Uma Aproximação para o Problema de Reabastecimento em Conjunto com Capacidades*

Miguel Ángel Marfurt Alarcón, Lehilton Lelis Chaves Pedrosa

[1]Instituto de Computação – Universidade Estadual de Campinas (Unicamp)

miguel.alarcon@students.ic.unicamp.br, lehilton@ic.unicamp.br

Abstract. *In the Joint Replenishment Problem (JRP), there is a set of retailers that face daily demands for an item in a planning horizon. The demands are satisfied by items currently held in inventory and replenished by making a joint order to a depot for a subset of retailers. The goal is to decide when to place orders and which retailers are satisfied by each order, so that the total ordering and holding costs are minimized. In the Tree JRP, the supply chain is represented by a tree whose leaves are the retailers, and the ordering cost for a subset of retailers is determined by the cost of the minimal subtree that connects them to the root. In this work, we start the study of the variant where orders have bounded capacity, and give a 6-approximation based on LP rounding.*

1. Introdução

Os denominados *Inventory Routing Problems* (IRP) [Bell et al. 1983] estão presentes na indústria desde pequenas até grandes empresas. Dado um ou mais depósitos e uma frota de veículos, o objetivo é encontrar uma melhor política de entregas de forma a servir um conjunto de varejistas. O que torna esses problemas particularmente desafiadores é que os custos de roteamento devem ser balanceados com os custos de armazenamento dos itens pelos varejistas em um plano de horizonte de demandas. Para as versões mais gerais de IRP, conhecemos apenas algoritmos de aproximação com fatores sub-logarítmicos para o caso com um único depósito [Nagarajan and Shi 2015]. Por esse motivo, diversos trabalhos voltaram-se para casos particulares, logrando aproximações com fatores constantes [Fukunaga et al. 2014, Cheung et al. 2015, Jiao and Ravi 2019].

Uma prática comum é modelar a cadeia de fornecimento como uma árvore, de forma que a raiz represente o depósito e que as folhas representem os varejistas. Desse modo, o custo de roteamento pode ser estimado pelo custo de conectar a raiz a um subconjunto de folhas dessa árvore, dando origem ao chamado *Tree Joint Replenishment Problem* (Tree JRP). Esse problema admite uma 3-aproximação baseada em arredondamento de programa linear (PL) [Cheung et al. 2015] e o melhor algoritmo de aproximação conhecido tem fator 2 [Pedrosa 2014]. Uma característica do Tree JRP é que se pressupõe um único veículo com capacidade de carregamento ilimitada, o que não é realista para determinadas aplicações. Neste trabalho, estudamos o *Capacitated Splittable Tree JRP*, quando o número itens de uma entrega é limitado, mas diversas ordens podem ser realizadas em um mesmo dia. Neste trabalho, apresentamos uma 6-aproximação para essa variante, também baseada no arredondamento de PL.

*Apoiado pela Fundação de Amparo à Pesquisa do Estado de São Paulo (FAPESP), processo 2015/11937-9, e Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq), processos 425340/2016-3, 422829/2018-8 e 312186/2020-7.

Definição dos problemas. A entrada do Tree JRP contém uma uma árvore $\mathcal{T}$, cuja raiz r representa o depósito e as folhas representam os varejistas, N. Associamos um número $K^{(j)} \geq 0$ a cada vértice j da árvore que representa o custo de passar por esse vértice. O custo de uma entrega para um subconjunto de varejistas $S \subseteq N$ é definido como $\sum_{j \in \bigcup_{i \in S} path(i)} K^{(j)}$, onde $path(i)$ é o conjunto de vértices da árvore no caminho de i até r. Há um inteiro positivo T representando o tamanho do horizonte de planejamento de forma que, para cada período $t \in \{1, ..., T\}$, existe uma demanda por d_{it} itens do varejista $i \in N$. Além disso, há também um número $h^i_{st} \geq 0$ que corresponde ao custo de armazenamento de um item do período s até t para o varejista i. A única suposição que temos sobre os custos de armazenamento é a monotonicidade, ou seja, para cada varejista i e quaisquer três períodos $q \leq s \leq t$, devemos ter $h^i_{qt} \geq h^i_{st}$. Uma solução consiste dos períodos em que se realizam entregas e do conjunto de varejistas reabastecidos em cada entrega, de forma a satisfazer todas as demandas com itens disponíveis no inventário. O objetivo é minimizar a soma dos custos de entrega e de armazenamento.

Como os custos de armazenamento são monotônicos, no Tree JRP é preferível satisfazer demandas com itens entregues mais recentemente. Isso implica que entregas são feitas apenas para varejistas com inventário vazio e, como não há limite no número de itens de um entrega, apenas um veículo precisa ser utilizado em um dia (cf. [Pedrosa 2014]). O *Capacitated Splittable Tree JRP* é a variante em que as entregas são feitas por uma frota de veículos ilimitada, mas cada veículo pode entregar apenas U unidades. Note que, nessa variante, os d_{it} itens de uma demanda podem ser distribuídos em várias entregas em um mesmo período.

2. Uma aproximação para o Capacitated Splittable Tree JRP

Assim como o algoritmo de Cheung et al., nosso algoritmo é baseado no arredondamento de PL. A seguir, denote por $\mathcal{T}_j$ a sub-árvore maximal de $\mathcal{T}$ enraizada em $j \in \mathcal{T}$ e seja $\mathcal{D}$ o conjunto de todos os pares (i,t) com $d_{it} > 0$. Defina também o valor fixo $H^i_{st} = h^i_{st} d_{it}$, i.e., H^i_{st} é o custo de armazenar os d_{it} itens do período s até t por um varejista i. No PL abaixo, temos uma variável inteira y^j_s que indica o número de vezes que o vértice j é visitado no período s. Além disso, temos uma variável racional x^i_{st} que representa a fração dos d_{it} itens da demanda $(i,t) \in \mathcal{D}$ que é satisfeita por uma entrega no período s. Uma solução para o programa abaixo é um limitante inferior para o valor de uma solução ótima.

$$\min \sum_{j \in V(\mathcal{T})} \sum_{s=1}^{T} K^{(j)} y^j_s + \sum_{(i,t) \in \mathcal{D}} \sum_{s=1}^{t} H^i_{st} x^i_{st}$$

(CST-JRP)

$$\text{s.a.} \quad \sum_{s=1}^{t} x^i_{st} = 1, \qquad (i,t) \in \mathcal{D} \tag{1}$$

$$x^i_{st} \leq y^j_s, \qquad (i,t) \in \mathcal{D}, s \leq t, j \in path(i) \tag{2}$$

$$\sum_{i \in V(\mathcal{T}_j) \cap N} \sum_{t=s}^{T} \frac{d_{it}}{U} x^i_{st} \leq y^j_s, \qquad s \leq T, j \in V(\mathcal{T}) \tag{3}$$

$$y^j_s \in \mathbb{Z}^+, x^i_{st} \geq 0, \qquad (i,t) \in \mathcal{D}, s \leq t, j \in V(\mathcal{T}) \tag{4}$$

A principal diferença para a formulação acima para a versão sem capacidades em [Cheung et al. 2015] é a adição das restrições (3). Elas limitam o número de vezes que cada vértice j deve ser visitado no período s em função do número de demandas servidas no período.

Nosso algoritmo consiste de duas etapas. Na primeira, determinamos em que períodos serão agendadas entregas e quais varejistas participarão de entregas nos períodos. Para isso, utilizamos uma sub-rotina de arredondamento de Cheung et al., mas adaptada para (CST-JRP). Na segunda, para cada período, determinaremos quantas entregas e quais varejistas serão incluídos em cada entrega, utilizando um algoritmo de roteamento.

Agendamento de entregas. Dada uma solução ótima da relaxação linear, iremos determinar em que períodos cada vértice será visitado. Primeiro, criamos um conjunto de entregas tentativas. Para cada cada $j \in V(\mathcal{T})$, criamos uma entrega tentativa em todo período $s \in \{1, 2, \ldots, T\}$ tal que o intervalo $(\sum_{u=1}^{s-1} y_u^j, \sum_{u=1}^{s} y_u^j]$ contém um inteiro. Intuitivamente, podemos pensar que o total de entregas fracionárias y_u^j acumuladas entre duas entregas tentativas consecutivas é de no máximo um. Depois, selecionamos entregas definitivas. Escolhemos as próprias entregas tentativas para a raiz r e processamos os demais vértices j iterativamente em um percurso *pré-ordem*. Seja j' o pai do vértice j (que já foi processado antes pelo algoritmo). Para cada entrega tentativa em um período s, escolhemos no máximo duas entregas definitivas em j: no último período de entrega definitivo de j' em $[1, s]$ e no primeiro período de entrega definitivo de j' em $(s, T]$.

Roteamento de entregas. Para cada período s, queremos construir um conjunto de entregas incluindo os vértices com entregas agendadas nesse período. Note que uma entrega corresponde a uma subárvore enraizada em r e que entregas diferentes podem conter vértices diferentes. O algoritmo irá construir todas as entregas simultaneamente processando a árvore de maneira *bottom-up*, i.e., processamos os vértices de $\mathcal{T}$ que têm entrega agendada em s em um percurso em *pós-ordem*. Para uma folha i de $\mathcal{T}$ (que corresponde a um varejista), seja s' o período da próxima entrega agendada para i, ou $T + 1$ se não houver. A demanda acumulada de i no período s é definida como $D_i^s := \sum_{t=s}^{s'-1} d_{it}$. Criamos $\lfloor \frac{D_i^s}{U} \rfloor$ entregas cheias para i com U itens cada e, possivelmente, uma entrega parcial com o número de itens restantes da demanda, diga-se, R_i^s. Para um vértice interno j, definimos D_j^s como a soma das demandas residuais R_ℓ^s de seus filhos ℓ. Analogamente, neste caso combinamos as entregas parciais dos filhos em $\lfloor \frac{D_j^s}{U} \rfloor$ entregas cheias e, possivelmente, uma entrega parcial com o restante da demanda, R_j^s. Por simplicidade, se um vértice j não tem entrega agendada para um período s, definimos $D_j^s := 0$ e $R_j^s := 0$.

O algoritmo constrói apenas entregas que respeitam a capacidade U, então para verificar que a solução gerada é viável, basta garantir que uma entrega é agendada na raiz antes de cada demanda. Isso segue do fato de que $y_s^j \leq y_s^{j'}$ sempre que j' é antecessor de j em $\mathcal{T}$, portanto o número de entregas "fracionárias" na raiz é maior do que o número de entregas em um varejista.

Em seguida, iremos analisar o custo da solução gerada, que pode ser dividido em custo de armazenamento (*holding cost*) e custo de entrega (*ordering cost*). Para limitar o custo de armazenamento, comparamos com o valor ótimo do problema dual. Seja B_t^i a variável dual ótima relacionada à restrição (1). O lema abaixo segue por folgas complementares porque uma demanda (i, t) é servida por uma entrega em s com $x_{st}^i > 0$.

Lema 1. *O custo de armazenamento é de no máximo* $\sum_{(i,t)\in\mathcal{D}} B^i_t$.

A análise do custo de entrega é mais elaborada. Observe que o algoritmo de agendamento implica que o número de períodos em que uma entrega é agendada é $\lfloor \sum_{u=1}^{T} y^r_u \rfloor$ para a raiz e no máximo $2\lfloor \sum_{u=1}^{T} y^j_u \rfloor$ para os demais vértices. No entanto, como uma entrega tem capacidade limitada, um vértice pode ser visitado múltiplas vezes no mesmo período. Estimamos o número de vezes que um vértice é visitado com o seguinte lema.

Lema 2. *Seja* n^s_j *o número de vezes que um vértice* j *é visitado pelo algoritmo no período* s. *Para todo* $j \in V(\mathcal{T})$, *vale* $n^s_j \leq \left\lfloor \sum_{i\in V(\mathcal{T}_j)\cap N} \frac{D^s_i}{U} \right\rfloor + 2$.

Lema 3. *O custo de entrega é no máximo* $5 \sum_{j\in V(\mathcal{T})} \sum_{s=1}^{T} K^{(j)} y^j_s$.

Demonstração. Fixe um vértice j e defina $\mathcal{S}_j$ como o conjunto de períodos em que é realizada alguma entrega j. Utilizando o resultado do Lema 2, o número total de entregas que incluem j, diga-se n_j, é limitado por

$$n_j \leq \sum_{s\in\mathcal{S}_j} \left(\sum_{i\in V(\mathcal{T}_j)\cap N} \frac{D^s_i}{U} + 2 \right) = \sum_{i\in V(\mathcal{T}_j)\cap N} \sum_{t=1}^{T} \frac{d_{it}}{U} + 2|\mathcal{S}_j|,$$

onde a igualdade vale por construção do algoritmo de roteamento e pode ser verificada usando indução na altura de j em $\mathcal{T}$. Observe que por construção do algoritmo de agendamento temos que

$$2|\mathcal{S}_j| \leq 2 \cdot 2 \left\lfloor \sum_{s=1}^{T} y^j_s \right\rfloor \leq 4 \sum_{s=1}^{T} y^j_s.$$

Então resta apenas limitar o valor do termo com a soma. Somando-se todas as restrições do tipo (3) do programa linear para todo t, e utilizando $\sum_{s=1}^{t} x^i_{st} = 1$ pela restrição (2),

$$\begin{aligned} \sum_{s=1}^{T} y^j_s &\geq \sum_{s=1}^{T} \sum_{i\in V(\mathcal{T}_j)\cap N} \sum_{t=s}^{T} \frac{d_{it}}{U} x^i_{st} \\ &= \sum_{i\in V(\mathcal{T}_j)\cap N} \sum_{t=1}^{T} \frac{d_{it}}{U} \sum_{s=1}^{t} x^i_{st} \\ &= \sum_{i\in V(\mathcal{T}_j)\cap N} \sum_{t=1}^{T} \frac{d_{it}}{U}. \end{aligned}$$

Juntando os dois limitantes e somando para todos os vértices, obtemos o lema. □

Como o valor ótimo do PL é um limitante para o ótimo, os Lemas 1 e 3 implicam:

Teorema 1. *Existe uma* 6*-aproximação para o Capacitated Splittable Tree JRP.*

Referências

Bell, W. J., Dalberto, L. M., Fisher, M. L., Greenfield, A. J., Jaikumar, R., Kedia, P., Mack, R. G., and Prutzman, P. J. (1983). Improving the distribution of industrial gases with an on-line computerized routing and scheduling optimizer. *Interfaces*.

Cheung, M., Elmachtoub, A., Levi, R., and Shmoys, D. (2015). The submodular joint replenishment problem. *Mathematical Programming*.

Fukunaga, T., Nikzad, A., and Ravi, R. (2014). Deliver or hold: Approximation algorithms for the periodic inventory routing problem. *LIPIcs*.

Jiao, Y. and Ravi, R. (2019). Inventory routing problem with facility location. *ArXiv*.

Nagarajan, V. and Shi, C. (2015). Approximation algorithms for inventory problems with submodular or routing costs. *Mathematical Programming*.

Pedrosa, L. L. C. (2014). *Approximation Algorithms for Facility Location Problems and Other Supply Chain Problems*. PhD thesis, Instituto de Compução, Unicamp.