{max} previsto e real nos experimentos realizados.

A convergência do C_{max} reflete a aderência do método em utilizar a previsão das ordens futuras, ainda não explorado pela literatura, baseando-se no conhecimento *priori* de sua distribuição de probabilidades e usando um conjunto de ordens estimadas para realização da programação de alocação das ordens. Desta forma, diferenciando o conjunto de ordens visíveis e estimadas, criando uma programação prevista que mesmo não sendo confirmada, pode levar a melhores decisões quanto à alocação dos ativos destinados ao seu atendimento.

IV. CONCLUSÃO

A complexidade das decisões tomadas em sistemas logísticos leva a busca por modelos matemáticos que suportem uma tomada de decisão em um ambiente de elevada incerteza. Neste contexto, o presente artigo explorou um procedimento de despacho que precisa ser dinâmico em virtude da falta de informação completa, no caso a lista de requisições a serem realizadas em momentos futuros, mas ao mesmo tempo precisa ser rápido para que o sistema possa realizar estimativas da conclusão de suas tarefas de maneira assertiva. A complexidade dos algoritmos de despacho dinâmico apresenta origens tanto em relação ao endereçamento das incertezas quanto à sua escala. Embora inspirado no algoritmo proposto por [12], muitos aspectos computacionais precisam ser explorados na fase de implementação do código. Além disso, é de profunda relevância identificar, de maneira objetiva, a aplicação direta do conceito criado, visando solução de problemas reais e relevantes para a sociedade.

Neste sentido, o problema explorado apresenta consonância com a realidade de um sistema urbano quando operações de transporte tanto de passageiros (exemplo: taxi), quanto de cargas como entregas de cargas (exemplo: entregadores de documentos, pacotes).

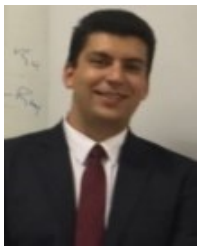
No ponto de vista das cidades, o aumento de interesse em temas como "cidades inteligentes" tem provocado o interesse de pesquisadores em desafios computacionais neste referido assunto. Um dos problemas mais relevantes neste contexto está o elevado nível de incerteza neste ambiente. Hoje, megacidades estão repletas de prestadores de serviço que compartilham a mesma via para diversos tipos de entregas (supermercado, alimentos, produtos farmacêuticos, documentos, móveis, transporte escolar, dentre outros). E por melhor que seja o planejamento de uma empresa, a complexidade gerada pela interação destes agentes torna a execução deste planejamento um verdadeiro caos. Desta forma, estudos desta natureza podem lançar luz aos problemas enfrentados por estes grandes centros urbanos, levando os sistemas que deles dependem a um ganho de eficiência e consequente melhor aproveitamento dos recursos empregados para esta prestação de serviços (combustível, poluição, ruídos, ocupação dos tempos dos cidadãos, qualidade de vida, etc.).

Por fim, acredita-se que este estudo contribui em apresentar uma forma objetiva de se tratar uma das fontes de incerteza neste caos urbano chamada cidade.

REFERÊNCIAS

[1] C. Leite, "São Paulo, megacidade e redesenvolvimento sustentável: uma estratégia propositiva," *Rev. Bras. Gestão Urbana*, 2010.
[2] M. L. Pinedo, *Planning and scheduling in manufacturing and services: Second edition*. 2009.
[3] P. van Hentenryck, R. Bent, and E. Upfal, "Online stochastic optimization under time constraints," *Ann. Oper. Res.*, 2010.
[4] M. Pinedo, *Scheduling: theory, algorithms, and systems*. Springer, 2012.
[5] L. R. Abreu and B. A. Prata, "A hybrid genetic algorithm for solving the unrelated parallel machine scheduling problem with sequence dependent setup times," *IEEE Lat. Am. Trans.*, 2018.
[6] P. H. da Silva Palhares and L. da Cunha Brito, "Constrained Mixed Integer Programming Solver Based on the Compact Genetic Algorithm," *IEEE Lat. Am. Trans.*, 2018.
[7] M. B. Wright, J. R. Daduna, and A. Wren, "Computer-Aided Transit Scheduling," *J. Oper. Res. Soc.*, 2006.
[8] J. P. Paixão and I. M. Branco, "A quasi-assignment algorithm for bus scheduling," *Networks*, 1987.
[9] C. Basnet, L. R. Foulds, and J. M. Wilson, "An exact algorithm for a milk tanker scheduling and sequencing problem," *Ann. Oper. Res.*, 1999.
[10] S. Lan, J.-P. Clarke, and C. Barnhart, "Planning for Robust Airline Operations: Optimizing Aircraft Routings and Flight Departure Times to Minimize Passenger Disruptions," *Transp. Sci.*, 2006.
[11] S. I. Gass, "Urban operations research," *Transp. Res. Part A Gen.*, 2002.
[12] R. Bent and P. Van Hentenryck, *Online Stochastic Combinatorial Optimization*. 2018.
[13] S. Dasgupta, C. H. Papadimitriou, and U. V. Vazirani, "Algorithms," 2006.
[14] J. F. Gonçalves, J. J. De Magalhães Mendes, and M. G. C. Resende, "A hybrid genetic algorithm for the job shop scheduling problem," *Eur. J. Oper. Res.*, 2005.
[15] R. Mellado Silva, C. Cubillos, and D. Cabrera Paniagua, "A constructive heuristic for solving the Job-Shop Scheduling Problem," *IEEE Lat. Am. Trans.*, 2016.
[16] M. Ziaee, "A heuristic algorithm for solving flexible job shop scheduling problem," *Int. J. Adv. Manuf. Technol.*, 2014.

[17] J. C. Ferreira, M. T. Arns Steiner, and M. Siqueira Guersola, "A Vehicle Routing Problem Solved Through Some Metaheuristics Procedures: A Case Study," *IEEE Lat. Am. Trans.*, 2017.
[18] D. C. Montgomery, *Applied Statistics and Probability for Engineers Third Edition*. 2003.



Daniel de Oliveira Mota possui graduação em Engenharia da Produção (2008) e Mestrado em Engenharia Industrial (2010) pela *North Carolina A&T State University* e Doutorado em Engenharia Naval (2016) pela Universidade de São Paulo. Professor do departamento de Engenharia de Produção da Universidade de São Paulo. Possui experiência em modelagem matemática de sistemas logísticos utilizando técnicas de Pesquisa Operacional. Particular interesse em problemas estocásticos no contexto urbano e hospitalar.