Sobre o Problema do Caixeiro Viajante com Drone*

Pedro H. D. B. Hokama[1], Carla N. Lintzmayer[2], Mário César San Felice[3]

[1]Instituto de Matemática e Computação – Universidade Federal de Itajubá – Brasil

[2]Centro de Matemática, Computação e Cognição – Universidade Federal do ABC – Brasil

[3]Departamento de Computação – Universidade Federal de São Carlos – Brasil

hokama@unifei.edu.br, carla.negri@ufabc.edu.br, felice@ufscar.br

Abstract. *We address the Flying Sidekick Traveling Salesman Problem, show a structural property respected by an optimal solution for any instance, and use it to provide the most efficient algorithm in the literature among those that create solutions based on inserting the drone into Hamiltonian cycles. Our algorithm takes linear time for cycles obtained by the Nearest Neighbor heuristic.*

Resumo. *Abordamos o problema do Caixeiro Viajante com Companheiro Voador, mostramos uma propriedade estrutural respeitada por uma solução ótima para qualquer instância, e usamos a mesma para propor o algoritmo mais eficiente da literatura dentre os que criam soluções baseadas na inserção do drone em ciclos Hamiltonianos. Nosso algoritmo leva tempo linear para ciclos obtidos pela heurística Nearest Neighbor.*

1. Introdução

No Problema do Caixeiro Viajante com Companheiro Voador (*FSTSP*, de *Flying Sidekick Traveling Salesman Problem*), definido por [Murray and Chu 2015], temos um caminhão que realiza uma rota para fazer entregas a clientes, podendo lançar um drone para fazer entregas para outros clientes com objetivo de minimizar o tempo de serviço. Sendo uma generalização do clássico TSP, o FSTSP também é NP-difícil, sendo que várias heurísticas já foram propostas na literatura para tratá-lo [Macrina et al. 2020]. Estamos interessados em heurísticas do tipo *ordenar primeiro, dividir depois*, em que primeiro constrói-se um ciclo Hamiltoniano sobre todos os clientes, e depois remove-se alguns clientes para serem atendidos pelo drone. Por isso, definimos o problema *h-FSTSP*, que consiste em resolver o FSTSP de forma que tanto a ordem da rota do caminhão quanto a ordem da rota do drone *respeitem* a ordem de um ciclo Hamiltoniano inicial h dado.

De fato, várias heurísticas conhecidas para o FSTSP [Agatz et al. 2018, Ha et al. 2018, Kundu et al. 2022], resolvem (otimamente) o h-FSTSP e têm tempo de execução $O(n^3)$ (dado que o ciclo Hamiltoniano já foi calculado), basicamente porque consideram todas as combinações dos n nós de h para lançamento, entrega e recuperação do drone. Neste trabalho, apresentamos a "Propriedade do Drone Preguiçoso", que garante que apenas algumas dessas combinações precisam ser consideradas por algoritmos que resolvem o h-FSTSP. Desenvolvemos um algoritmo exato para o h-FSTSP que usa esta propriedade e mostramos que seu tempo de execução é $O(n)$ se h for gerado pela heurística *Nearest Neighbor* [Bellmore and Nemhauser 1968].

*apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Código de Financiamento 001. C. N. Lintzmayer foi parcialmente financiada pelo CNPq - Proc. 312026/2021-8, e por L'ORÉAL-UNESCO-ABC Para Mulheres na Ciência.

2. Notação e Definições

Dado um inteiro $n > 0$, consideraremos que o conjunto $N = \{0, 1, \ldots, n+1\}$ contém todos os nós da nossa rede, sendo $1, \ldots, n$ os clientes e $0 \equiv n+1$ o depósito. Dados $i \in N \setminus \{n+1\}$ e $j \in N \setminus \{0\}$, denotamos por $t(i,j)$ o tempo que o caminhão leva para ir de i a j. Ademais, dado um número $\alpha \in \mathbb{R}^+$, consideramos que $t(i,j)/\alpha$ é o tempo que o drone leva para ir de i a j. Chamamos α de *velocidade do drone*. Consideramos o tempo do drone como uma proporção do tempo do caminhão devido às principais instâncias que são usadas na literatura do FSTSP [Bouman et al. 2018]. Porém, originalmente o problema define duas funções de tempo, uma para cada veículo.

Uma *operação* o é um par (r, d) de sequências de nós, onde r é associada ao caminhão e d é associada ao drone. A sequência $r = (v_1^r, \ldots, v_{|r|}^r)$ descreve a *rota do caminhão* desde um nó inicial $v_1^r \in N \setminus \{n+1\}$ até um nó final $v_{|r|}^r \in N \setminus \{0\}$, com $|r| \geq 2$ e leva tempo $t(r) = \sum_{\ell=1}^{|r|-1} t(v_\ell, v_{\ell+1})$ para acontecer. A sequência d pode ser vazia, indicando que o caminhão está carregando o drone naquela operação. Se não for vazia, então $d = (v_1^r, v^d, v_{|r|}^r)$ descreve a *rota do drone*, com seu primeiro e último nós coincidindo com aqueles de r, e ela leva tempo $t(d) = (t(v_1^r, v^d) + t(v^d, v_{|r|}^r))/\alpha$. O nó v^d é chamado *nó de drone*, v_1^r é chamado de *nó de lançamento* e $v_{|r|}^r$ é chamado de *nó de encontro*. Finalmente, o tempo da operação $o = (r, d)$, denotado $t(o)$, é dado por $\max\{t(r), t(d)\}$ se d não é vazia, ou por $t(r)$, caso contrário.

Por fim, consideraremos $h = (v_0, v_1, \ldots, v_{n+1})$ como sendo um ciclo Hamiltoniano sobre N em que $v_0 = 0$ e $v_{n+1} = n+1$. No h-FSTSP, dada a entrada (n, h, t, α), deseja-se encontrar uma sequência $S = (o_1, \ldots, o_{|S|})$ de operações que satisfaz: (i) o primeiro nó de r_1 e o último nó de $r_{|S|}$ são o depósito, que não aparece em nenhuma outra operação; (ii) para cada $\ell \in \{1, \ldots, |S|-1\}$, o último nó de r_ℓ é igual ao primeiro de $r_{\ell+1}$; (iii) todos os outros nós da rede aparecem exatamente uma vez em alguma sequência r_ℓ, ou como nó de drone em alguma d_ℓ; (iv) a ordem de h é respeitada por todas as operações; e (v) o tempo total $t(S) = \sum_{\ell=1}^{|S|} t(o_\ell)$ é o menor possível.

3. Propriedade do Drone Preguiçoso

Dado o ciclo Hamiltoniano $h = (v_0, v_1, \ldots, v_{n+1})$, para quaisquer três nós v_i, v_j, v_k tais que $0 \leq i < j < k \leq n+1$, definimos $o_{i,j,k} = (r, d)$, com $r = (v_i, \ldots, v_{j-1}, v_{j+1}, \ldots, v_k)$ e $d = (v_i, v_j, v_k)$, como uma operação em que usamos o drone para atender o nó v_j. Assim, por definição, $t(o_{i,j,k}) = \max\{t(r), t(d)\}$, isto é,

$$t(o_{i,j,k}) = \max\left\{\sum_{\ell=i}^{j-2} t(v_\ell, v_{\ell+1}) + t(v_{j-1}, v_{j+1}) + \sum_{\ell=j+1}^{k-1} t(v_\ell, v_{\ell+1}),\ \frac{t(v_i, v_j) + t(v_j, v_k)}{\alpha}\right\}.$$

Para dois trios $i < j < k$ e $i' < j' < k'$, dizemos que $o_{i,j,k}$ *está contida em* $o_{i',j',k'}$ *com respeito a* h se $i' \leq i$ e $k' \geq k$. Ademais, dizemos que $o_{i,j,k}$ *domina* $o_{i',j',k'}$ *com respeito a* h se esta está contida naquela e se $t(o_{i',j',k'}) \geq \sum_{\ell=i'}^{i-1} t(v_\ell, v_{\ell+1}) + t(o_{i,j,k}) + \sum_{\ell=k}^{k'-1} t(v_\ell, v_{\ell+1})$. Note que ambos os lados dessa inequação correspondem aos custos de visitar o mesmo conjunto de nós, e que ambos começam e terminam nos mesmos nós. O seguinte resultado mostra que operações que são dominadas por outras não são necessárias para que se chegue a uma solução ótima.

Lema 1. *Existe uma solução ótima para o FSTSP (ou h-FSTSP) que não possui operações dominadas*[1].

Dizemos que o drone é *rápido na operação* $o_{i,j,k}$ se $t(d) \leq t(r)$, isto é, o drone não deixa o caminhão esperando. Isso implica que $t(o_{i,j,k}) = t(r)$. O resultado a seguir mostra que se o drone é rápido na operação $o_{i,j,k}$, então, para qualquer $i' \leq i$ e $k' \geq k$, substituir $o_{i,j,k}$ por $o_{i',j,k'}$ em uma solução não pode diminuir o custo da solução.

Lema 2. *Sejam v_i, v_j, v_k três nós tais que $0 \leq i < j < k \leq n+1$. Se o drone é rápido na operação $o_{i,j,k}$, então para qualquer par de nós $v_{i'}, v_{k'}$ com $i' \leq i$ e $k' \geq k$, a operação $o_{i,j,k}$ domina $o_{i',j,k'}$ com respeito a h.*

O resultado anterior estabelece uma condição para identificar uma família de operações dominadas. O resultado a seguir mostra como essa ideia pode ajudar a deixar um algoritmo para o h-FSTSP mais rápido ignorando operações desnecessárias.

Teorema 3 (*Propriedade do Drone Preguiçoso*)**.** *Sejam v_i, v_j, v_k três nós do ciclo Hamiltoniano h tais que $i < j < k$. Se o drone é rápido em $o_{i,j,k}$, então qualquer outra operação $o_{i',j,k'}$ com $i' \leq i$ e $k' \geq k$ pode ser ignorada por um algoritmo para o h-FSTSP.*

4. Algoritmo SplitLazy

[Ha et al. 2018] propuseram uma heurística do tipo *ordenar primeiro, dividir depois* para o FSTSP com uma função objetivo diferente. A parte de dividir da heurística deles é o algoritmo Split, que resolve o h-FSTSP. [Kundu et al. 2022] também apresentaram uma heurística parecida com o Split, mas com algumas modificações que melhoram os resultados empíricos, sendo atualmente o algoritmo mais rápido da literatura para o h-FSTSP. Ambos algoritmos são conceitualmente similares ao de [Agatz et al. 2018], que foi a primeira heurística do tipo para o FSTSP, e os três têm tempo de execução $O(n^3)$.

Nosso algoritmo SplitLazy possui similaridades com os algoritmos mencionados, porém faz uso da Propriedade do Drone Preguiçoso para resolver o h-FSTSP muito mais rápido na prática, apesar de ainda ter complexidade de tempo $O(n^3)$ no pior caso. O SplitLazy constrói um digrafo G com conjunto de nós N no qual cada arco v_iv_k corresponde a uma operação e pode ser associado a um nó de drone. Se nenhum nó de drone for associado, a rota do caminhão dessa operação é apenas $r = (v_i, v_{i+1}, \ldots, v_k)$, sendo a rota do drone vazia. Se um nó de drone v_j for associado ao arco v_iv_k, então $i < j < k$ e a rota do caminhão é $r = (v_i, v_{i+1}, \ldots, v_{j-1}, v_{j+1}, \ldots, v_k)$ enquanto a rota do drone é $d = (v_i, v_j, v_k)$. O custo de um arco é o tempo da operação.

Inicialmente, o algoritmo adiciona a G os arcos de h. Seus três laços aninhados testarão triplas (i, j, k) onde v_i será um nó de lançamento, v_k um nó de encontro e v_j um nó de drone. O primeiro laço considera cada candidato v_j, enquanto o segundo e terceiro laços consideram pares (v_i, v_k) que gradualmente se distanciam de v_j, isto é, o índice i decresce de $j-1$ a 0, enquanto o índice k aumenta de $j+1$ a $n+1$. Se o algoritmo detectar que o drone é rápido em uma operação $o_{i,j,k}$, ele desiste de verificar operações dominadas por $o_{i,j,k}$ que tenham nó de drone v_j, i.e., com nó de lançamento com índice menor do que i e nó de encontro maior que k. Por outro lado, uma operação rápida $o_{i,j,k}$ com $k > j+1$ não domina operações com nó de lançamento menor do que i e nó de encontro menor do que k. Por isso, essas operações não dominadas precisam ser consideradas pelo algoritmo.

[1] As demonstrações deste artigo foram omitidas por limitação de espaço.

Quando algum nó de drone v_j está sendo considerado pelo algoritmo e este encontra uma operação $o_{i,j,k}$ que o faz adicionar o arco v_iv_k, pode acontecer de em uma iteração posterior, sobre um nó de drone $v_{j'}$, a operação $o_{i,j',k}$ ter custo menor. Nesse caso, o custo do arco v_iv_k é alterado, bem como o nó de drone associado a ele. É por isso que no algoritmo nós precisamos atualizar arcos ao invés de apenas adicioná-los.

Por fim, após a construção de G, o algoritmo encontra um caminho mínimo em G e o usa para construir uma solução para o FSTSP. Ela é ótima para o h-FSTSP considerando o ciclo h e não considera todas as $\binom{n}{3}$ possíveis operações para construir G porque usa a Propriedade do Drone Preguiçoso para evitar operações que são dominadas por outras.

5. 1-Localidade e Eficiência do SplitLazy para Nearest Neighbor

Dizemos que um nó v_j do ciclo Hamiltoniano $h = (v_0, v_1, \ldots, v_{n+1})$ tem 1-*localidade* se $t(v_{j-1}, v_{j+1}) \geq \frac{1}{\alpha}\big(t(v_{j-1}, v_j) + t(v_j, v_{j+1})\big)$, i.e., a operação $o_{j-1,j,j+1}$ domina todas as operações com nó de drone v_j. Pela forma como o algoritmo SplitLazy funciona, podemos garantir um melhor desempenho de pior caso sobre ciclos Hamiltonianos cujos nós possuam 1-localidade.

Teorema 4. *O algoritmo* SplitLazy *tem tempo de execução $O(n)$ sobre instâncias (n, h, t, α) em que todos os nós de h, exceto talvez pelo último cliente, têm 1-localidade.*

Ideia da prova. O algoritmo SplitLazy leva tempo constante em cada nó com localidade 1 e, no máximo, tempo linear no nó do último cliente. □

A clássica heurística Nearest Neighbor [Bellmore and Nemhauser 1968] para o TSP começa do depósito e, em cada iteração, estende o caminho sendo construído com o nó mais próximo do último nó adicionado. Conseguimos mostrar que esta heurística gera ciclos Hamiltonianos favoráveis para nosso algoritmo.

Lema 5. *Em grafos métricos, a heurística Nearest Neighbor produz ciclos Hamiltonianos em que todos os nós, exceto talvez pelo último cliente, têm 1-localidade quando $\alpha \geq 3$.*

Ideia da prova. A escolha gulosa da heurística garante que, para quaisquer três nós consecutivos, o primeiro não pode estar mais próximo do terceiro que do segundo. □

É interessante notar que, existem instâncias que sempre possuem um nó com localidade maior que 1 para qualquer ciclo Hamiltoniano. Curiosamente, no caso da heurística Nearest Neighbor este nó sempre é o último cliente selecionado.

Lema 6. *Existem instâncias para as quais todo ciclo Hamiltoniano tem pelo menos um nó com localidade maior que 1.*

Ideia da prova. Tome uma instância em que um nó está distante de todos os demais. □

6. Conclusão e Trabalhos Futuros

Neste trabalho apresentamos o algoritmo SplitLazy, que é o algoritmo mais eficiente para resolver o h-FSTSP, e mostramos que ele leva tempo linear quando h é obtido pela heurística Nearest Neighbor com $\alpha \geq 3$, caso em que h apresenta 1-localidade. Podemos generalizar o conceito de localidade, bem como analisar as características estruturais de circuitos Hamiltonianos produzidos por outras heurísticas para o TSP, a fim de provar melhores garantias de eficiência de pior caso para nosso algoritmo. Também temos interesse em generalizar a propriedade do Drone Preguiçoso para problemas relacionados.

Referências

Agatz, N., Bouman, P., and Schmidt, M. (2018). Optimization Approaches for the Traveling Salesman Problem with Drone. *Transportation Science*, 52(4):965–981.

Bellmore, M. and Nemhauser, G. L. (1968). The traveling salesman problem: A survey. *Operations Research*, 16(3):538–558.

Bouman, P., Agatz, N., and Schmidt, M. (2018). Instances for the TSP with Drone (and some solutions).

Ha, Q. M., Deville, Y., Pham, Q. D., and Hà, M. H. (2018). On the min-cost Traveling Salesman Problem with Drone. *Transportation Research Part C: Emerging Technologies*, 86:597–621.

Kundu, A., Escobar, R. G., and Matis, T. I. (2022). An efficient routing heuristic for a drone-assisted delivery problem. *IMA Journal of Management Mathematics*, 33(4):583–601.

Macrina, G., Di Puglia Pugliese, L., Guerriero, F., and Laporte, G. (2020). Drone-aided routing: A literature review. *Transportation Research Part C: Emerging Technologies*, 120:102762.

Murray, C. C. and Chu, A. G. (2015). The flying sidekick traveling salesman problem: Optimization of drone-assisted parcel delivery. *Transportation Research Part C: Emerging Technologies*, 54:86–109.

A. Provas dos Lemas 1 e 2

Demonstração do Lema 1. Considere uma solução ótima S^* para o FSTSP com uma operação $o_{i',j',k'} = (r', d')$, que é dominada por $o_{i,j,k} = (r, d)$ com relação a uma sequência $h = (v_0, \ldots, v_{n+1})$. Considere outra solução S construída para S^* que substitui $o_{i',j',k'}$ pelas operações $((v_{i'}, \ldots, v_i), ())$, (r, d), e $((v_k, \ldots, v_{k'}), ())$. Note que

$$t(S^*) - t(S) = t(o_{i',j',k'}) - \left(\sum_{\ell=i'}^{i-1} t(v_\ell, v_{\ell+1}) + t(o_{i,j,k}) + \sum_{\ell=k}^{k'-1} t(v_\ell, v_{\ell+1}) \right) \geq 0 \ ,$$

sendo que a última desigualdade vale porque $o_{i,j,k}$ domina $o_{i',j',k'}$ com respeito à sequência h. □

Demonstração do Lema 2. Por definição,

$$t(o_{i',j,k'}) = \max \left\{ \begin{array}{l} \displaystyle\sum_{\ell=i'}^{j-2} t(v_\ell, v_{\ell+1}) + t(v_{j-1}, v_{j+1}) + \sum_{\ell=j+1}^{k'-1} t(v_\ell, v_{\ell+1}) \, , \\ (t(v_{i'}, v_j) + t(v_j, v_{k'}))/\alpha \end{array} \right\} \ .$$

Assim, claramente,

$$t(o_{i',j,k'}) \geq \sum_{\ell=i'}^{j-2} t(v_\ell, v_{\ell+1}) + t(v_{j-1}, v_{j+1}) + \sum_{\ell=j+1}^{k'-1} t(v_\ell, v_{\ell+1}) \ . \tag{1}$$

Como o drone é rápido na operação $o_{i,j,k}$, nós sabemos que

$$t(o_{i,j,k}) = \sum_{\ell=i}^{j-2} t(v_\ell, v_{\ell+1}) + t(v_{j-1}, v_{j+1}) + \sum_{\ell=j+1}^{k-1} t(v_\ell, v_{\ell+1}) \ ,$$

o que, junto da Eq. (1), e porque $i' \leq i$ e $k' \geq k$, significa que

$$t(o_{i',j,k'}) \geq \sum_{\ell=i'}^{i-1} t(v_\ell, v_{\ell+1}) + t(o_{i,j,k}) + \sum_{\ell=k}^{k'-1} t(v_\ell, v_{\ell+1}) \ .$$

Portanto, por definição, $o_{i,j,k}$ domina $o_{i',j,k'}$. □

B. Pseudocódigo do algoritmo apresentado na Seção 4

Algoritmo 1 SPLITLAZY(n, h, t, α)

Input: inteiro $n > 0$, ciclo Hamiltoniano $h = (v_0, v_1, \ldots, v_{n+1})$ sobre N, $t(u, v)$ para qualquer par $u, v \in N$ e real $\alpha > 0$
Output: uma solução S para o FSTSP

seja G um digrafo com $V(G) = N$ e $E(G) = \emptyset$ ▷ *Cada arco corresponderá a uma operação*
Para $i = 0$ até n **faça** ▷ *Inicialmente, cada operação não está usando o drone*
    adicione $v_i v_{i+1}$ a $E(G)$ com custo $t(v_i, v_{i+1})$ e nenhum nó de drone associado
Para $j = 1$ até n **faça** ▷ *Considere cada nó v_j como possível nó de drone*
    base_cost $\leftarrow t(v_{j-1}, v_{j+1})$
    pre_cost $\leftarrow -t(v_{j-1}, v_j)$
    $k_{max} \leftarrow n + 1$
    Para $i = j - 1$ até 0 **faça** ▷ *Considere cada possível nó de lançamento v_i*
        $k \leftarrow j + 1$ ▷ *Considere o nó de encontro mais próximo v_k*
        drone_cost $\leftarrow \frac{1}{\alpha}(t(v_i, v_j) + t(v_j, v_k))$
        pre_cost $\leftarrow$ pre_cost $+ t(v_i, v_{i+1})$
        pos_cost $\leftarrow 0$
        truck_cost $\leftarrow$ pre_cost $+$ base_cost $+$ pos_cost
        Se o arco $v_i v_k \notin E(G)$ ou seu custo é maior do que $\max\{$drone_cost, truck_cost$\}$ **então**
            adicione ou atualize o arco $v_i v_k$ com custo $\max\{$drone_cost, truck_cost$\}$ e nó de drone v_j
        Se drone_cost $\leq$ truck_cost **então** ▷ *Encontrou uma operação $o_{i,j,k}$ em que o drone é rápido*
            break
        Para $k = j + 2$ até k_{max} **faça** ▷ *Considere cada possível nó de encontro v_k com $k \geq j + 2$*
            drone_cost $\leftarrow \frac{1}{\alpha}(t(v_i, v_j) + t(v_j, v_k))$
            pos_cost $\leftarrow$ pos_cost $+ t(v_{k-1}, v_k)$
            truck_cost $\leftarrow$ pre_cost $+$ base_cost $+$ pos_cost
            Se o arco $v_i v_k \notin E(G)$ ou seu custo é maior do que $\max\{$drone_cost, truck_cost$\}$ **então**
                adicione ou atualize $v_i v_k$ com custo $\max\{$drone_cost, truck_cost$\}$ e nó de drone v_j
            Se drone_cost $\leq$ truck_cost **então** ▷ *Encontrou uma operação $o_{i,j,k}$ em que o drone é rápido*
                $k_{max} \leftarrow k - 1$
                break
Encontre um caminho mínimo P de v_0 a v_{n+1} em G
Construa uma solução S com as operações correspondentes a cada arco em P
Devolve S

C. Resultados comparando o Algoritmo SplitLazy com outros da literatura.

A Tabela 1 mostra os resultados dos algoritmos quando particionados por tamanho e por valor de α. A amplitude média de redução dos custos se relaciona com o valor de α, o que não é surpreendente, já que um α menor significa um drone mais lento. As colunas AGATZ, SPLIT e KUNDU apresentam os tempos dos algoritmos propostos por [Agatz et al. 2018], [Ha et al. 2018] e [Kundu et al. 2022], respectivamente. As colunas LAZYMATRIX e LAZYLISTS apresentam os tempos de implementações do algoritmo SPLITLAZY usando, respectivamente, matriz e listas de adjacência. Podemos observar que as implementações do Algoritmo SPLITLAZY são muito mais velozes que os demais algoritmos da literatura para o h-FSTSP, e que isso se deve ao nosso algoritmo precisar verificar muito menos triplas que os demais.

Tabela 1. Comparação entre algoritmos para o h-FSTSP partindo de um ciclo produzido pela heurística Nearest Neighbor

# de Nós	α	# de Inst.	Redução de Custo	AGATZ (μs)	SPLIT (μs)	KUNDU (μs)	LAZYMATRIX (μs)	LAZYLISTS (μs)	Triplas / $(n+1)$ Média	Triplas / $(n+1)$ Desv.Pad.
≤ 10	*	540	-29.04%	5,795	59	14,495	31	26	1.88	1.33
20		90	-25.84%	6,655	541	14,259	56	37	2.46	2.06
50		90	-26.66%	13,065	7,751	16,749	139	72	2.78	2.48
75		90	-23.98%	25,689	26,157	17,350	244	101	2.80	2.52
100		90	-22.55%	35,824	61,499	19,480	373	119	2.84	2.58
175		90	-22.32%	76,165	327,691	24,203	889	202	2.85	2.57
250		180	-22.02%	140,757	960,616	28,001	1,639	267	2.83	2.55
375		127	-22.68%	361,981	3,236,189	36,282	3,505	383	2.82	2.53
500		92	-21.85%	798,070	7,689,411	47,120	6,002	507	2.83	2.52
≤ 75	1	270	-21.19%	8,674	3,835	14,431	84	58	4.43	1.27
≥ 100 and ≤ 250		120	-17.46%	98,967	577,095	23,539	1,291	384	6.43	0.38
≥ 375		73	-17.69%	547,476	5,124,654	40,601	4,879	815	6.37	0.28
≤ 75	2	270	-30.83%	9,098	3,840	15,337	63	32	1.08	0.19
≥ 100 and ≤ 250		120	-24.45%	98,430	577,693	25,499	1,053	135	1.09	0.04
≥ 375		74	-24.47%	543,088	5,106,614	41,458	4,436	249	1.10	0.02
≤ 75	3	270	-31.57%	8,955	3,926	15,341	61	32	0.92	0.08
≥ 100 and ≤ 250		120	-24.76%	97,730	578,028	25,727	1,061	122	1.00	0.00
≥ 375		72	-24.84%	544,996	5,089,344	40,433	4,345	240	1.00	0.00