Universidade Estadual de Londrina

LUAN VICTOR DE ALMEIDA

ALGORITMO DE CRESCIMENTO GRADATIVO DE DIÂMETRO PARA DETECÇÃO DE K-*FLOCKS* EM DADOS DE TRAJETÓRIAS

LONDRINA
2022

LUAN VICTOR DE ALMEIDA

ALGORITMO DE CRESCIMENTO GRADATIVO DE DIÂMETRO PARA DETECÇÃO DE *K-FLOCKS* EM DADOS DE TRAJETÓRIAS

Trabalho de Conclusão de Curso apresentado ao curso de Bacharelado em Ciência da Computação da Universidade Estadual de Londrina para obtenção do título de Bacharel em Ciência da Computação.

Orientador: Prof(a). Dr(a). Daniel dos Santos Kaster

LONDRINA

2022

LUAN VICTOR DE ALMEIDA

ALGORITMO DE CRESCIMENTO GRADATIVO DE DIÂMETRO PARA DETECÇÃO DE K-*FLOCKS* EM DADOS DE TRAJETÓRIAS

Trabalho de Conclusão de Curso apresentado ao curso de Bacharelado em Ciência da Computação da Universidade Estadual de Londrina para obtenção do título de Bacharel em Ciência da Computação.

BANCA EXAMINADORA

Orientador: Prof(a). Dr(a). Daniel dos Santos Kaster
Universidade Estadual de Londrina

Prof. Dr. Evandro Bacarin
Universidade Estadual de Londrina

Ms. Denis Evangelista Sanches
Universidade Estadual de Londrina

Londrina, 27 de junho de 2022.

Este trabalho é dedicado às crianças adultas
que, quando pequenas, sonharam em se
tornar cientistas.

"Não vos amoldeis às estruturas deste mundo, mas transformai-vos pela renovação da mente, a fim de distinguir qual é a vontade de Deus: o que é bom, o que Lhe é agradável, o que é perfeito.
(Bíblia Sagrada, Romanos 12, 2))

ALMEIDA, L. V.. **Algoritmo de Crescimento Gradativo de Diâmetro para Detecção de *k*-*Flocks* em Dados de Trajetórias**. 2022. 34f. Trabalho de Conclusão de Curso (Bacharelado em Ciência da Computação) – Universidade Estadual de Londrina, Londrina, 2022.

RESUMO

Há uma demanda crescente por dispositivos para coletar e analisar informações relacionadas à geolocalização, incluindo a descoberta de padrões de comovimento de objetos móveis. Entre os padrões de comovimento mais relevantes está o padrão *Flock*. Um *flock* é um conjunto de objetos que se movem juntos, delimitados por um disco móvel de diâmetro fixo, por um período consecutivo de instantes de tempo. Recentemente, foi proposta uma variação desse padrão, denominada padrão *k-flocks*, que dispensa a definição do parâmetro distância. O algoritmo existente para detectar *k-flocks* segue uma abordagem *top-down*, começando por um único *flock* mais extenso e realizando subdivisões sucessivas até encontrar *k-flocks* com o menor diâmetro possível. Este trabalho propõe um novo algoritmo *bottom-up* para o problema *k-flocks*, baseado no agrupamento iterativo de elementos próximos no espaço até detectar os *k-flocks* e também k_ϵ-*Flocks* (diâmetro maximal). O trabalho descreve os fundamentos da solução, discute sua correção e apresenta o algoritmo proposto.

Palavras-chave: Padrões de comovimento; Trajetórias de objetos móveis; Padrão *Flock*.

ALMEIDA, L. V.. **Proposed Diameter Growth Algorithm for Detection of k-*Flocks* in Trajectory Data**. 2022. 34p. Final Project (Bachelor of Science in Computer Science) – State University of Londrina, Londrina, 2022.

ABSTRACT

There is a growing demand for devices to collect and analyze information related to geolocation, including moving object comovement patterns' discovery. Among the most relevant comovement patterns is the Flock Pattern. A flock is a set of objects moving together defined by a moving disk of fixed diameter for a consecutive amount of time instants. Recently, a variation of this pattern was proposed, called the *k-flocks* pattern, which eliminates the need to define the distance parameter. The existing algorithm to detect *k-flocks* follows a top-down approach starting from a single flock candidate and performing successive subdivisions until finding k flocks with the smallest diameter possible. This work proposes a new bottom-up algorithm for the *k-flocks* problem based on iteratively clustering elements close in the space until reaching the *k-flocks* and k_ϵ-*Flocks* (maximal diameter). The work describes the foundations of the solution, discusses its correctness, and presents the proposed algorithm.

Keywords: Comovement Patterns; Moving object trajectories; Flock Pattern.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

DBSCAN	Density-Based Spatial Clustering of Applications with Noise
GPS	Global Positioning System
HDBSCAN	Hierarquical Density-Based Spatial Clustering of Applications with Noise
MO	Moving Objects
MST	Minimal Spanning Tree

SUMÁRIO

1 INTRODUÇÃO

A presença crescente de sensores de posicionamento, notadamente GPS (*Global Positioning System*), tem potencializado a coleta e análise de dados de trajetórias de objetos móveis (*Moving Objects* – MOs). Uma categoria importante de análises concentra-se em objetos móveis que se movimentam próximos uns dos outros por instantes consecutivos de tempo, caracterizados como objetos móveis em *comovimento*. Os MOs podem ser pessoas, automóveis, animais ou qualquer outro dispositivo que tenha algum sensor de posicionamento instalado. Situações de comovimento incluem, por exemplo, uma carreata, animais migrando em conjunto, multidões em um protesto, etc., cujas detecções em um banco de dados de trajetórias podem ser de interesse para várias aplicações [3, 4].

Ilustrando essa ideia, em Markovic et al.[4], são explicitadas aplicações interessantes para problemas de análise e agrupamento de MOs, onde é necessário a verificação dos dados estatísticos coletados sobre o contexto da movimentação de pessoas em grupo, incluindo a modelagem do comportamento dos indivíduos. Em grandes multidões, constantemente novos indivíduos ou até grupos inteiros são inseridos em grupos que já estão em movimento, sendo essa análise um exemplo de muita importância na classificação dos caminhos escolhidos pelos indivíduos durante o seu trajeto.

Os padrões de comovimento mais conhecidos na literatura são baseados em disco de diâmetro fixo que engloba os objetos móveis [5, 6], ou em agrupamento por densidade [3]. Dentre os que utilizam discos para a análise dos agrupamentos, o padrão *flock* é um dos mais relevantes. O padrão *flock* [7, 5, 6, 1] é dado por um grupo de pelo menos μ MOs se movendo juntos cuja distância máxima está limitada por um disco fixo de diâmetro ϵ, durante um intervalo de tempo de no mínimo δ instantes de tempo consecutivos. Todavia, a escolha do parâmetro adequado que defina um disco fixo dependente do conjunto de dados, cuja definição imprecisa provoca mudanças impactantes para a análise do problema [8]. Isto torna complicado para o usuário explorar um conjunto de dados utilizando este tipo de padrão sem que tenha pleno conhecimento da distribuição de distâncias dos pontos das trajetórias. Por isso, alguns autores propuseram novas visões acerca da mesma abordagem de disco, com intuito de deixar o parâmetro de distância menos rígido.

Em um trabalho anterior [1], foi introduzida a ideia de *k-Patterns*, ou seja, a descoberta dos k-padrões de comovimento, utilizando um critério de ranqueamento no encontro das k respostas. Em particular, o padrão proposto k_ϵ-*Flocks* retorna k *flocks* de tamanho mínimo, onde k é o número de *flocks* desejado e o diâmetro do disco é um parâmetro livre, sendo adequado automaticamente ao objetivo buscado. Nesse mesmo trabalho, foi apresentado um algoritmo *top-down* para detectar k_ϵ-*Flocks*, baseado na

subdivisão iterativa de *flocks* maximais até que se retorne os *top-k* mais relevantes em janela de tempo w, com δ instantes de tempo. Observando o algoritmo *top-down* proposto, percebe-se que a visão de possuir inicialmente um agrupamento que englobe todos os MOs, realizando divisões sucessivas no agrupamento, não é a opção mais eficiente para muitos casos, pelo fato de que o resultado procurado é baseado em discos de diâmetro mínimo. Este trabalho de conclusão de curso propõe um novo algoritmo para o Padrão k_ϵ-*Flock*, baseado em agrupamento sucessivo de objetos móveis próximos no espaço até encontrar os k_ϵ-*Flocks*.

Para manter o parâmetro de distância livre a proposta constrói, gradativamente, um conjunto de grafos, um para cada instante de tempo, e identifica *flocks* candidatos à medida que é formado um subgrafo completo com μ vértices em todos os instantes de tempo. Os fundamentos do algoritmo são a ordenação parcial de arestas com base na distância entre pares de pontos e a construção incremental de subgrafos, inserindo iterativamente arestas de menor distância e identificando cliques por meio de varreduras locais nos subgrafos. O trabalho introduz os fundamentos da solução, discute sua corretude e apresenta o algoritmo proposto.

Este trabalho está organizado como segue. A Seção 2 apresenta as definições iniciais para entendimento da contribuição, juntamente com o contexto das discussões presentes na literatura dos padrões já abordados. A Seção 3 descreve a proposta do projeto. A Seção 4 descreve os resultados preliminares já atingidos juntamente com o pseudocódigo desenvolvido. A Seção 5 apresenta os resultados obtidos e as considerações finais desde trabalho.

2 FUNDAMENTAÇÃO TEÓRICA

2.1 Descoberta de Padrões em Trajetórias de Objetos Móveis

Uma trajetória, de acordo com SPACCAPIETRA, S. et al. [9], é definida como sendo segmentos de caminho de objetos móveis, que representam a evolução espaço-temporal dos MOs explicitados no espaço como pontos, desde sua partida até sua chegada. As trajetórias são coletadas, por exemplo, por dispositivos que possuem tecnologias embutidas para captura dos movimentos dos MO com precisão, como o GPS (*Global Positioning System*). Durante um período de tempo, os dispositivos coletam a posição do MO enquanto ele se movimenta, coletando de forma consecutiva várias posições e descrevendo sua trajetória. Com isso, é possível encontrar muitas aplicações utilizando as análises de dados provenientes do de trajetórias, desde análise de dados de viagens [4] até evolução de ciclones [10].

Dentro do estudo de trajetórias, existem áreas que estudam os padrões de movimento dos objetos móveis, principalmente quando estes se movimentam em conjunto, caracterizados como estando em comovimento. Para mineração dos padrões de comovimento duas técnicas de agrupamento se destacam na literatura: o agrupamento baseado em disco [6, 5] e o agrupamento baseado em densidade [11]. O agrupamento em disco é caracterizado por criar um disco que englobe os MOs, mas possui a restrição espacial do disco e a dificuldade de escolher o seu diâmetro. Já agrupamento baseado em densidade propõe uma solução para a limitação do disco, agrupando os pontos baseando-se em *clustering* por densidade, tipicamente o DBSCAN (*Density-Based Spatial Clustering of Applications with Noise*) [11]. Porém, o DBSCAN ainda requer a definição de um raio de conectividade entre os pontos para classificar a região de vizinhança, além de ter tipicamente um alto custo computacional.

2.2 Padrão *flock*

O foco deste trabalho é em padrões com agrupamento por disco. Nesta categoria, o padrão mais conhecido é o Padrão *Flock*, definido como um conjunto de pelo menos μ MOs se locomovendo em grupo por um número mínimo de δ instantes de tempo consecutivos, onde a distância que os caracteriza como em comovimento é delimitada por um disco de diâmetro ϵ [12, 5].

Como já foi mencionado, o disco que engloba os MOs observados na definição de *flocks* precisa de um valor fixo de diâmetro ϵ, que é difícil de ser definido de forma precisa. A área que o disco engloba implica uma limitação espacial na obteção dos pontos,

sabendo que os pontos precisam pertencer ao disco durante todos os instantes de tempo da janela de análise. Tal fator dificulta o encontro dos padrões importantes, pois a noção que caracteriza os pontos como "estando juntos" pode ser inespecífica caso não se conheça a fundo a distribuição de distâncias dos pontos das trajetórias. Por exemplo, para cada base de dados, seriam necessários "chutes" sucessivos de diâmetro, que muitas vezes retornarão mais agrupamentos que o esperado, ou até mesmo nenhum, até se chegar em um valor adequado. Outro exemplo compreende situações que a distribuição das distâncias varia muito, ou seja, quando o diâmetro do disco é muito alterado entre as medições dos MO, como por exemplo carros em uma estrada variando constantemente sua velocidade e distância, tornando-se difícil a escolha do diâmetro ideal para o disco.

2.3 O Padrão k_ϵ-*Flocks*

Com intuito de facilitar a obtenção de *flocks* e tornar o parâmetro de distância menos rígido, vários padrões que flexibilizam o parâmetro de distância já foram propostos na literatura. Entre eles, o padrão *flock* com Grau de Liberdade, explicitado em [6], como extensão do padrão *flock*, mas com foco em agrupamentos em que as trajetórias são menos comportadas. Esse padrão permite que as entidades possuam um grau de liberdade definido matematicamente e possam desconectar-se do grupo por breves instantes de tempo. De forma semelhante, o padrão Enxame [13] propõe que o grupo pode se dispersar no decorrer dos instantes de tempo, desde que convirjam novamente em instantes de tempo posteriores. Já Sanches et al. [1] propuseram o padrão k_ϵ-*Flock*, uma variação do padrão *Flock* que elimina a necessidade de se fornecer o parâmetro de distância. O padrão k_ϵ-*Flock* retorna os k *flocks* de diâmetro mínimo em cada janela temporal w de uma *stream* de dados de trajetórias, onde o diâmetro mínimo é dado pela Definição 1.

Definição 1 (Diâmetro Mínimo de um *Flock*) *O diâmetro mínimo de um flock $f_w(\mu, \epsilon)$, em uma janela w, é o menor valor $\epsilon' \in \mathbb{R}$ tal que $f_w(\mu, \epsilon') = f_w(\mu, \epsilon)$.*

O padrão k_ϵ-*Flock* não usa o diâmetro fixo, mas identifica *flocks* com o menor diâmetro possível que englobe a quantidade mínima de pontos μ para formar um *flock*. Assim, em vez da necessidade de inserir o parâmetro de distância não trivial, o algoritmo requer um valor k mais simples, sendo o número de *flocks* presentes no conjunto resposta ao fim da busca. Formalmente, k_ϵ-*Flocks* é dado pela Definição 2.

Definição 2 (k_ϵ-*Flocks*) *k_ϵ-Flocks em respeito a uma janela temporal w e um número mínimo de trajetórias $\mu > 1$ ($\mu \in \mathbb{N}$), representado como k_ϵ-Flocks(μ, w), é o conjunto $\mathcal{F}_w^k$ contendo k ($k \in \mathbb{N}$) flocks tal que para cada flock $f_w(\mu, \epsilon) \notin \mathcal{F}_w^k$ temos que $\epsilon \geq \epsilon_k$, onde ϵ_k é o menor diâmetro do flock mais extenso em $\mathcal{F}_w^k$. Caso não exista flocks suficientes na janela, menos de k flocks são reportados.*

2.4 Algoritmo Top-Down para detecção de K_ϵ-*Flocks*

Foi proposto por Sanches et al. [1] um algoritmo *top-down* para detecção de k_ϵ-*Flocks*. A ideia do algoritmo é identificar k_ϵ-*Flocks* a partir de subdivisões consecutivas do maior *flock*, até que se encontre o *flock* de tamanho mínimo em cada instante de tempo da janela.

Inicialmente, tem-se um único *flock* que engloba todas as entidades daquele instante de tempo, que é subdividido em pelo menos dois subflocks menores que o original, com um MO diferente entre eles. O *flock* poderá ser dividido em subflocks menores, dependendo da quantidade de pontos de borda presentes no disco e do μ escolhido para o teste. A Figura 1 ilustra em determinado instante de tempo, o split do *flock* mais extenso formado por um disco $\overline{f_w}$. O disco é subdividido em dois discos menores (f_w^1 e f_w^2), devido à presença de dois pontos de borda, formados pelas mesmas entidades do *flock* original, com exceção do ponto de borda utilizado na divisão. Na segunda etapa, à direita, o mesmo processo é realizado, com a diferença que o antigo disco f_w^1 agora é o *flock* mais extenso $\overline{f_w}$ em questão, contendo três pontos de borda e formando, por consequência, três novos *subflocks* menores. O algoritmo então realiza todos os passos de divisão dos agrupamentos buscando retornar o conjunto resposta com os *flocks* de tamanho mínimo.

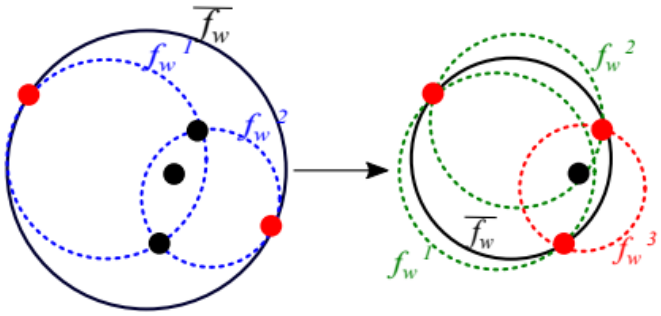


Figura 1 – Exemplo da divisão em subflocks de dois discos $\overline{f_w}$ em duas iterações diferentes do algoritmo.(retirado de [1])

Porém, mesmo funcional, o algoritmo pode ser bastante ineficiente, dependendo das características do conjunto de dados. Para instantes de tempo com poucas entidades e o μ desejado estando próximo a quantidade total de entidades, o *top-down* pode ser considerado eficiente. Porém, quanto menor o parâmetro μ e maior o número de entidades de cada instante de tempo, mais prejudicado fica o seu desempenho. Como casos em que o μ próximo ao valor total de entidades do instante de tempo são pouco comuns, uma nova abordagem do problema foi proposta neste trabalho.

3 PROPOSTA DE UM ALGORITMO POR AGRUPAMENTO DE PONTOS PARA DETECÇÃO DE K_ϵ-*FLOCKS*

O objetivo deste trabalho é desenvolver um algoritmo para descoberta de k_ϵ-*Flocks* em uma *stream* de dados de trajetórias baseado em agrupamento sucessivo de pontos. A ideia fundamental do algoritmo é baseada na observação de situações típicas, onde o número mínimo de MOs fornecido para indicar um *flock* (μ) é pequeno, se comparado ao total de MOs presentes na janela de tempo em análise. Esta seção define o problema, apresenta a proposta de solução e os métodos e técnicas adotados.

3.1 Definição do problema

O algoritmo *top-down* demanda um número grande de iterações de subdivisão de *flocks* candidatos até chegar a *subflocks* com o número desejado de entidades. A abordagem proposta, ao contrário, parte de MOs isolados e vai formando e aumentando grupos de MOs próximos no espaço até que se chegue em candidatos com o número mínimo de entidades para configurar um *flock*. Sendo μ pequeno em relação ao número de MOs na janela, o número de iterações desta abordagem tende a ser muito menor que da abordagem *top-down*, possibilitando reduzir consideravelmente o tempo de execução do algoritmo.

Para ilustrar a ideia, a Figura 2 mostra um conjunto de pontos em um instante de tempo e o candidato a *flock* de menor diâmetro, considerando-se $\mu = 3$. Neste caso, o método *top-down* teria, inicialmente, o candidato a *flock* mais extenso englobando todas as 7 entidades, com diâmetro igual a duas vezes a distância entre o centro do círculo e o ponto B, a ser subdividido sucessivamente até chegar no candidato $\{a, b, c\}$. Contudo, o número total de iterações é muito maior, pois cada subdivisão produz novos subgrupos, denotados pelos círculos pontilhados verde, vermelho e amarelo, que precisam ser explorados independentemente. Já a abordagem por aglutinação sucessiva de entidades próximas entre si, inicialmente, agruparia as entidades $\{b, c\}$ e, em um segundo passo, $\{a, b, c\}$, detectando o *flock* candidato. Embora este exemplo ilustre um caso ideal da proposta do trabalho, pode-se perceber que casos semelhantes são frequentes. A detecção de *flocks* se aplica, afinal, a situações em que se quer detectar o comovimento de um número reduzido de entidades dentre o total existente no conjunto de trajetórias.

3.2 Proposta de solução

Este trabalho propõe uma variação do método presente na literatura com abordagem decremental de diâmetro e subdivisão de agrupamentos por um método incremental

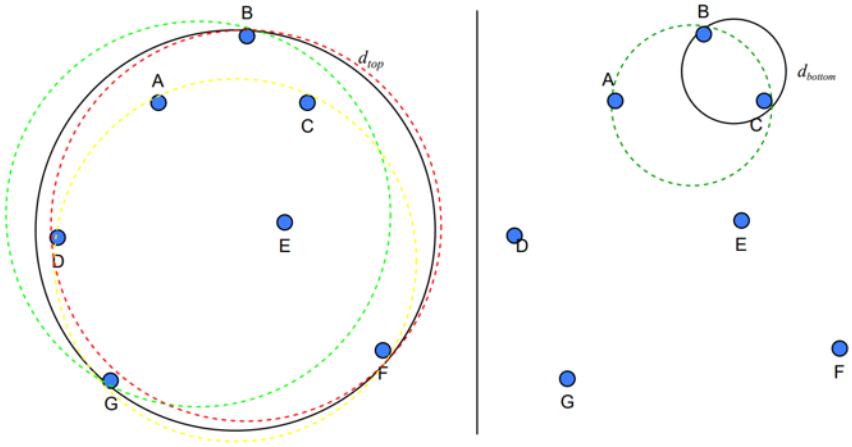


Figura 2 – Exemplo de uma mesma base de dados bidimensional Euclidiana com 7 pontos, em um instante de tempo qualquer, sendo analisada pela abordagem *top-down* (à esquerda) e por uma abordagem *bottom-up* (à direita), ambas buscando encontrar o *flock* {a, b, c}.

gradativo do diâmetro à medida que são aglutinados pontos próximos até que se encontre um agrupamento com μ entidades presente simultaneamente em todos os instantes de tempo da janela de análise.

3.3 Procedimentos metodológicos/Métodos e técnicas

Este trabalho é a sequência de um projeto de Iniciação Científica [14] com a mesma proposta voltada para a área de agrupamento de objetos móveis. A proposta consiste na detecção de k_ϵ-*Flocks* em uma janela temporal, baseada em sucessivas aglutinações de trajetórias para formação de *flocks* em um dado número de instantes de tempo consecutivos.

Primeiramente, foi realizada uma nova busca por referências bibliográficas que possam contribuir no estudo do comovimento de MO e também em áreas mais específicas abordadas nesse trabalho de coclusão de curso, como algoritmos desenvolvidos e documentados que melhorem o desempenho das etapas cruciais de desenvolvimento. Dessa forma, foram gerados k_ϵ-*Flocks* através do algoritmo *top-down* para estudo do comportamento e análise de resultados obtidos, juntamente com o aprofundamento nos estudos da área de encontro de subgrafos.

Para desenvolver o método incremental de encontro de subgrafos, foi desenvolvido um psudocódigo inicial, já contendo os métodos de agrupamentos que compõe a proposta sujerida, como o método do Clique por exemplo. Com o pseudocódigo formalizado, foi realizada a implementação do algoritmo proposto de incremento de diâmetro, resolvendo o problema utilizando a teoria de Cliques de forma menos custosa que o método trivial, de custo computacional NP-Completo.

Na etapa de testes, foi feita uma sequência de compilações para obter os resultados do método *top-down* em diversas bases de dados conhecidas. Em seguida, os mesmos testes foram realizados para a proposta *bottom-up*, para comparação de desempenho. A análise dos resultados buscou identificar a diferença de custo de cada método.

4 ALGORITMO BOTTOM-UP PARA DETECÇÃO DE K_ϵ-*FLOCKS*

Nesta seção é mostrada a descrição de todas as etapas de execução já definidas, explicando de forma detalhada as decisões tomadas mediante as dificuldades encontradas com o desenvolvimento dos estudos e formulação do algoritmo proposto.

O algoritmo proposto depende de duas coisas. A primeira é uma forma de agrupar gradativamente elementos mais próximos entre si. Isso pode ser resolvido por meio de um acesso a pares de elementos ordenados pela distância entre os elementos. A alternativa adotada foi calcular as distâncias entre todos pares de elementos em cada instante de tempo e estabelecer uma ordenação global compreendendo todos os instantes de tempo da janela. Mas, há outras possibilidades que permitem reduzir o custo desta tarefa. A segunda coisa é definir uma condição de parada que garanta a corretude do algoritmo. As seções a seguir discutem duas possibilidades, uma que não atende à necessidade do problema e uma segunda, que foi a escolhida neste trabalho.

4.1 Definição da condição de parada

A abordagem proposta tem semelhança com a ideia do algoritmo de agrupamento por densidade HDBSCAN (*Hierarquical Density-Based Spatial Clustering of Applications with Noise*) [5]. Assim, uma das possibilidades de condição de parada para a abordagem proposta seria utilizar a estratégia adotada pelo HDBSCAN demonstrada na Figura 3, baseada na formação de árvores espalhadas mínimas (*Minimal Spanning Tree* – MST) e na estabilidade de *clusters*.

Entretanto, essa abordagem não se aplica ao problema, pois pode acontecer do valor de ε ser muito incrementado, aglutinando, de um nível para outro do dendograma, agrupamentos com muito mais que μ elementos. Portanto, seria necessário fazer um pós-processamento, utilizando a estratégia *top-down* para subdividir os *flocks* até chegar à resposta correta.

Contornando os problemas da alternativa anterior, a proposta deste trabalho é buscar k agrupamentos a partir da identificação de cliques de tamanho μ em cada instante de tempo da janela ω. Um clique [15] é um subgrafo completo, formado por vértices conectados por arestas não direcionadas, onde cada vértice é conectado com todos os seus adjacentes.

O problema do encontro de um subgrafo completo é conhecido na literatura como o Problema do Clique e tem algumas variações. As variações incluem encontrar o clique com maior número de vértices em dado instante de tempo, o clique de maior valor em

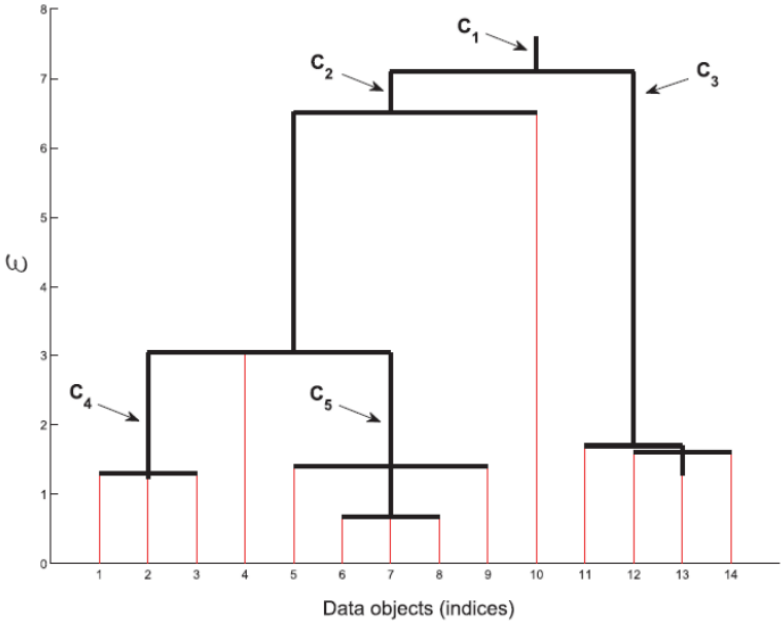


Figura 3 – Dendograma correspondente à hieraquia do HDBSCAN com $\mu = 3$. (Retirado de [2])

um grafo com pesos ou, dado um valor μ, encontrar os subgrafos completos (um dos 21 problemas NP-Completos de Kart [16]) contendo μ vértices, como é utilizado neste trabalho. Um clique é explicitado pela Definição 3 [17], como segue.

Definição 3 (Clique) *Dado um grafo G não direcionado, C é clique, tal que C seja um subconjunto de G e $G(C)$ seja completo, ou seja, todo vértice de C é adjacente a todos os outros vértices de C.*

Para inserir cliques na detecção de k_ϵ-*Flocks*, este trabalho propõe um algoritmo que encontra os cliques em subgrafos construídos gradativamente para cada instante de tempo, inserindo-se consecutivamente as arestas de menor peso global, ou seja, de menor distância entre os vértices. Isto é, cada instante de tempo é formado inicialmente por um grafo vazio. A cada iteração, identifica-se a aresta de menor peso atual, que corresponde ao par de pontos mais próximos entre si, em qualquer instante de tempo, que ainda não foi conectado por uma aresta, e insere-se estes vértices e esta aresta no grafo do instante de tempo respectivo. Como as arestas são adicionadas de forma crescente, sempre selecionando a menor aresta atual, é garantido que o primeiro agrupamento definido por um clique de tamanho μ corresponde ao *flock* de diâmetro mínimo com μ entidades na janela, o segundo clique sendo o segundo *flock* e assim sucessivamente. Esta estratégia permite definir uma condição de parada correta para o problema, interrompendo o algoritmo quando os mesmos k cliques forem encontrados em todos os grafos dos instantes de tempo da janela ω.

O principal desafio dessa abordagem é que, em sua versão clássica, o Problema do Clique é NP-Completo. Para contornar esse problema, a proposta consiste no encontro de cliques de forma incremental, à medida que novas arestas vão sendo adicionadas nos grafos, amortizando consideravelmente o custo da detecção de cliques. A próxima seção detalha o algoritmo proposto.

4.2 Descrição do algoritmo

Algoritmo 1: Algoritmo Bottom-Up para encontro de k_ϵ-*Flocks* agrupamentos de menor diâmetro mínimo

Entrada: ω: janela temporal contendo o conjunto de trajetórias $\mathcal{T}$ onde t_i refere-se ao i-ésimo instante de tempo de w
μ: número mínimo de entidades para formação de um *flock*
k: quantidade de ϵ-flocks de retorno
Saída: F_w^k: Conjunto de k *flocks* de tamanho mínimo

```
1  início
2      Q ← buildPQueue(ω)
3      F_w^k ← ∅
4      para cada t_i ∈ ω faça
5          G[t_i] ← empty_graph
6          C[t_i] ← ∅
7      fim
8      enquanto |F_k| < k e Q ≠ ∅ faça
9          /* Escolhe a menor aresta entre todos os instantes de tempo da janela */
10         (u, v, t_i) ← Q.removeMin()
11         /* Insere os vértices e a aresta no grafo local do respectivo instante de tempo */
12         G[t_i].addEdge(u, v)
13         /* Verifica se a aresta gera novos cliques de tamanho μ em G[t_i] */
14         C_new ← findNewCliques(G[t_i], (u, v), μ)
15         para cada c ∈ C_new faça
16             /* Verifica se o novo clique já aparece nos demais instantes */
17             candidate ← true
18             para cada t_j ∈ {ω − t_i} faça
19                 se c ∉ C[t_j] então candidate ← false ;
20             fim
21             /* Se aparece em todos os instantes, é um candidato a k_ε-Flock */
22             se candidate então
23                 /* Computa discos e o diâmetro do flock candidato */
24                 D ← ∅
25                 ε ← 0
26                 para cada t_j ∈ ω faça
27                     D[t_j] ← computeMinDisk(c_j)
28                     se ε > diameter(D[t_j]) então ε ← diameter(D[t_j]) ;
29                 fim
30                 f_ω ← buildFlock(c, disk)
31                 F_w^k ← F_w^k ∪ f_ω
32             fim
33         fim
34     fim
35     /* Se o conjunto resposta ficou com mais de k respostas, remove os k'-ésimo flocks tais que k' > k
         */
36     se |F_w^k| > k então
37         F_w^k.removeLargest(k)
38     fim
39 fim
```

O Algoritmo 1 apresenta o pseudocódigo da solução proposta. A partir de uma janela ω de um conjunto de trajetórias contendo as posições de MOs, é construída uma fila de prioridade Q (linha 2), com todos os pares de pontos de cada instante de tempo,

organizada de forma crescente pela distância entre eles. Sem otimizações que reduzam o número de cálculos de distância entre pontos, esta fila contém todos os pares que simulam as arestas de um grafo completo para cada instante de tempo. Também são inicializados o conjunto resposta F_w^k e os conjuntos G e C (linhas 3, 5 e 6), para armazenar, respectivamente, os subgrafos e os conjuntos de cliques para os instantes de tempo da janela.

Em seguida, a partir de iterações consecutivas, é obtido o par de elementos mais próximos entre si, dentre os pares em Q. Os vértices incidentes nesta aresta são adicionados ao grafo $G[t_i]$, caso ainda não façam parte do grafo, bem como a aresta, onde t_i é o instante de tempo correspondente ao par. A Figura 4 mostra o estado dos subgrafos após uma sequência de quatro iterações.

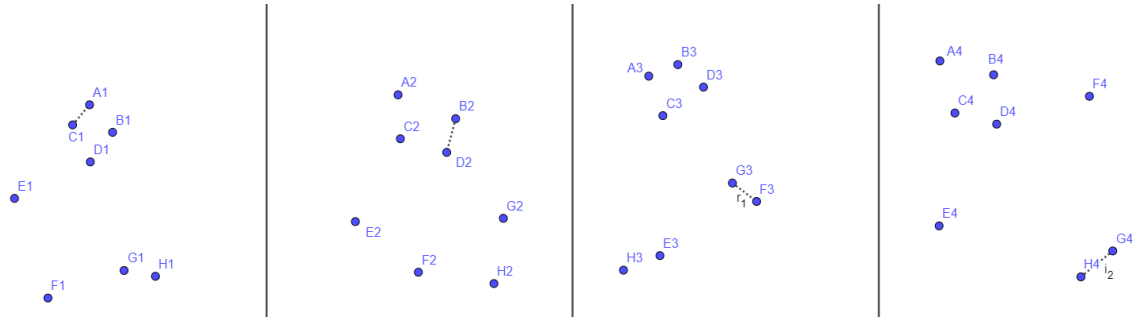


Figura 4 – Representação da escolha das menores aresta de uma base de dados de 4 instantes de tempo com 7 entidades cada. Menor aresta de cada instante de tempo representada por uma linha pontilhada.

Para cada aresta adicionada, é feita uma verificação se um ou mais novos cliques foram formados com a adição da aresta no grafo $G[t_i]$ (linha 14). O retorno desses cliques é feito por um algoritmo adaptado, baseado na teoria de encontro de cliques maximais de Bron-Kerbosh [18], pois esse não possuia uma verificação da quantidade μ de objetos presentes no retorno do flock maximal. Os cliques encontrados em determinado instante de tempo são salvos na lista de cliques local $C[t_i]$, para que não haja recálculos desnecessários em iterações posteriores. Caso um ou mais novos cliques forem formados com a adição da nova aresta, verifica-se se o(s) mesmo(s) clique(s) já foi/foram encontrado(s) nos demais instantes de tempo, por meio de uma varredura dos vértices presentes no(s) novo(s) clique(s) reportado(s) com os demais já retornados (linha 15). Caso encontre em todos os instantes de tempo da janela grafos completos com esses vértices em comum, um candidato a k_ϵ-*Flock* foi detectado (linha 22), como ilustra a Figura 5. Em seguida, para cada instante de tempo, o algoritmo computa os discos envolventes de tamanho mínimo que cobrem os pontos no clique. O diâmetro ϵ do k_ϵ-*Flock* candidato é o maior diâmetro dentre esses discos. Essa operação identifica o *Smallest Enclosing Circle*[1] (Circulo de Englobamento Mínimo) que engloba todos os pontos do clique para cada instante de tempo da janela.

[1] <https://www.nayuki.io/page/smallest-enclosing-circle>

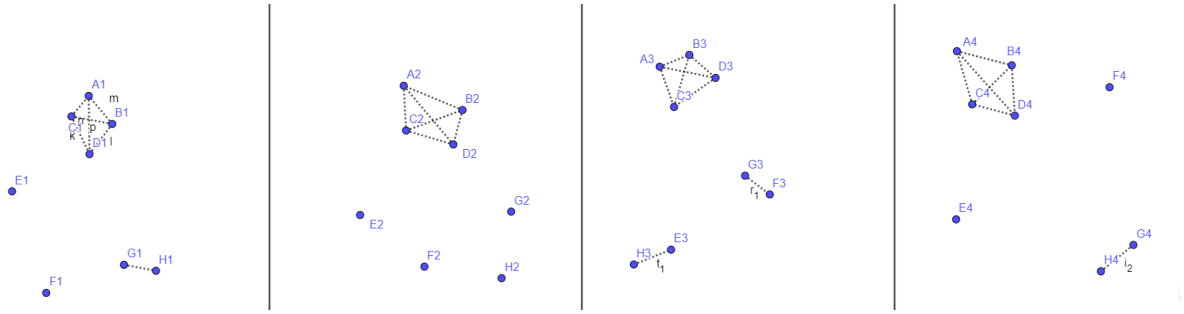


Figura 5 – Exemplo de clique encontrado nos elementos {A, B, C, D} com $\mu = 4$, $k = 1$ e $\delta = 4$.

Os pontos que estiverem na interseção dos pontos nos discos de todos os instantes de tempo definem as entidades que formam o *flock* f_ω, que é um dos k_ϵ-*Flocks*. Todo o ciclo é repetido até que se tenha na lista de retorno k *flocks* ou todas as arestas em Q tenham sido processadas, retornando os k_ϵ-*Flocks* na janela. Caso a quantidade F_w^k de flocks retornados seja superior a k flocks, a função $removeLargest(k)$ é utilizada para o descarte dos flocks mais extensos do conjunto F_w^k.

Porém, podem existir outros pontos que não foram processados no clique, mas que também estão sendo cobertos pelo disco definido pelo diâmetro do maior *flock* candidato da janela em todos os instantes de tempo. Para isso, denota-se o Algoritmo 2 de pós processamento para encontro de cliques maximais.

4.3 Algoritmo BOTTOM-UP para encontro de *Cliques* Maximais

O algoritmo inicialmente observa os cliques que foram reportados ao fim do processamento e seleciona aqueles que possuem o mesmo diâmetro porém com MOs diferentes. Nessa etapa, o algoritmo cria um novo clique auxiliar a partir da junção das entidades presentes em cada grupo de cliques de mesmo diâmetro selecionados no passo anterior. É construído portanto uma lista de *flocks* contendo μ entidades cada e formados pelo resultado do processo de combinação denotado pela fórmula matemática $C_k^n = \frac{n!}{k!(n-k)!}$, feito acerca dos elementos que formam o cliques auxiliar.

Verifica-se, portanto, se todos os elementos dessa lista já foram reportados em algum momento anterior na execução do algoritmo, e em caso verdadeiro o clique auxiliar é reportado como uma resposta e os flocks de mesmo diâmetro que foram agrupados são desconsiderados na resposta final.

Vale ressaltar que através dos testes realizados na base de dados *Caribous* que será descrita em detalhes mais a frente, o algoritmo referente ao encontro dos cliques maximais não apresentou uma divergência elevada em tempo de execução e custo computacional

Algoritmo 2: Algoritmo *Bottom-up* para encontro de *Cliques* Maximais

Entrada: S: *stream*/conjunto de dados contendo o conjunto de trajetórias $\mathcal{T}$
μ: número mínimo de entidades para formação de um *flock*
δ: número de instantes de tempo consecutivos para formar um *flock*
k: quantidade de flocks de retorno
Saída: F_w^k: Conjunto de k *flocks* para cada $\omega \in S$

```
1  início
2      /* Cria lista de cliques auxiliares*/
3      F'_k ← agrupaDiametrosIguais(F_w^k)
4      para cada f ∈ F'_k faça
5          C ← combinacao(f)
6          se /*Todos elementos em C já foram reportados*/ então
7              removeElementos(F_w^k)
8              F_w^k ← insere(C)
9          fim
10     fim
11 fim
```

para realizar esse pós-processamento quando comparado com a execução sem ele (gráficos referentes aos testes são demonstrados na seção 5).

4.4 Algoritmo BOTTOM-UP em Janela Deslizante

O algoritmo *Bottom-Up* descrito na Seção 4.2 juntamente com o algoritmo na Seção 4.3 retorna os F_w^k flocks em uma única janela temporal ω fornecida. Porém, com o objetivo de realizar a consulta completa dos k_ϵ-*Flocks* em uma stream de dados, é apresentado o Algoritmo *Bottom-Up* para análise e processamento dos demais instantes de tempo do conjunto de dados fornecido caso esse possua mais que ω. Assim, o Algoritmo 3, utiliza além do parâmetro ω o comprimento δ das janelas a serem processadas bufferizando os dados referentes ao intervalo t_0 ao $t_{\delta-1}$ que compõe os instantes de tempo presentes na janela.

Algoritmo 3: Algoritmo *Bottom-up* em Janela Deslizante

Entrada: D: *stream*/conjunto de dados contendo o conjunto de trajetórias $\mathcal{T}$
μ: número mínimo de entidades para formação de um *flock*
δ: número de instantes de tempo consecutivos para formar um *flock*
k: quantidade de flocks de retorno
Saída: F^k: Conjunto de k *flocks* para cada $\omega \in D$

```
1  início
2      F_k ← ∅
3      /* Recebe e Desliza a Janela*/
4      enquanto ω = bufferWindow(D, δ) faça
5          F_k ← F_k∪ Bottom-Up (ω, μ, δ)
6      fim
7      return F_k
8  fim
```

Após o processamento completo dessa janela, o deslizamento é feito e os dados referentes ao primeiro instante de tempo são descartados, assim como sua lista de arestas incremental, seu grafo local e os cliques locais reportados nesse instante de tempo pela janela anterior. De fato, os dados processados nos instantes de tempo mantidos na mesma

janela são guardados, contendo cada instante de tempo a lista de arestas restantes para inserção, seu grafo local preenchido com arestas e vértices e a lista de cliques locais. Com o deslocamento da janela t_1 ao $t_{\omega+1}$ as distâncias de todos os objetos do novo instante de tempo são calculadas e o processamento incremental de arestas é retomado.

5 EXPERIMENTOS E RESULTADOS

Neste capítulo são demonstrados os experimento que foram realizados para avaliar as contribuições deste trabalho. A Seção 5.1 mostra os conjuntos de dados utilizados no processamento e as configurações escolhidas no experimento. A Seção 5.2 mostra os resultados obtidos baseados nos experimentos realizados com o algoritmo *Top-Down*. A Seção 5.3 mostra os benefícios observados na utilização do novo método proposto.conhecida

5.1 Conjuntos de Dados e Configurações

A partir dos conceitos apresentados e da implementação do algoritmo, foi escolhida para testes uma base de dados amplamente utilizada na área conhecida como *Caribous*, do projeto *Porcupine Caribou Herd Satellite Collar PRoject*. Esse conjunto é caracterizado por apresentar a movimentação de 44 renas em processo de migração no noroeste do Canadá. A partir da análise desses movimentos e de suas trajetórias no espaço o conjunto de dados foi utilizado neste projeto variando os parâmetros de execução com o intúito de abranger uma maior quantidade de casos dentro desse conjunto de dados.

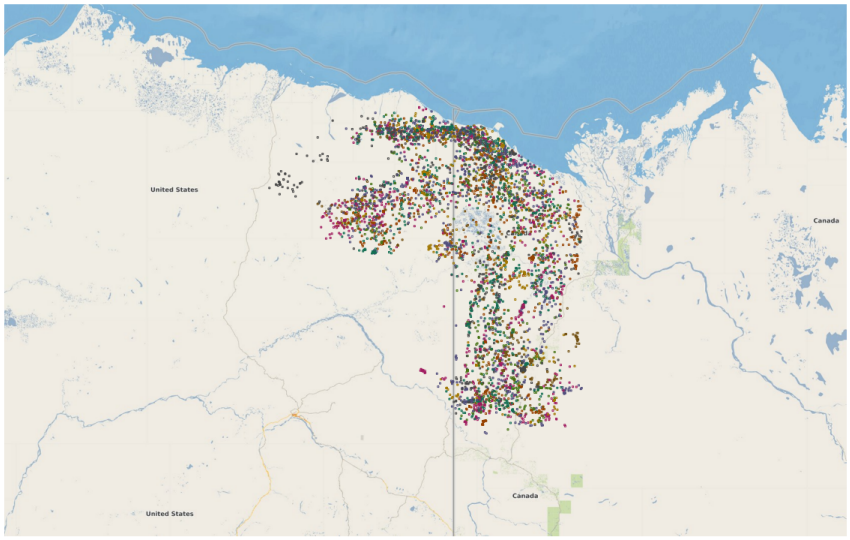


Figura 6 – Representação das renas em migração do conjunto de dados *Caribous* no noroeste do Canadá (Retirado de [1])

A Tabela 1 resume todos os dados disponíveis referentes ao conjunto de dados.

5.2 Configuração dos experimentos

Os algoritmos propostos foram desenvolvidos em C++ e compilados com o GCC v7.5. Para realização dos experimentos foi utilizada a variação de parâmetros descritos

Tabela 1 – Conjunto de dados utilizado nos testes do algoritmo.

Conjunto de dados	**Nº de pontos**	**Nº de trajetórias**	**Nº de timestamps**	**Taxa de amostragem**	**Granularidade**
Caribous	15796	44	359	-	44

Fonte: Retirado de [1].

na Tabela 2. Os experimentos visam analisar o tempo de execução comparando com o algoritmo precursor em [1] e também os resultados obtidos.

Tabela 2 – Configuração dos experimentos

Parâmetros	**Valores avaliados**	**Valor padrão**
μ	3, 5	3, 5*
k	1, 3, . . . , 13	5
δ	3, 5, . . . , 15	10

* Dois conjuntos de execuções realizadas, a primeira utilizando valor de μ fixo em 3 e a segunda fixo em 5.

Os experimentos foram realizados em um computador do Departamento de Computação (DC/UEL) disponibilizado pelo orientador com um processador Intel(R) Core(TM) i7-8700 CPU @ 3.20GHz, com 32GB de memória RAM, no sistema Ubuntu 20.04.2 LTS Focal x64.

5.3 Análise dos resultados obtidos

Como foi observado conforme o trabalho do *padrão* k_ϵ*-Flocks*, definir um diâmetro para o *flock* não é uma tarefa trivial. Mesmo para o cientista de dados mais experiente da área, muitos são os fatores que acabam causando divergências na base de dados, como por exemplo o tipo do indivíduo ou até variações de comportamento, o que dificulta cada vez mais a escolha de um diâmetro ideal.

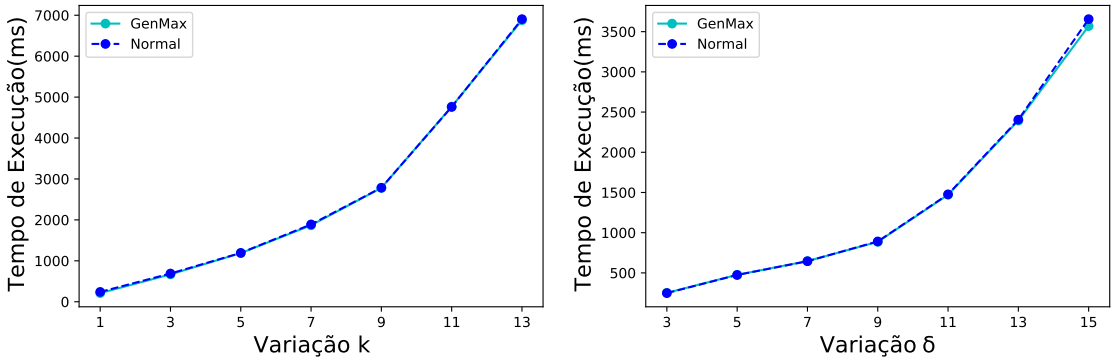
Com isso, foram realizados experimentos visando mostrar as diferenças dos resultados para cada algoritmo estudado e os respectivos parâmetros utilizados nas execuções. Devido a isso, foi escolhido para este trabalho uma abordagem na qual os testes realizados variam em número de entidades no *flock* μ, comprimento δ da janela de análise e a quantidade k de respostas por janela. Isso traz uma variedade de abordagens para o problema onde o usuário pode analisar o comportamento do conjunto de dados mediante a diversas variações nos parâmetros de entrada. Como já foi mencionado, o código *Bottom-Up* é focado em situações mais típicas onde a quantidade μ de objetos por instante de tempo tende a ser pequena, portanto esse será o foco das análises dessa seção.

5.4 Comparação dos algoritmos *Top-Down* e *Bottom-Up*

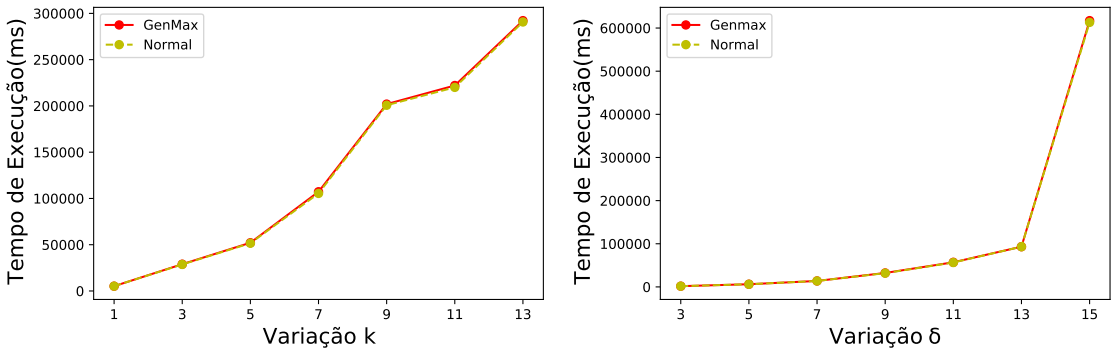
Dentre os testes executados nos algoritmos *Top-Down* e *Bottom-Up* foi observado um avanço interessante acerca dos resultados de ambas as execuções medindo o tempo de execução com cada alteração de parâmetro.

5.4.1 Quantidade de respostas e Flock Maximal

Quando o foco é o número de respostas, foi observado que o *Bottom-Up* analisou algumas respostas abordadas pelo *Top-Down* de forma diferente. Devido à natureza de segmentar os flocks pelos pontos de borda do *Top-Down*, ocorrem casos onde o circulo que engloba os flocks encontrados é segmentado, porém o diâmetro do flock permanece o mesmo. Para esses casos, levando em consideração a vertente do flock maximal, é necessário que o objeto permaceça no flock, pois sua retirada não reduziu o tamanho do diâmetro. Vale ressaltar que essa análise é feita para casos onde esse flock já possui μ entidades após essa segmentação.



(a)



(b)

Figura 7 – Tempo de execução em relação à variação dos parâmetros de entrada dos testes realizados com e sem a utilização do algoritmo para encontro de cliques maximais. Em (a) com $\mu = 3$ e em (b), com $\mu = 5$. $\delta = 10$ e k variável nos gráficos à esquerda, e $k = 5$ e δ variável nos gráficos a direita.(Fonte: próprio autor)

Dessa forma, foi necessário a utilização do pós-processamento de flocks maximais,

ajustando as respostas para que possuam todos os objeto necessários. Na Figura 7 podemos observar a comparação dos tempos de execução em relação a variação dos parâmetros de entrada dos testes realizados com e sem a utilização do algoritmo. Fixando dois parâmetros, é mostrando na figura a diferença praticamente mínima entre as duas execuções.

5.4.2 Tempo de execução

Os tempos de execução do algoritmo *Bottom-Up* para os parâmetros informados foram consideravelmente menores que os do *Top-Down* como é possível observar de forma clara nas Figuras 8 à 11. Nos gráficos em azul representam as execuções do *Top-Down* e em verde do *Bottom-Up*. É denotado no eixo vertical o tempo de execução em escala logaritmica de ambas das execuções, e no eixo horizontal a variação dos parâmetros de entrada k e δ para um dado μ fixo.

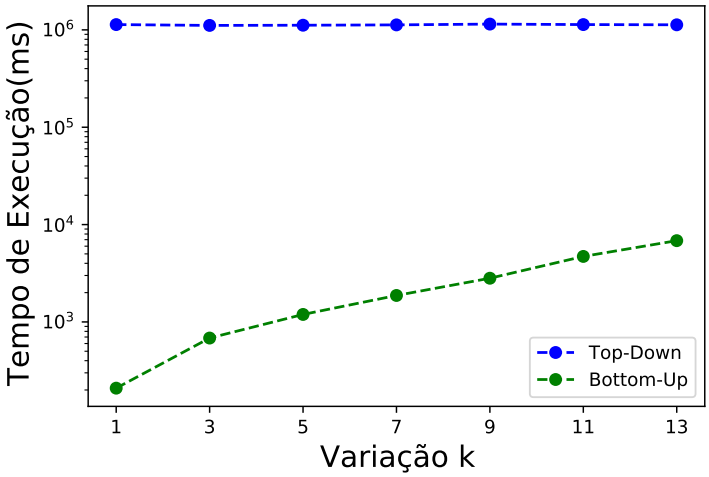


Figura 8 – Comparação de tempo de execução entre os algoritmos *Top-Down* e *Bottom-Up* com $\mu = 3$ e $\delta = 10$.(Fonte: próprio autor)

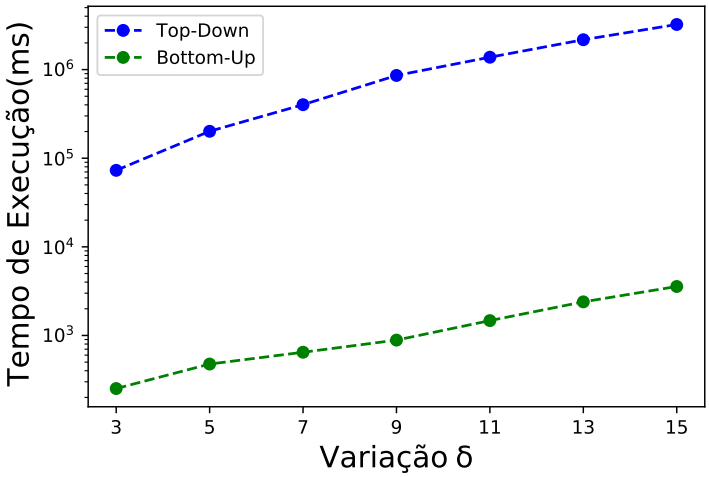


Figura 9 – Comparação de tempo de execução entre os algoritmos *Top-Down* e *Bottom-Up* com $\mu = 3$ e $k = 5$.(Fonte: próprio autor)

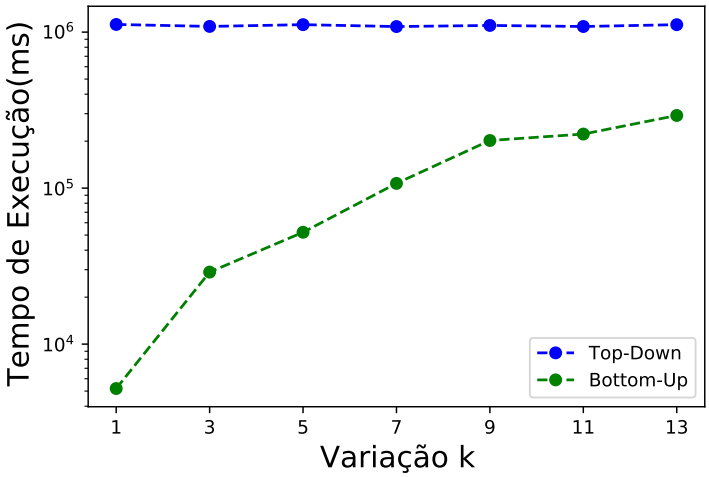


Figura 10 – Comparação de tempo de execução entre os algoritmos *Top-Down* e *Bottom-Up* com $\mu = 3$ e $\delta = 10$.(Fonte: próprio autor)

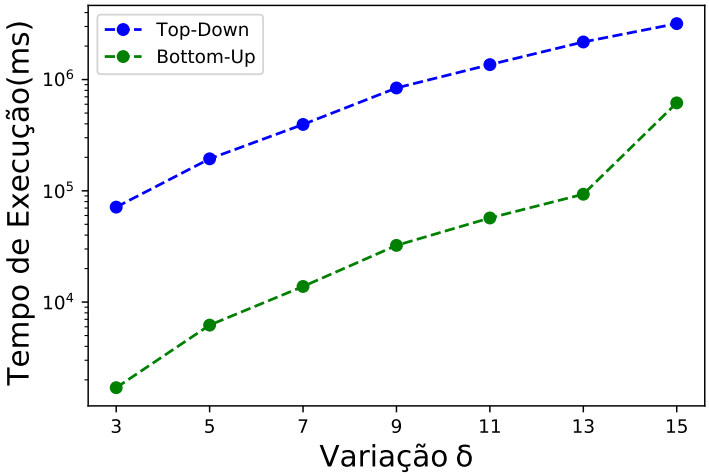


Figura 11 – Comparação de tempo de execução entre os algoritmos *Top-Down* e *Bottom-Up* com $\mu = 3$ e $k = 5$.(Fonte: próprio autor)

6 CONCLUSÃO

O grande avanço no número de tecnologias baseadas nas informações obtidas por sistemas e serviços de localização espaço-temporal, proporcionou uma crescente demanda de estudos relevantes acerca dos dados obtidos por esses sistemas (*GPS*). Dentre as áreas mais relevantes, destaca-se a área de objetos móveis em comovimento, ou seja, se movimentando em conjunto por dado período de tempo. Com isso tem-se o Padrão *Flock* conhecido na literatura por identificar grupos de comovimento dado um disco de diâmetro fixo que englobe esses grupos ao longo de todos os instantes de tempo de análise.

Porém, esse disco requer um nível de precisão elevado quando trata-se de escolher seu diâmetro, pois cada base de dados possui suas nuances. Um mesmo raio pode ser ótimo para alguns casos e irrelevante para outros, trazendo a tona diversos estudos para contornar esse problema. Obviamente para casos onde existe um conhecimento prévio das informações, o procedimento tem potencial para ser muito mais vantajoso no retorno das respostas, mas ainda não é algo trivial para se ter domínio. Além dos padrões de análise de cada tecnologia e como os dados que ela disponibiliza devem ser computados para realizar esse tipo de agrupamento, dependendo de muitos fatores como o tipo do objeto, seu padrão de movimento, possíveis ruídos, entre outros.

Devido as dificuldades apresentadas, o algoritmo *Top-Down* para encontro de diâmetros mínimos foi implementado, com a ideia de manter o diâmetro como um parâmetro livre, e agora solicitar ao usuário um valor k, que representa a quantidade de respostas que o algoritmo deve apresentar. Essa proposta tem como principal foco a descoberta de k_ϵ-*Flocks* de forma exploratória, onde os *flocks* são subdivididos até que possúam a quantidade de objetos desejada. Mas essa proposta mesmo funcional, foi entendida como pouco otimizada para casos onde o número de entidades na base de dados é grande, mas a quantidade desejada ao fim do processamento é pequena.

Com isso, este trabalho traz um novo algoritmo para detecção de k_ϵ-*Flocks* em uma janela temporal, baseado no agrupamento sucessivo de pontos próximos, encontrando agrupamentos correspondentes aos k_ϵ-*Flocks* ao serem mapeados no espaço como grafos e detectando cliques incrementalmente. Foi observado que o tempo de execução do algoritmo *Bottom-Up* foi inferior ao de seu precursor *Top-Down* para os teste realizados na base de dados escolhida, e com a variação de parâmetros adotada. Como já foi mencionado na Secção 3.1, os casos mais interessantes são baseados em valores menores para os parâmetros de entrada referentes ao número de entidades.

Portanto, seguem as contribuições desde trabalho:

- apresenta os conceitos relacionados à descoberta de padrões de comovimento em

objetos móveis com ênfaze na descoberta de **k-padrões** de comovimento, onde k representa a quantidade de respostas desejadas e o parâmetro de distância é livre;

- apresenta o conceito de k_ϵ-*Flocks* que utilizam os parâmetros do Padrão *flock*, porém buscando agrupamentos de diâmetro mínimo em uma dada janela temporal ω;
- apresenta o novo **Algoritmo *Bottom-Up*** para análise e detecção de k_ϵ-*Flocks* em uma janela temporal ω e também o **Algoritmo *Bottom-Up* em Janela Deslizante** para uma stream de dados de trajetórias;
- apresenta o **Algoritmo *Bottom-up* para encontro de *Cliques* Maximais** como forma de pós-processamento, aglutinando possíveis respostas que represetem o mesmo *flock*.

A partir das contribuições mencionadas, é possível construir um algoritmo que retorna uma quantidade pré-determinada de agrupamentos de objetos móveis a partir de um conjunto de dados de trajetórias desconhecido, sem que seja necessário definir um parâmetro de distância. É possível também, avaliar as respostas e distinguir quais na realidade pertencem ao mesmo agrupamento maximal, reavaliando-o e retornando esse novo elemento maximal de mesmo diâmetro.

Para trabalho futuros, é valido avaliar novas formas de otimizar a busca de cliques nos grafos locais de cada instante de tempo, pois o Problema do Clique, muito conhecido na literatura por ser um problema NP-Completo, é muito custoso e determinante na velocidade do processamanto dos dados dos objetos móveis. Assim como, a busca por novos métodos de análise dos cliques maximais para o pós-processamento das respostas retornadas ao fim algoritmo.

REFERÊNCIAS

[1] SANCHES, D. E. et al. A top-down algorithm with free distance parameter for mining top-k flock patterns. In: MANSOURIAN, A. et al. (Ed.). *Geospatial Technologies for All - Selected Papers of the 21st AGILE Conference on Geographic Information Science, Lund, Sweden, 12-15 June 2018*. Springer, 2018. (Lecture Notes in Geoinformation and Cartography), p. 233–249. Disponível em: <https://doi.org/10.1007/978-3-319-78208-9_12>.

[2] CAMPELLO, R. J. G. B. et al. Hierarchical density estimates for data clustering, visualization, and outlier detection. *ACM Trans. Knowl. Discov. Data*, v. 10, n. 1, p. 5:1–5:51, 2015. Disponível em: <https://doi.org/10.1145/2733381>.

[3] WANG, Y.; LIM, E.; HWANG, S. Efficient mining of group patterns from user movement data. *Data Knowl. Eng.*, v. 57, n. 3, p. 240–282, 2006. Disponível em: <https://doi.org/10.1016/j.datak.2005.04.006>.

[4] MARKOVIC, N. et al. Applications of trajectory data from the perspective of a road transportation agency: Literature review and maryland case study. *IEEE Trans. Intell. Transp. Syst.*, v. 20, n. 5, p. 1858–1869, 2019. Disponível em: <https://doi.org/10.1109/TITS.2018.2843298>.

[5] VIEIRA, M. R.; BAKALOV, P.; TSOTRAS, V. J. On-line discovery of flock patterns in spatio-temporal data. In: AGRAWAL, D. et al. (Ed.). *17th ACM SIGSPATIAL International Symposium on Advances in Geographic Information Systems, ACM-GIS 2009, November 4-6, 2009, Seattle, Washington, USA, Proceedings*. ACM, 2009. p. 286–295. Disponível em: <https://doi.org/10.1145/1653771.1653812>.

[6] CAO, Y.; ZHU, J.; GAO, F. An algorithm for mining moving flock patterns from pedestrian trajectories. In: MORISHIMA, A. et al. (Ed.). *Web Technologies and Applications - APWeb 2016 Workshops, WDMA, GAP, and SDMA, Suzhou, China, September 23-25, 2016, Proceedings*. [s.n.], 2016. (Lecture Notes in Computer Science, v. 9865), p. 310–321. Disponível em: <https://doi.org/10.1007/978-3-319-45835-9_27>.

[7] BENKERT, M. et al. Reporting flock patterns. *Comput. Geom.*, v. 41, n. 3, p. 111–125, 2008. Disponível em: <https://doi.org/10.1016/j.comgeo.2007.10.003>.

[8] ONG, R. et al. Parameter estimation and pattern validation in flock mining. In: APPICE, A. et al. (Ed.). *New Frontiers in Mining Complex Patterns - Second International Workshop, NFMCP 2013, Held in Conjunction with ECML-PKDD 2013, Prague, Czech Republic, September 27, 2013, Revised Selected Papers*. Springer, 2013. (Lecture Notes in Computer Science, v. 8399), p. 3–17. Disponível em: <https://doi.org/10.1007/978-3-319-08407-7_1>.

[9] SPACCAPIETRA, S. et al. A conceptual view on trajectories. *Data Knowl. Eng.*, v. 65, n. 1, p. 126–146, 2008. Disponível em: <https://doi.org/10.1016/j.datak.2007.10.008>.

[10] PÉREZ, I. A. et al. Applications of air mass trajectories. *Advances in Meteorology*, Hindawi, v. 2015, 2015.

[11] ESTER, M. et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In: SIMOUDIS, E.; HAN, J.; FAYYAD, U. M. (Ed.). *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD-96), Portland, Oregon, USA*. AAAI Press, 1996. p. 226–231. Disponível em: <http://www.aaai.org/Library/KDD/1996/kdd96-037.php>.

[12] GUDMUNDSSON, J.; KREVELD, M. J. van. Computing longest duration flocks in trajectory data. In: BY, R. A. de; NITTEL, S. (Ed.). *14th ACM International Symposium on Geographic Information Systems, ACM-GIS 2006, November 10-11, 2006, Arlington, Virginia, USA, Proceedings*. ACM, 2006. p. 35–42. Disponível em: <https://doi.org/10.1145/1183471.1183479>.

[13] LI, Z. et al. Swarm: Mining relaxed temporal moving object clusters. *Proc. VLDB Endow.*, v. 3, n. 1, p. 723–734, 2010. Disponível em: <http://www.vldb.org/pvldb/vldb2010/pvldb_vol3/R65.pdf>.

[14] ALMEIDA, L. V. de; VERDADE, V. E.; KASTER, D. S. Proposta de algoritmo por crescimento gradativo de diâmetro para detecção de k-flocks em dados de trajetórias. In: SBC. *Anais do XVI Escola Regional de Banco de Dados*. [S.l.], 2021. p. 119–128.

[15] WU, Q.; HAO, J. A review on algorithms for maximum clique problems. *Eur. J. Oper. Res.*, v. 242, n. 3, p. 693–709, 2015. Disponível em: <https://doi.org/10.1016/j.ejor.2014.09.064>.

[16] KARP, R. M. Reducibility among combinatorial problems. In: JÜNGER, M. et al. (Ed.). *50 Years of Integer Programming 1958-2008 - From the Early Years to the State-of-the-Art*. Springer, 2010. p. 219–241. Disponível em: <https://doi.org/10.1007/978-3-540-68279-0_8>.

[17] BOMZE, I. M. et al. The maximum clique problem. In: DU, D.; PARDALOS, P. M. (Ed.). *Handbook of Combinatorial Optimization*. Springer, 1999. p. 1–74. Disponível em: <https://doi.org/10.1007/978-1-4757-3023-4_1>.

[18] BRON, C.; KERBOSCH, J. Finding all cliques of an undirected graph (algorithm 457). *Commun. ACM*, v. 16, n. 9, p. 575–576, 1973.