Implementação de um algoritmo construtivo para o Problema de Distritamento

Arthur Henrique Sousa Cruz

Universidade Federal de Lavras

1 Introdução

Problemas de Distritamento consistem na divisão de unidades territoriais (ou unidades básicas) em distritos que são balanceados de acordo com certos atributos, como a qualidade do solo de um terreno agrícola ou a demanda de produtos em um centro comercial Kalcsics and Ríos-Mercado (2019); Gliesch et al. (2017, 2018). Essa classe de problemas pode ser encontrada em diversas áreas, como, por exemplo, em assistência médica domiciliar Blais et al. (2003), distribuição de energia Bergey et al. (2003), reforma agrária Gliesch et al. (2017), divisão de setores eleitorais Bozkaya et al. (2003) e distribuição de mercadorias Gliesch et al. (2018).

No Problema de Distramento Comercial (PDC) tem-se como unidades básicas blocos de uma cidade e o objetivo é a geração de distritos conexos balanceados de acordo com a carga de trabalho, a demanda e a quantidade de clientes. Como objetivo tem-se a geração de distritos tão compactos quanto possível Kalcsics and Ríos-Mercado (2019); Gliesch et al. (2018). Em Gliesch and Ritt (2019), os autores apresentam um *framework* baseado no método proposto por Gliesch et al. (2018) para a resolução de problemas de distritamento. O algoritmo conta com três possibilidades para a função objetivo, sendo duas delas baseadas no centro do distritos e a terceira no diâmetro. Foi utilizada uma Busca Tabu com diferentes buscas locais que focam em melhorar somente o balanceamento ou somente a compacidade. A meta-heurística foi capaz de encontrar resultados factíveis para todas as instâncias testadas, tornando-se assim o estado da arte para o problema.

O trabalho de Vidal et al. (2019) aponta que trabalhos que envolvem tanto problemas de distritamento quanto o PRV estão em emergência. Gliesch et al. (2020) aborda o Problema de Distritamento Comercial tendo como objetivo minimizar as rotas dos distritos gerados, enquanto Bender et al. (2020) explora o distritamento para obter distritos em um cenário de otimização diário.

Neste trabalho será apresentada uma implementação de uma simplificação da heurística construtiva apresentada em Gliesch et al. (2018) para o solução do Problema de Distritamento Comercial. O intuito é a utilização do algoritmo completo como parte do problema estudado pelo Projeto de Mestrado, cujo objetivo é a solução do Problema de *Pickup* e *Delivery* com Janelas de Tempo (Ropke and Pisinger, 2006; Sartori and Buriol, 2020) em um contexto onde há grande quantidade de reotimizações ao longo do dia. Para isso, foi iniciada sua implementação durante a disciplina.

2 Definição Formal

Baseando-se em Gliesch and Ritt (2019), define-se uma simplificação do PDC como se segue. Dado um grafo planar não-direcionado $G = (V, A)$, tal que $\mid V \mid = n$ e $\mid A \mid = m$, tem-se como objetivo dividir as unidades básicas $v \in V$ em $p \leq n$ distritos. Cada distrito deve ser conexo e balanceado de acordo com o número de vértices. Uma aresta $(u, v) \in A$ com $u, v \in V$ tem uma distância definida por d_{uv}.

Uma solução S é uma partição de nós nos distritos, de forma que $S_1 \cup ... \cup S_p \subseteq V$, sendo S_i o i-ésimo distrito. Uma solução completa é uma solução S tal que $\cup_{i=1}^{p} S_i = V$, sendo incompleta caso contrário. Diz-se que $S(v) = i$ se o vértice $v \in V$ está alocado no distrito i e, caso não esteja alocado a nenhum distrito, dizemos que ele está desalocado. Uma solução é incompleta se, e somente se, ela contém ao menos um vértice desalocado.

Cada distrito i contém uma borda $\delta S_i = \{u : S_i \iff \exists v : (u, v) \in A, v \notin S_i\}$, ou seja, vértices pertencentes ao distrito i que são adjacentes a vértices que não pertencem a i. Ao mesmo tempo, define-se os candidatos de um distrito $\delta S_i = \{u \notin S_i, v \in \partial S_i : (u, v) \in A\}$ como os vértices que são adjacentes à borda de S_i. Diz-se que dois distritos i e j são vizinhos se $\delta S_i \cap \partial S_j \neq \varnothing$.

Um distrito é dito balanceado se o desvio de seu número de vértices não exceder um fator τ em relaçao a um valor de balanceamento ideal dado por $\lambda = n/p$. Assim, define-se que o balanceamento do distrito μ_i é dado por

$$\mu_i = \frac{\mid \lambda - n_i \mid}{\lambda}, \tag{1}$$

sendo n_i o número de vértices contido em S_i. A divisão por λ é feita para que os valores sejam normalizados em relação à média ideal de vértices por distrito, permitindo assim a comparação com o valor τ. Diz-se que um distrito está balanceado se, e somente se, $\mu_i \leq \tau$. Uma solução é considerada balanceada se todos os seus distritos são balanceados.

O diâmetro de um distrito é dado pela maior entre as menores distâncias eucldianas de todos os vértices, sendo assim, temos que o diâmetro de um distrito i é $D_i = \arg\max \{d_{u,v} u, v \in S_i\}$. O objetivo a ser minimizado neste trabalho são os diâmetros dos distritos e, portanto, define-se o diâmetro da solução como $D = \arg\max D_i$, com $i = 1, ..., p$. Se uma solução S é balanceada e tem todos os distritos completos e conexos, dizemos que S é factível. Caso não exista qualquer distribuição de distritos que possa diminuir o valor do diâmetro em uma solução factível, dizemos que ela é ótima.

3 Metodologia

Como já dito, para resolver o problema apresentado, foi implementada uma simplificação da heurística construtiva utilizada por Gliesch and Ritt (2019) e proposta por Gliesch et al. (2018). O Algoritmo 1 apresenta o pseudocódigo do algoritmo proposto. Inicialmente, é necessário selecionar sementes para cada

um dos distritos (linha 1). Para isso, primeiro sorteia-se a semente do primeiro distrito e adiciona-se ao conjunto de sementes π. Após isso são escolhidas aleatoriamente $\sqrt{n}$ vértices candidatos a sementes e o que será adicionado à π será o vértice v tal que $d_{uv} = \arg\max_{w \in \pi} d_{wv}$, com $u \in \pi$.

Após a escolha das sementes, os distritos são inicializados e adicionados a uma *Heap* ordenada crescentemente pelo balanceamento dos distritos (linhas 2 à 3). Com os distritos criados, inicia-se a fase de expansão (linhas 4 até 10), que persiste enquanto houver distritos na *Heap*. No laço, é selecionado o distrito no topo da *Heap* e, caso ele tenha candidatos, é selecionado o candidato com menor distância para seu adjacente pertencente ao conjunto de boradas do distrito. O candidato é alocado ao distrito e o distrito readicionado à *Heap*. Caso o distrito removido da *Heap* não tenha candidatos, significa que não há mais como expandí-lo e, por isso, ele se mantém fora da *Heap*.

Algorithm 1 Heurísrica construtiva

```
1: π = seleciona_sementes(p)
2: S = cria_distritos(sementes)
3: H = inicializa_heap(S)
4: while H ≠ ∅ do
5:    Dist = H.remove()
6:    if ∂Dist ≠ ∅ then
7:       v = melhor_candidato(∂Dist)
8:       aloca(v, Dist)
9:       H.adiciona(Dist)
10:   end if
11: end while
```

É fácil notar a baixa complexidade deste algoritmo. A seleção de sementes tem custo de $O(p^2\sqrt{n})$, visto que, para cada distrito, são selecionados $\sqrt{n}$ vértices e sua distância é calculada para $1, 2, ..., p$ sementes. A criação de cada distrito tem o custo de alocação de um vértice, que será discutido mais à frente. Para a criação da *Heap* faz-se a inserção de um distrito de cada vez e, como a estrutura tem um tempo de inserção logarítmico, a complexidade total se dá por $O(p \log p)$.

O laço é repetido enquanto houver vértices e, como é adicionado um vértice por iteração, ele executas n vezes. Assim, o número de vezes que se realiza a remoção da *Heap* e o teste para verificar se há candidatos é de $O(n \log p)$ e $O(n)$ respectivamente. O tempo de seleção do melhor candidato depende do número de candidatos que há no distrito. Segundo Gliesch and Ritt (2019), este número costuma ser próximo de $\sqrt{n}$ em situações práticas, logo, considerando a utilização de uma lista, assume-se que o tempo gasto nas buscas pelo melhor candidato é de aproximadamente $O(n\sqrt{n})$. A alocação de um vértice tem o custo de adicionar seus candidatos. Como um grafo planar tem grau médio de no máximo 6, tem-se, em m édia, um tempo constante para a atualização dos candidatos, visto que,

mantendo-se uma lista, é possível inserir cada um deles também em $O(1)$. Assim, a alocação de todos os vértices tem complexidade de $O(n)$

Por fim, as readições na *Heap* têm custo total de $O(n \log p)$, já que adicionar um único distrito tem custo $O(\log p)$. A complexidade de tempo do algoritmo seria, então, $O(p^2\sqrt{n} + p \log p + n \log p + n\sqrt{n})$, ou seja, $O((p^2 + n)\sqrt{n} + (n + p) \log p)$. Como tem-se um valor de n extremamente maior que p para casos práticos, podemos considerar que p não causa grande influência no tempo de execução, o que nos dá uma complexidade de $O(n\sqrt{n} + n) = O(n\sqrt{n})$.

Como o algoritmo tem uma grande quantidade de verificações de vizinhança e também de acesso aos vizinhos, foi mantida em memória uma Lista de Adjacência e uma Matriz de Adjacência, para aproveitar as vantagens de ambas.

4 Testes e Resultados

Para a verificação do funcionamento da heurística ela foi implementada em $C++$ e testadas em instâncias da literatura propostas por Rio e Gliesch et al. (2018). As instâncias utilizadas se encontram em Cruz (2020a). O código fonte pode ser encontrado em Cruz (2020b) e os resultados obtidos podem ser vistos em Cruz (2020c).

Cada uma das instâncias foi executada 5 vezes e a Tabela 1 mostra os resultados obtidos para as execuções. A primeira coluna indica o nome da instância. A segunda coluna, $\mid C \mid$, indica o número de vezes que foram obtidas soluções com todos os distritos conexos. A terceira coluna ($\mid B \mid$) indica quantas das soluções foram balanceadas, enquanto a quarta, $T(s)$, indica o tempo médio das 5 execuções. Por fim a coluna D indica o valor médio obtido para o diâmetro e μ a média do desbalanceamento total das soluções (sendo que o desbalanceamento total é o somatório do desbalanceamento dos distritos).

Tabela 1: Resultados obtidos após a execução das instâncias.

	$\mid C \mid$	$\mid B \mid$	$T(s)$	D	μ
d500-01	5	0	0.061	134.193	3.16
d500-20	5	0	0.079	123.071	9.424
del-n10000-k160-s7725	5	0	14.448	1400.989	262.64
del-n10000-k50-s2196	5	0	15.296	1395.687	66.092
del-n1000-k5-s2292	5	5	0.382	1410.732	0.238
del-n2500-k12-s12455	5	0	1.488	1379.541	6.797
del-n5000-k25-s17706	5	0	4.511	1398.516	23.386
DT500-01	5	0	0.068	690.376	7.0
DT500-20	5	0	0.075	702.179	9.32

A coluna $\mid C \mid$ mostra que todos os distritos são conexos para todas as soluções, o que já era esperado devido a natureza do método, que anexa vizinhos dos vértices do distritos a partir de uma semente. Também era esperado o baixo

custo de tempo devido a complexidade assintótica do algoritmo. Como pode ser visto na coluna $T(s)$, somente as instâncias com 10000 vértices demoraram mais de 5 segundos para finalizar a execução. Quanto ao balanceamento, não havia qualquer garantia de que seriam obtidas soluções factíveis, sendo uma surpresa a obtenção de soluções factíveis para a instância *del*-n1000-k5-s2292, assim como o baixo valor para seu desbalanceamento total (coluna μ).

Mesmo com resultados ruins para o balanceamento e para o diâmetro, a heurística pode ser considerada um sucesso. O objetivo deste algoritmo não é encontrar uma solução que seja factível ou ótima, mas sim uma solução construída rapidamente que seja conexa, para que sejam aplicadas buscas locais ou outras formas de melhorar o resultado alcançado pela etapa de construção.

5 Conclusão e Trabalhos Futuros

O Problema de Distritamento Comercial (PDC), consiste na divisão de um território comercial de forma a gerar distritos conexos, balanceados e compactos. Neste trabalho foi implementada uma simplificação da heurística construtiva proposta por Gliesch et al. (2018) que é parte de uma meta-heurística que soluciona o PDC.

O balanceamento dos distritos foi feito com base no número de vértices do distrito e, a fim de melhorar a compacidade, a função objetivo visa reduzir o diâmetro dos distritos. A heurística foi implementada e testada com instâncias da literatura e, como resultado, foram obtidas soluções conexas em um curto período de tempo. Apesar da maioria dos distritos gerados serem desbalanceados e terem grande valor para o diâmetro, a heurística pode ser considerada um sucesso, já que seu objetivo consiste em gerar soluções iniciais.

Como continuação deste projeto tem-se a continuação da implementação da meta-heurística proposta por Gliesch and Ritt (2019), que pode ser vista como uma extensão de Gliesch et al. (2018). Ao fim, espera-se poder utilizar a meta-heurística como parte da solução do Problema de *Pickup* e *Delivery* com Janela de Tempo, proposto como o Projeto do Programa de Pós-Graduação.

Referências Bibliográficas

Matthias Bender, Jörg Kalcsics, and Anne Meyer. Districting for parcel delivery services–a two-stage solution approach and a real-world case study. *Omega*, page 102283, 2020.

Paul K Bergey, Cliff T Ragsdale, and Mangesh Hoskote. A simulated annealing genetic algorithm for the electrical power districting problem. *Annals of Operations Research*, 121(1-4):33–55, 2003.

Marko Blais, Sophie D Lapierre, and Gilbert Laporte. Solving a home-care districting problem in an urban setting. *Journal of the operational research society*, 54(11):1141–1147, 2003.

Burcin Bozkaya, Erhan Erkut, and Gilbert Laporte. A tabu search heuristic and adaptive memory procedure for political districting. *European journal of operational research*, 144(1):12–26, 2003.

Arthur H. S. Cruz. Instâncias utilizadas. `https://github.com/thuzax/ED-2020/tree/master/trab-final/Algoritmo-Construtivo/test_instances/examples`, 2020a. Acessado em setembro de 2020.

Arthur H. S. Cruz. Código fonte. `https://github.com/thuzax/ED-2020/tree/master/trab-final/Algoritmo-Construtivo/`, 2020b. Acessado em setembro de 2020.

Arthur H. S. Cruz. Resultados obtidos. `https://github.com/thuzax/ED-2020/tree/master/trab-final/Algoritmo-Construtivo/results`, 2020c. Acessado em setembro de 2020.

Alex Gliesch and Marcus Ritt. A generic approach to districting with diameter or center-based objectives. In *GECCO 2019 - Proceedings of the 2019 Genetic and Evolutionary Computation Conference*, pages 249–257, Prague, 2019. Association for Computing Machinery, Inc. ISBN 9781450361118. https://doi.org/10.1145/3321707.3321874.

Alex Gliesch, Marcus Ritt, and Mayron C.O. Moreira. A genetic algorithm for fair land allocation. In *GECCO 2017 - Proceedings of the 2017 Genetic and Evolutionary Computation Conference*, pages 793–800, Berlin, 2017. Association for Computing Machinery, Inc. ISBN 9781450349208. https://doi.org/10.1145/3071178.3071313.

Alex Gliesch, Marcus Ritt, and Mayron C. O. Moreira. A multistart alternating tabu search for commercial districting. In Arnaud Liefooghe and Manuel López-Ibáñez, editors, *Evolutionary Computation in Combinatorial Optimization*, pages 158–173, Cham, 2018. Springer International Publishing. ISBN 978-3-319-77449-7.

Alex Gliesch, Marcus Ritt, Arthur H S Cruz, and Mayron C O Moreira. A hybrid heuristic for districting problems with routing criteria. In *Proc. of the IEEE WORLD CONGRESS ON COMPUTATIONAL INTELLIGENCE (WCCI) 2020 (to appear)*, pages 1–9, Glasgow, 2020.

Jörg Kalcsics and Roger Z Ríos-Mercado. Districting problems. In *Location science*, pages 705–743. Springer, 2019.

Stefan Ropke and David Pisinger. An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. *Transportation Science*, 40(4):455–472, 2006. ISSN 15265447. https://doi.org/10.1287/trsc.1050.0135.

Carlo S Sartori and Luciana S Buriol. A study on the pickup and delivery problem with time windows: Matheuristics and new instances. *Computers & Operations Research*, page 105065, 2020.

Thibaut Vidal, Gilbert Laporte, and Piotr Matl. A concise guide to existing and emerging vehicle routing problem variants. *European Journal of Operational Research*, pages 1–16, 2019. ISSN 03772217. https://doi.org/10.1016/j.ejor.2019.10.010.