Universidade Estadual de Londrina

DENIS EVANGELISTA SANCHES

DETECÇÃO ON-LINE DE K-FLOCKS COM DIÂMETRO COMO PARÂMETRO LIVRE

Londrina
2018

DENIS EVANGELISTA SANCHES

DETECÇÃO ON-LINE DE K-FLOCKS COM DIÂMETRO COMO PARÂMETRO LIVRE

Dissertação apresentada ao Programa de Mestrado em Ciência da Computação da Universidade Estadual de Londrina para obtenção do título de Mestre em Ciência da Computação.

Orientador: Prof. Dr. Daniel dos Santos Kaster
Coorientador: Dr. Marcos Rodrigues Vieira

Londrina
2018

Ficha de identificação da obra elaborada pelo autor, através do Programa de Geração Automática do Sistema de Bibliotecas da UEL

Evangelista Sanches, Denis.
Detecção On-line de k-Flocks com Diâmetro como Parâmetro Livre / Denis Evangelista Sanches. - Londrina, 2018.
82 f. : il.

Orientador: Daniel dos Santos Kaster.
Coorientador: Marcos Rodrigues Vieira.
Dissertação (Mestrado em Ciência da Computação) - Universidade Estadual de Londrina, Centro de Ciências Exatas, , 2018.
Inclui bibliografia.

1. Padrão Flock - Tese. 2. Descoberta de Padrões de k-comovimento - Tese. 3. Parâmetro de Distância Livre - Tese. I. dos Santos Kaster, Daniel. II. Rodrigues Vieira, Marcos. III. Universidade Estadual de Londrina. Centro de Ciências Exatas. . IV. Título.

DENIS EVANGELISTA SANCHES

DETECÇÃO ON-LINE DE K-FLOCKS COM DIÂMETRO COMO PARÂMETRO LIVRE

Dissertação apresentada ao Programa de Mestrado em Ciência da Computação da Universidade Estadual de Londrina para obtenção do título de Mestre em Ciência da Computação.

BANCA EXAMINADORA

Orientador: Prof. Dr. Daniel dos Santos Kaster
Universidade Estadual de Londrina – UEL

Prof. Dr. Renato Fileto
Universidade Federal de Santa Catarina – UFSC

Prof. Dr. Adilson Luiz Bonifácio
Universidade Estadual de Londrina – UEL

Londrina, 29 de agosto de 2018.

À minha mãe, com carinho.

AGRADECIMENTOS

A Deus.

À minha mãe Lourdes Maria Evangelista, mulher corajosa e batalhadora, por todo apoio incondicional, não apenas durante o período deste trabalho, mas por toda a minha vida.

Ao meu orientador Prof. Dr Daniel dos Santos Kaster, por todo ensinamento, auxílio, estímulo e paciência durante todo o percurso deste mestrado.

Aos professores que diretamente me ajudaram e me recepcionaram no Departamento de Informática e Estatística (UFSC-CTC-INE)[1] durante minha estadia no segundo semestre de 2016. Em especial à Prof.ª Dra. Vania Bogorny e ao Prof. Dr. Luis Otavio Alvares por terem me recebido de braços abertos e terem colaborado com o direcionamento deste trabalho. Não poderia de deixar de agradecer também aos doutorandos da época, Carlos Andres Ferrero e André Salvaro Furtado pelas inúmeras reuniões construtivas. Gostaria também de agradecer nominalmente o Prof. Dr. Renato Fileto que da mesma forma me instruiu direta e indiretamente enquanto aluno de sua disciplina.

Aos meus amigos de laboratório, companheiros de tempos estressantes de trabalho mas também de grandes momentos de divertimento. Também aos demais amigos do nosso grupo de amizade que sem dúvida alguma colaboraram com bons momentos neste período.

Aos demais amigos próximos e familiares que me acompanharam pacientemente nesse meu percurso, dando apoio e motivação!

[1] <http://ine.ufsc.br/>

"Não fui eu que ordenei a você? Seja forte e corajoso! Não se apavore nem desanime, pois o Senhor, o seu Deus, estará com você por onde você andar."
(Bíblia Sagrada, Josué 1:9)

SANCHES, D. E. **Detecção on-line de *k-Flocks* com diâmetro como parâmetro livre**. 2018. 82 f. Dissertação (Mestrado em Ciência da Computação) – Universidade Estadual de Londrina, Londrina, 2018.

RESUMO

É notória a ubiquidade dos dados espaço-temporais nos dias de hoje, demandando algoritmos eficientes para minerar informações importantes. Neste cenário, destaca-se a busca por padrões de comovimento de objetos móveis, pois possuem inúmeras aplicações em diversas áreas. O padrão *flock*, um dos mais conhecidos na literatura, por exemplo, identifica grupos de objetos móveis que se movimentam juntos, próximos entre si por uma distância máxima de um disco de diâmetro fixo predefinido. Inúmeros trabalhos apresentam variações e até mesmo outros padrões para mitigar as conhecidas limitações do padrão *flock* que são a consecutividade temporal e a exigência do grupo em permanecer junto limitado a um disco de tamanho predefinido. Para essa limitação do disco do padrão *flock* diversos trabalhos utilizam a busca por densidade argumentando ser mais flexível e menos restritiva. Contudo, tanto a abordagem de busca baseada em disco quanto a baseada em densidade requerem uma distância fixa como entrada para seus algoritmos, portanto, não resolvendo por completo o problema da dificuldade de parametrização desses algoritmos. Este trabalho de mestrado apresenta o conceito da descoberta de k-padrões de comovimento que, por meio de uma consulta exploratória exata, um critério de ranqueamento para as respostas e a liberação do usuário de fornecer um parâmetro fixo a uma condição dinâmica sobre o tempo, retorne a quantidade de padrões mais importantes segundo esse critério. Especificamente para a identificação de *flock*s, são apresentadas novas definições de padrões para encontrar os k *flocks* mais significativos, usando como critério de ranqueamento, ou seja, considerando como os *flocks* mais relevantes, aqueles de tamanho e extensão mínimos (de diâmetros mínimos), e com a distância como parâmetro livre ($k\epsilon$-*Flock*s). Define também um padrão para retornar $k\epsilon$-*Flocks* que respeitem uma dada distância máxima para refinamento das respostas. Respondendo a esses novos padrões, três algoritmos exploratórios com abordagem *top-down* são apresentados e avaliados. Um para detecção de $k\epsilon$-*Flock*s apenas em uma janela temporal de dados, outro com estratégia de janela deslizante para *stream* e conjuntos de dados inteiros, ambos livres do parâmetro de distância, e o terceiro de janela deslizante para consultar os $k\epsilon$-*Flocks* que respeitem a distância limite fornecida (*Filtered-*$k\epsilon$*-Flock*s). Por fim, os experimentos demostram a importância da aplicação desse novo conceito e padrões para uma consulta exploratória, e uma análise de desempenho dos algoritmos.

Palavras-chave: Padão *Flock*. descoberta de *flocks*. *k-Flocks*. Descoberta de padrões de k-comovimento. Parâmetro de distância livre. Algoritmo de agrupamento.

SANCHES, D. E. **Online detection of k-Flocks with diameter as a free parameter**. 2018. 82 p. Thesis (Master in Science in Computer Science) – Universidade Estadual de Londrina, Londrina, 2018.

ABSTRACT

The ubiquity of spatiotemporal data is noticeable nowadays, demanding efficient algorithms to mine important information. In this scenario, mining co-movement patterns of moving objects stands out, since it has several applications in different areas. The flock pattern, a well-known pattern in the literature, identifies groups of moving objects that move together within a predefined distance of a diameter disk over a period of time. Several works present variations and even other patterns to mitigate the limitations related to flock pattern i.e. the temporal consecutiveness and the requirement of the group to remain together bounded to a disk of predefined size. To outline disk limitation of the flock pattern several works use the density-based clustering in order to be more flexible and less restrictive. However, both the disk-based and density-based clustering approaches require a fixed distance boundary as input to their algorithms, thus not fully solving the problem of the parameterization in the algorithms. This master's work presents the concept of the discovery of k-patterns of co-movement that by means of an exact exploratory query, a ranking criterion for the answers and relinquishing the user from providing a fixed parameter to a dynamic condition over time, return the number of the most important patterns with respect to this criterion. Specifically for mining flock patterns, new patterns definitions are presented to find the most significant k flocks, with the ranking criterion being those flocks of minimum size and minimum extension the most relevant (minimum diameters), and with distance as free parameter (*kϵ-Flocks*). A new pattern to return *kϵ-Flocks* that respect a given maximum distance for refinement of the answers is also defined. Answering these new patterns, three exploratory top-down algorithms are presented and evaluated. One for mining *kϵ-Flocks* only in one temporal data window, another with sliding window strategy for stream and entire datasets, both free of the distance parameter and the third also with sliding window to mine *kϵ-Flocks* that respect the distance limit provided (*Filtered-kϵ-Flocks*). Finally, the experiments demonstrate the importance of applying this new concept for exploratory queries, and a performance analysis of these algorithms.

Keywords: Flock pattern. Flock discovery. *k*-Flocks. k-co-movement patterns discovery. Free distance parameter. Clustering algorithm.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

BFE	*Basic Flock Evaluation Algorithm*
BFS	*Breadth-First Search*
CFE	*Cluster Filtering Evaluation*
CMC	*Coherent Moving Cluster*
CRE	*Continuous Refinement Evaluation*
CuTS	*Convoy Discovery using Trajectory Simplification*
DBSCAN	*Density-Based Spatial Clustering of Applications with Noise*
DP	Algoritmo de *Douglas-Peucker*
GPS	*Global Positioning System*
LBS	*Location-Based Services*
LCM	*Linear time Closed itemset Miner*
MBR	*Minimum Bounding Rectangle*
MO	*Moving Object*
NOPW	*Normal Opening Window*
PFE	*Pipe Filter Evaluation*
PSI	*Plane sweeping, Signatures and Indexes*
REMO	*RElative MOtion*
TDE	*Top Down Evaluation*

LISTA DE SÍMBOLOS

T_j	j-ésima trajetória
$\mathcal{T}$	conjunto de trajetórias
t_i	i-ésimo instante de tempo
$p_j^{t_i}$	ponto de localização da T_j em t_i
μ	número mínimo de trajetórias ($\mu \in \mathbb{N}$)
δ	duração mínima $\delta > 1$ ($\delta \in \mathbb{N}$)
k	quantidade de respostas solicitadas
w	janela temporal de comprimento δ ($\|w\| = \delta$)
ϵ	distância máxima $\epsilon > 0$ (diâmetro)
ε	raio de conectividade para algoritmos de densidade (DBSCAN)
f_k	k-ésimo *flock*
$\mathcal{F}$	conjunto de *flocks*
$c_k^{t_i}$	centro de disco do f_k em t_i
$\overline{f_w}$	*flock* mais extenso (de maior diâmetro) em w
$\mathcal{F}_w^k$	k_ϵ-*Flocks* de uma única janela temporal w
$\mathcal{F}^k$	conjunto de todos $\mathcal{F}_w^k$ de todas as janelas temporais
$\mathcal{F}_w^{filtered-k}$	*Filtered-*k_ϵ*-Flocks* de uma única janela temporal w
$\mathcal{F}^{filtered-k}$	conjunto de todos $\mathcal{F}_w^{filtered-k}$ de todas janelas temporais

SUMÁRIO

1 INTRODUÇÃO

A ubiquidade dos serviços baseados em localização, do inglês *Location-Based Services* (LBS), é notória. A demanda por esses serviços, que utilizam o rastreamento de localização de objetos móveis, é crescente graças à evolução tecnológica, como os dispositivos móveis com GPS (*Global Positioning System*) embutido, ao barateamento dessas tecnologias e às inúmeras aplicações possíveis [6, 11, 12]. Como exemplos nos dias de hoje há os aplicativos Waze[1] e Uber[2]. O primeiro é um aplicativo de trânsito, em que os usuários podem, por exemplo, visualizar e calcular rotas, identificar congestionamentos e acidentes. Já no aplicativo Uber é possível, a partir da localização atual, solicitar algum táxi privado próximo, além de conseguir acompanhar seu trajeto até o local solicitado.

Como entrada de dados desses LBSs têm-se os caminhos dos objetos móveis — *Moving Objects* (MOs) — sendo rastreados. Esses são persistidos como sequências de pontos de localização, coletados por meio de dispositivo GPS, respeitando uma taxa de amostragem, contendo as coordenadas do MO no exato momento de cada coleta. Esses MOs podem ser pessoas com *smartphones*, animais com coleiras com GPS, carros, etc. No entanto, a análise exaustiva dos caminhos traçados por todos esses objetos, 24 horas por dia, 7 dias por semana, muitas vezes não é imprescindível [13]. Então, de acordo com a necessidade, a aplicação pode requerer uma taxa de amostragem de pontos diferente. Ou seja, para acompanhar a rota em tempo real de um carro, quanto maior a taxa de amostragem desses pontos, maior a confiabilidade e precisão na análise e visualização. Por outro lado, se a intenção for apenas identificar os principais pontos de parada de um caminhão em um dia, apenas alguns pontos de localização diários já serão satisfatórios.

A partir dessa crescente quantidade de dados espaço-temporais cada vez mais aumenta a demanda por algoritmos capazes de extrair informações relevantes. Dentre as inúmeras possibilidades, há a subárea de identificação de padrões de MOs em comovimento, que basicamente são grupos de MOs se movendo próximos entre si por um período. Esses algoritmos são divididos em dois paradigmas: os algoritmos on-line, em que os dados são processados assim que são coletados, ou em janelas temporais de dados, em que o desempenho se torna requisito crítico por demandar processamento em *stream* de dados; e os algoritmos off-line, tal que todo o conjunto de dados é exigido como entrada para o processamento. Os algoritmos de identificação de padrões de comovimento também são divididos com respeito à estratégia de agrupamento utilizada. Existem, então, basicamente duas abordagens: os algoritmos que utilizam agrupamento baseado em distância/disco, e aqueles baseados no conceito de densidade, muitas vezes utilizando o

[1] www.waze.com
[2] www.uber.com

algoritmo de agrupamento *Density-Based Spatial Clustering of Applications with Noise* (DBSCAN) ou alguma de suas variações.

Na estratégia com busca baseada em discos, o Padrão *Flock* é um dos mais conhecidos na literatura. Por definição, um *flock* é um grupo de pelo menos μ MOs se movendo juntos, limitados a uma distância máxima de um disco de diâmetro ϵ predefinido e por um intervalo de tempo mínimo δ. No entanto, como apontado em alguns trabalhos [14, 9, 8], o Padrão *flock* requer condições muito restritas para algumas aplicações, como a limitação espacial definida pela forma de um disco de tamanho fixo e a restrição de consecutividade temporal, isto é, a necessidade de haver pontos de localização de todos os MOs do grupo em todos os instantes de tempo deste período, o que limita encontrar padrões importantes. Desse modo, trabalhos são encontrados na literatura propondo flexibilizações a esse padrão [15], assim como a definição de outros padrões [14, 7, 16, 9, 10, 17] que, em relação à limitação espacial do disco e seu diâmetro fixo, utilizam a abordagem por densidade como solução.

Contudo, embora seja verdade que a definição de um diâmetro fixo para encontrar padrões de comovimento, no caso *flocks*, seja uma tarefa difícil até mesmo para especialistas de domínio, a abordagem por agrupamento (DBSCAN) também requer o fornecimento de um parâmetro de distância que é sensível à distribuição dos MOs no espaço. Mesmo que alguns trabalhos [8, 18] apresentem análises para estimar esse parâmetro de distância, ainda há a exigência de que esse parâmetro seja fornecido aos algoritmos. Além disso, ambas as estratégias com disco ou por densidade são sensíveis às peculiaridades de cada domínio. Em um monitoramento de tráfego, por exemplo, esse problema é bem evidente, pois há inúmeros tipos de veículos sendo monitorados, como diferentes modelos de carros, caminhões, motos, etc, que se movem com velocidades distintas. O meio ainda impacta nesses objetos, como semáforos, quebra-molas, acidentes, desvios, congestionamentos e clima. Adicionalmente, cada um dos veículos pode possuir um dispositivo com taxa de amostragem diferente, uns coletando suas localizações a cada dois segundos enquanto outros apenas a cada certa distância percorrida. Neste cenário, qual a distância fixa para se encontrar grupos de veículos se movendo juntos a qualquer hora e local? Inviável.

Este trabalho apresenta uma nova abordagem de descoberta de k-padrões de comovimento, que é a identificação dos k padrões em janelas temporais independentemente da variação de certo comportamento durante o tempo, onde k-respostas são as mais relevantes segundo um critério desejado. Assim uma busca exploratória exata é realizada para identificar essas *top-k* respostas. Essa abordagem pode variar de acordo com o padrão desejado e o parâmetro para definir esse critério de ranqueamento.

Aplicando essa nova abordagem especificamente ao Padrão *Flock*, adotando a distância como comportamento variável durante o tempo, dispensando assim o fornecimento desse parâmetro pelo usuário, e sendo os *flocks* de menor extensão o critério de ranquea-

mento, este trabalho apresenta o novo conceito de k_ϵ-*Flocks* que são os k-*flocks* de diâmetro mínimo em uma dada janela temporal e a definição do novo Padrão k_ϵ-*Flock* que retorna o conjunto de todos esses k_ϵ-*Flocks* para todas as janelas temporais processadas, seja a entrada todo um conjunto de dados ou uma *stream* de dados. Da mesma forma, é definido o conceito de *Filtered*-k_ϵ-*Flocks*, que são os k_ϵ-*Flocks* de até um diâmetro máximo, e o Padrão *Filtered*-k_ϵ-*Flocks* para identificação de todos os *Filtered*-k_ϵ-*Flocks* de uma entrada de dados, utilizando uma distância limite para essa busca exploratória.

Um algoritmo exploratório exato (Algoritmo *Top-Down*) para uma janela temporal e uma variação (Algoritmo *Top-Down* em Janela Deslizante) para uma entrada de dados são apresentados e analisados para entendimento inicial de conjuntos de dados pelo usuário. Uma variação mais eficiente em termos de custo, o *Filtered Top-Down*, também é apresentado para a identificação dos mesmos k_ϵ-*Flocks* quando já há uma noção prévia do usuário para delimitar a busca, limitando a extensão máxima do diâmetro das respostas a um valor previamente definido e fornecido.

A sequência desta dissertação está organizada da seguinte forma. O Capítulo 2 apresenta os conceitos e definições básicas que sustentam o restante deste trabalho. O Capítulo 3 descreve os principais padrões de comovimento de objetos móveis da literatura, destacando suas flexibilizações em relação às limitações do Padrão *Flock* e aborda o impacto e análise da parametrização sobre este padrão. O Capítulo 4 apresenta a definição do problema e os novos conceitos e algoritmos propostos. O Capítulo 5 descreve os experimentos e resultados. Por fim, o Capítulo 6 apresenta a conclusão e as considerações finais deste trabalho.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo são apresentados os conceitos e as definições básicas, fundamentais para o entendimento do restante deste trabalho. Além disso, são abordados alguns algoritmos para os padrões foco deste trabalho, *flock* e comboio, diferenciando suas estratégias de busca adotadas, baseadas em distância/disco e em densidade, respectivamente, e monitoramento contínuo de agrupamentos de objetos em movimento.

2.1 Trajetórias de Objetos Móveis

Spaccapietra et al.[1] definem trajetórias como segmentos — pedaços — de caminhos de objetos móveis que representam a evolução espaço-temporal desde um ponto de partida até um final, sendo persistidos em formato de pontos, denotadas como $trajetória(T) : [t_{início}, t_{fim}] \rightarrow$ espaço. Para formalizar esses conceitos seguem as Definições 1, 2 e 3, adaptadas de [19], considerando que este trabalho utiliza o espaço Euclidiano de duas dimensões para representar localizações, com latitude e longitude.

Definição 1 *Um ponto p é uma tupla* (x, y, t) *onde* x *e* y *são coordenadas espaciais representando um lugar e* t *o instante de tempo em que este ponto foi coletado.*

Definição 2 *A trajetória T de um objeto móvel (MO) é a sequência de pontos* $<p_1, p_2, \ldots, p_n>$ *amostrados, onde* $p_i = (x_i, y_i, t_i)$ *e* $t_i < t_{i+1}$ *para* $1 \leq i \leq n$.

Definição 3 *Uma subtrajetória T' de T é uma lista de pontos consecutivos* $<p_j, p_{j+1}, \ldots, p_{j+l}>$, *onde* $p_j \subset T, j \geq 1,$ *e* $j + l \leq n$.

Esses pontos são coletados por dispositivos com tecnologias capazes de reportar, com certa precisão, a localização em determinado momento de um MO, como exemplo, e principalmente, dispositivos com GPS (*Global Positioning System*) embutido. Essas localizações são coletadas de acordo com uma taxa de amostragem (*sampling rate*), normalmente denotado por R, de tal forma que a cada intervalo de tempo, um ponto de localização é coletado e reportado pelo dispositivo. As taxas de amostragem são variadas pois dependem dos dispositivos e do domínio da aplicação. Ong et al.[18] afirmam que embora uma alta taxa de amostragem represente melhor o caminho de um objeto, o processamento necessário para se trabalhar com esses dados também deve ser maior.

Esses dados espaço-temporais requerem tratamentos específicos (pré-processamento) antes de serem utilizados diretamente em aplicações. Na literatura, várias técnicas são

mencionadas para essa etapa. Uma delas, a limpeza (*cleaning*), busca identificar e remover dados inconsistentes, ruidosos e *outliers*. A compressão/simplificação (*compression/simplification*), quando da necessidade de filtrar um conjunto menor de dados para processamento, elimina pontos redundantes e desnecessários. Como exemplo, um algoritmo muito utilizado para isso na literatura é o de *Douglas-Peucker* (DP) [13]. Consequentemente, após a aplicação de qualquer técnica que visa reduzir a quantidade de dados de uma trajetória chama-se a nova representação de trajetória aproximada (*aproximate trajectory*), não podendo esta se desviar demais da original.

Por outro lado, pode ser necessário a complementação (*completion*), para reduzir incertezas em trajetórias com baixa taxa de amostragem, falhas e/ou ruídos. Nesse contexto, algoritmos de interpolação, principalmente a linear, são utilizados para calcular pontos aproximados conforme necessidade. Já a calibração (*calibration*) das trajetórias busca homogeneizar suas taxas de amostragem para que sejam facilitadas comparações entre essas trajetórias em tempos predeterminados, já que cada dispositivo pode possuir sua estratégia de amostragem. Por fim, as longas trajetórias dos MOs muitas vezes precisam ser quebradas em trajetórias/subtrajetórias para que façam mais sentido para as respectivas aplicações, sendo essa técnica denominada segmentação (*segmentation*) [13, 20, 21]. Um exemplo é a quebra de trajetórias de pessoas por dias, ou seja, seu caminhos diários como únicas trajetórias, ou ainda de forma mais semântica, entre trajetórias que descrevam caminhos entre casa e trabalho pela manhã, trabalho e restaurante na hora do almoço, do restaurante ao trabalho, e outra do trabalho à casa no fim do dia.

A Figura 1 apresenta boa parte dessas definições e conceitos. Dado todo o caminho de um MO sendo monitorado, do instante t_0 ao atual t_{agora}, seu caminho é representado por todos os pontos coletados seguindo uma taxa de amostragem. Contudo, conforme necessidade do contexto da aplicação, partes desse caminho são ignorados, no caso, subtrajetórias irrelevantes, como por exemplo, entre os instantes de tempo t_1 e t_2, t_4 e t_5, t_6 e t_7, e t_8 e t_{agora}, após um processo de segmentação.

Esses dados coletados apenas como pontos de localização e o identificador de seus respectivos objetos móveis são denominados dados brutos, compondo, assim, trajetórias brutas (*raw trajectories*). Quando a esses dados são acrescentados informações extras, normalmente de outras fontes de dados, diz-se que essas trajetórias são enriquecidas pelo processo de anotação. Esta abordagem, tendo como precursores Spaccapietra et al.; Alvares et al.[1, 22], trabalha com trajetórias ditas enriquecidas semanticamente (trajetórias semânticas), não sendo foco deste trabalho.

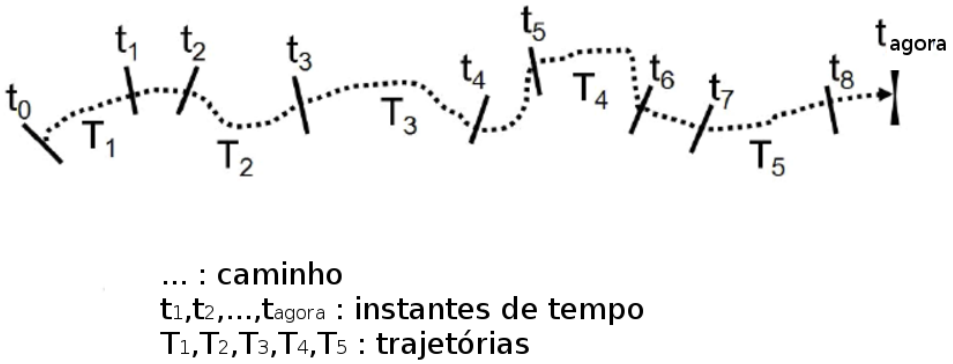


Figura 1 – Representação do caminho de um objeto móvel, segmentado semanticamente em trajetórias. Neste exemplo, algumas partes do caminho — pontos não pertencentes às trajetórias — são irrelevantes à aplicação. (Fonte: Adaptada de 1)

2.2 Padrões de Grupos de Objetos Móveis

Com a evolução tecnológica e consequente aumento da disponibilidade de dados espaço-temporais a busca em identificar informações nessas grandes massas de dados, e até mesmo on-line, tem demandado grande interesse da academia e indústrias. Encontrar correlações entre grupos de objetos móveis (MOs) ou até mesmo entre as próprias subtrajetórias de um único MO é de fundamental importância para vários domínios de problemas e aplicações [23]. Dentre os diversos exemplos de aplicações com a identificação desses padrões, Zheng e Zhou[23] citam otimização de aplicações de transporte e logística, métodos de predição e estudos de comportamento de animais em seus habitats naturais. Spaccapietra et al.[1], por sua vez, mencionam aplicações também em ciências sociais, biologia e medicina.

Como previamente mencionado, os padrões podem se referir a uma única trajetória ou a um grupo. Esses padrões, na literatura, são normalmente classificados como agrupamento de trajetórias, padrões sequenciais, padrões periódicos e padrões de grupos se movendo juntos [20], este foco deste trabalho. A Figura 2, com trajetórias para quatro objetos (MOs) — uma cor para cada — em vinte instantes de tempo, demonstra alguns exemplos desses possíveis padrões: um *flock* com os objetos vermelho, azul e verde por cinco instantes de tempo; um padrão periódico com o MO verde, ocorrendo o mesmo comportamento espaço-temporal com alguma periodicidade; um padrão de encontro, onde os MOs azul, vermelho e laranja se encontram por quatro instantes de tempo e; um padrão de local frequente, refletindo uma região frequentemente visitada pelo MO laranja.

Para a mineração desses padrões de MOs há dois paradigmas distintos, os algoritmos off-line e on-line. A maior diferença entre esses paradigmas é que para o primeiro, todo o conjunto de dados é necessário já como entrada para o processamento. Em contrapartida, os algoritmos on-line processam os dados assim que estes vão chegando — *stream*

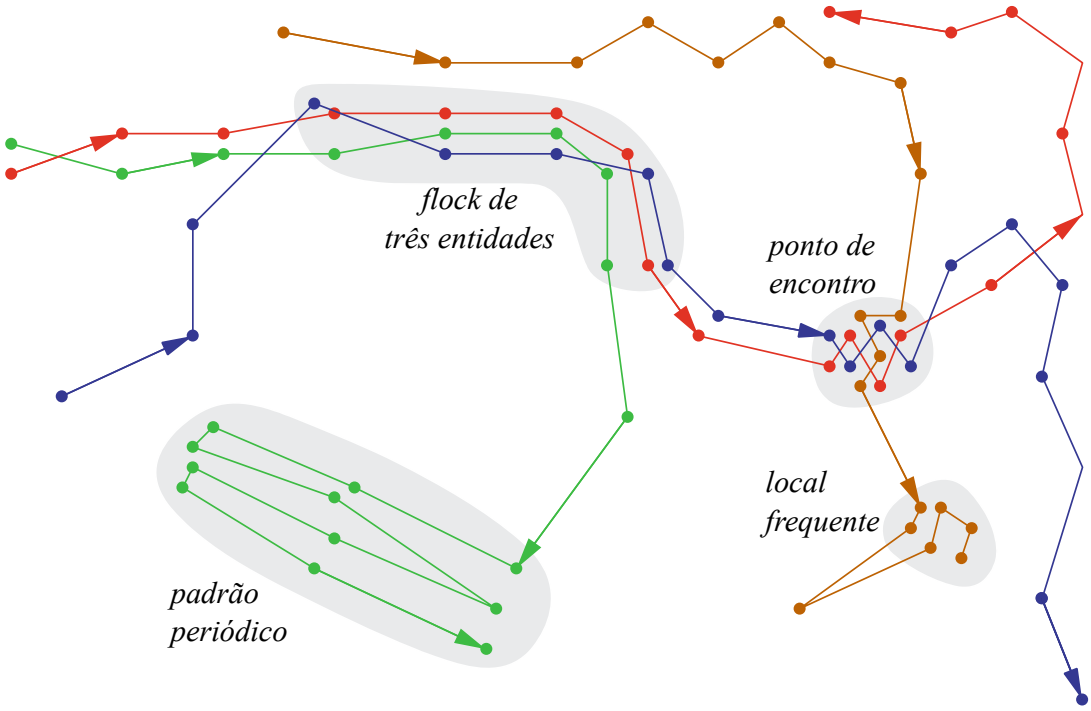


Figura 2 – Exemplos de padrões a partir de trajetórias. (Fonte: 2, traduzida de 1)

de dados —, seja o processamento de cada ponto de localização reportado ou até mesmo quando os dispositivos reportam "lotes" (*batches*) de dados [23, 20]. Ressalta-se, também, que um algoritmo para suportar o paradigma on-line requer processamento eficiente, "*on the fly*" [17, 24].

Outra diferenciação em se tratando de mineração desses padrões de grupos de MOs é justamente a estratégia de identificação/caracterização desses grupos, ou seja, a técnica de agrupamento. Para isso, duas técnicas se sobressaem em relação à questão de similaridade espacial entre MOs: a busca baseada em distância, também conhecida como busca baseada em disco (*range query*) e o agrupamento baseado em densidade [23, 21]. A primeira estratégia, baseada em discos, embora seja mais simples e computacionalmente mais barata traz limitações na identificação de grupos de diferentes naturezas, como a limitação do formato espacial, restringido por um disco, e a dificuldade em se identificar raios ideais como entrada aos algoritmos. Por outro lado, a estratégia baseada em densidade, embora seja abordada como solução às limitações do disco, ainda requer o raio de conectividade espacial, de forma análoga ao diâmetro da abordagem com discos, além de possuir maior custo computacional [25, 9, 14, 14, 17, 26].

As próximas seções apresentam os padrões *flock* e comboio, que são base para diversas outras propostas de padrões de comovimento, que serão apresentados no Capítulo 3. Também será introduzido brevemente o conceito de agrupamentos em movimento (*Moving Clusters*), que é um tema correlato mas não diretamente abordado neste trabalho.

2.3 Padrão *Flock*

Um dos padrões de grupo de objetos móveis em comovimento mais conhecidos é o Padrão *Flock*. Este padrão é formado por conjuntos maximais de objetos se movendo em grupo por um tempo predeterminado em que a distância entre os membros de cada grupo é delimitada por um disco. Na literatura, diferentes definições são encontradas para o Padrão *Flock*. Sua primeira referência foi o trabalho de Laube e Imfeld[27], precursor na definição de vários padrões de movimentos com o *framework* REMO (*RElative MOtion*), definindo um Padrão *Flock* como um grupo de posições de objetos se movendo na mesma direção. Em um trabalho posterior dos mesmos autores, um *flock* torna-se restrito espacialmente, sendo identificado um padrão apenas nos objetos de mesma direção, desde que respeitando uma distância máxima delimitada por, um disco de raio predefinido por exemplo [28]. Embora este trabalho tenha levantado a necessidade da restrição espacial para a identificação de padrões *flock* mais interessantes, apenas em um trabalho posterior [29] a definição passou a estabelecer de fato a questão de proximidade, exigindo que os objetos do *flock* respeitem uma região circular de raio predefinido. No entanto, todas as definições até então apenas levam em consideração um único instante de tempo, não contemplando de fato a identificação de objetos móveis andando em grupo durante um intervalo de tempo. Essa restrição temporal ao padrão foi adicionada nos trabalhos de Gudmundsson e Kreveld; Benkert et al.; Benkert et al.[30, 31, 32], ou seja, tendo o grupo se movimentando por um intervalo mínimo de tempo. O padrão adotado neste trabalho é dado pela Definição 4.

Definição 4 *Dados um conjunto de trajetórias $\mathcal{T}$, um número mínimo de trajetórias $\mu > 1\,(\mu \in \mathbb{N})$, uma distância máxima $\epsilon > 0$ definida por uma função de distância d, e um tempo de duração mínima $\delta > 1\,(\delta \in \mathbb{N})$. Um padrão $Flock(\mu, \epsilon, \delta)$ retorna todas as coleções $\mathcal{F}$ de tamanho máximo de trajetórias onde: para cada $f_k \in \mathcal{F}$, o número de trajetórias em f_k é maior ou igual a μ $(|f_k| \geq \mu)$ e existem δ instantes de tempo consecutivos tal que para todo instante de tempo $t_i \in [f_k^{t_1}..f_k^{t_{1+\delta}}]$, há um disco de centro $c_k^{t_i}$ e raio $\epsilon/2$ cobrindo todos os pontos $p_j^{t_i}$ em $f_k^{t_i}$. Ou seja: $\forall f_k \in \mathcal{F}$, $\forall t_i \in [f_k^{t_1}..f_k^{t_{1+\delta}}]$, $\forall T_j \in f_k : |f_j^{t_i}| \geq \mu$, $d(p_j^{t_i}, c_k^{t_i}) \leq \epsilon/2$.*

Para exemplificar, a Figura 3 ilustra a evolução espacial das trajetórias $T_1, \ldots, T_7$, especificadas pelos pontos em preto, em quatro instantes de tempo consecutivos, do 1 ao 4. Tomando $\mu = 3$ e $\delta = 3$, o único *flock* f encontrado é $f_1 = \{T_1, T_2, T_3\}$, de cores azul, verde e vermelho respectivamente, do intervalo de tempo $[1, 3]$, pois todos os pontos de localização dessas três trajetórias estão na área de um disco de centro $c_k^{t_i}$, e diâmetro ϵ, para o *flock* f_k no tempo t_i. O mesmo não acontece com o grupo $\{T_4, T_5, T_6\}$, representado pelos discos tracejados no intervalo de tempo $[2, 4]$. Como a trajetória T_6 dispersa das outras duas no instante $t = 3$, não há um disco de diâmetro ϵ que consiga cobrir todas essas

trajetórias nesse instante de tempo, acarretando a interrupção temporal desse candidato, sendo, consequentemente, descartado. Esse exemplo também mostra a sensibilidade do padrão quanto, principalmente, ao tamanho ideal do disco a ser utilizado. Se por um lado a escolha de um ϵ pequeno pode acarretar a perda de candidatos importantes, por outro, um grande demais poderia filtrar objetos distantes como sendo do mesmo grupo.

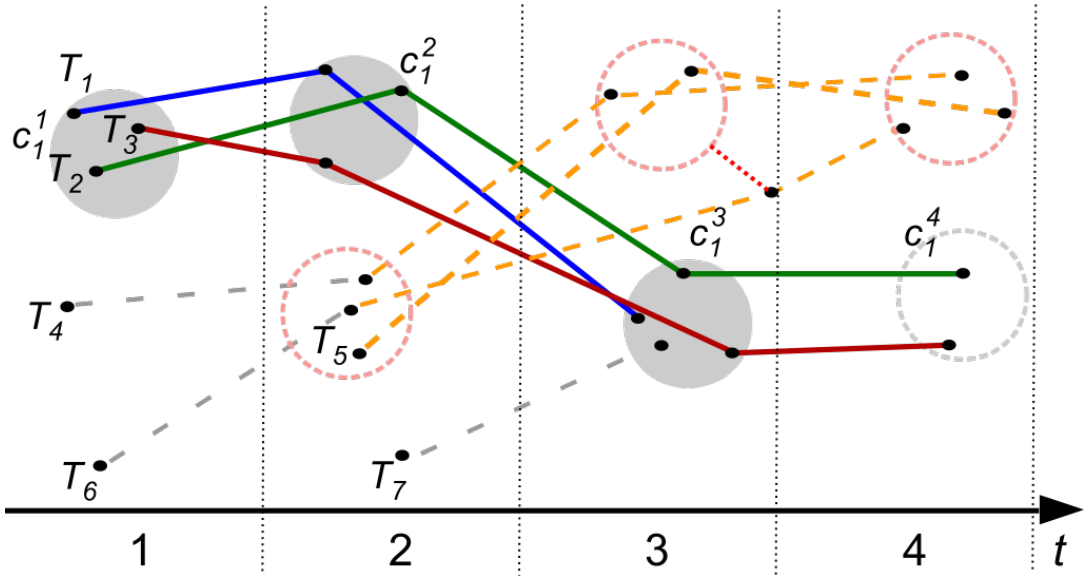


Figura 3 – Exemplo de Padrão *Flock*. (Fonte: Adaptada de 3)

Vale ressaltar que a Definição 4 adotada neste trabalho reporta *flocks* quando atingem comprimento δ (instantes de tempo consecutivos). Essa estratégia reporta *flocks* conhecidos como de comprimento fixo. Por outro lado, trabalhos como [30, 33, 34] buscam *flocks* de comprimento maximal, ou seja, aqueles de maior comprimento, retornando assim, menos respostas.

2.3.1 Principais Algoritmos para Detecção do Padrão *Flock*

Esta subseção apresenta três algoritmos conhecidos na literatura, utilizados como base para outros algoritmos e benchmarking, para a identificação de padrões *flock*.

2.3.1.1 *Basic Flock Evaluation Algorithm* (BFE)

Vieira, Bakalov e Tsotras[3] propuseram o primeiro algoritmo exato para detecção de padrões *flock* — seguindo a Definição 4 —, o BFE (*Basic Flock Evaluation Algorithm*). Além desse algoritmo ser em tempo polinomial e on-line, os autores propuseram quatro heurísticas visando a melhora do custo computacional com a redução de processamento de *flocks* candidatos desnecessários. O BFE pode ser dividido em duas etapas: (i) a identificação de discos — *flocks* candidatos a cada instante de tempo t_i —, e (ii) a junção desses candidatos com aqueles resultantes do instante anterior t_{i-1}.

Teorema 1 *Se para um dado instante de tempo t_i existir um ponto no espaço $c_k^{t_i}$ tal que:*

$$\forall T_j \in f, d(p_j^{t_i}, c_k^{t_i}) \leq \epsilon/2$$

então existe outro ponto no espaço $c'^{t_i}_k$ tal que

$$\forall T_j \in f, d(p_j^{t_i}, c'^{t_i}_k) \leq \epsilon/2$$

e existem pelo menos trajetórias $T_a \in f$ e $T_b \in f$ tal que

$$\forall T_j \in \{T_a, T_b\}, d(p_j^{t_i}, c'^{t_i}_k) = \epsilon/2$$

Na primeira etapa, a cada instante de tempo, é construído um índice de grade, com células do tamanho do diâmetro dos discos (ϵ), alocando todos os pontos em suas respectivas células de acordo com suas posições no espaço. Essa estratégia tem como finalidade reduzir o espaço de busca para processar apenas pontos próximos entre si na detecção de possíveis discos candidatos. Visando também a redução espacial de busca, o Teorema 1 do trabalho de Vieira, Bakalov e Tsotras[3] tem papel fundamental. Este define que, embora os objetos móveis (MOs) possuam movimentos livres no espaço-tempo, de tal forma que seriam infinitas as possibilidades de centros de discos cobrindo esses MOs, é possível ter um número finito de centros de discos, tomando dois pontos não mais distantes que ϵ e desenhando dois discos, um para cada lado, tal que esses dois pontos fazem parte da circunferência de ambos os discos. A Figura 4 mostra esses dois discos possíveis, de raio $\epsilon/2$, a partir dos pontos p_1 e p_2, com $dist(p_1, p_2) \leq \epsilon$.

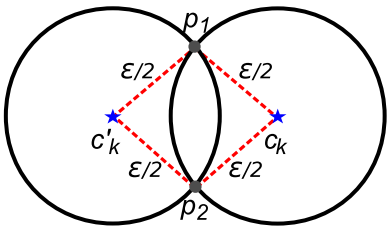


Figura 4 – Discos para $\{p_1, p_2\}, d(p_1, p_2) \leq \epsilon$. (Fonte: 3)

O processamento então é feito varrendo essa estrutura de grade. Como demonstrado na Figura 5, a cada célula $g_{x,y}$ não vazia, varre-se todos os seus pontos, emparelhando-os dois a dois, quando em até ϵ de distância. A Figura 5 mostra esse processo com o ponto p_1, no qual é feita uma busca por abrangência com raio ϵ. Para isso, são levados em consideração apenas os pontos da própria célula e das oito adjacentes, pois apenas pontos desta área limitada podem estar a até ϵ de p_1, no caso, os pontos pretos. Vale ressaltar que se não houver ao menos μ pontos nessas nove células, o processamento parte para

outra, visto que não é possível encontrar um candidato nesta área. Ainda nessa figura, p_1 é emparelhado com p_2, construindo assim os discos c_1 à esquerda e c_2 à direita, ambos de raio $\epsilon/2$. Desse modo, após estes processamentos e de um passo extra para manter apenas os discos de tamanhos maximais, a cada instante de tempo são encontrados todos os possíveis candidatos.

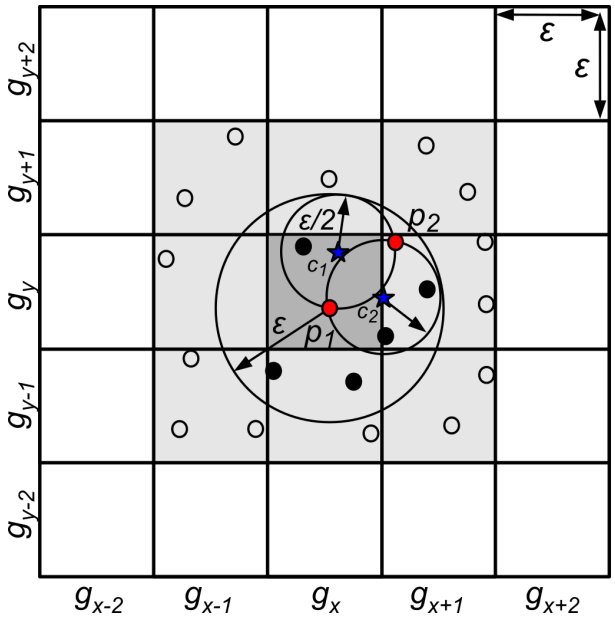


Figura 5 – Exemplo de processamento com base no índice de grade. (Fonte: 3)

Na segunda etapa, após identificar os candidatos no instante de tempo atual, é feita a "junção" de candidatos de t_i com aqueles em t_{i-1}, desde que possuam ao menos μ MOs em comum. Todos os candidatos do instante atual são considerados candidatos iniciais, de comprimento 1, ou seja, de um instante de tempo (t_i). Se esses candidatos tiverem correspondentes com aqueles no tempo anterior, ou seja, se tiverem ao menos μ MOs em comum, então são mantidos com esses MOs em comum, tendo seu tempo de vida acrescido em 1. Ao final, mantendo apenas os candidatos maximais, são reportados como $flocks$ aqueles que tiverem comprimento igual a δ, visto que o BFE retorna $flocks$ de tamanho fixo. Caso isso aconteça, o mesmo $flock$ reportado se torna novamente candidato no algoritmo, mas com o seu tempo de início aumentado em 1, reduzindo seu comprimento para $\delta - 1$ novamente, de tal forma que se este tiver outro correspondente em t_{i+1}, será reportado outra vez, mas agora com outro intervalo de tempo.

Já os demais quatro algoritmos que trabalham sobre um *buffer* de tamanho δ — janela w ($|w| = \delta$), implementados com heurísticas para se reduzir processamento e melhorar desempenho, são brevemente apresentados a seguir.

TDE (*Top Down Evaluation*) A ideia desta heurística é a suposição de que entre dois instantes de tempo consecutivos há pouca variação, fazendo com que o BFE

processe candidatos que em instantes de tempo seguintes seriam descartados. Assim, o TDE armazena os pontos de toda a janela w de comprimento δ e identifica os discos tanto no primeiro instante de tempo da janela w (t_i) quanto no último ($t_{i+\delta-1}$). Dessa forma, apenas as trajetórias dos MOs dos possíveis candidatos após a junção desses discos são processados por toda a janela w com o BFE.

PFE (*Pipe Filter Evaluation*) Essa heurística aplica um filtro nas trajetórias dentro da janela w, processando posteriormente no BFE apenas aquelas que tiverem ao menos μ pontos de localização a uma distância máxima ϵ durante toda a janela w. Por isso o nome da heurística, pois só são processados no BFE trajetórias que estiverem nesses "tubos" de diâmetro ϵ e com ao menos μ MOs durante w.

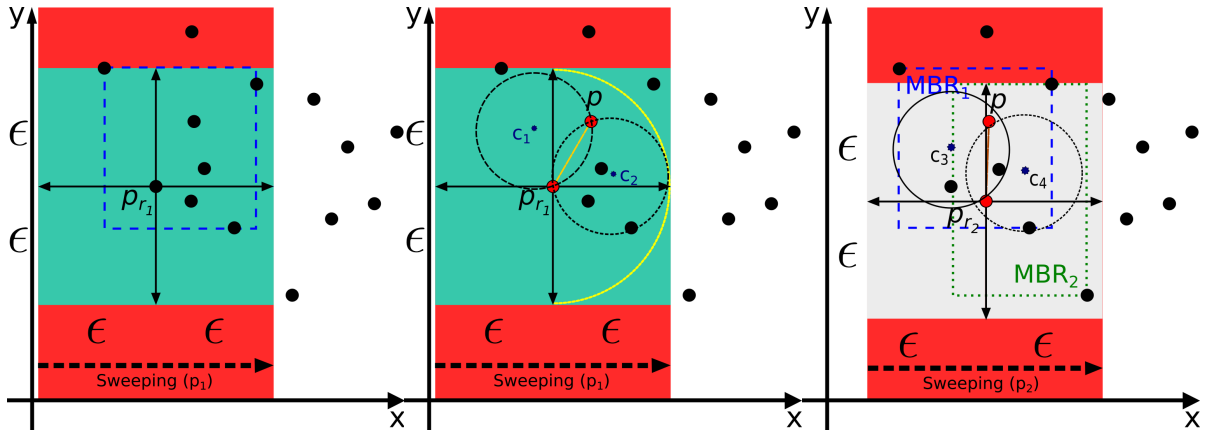
CRE (*Continuous Refinement Evaluation*) A abordagem desta heurística é, como seu próprio nome diz, aplicar um refinamento contínuo. Assim, no primeiro instante t_i da janela w, todos os discos de candidatos são encontrados. No entanto, para o instante de tempo t_{i+1} são processados apenas as trajetórias em candidatos do tempo anterior, aplicando esse refinamento incremental até o fim da janela w, utilizando assim o BFE sobre esse conjunto de trajetórias respectivamente.

CFE (*Cluster Filtering Evaluation*) A quarta e última heurística aplica a cada instante de tempo t_i o algoritmo DBSCAN (Seção 2.4) com $\varepsilon = \epsilon$ e $MinPts = \mu$. Posteriormente, verifica-se quais candidatos permanecem do instante anterior, desde que contenham ao menos μ MOs em comum com os agrupamentos encontrados em t_i. As trajetórias que participarem desses agrupamentos após as junções durante toda a janela w são processados pelo BFE. Vale ressaltar que o CFE apresentou a pior desempenho nos experimentos realizados em [3] por causa justamente do alto custo em se aplicar um algoritmo de agrupamento como o DBSCAN em todos os instantes de tempo.

2.3.1.2 *Plane sweeping, Signatures and Indexes* (PSI)

O algoritmo PSI (***P****lane sweeping,* ***S****ignatures and* ***I****ndexes*) primeiramente apresentando em [35] e posteriormente aprimorado em [4], consiste na aplicação de técnicas em certas etapas do próprio BFE para ganho de desempenho. Essas técnicas, em síntese, e que são explicadas com mais detalhes a seguir, são: (i) varredura de plano (*plane sweeping*) em vez de se trabalhar com o índice grade, (ii) utilização de assinatura binária (*binary signatures*) para reduzir as custosas operações de intersecção de conjuntos de trajetórias na junção de discos/candidatos de mesmo instante de tempo e (iii) a utilização de um índice invertido (*inverted index*) para agilizar junções de candidatos entre dois instantes de tempo consecutivos.

A Figura 6 apresenta as etapas do processamento de um instante de tempo no PSI. Em vez de construir o índice de grade e alocar os pontos em células como no BFE, o PSI ordena os pontos em cada instante de tempo pelo eixo x. Assim, a varredura é feita da esquerda para a direita, para todos os pontos, tomando o ponto p_r como centro de uma caixa (*box*) de tamanho 2ϵ x 2ϵ, conforme Figura 6a. No segundo momento, é feita uma busca por abrangência com ϵ apenas para o lado direito dessa caixa para identificar os pontos para emparelhar e criar discos, também conforme Teorema 1 em [3], não levando em consideração os pontos à esquerda pois já foram processados anteriormente (Figura 6b). Caso seja encontrado algum disco dentro dessa caixa, é construída uma MBR (*Minimum Bounding Rectangle*) que englobe todos os pontos da respectiva caixa, marcando essa MBR como ativa, assim como as MBR_1 e MBR_2 da Figura 6c. Após a identificação desses discos, antes da verificação da existência de discos que sejam subconjuntos de outros, por meio de operações de conjunto, assim como é feito no BFE, o PSI varre essas MBRs ativas encontradas e verifica aquelas que se intersectam.



(a) Varredura de plano em x. (b) Identificação de discos. (c) Construção de MBRs.

Figura 6 – Busca por discos maximais em cada instante de tempo com varredura de plano. (Fonte: 4)

Os discos em MBRs ativas sem intersecção são considerados *flocks* candidatos, visto que isso garante que não existam discos que contenham os discos dessas MBRs. Já aqueles que estão em MBRs interseccionadas, assim como as MBR_1 e MBR_2 da Figura 6c, passam para uma segunda etapa, que é a verificação das assinaturas binárias. Toda vez que um disco com pelo menos μ MOs é encontrado, são aplicadas funções *hash* sobre seus objetos, de tal modo que são mapeados em posições de um vetor binário (filtro de Bloom). Por conseguinte, dadas duas MBRs que se intersectam é realizada a operação AND sobre esses vetores de cada pares de discos. Se o resultado for igual a um dos dois vetores binários, é provável que este seja subconjunto do outro, necessitando, apenas a partir desse momento, realizar a custosa operação de subconjuntos entre esses discos. Ressalta-se que essa verificação é necessária pois é sabido que o filtro de Bloom pode

gerar falso-positivos, no entanto, garante a inexistência de falso-negativos, ou seja, se após essa operação AND sobre os dois vetores o resultando não foi igual a nenhum dos dois, é garantido que não se tratam de um disco subconjunto com outro.

Por último, resta o processamento de junção entre os discos encontrados em t_i com aqueles no tempo imediatamente anterior. Para isso, o PSI utiliza a estrutura de um índice invertido (*inverted index*) para saber exatamente com quais discos em t_{i-1} um disco em t_i precisa realizar essa operação, em vez de realizar a operação custosa de junção de todos os discos encontrados com os candidatos do tempo anterior como no BFE. O índice invertido, então, guarda em sua estrutura os identificadores de cada MOs (OIDs) e aponta para a lista de quais discos em t_{i-1} eles se encontram. Assim, dado um disco em t_i, é realizada a busca para saber em quais discos seus MOs estão nessa estrutura, retornando uma lista de discos de cada MO. Os discos que aparecerem ao menos μ vezes dentre essas listas são os discos a serem feitos a operação de junção entre os instantes de tempo consecutivos.

2.3.1.3 *LCMFlock*

Uma outra modalidade de algoritmo, mas estritamente aplicados a conjunto de dados off-line, são os que utilizam a mineração de itens frequentes, como em [36, 37, 10, 38]. Exemplificando essa abordagem, Turdukulov et al.[37] apresentam o algoritmo $LCMFlock$. O *framework* proposto para essa implementação consiste em quatro etapas:

a) identificação de todos os discos/agrupamentos possíveis em cada instante de tempo;

b) construção de uma versão transacional dos dados de trajetórias baseada nos discos visitados por cada trajetória;

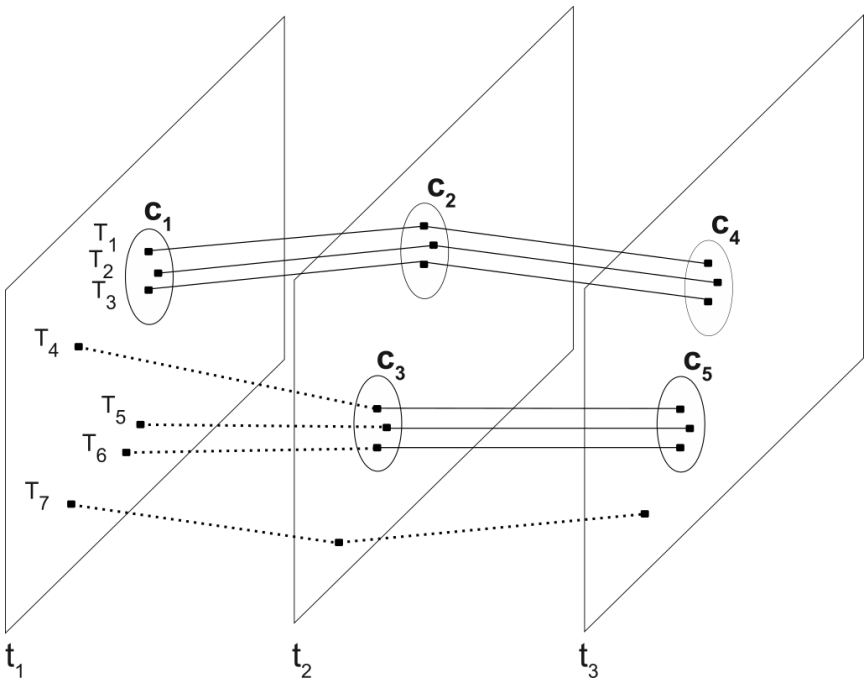
c) aplicação de um algoritmo de mineração de padrões frequentes sobre a versão transacional; e

d) realização de pós-processamento para averiguação da consecutividade temporal dos discos, a poda de duplicados e retorno dos *flocks* encontrados.

Para a realização do Item a) os autores propõem a utilização da primeira etapa do algoritmo BFE — Subsubseção 2.3.1.1 — por demonstrar resultados satisfatórios em termos de desempenho nos experimentos realizados com grandes conjuntos de dados.

Já para a sequência do *framework* com a mineração de padrões frequentes, é necessário, primeiramente, o mapeamento dos dados das trajetórias e de seus respectivos discos — Item b). A Figura 7 demonstra um exemplo desse mapeamento. A Figura 7b apresenta a versão transacional proposta dos dados da Figura 7a. Como para os algoritmos de mineração de padrões frequentes a entrada é no formato $\{TID : itemset\}$, onde TID é o identificador da transação e $itemset$ os identificadores dos itens, Turdukulov et

al.[37] propõem utilizar o OID dos MOs (trajetórias) como identificador das transações e um identificador único para cada disco encontrado para os *itemsets*.



(a) Exemplo de dataset de trajetórias.

TID	ID dos discos
T_1	$\langle C_1, C_2, C_4 \rangle$
T_2	$\langle C_1, C_2, C_4 \rangle$
T_3	$\langle C_1, C_2, C_4 \rangle$
T_4	$\langle C_3, C_5 \rangle$
T_5	$\langle C_3, C_5 \rangle$
T_6	$\langle C_3, C_5 \rangle$
T_7	$\emptyset$

(b) Versão transacional das trajetórias.

Figura 7 – Mapeamento das trajetórias em versão transacional. (Fonte: 5)

Esses dados são, então, seguindo o Item c), exportados para um algoritmo de mineração de padrões frequentes, no caso o LCM (*Linear time Closed itemset Miner*) [39, 40, 41], com o suporte sendo igual ao tamanho mínimo de um *flock* (μ). Este algoritmo foi escolhido por ter demonstrado mais eficiência com parâmetros de entrada similares àqueles utilizados para encontrar *flocks*. Vale ressaltar que o LCM apresenta duas variações, uma para identificar padrões frequentes maximais e outra para encontrar os padrões frequentes fechados, tal que o *framework* possibilita utilizar o mais adequado ao tipo de resposta esperada e domínio do problema.

Como última etapa — Item d) — o *LCMFlock* realiza pós-processamento sobre os resultados do LCM para filtrar apenas os padrões frequentes que estejam em instantes

de tempo consecutivos e que não sejam duplicados, retornando assim os padrões $flock$ identificados.

2.4 Padrão Comboio

O Padrão Comboio (*Convoy Pattern*) [14, 7], ou simplesmente Consulta de Comboio (*Convoy Query*), também retorna grupos de objetos que se movimentam juntos por um intervalo de tempo. Porém, diferentemente do Padrão *Flock*, em que a distribuição espacial de um grupo é definida por um disco, o Padrão Comboio define grupos a partir de objetos móveis agrupados por densidade. A principal definição de agrupamento baseado em densidade é dada pelo algoritmo DBSCAN [42].

O algoritmo DBSCAN (*Density-Based Spatial Clustering of Applications with Noise*) [42], baseia-se no conceito de densidade para identificar agrupamentos. A ideia principal por trás dessa densidade é que para cada ponto em um agrupamento, dado um raio ε (Eps), deveria conter ao menos um número mínimo de pontos $MinPts$ (μ) em sua vizinhança — *ε-neighborhood* (Definição 5) — com qualquer que seja a função de distância *dist* adotada. Entretanto, essa abordagem falha porque um agrupamento possui dois tipos de pontos: pontos *core*, presentes no interior de uma região densa, com pelo menos μ pontos em um raio ε ao redor do objeto; e pontos *border*, que estão na fronteira de regiões densas, na vizinhança de um ponto *core*, mas não sendo um. Assim, pontos de fronteira contêm menos pontos em sua vizinhança que um *core*. Os conceitos de densidade do DBSCAN são definidos a seguir.

Definição 5 *Dado um conjunto de dados D, a vizinhança ε (ε-neighborhood) de um ponto p, denotado por $N_\varepsilon(p)$, é definida por $N_\varepsilon(p) = \{q \in D \mid dist(p, q) \leq \varepsilon\}$.*

Definição 6 *Um ponto p é diretamente alcançável pela densidade (directly density-reachable) a partir de outro ponto q, respeitando μ e ε se:*

1. *$p \in N_\varepsilon(q)$ e*
2. *$|N_\varepsilon(q)| \geq \mu$ (ponto core).*

Definição 7 *Um ponto p é alcançável por densidade (density-reachable) se a partir de um ponto q, respeitando ε e μ, existir uma cadeia de pontos $p_1, \ldots, p_n$, $p_1 = q, p_n = p$ tal que p_{i+1} é diretamente alcançável por densidade a partir de p_i, $1 \leq i < n$.*

Definição 8 *Um ponto p é conectado por densidade (density-connected) a um ponto q, respeitando ε e μ, se existir um ponto p_j tal que ambos p e q sejam alcançáveis por densidade a partir de p_j.*

Definição 9 *Seja D um banco de dados de pontos. Um agrupamento C, respeitado ε e μ, é um subconjunto não vazio de D que satisfaz duas condições:*

1. *$\forall p, q$: se $p \in C$ e q é alcançável por densidade a partir de p, então $q \in C$.*
2. *$\forall p, q \in C$: p é conectado por densidade com q.*

Dessa forma, para se formar um agrupamento C por densidade, faz-se necessário que para cada ponto p em C tenha um ponto q, também no agrupamento C, tal que p esteja na vizinhança ε de q e que $N_\varepsilon(q)$ contenha ao menos μ pontos, conforme as definições dadas e representadas na Figura 8.

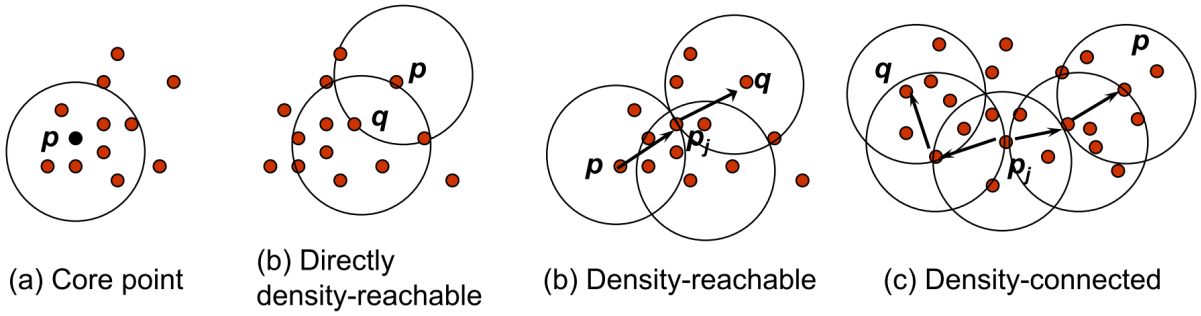


Figura 8 – Principais conceitos do DBSCAN. (Fonte: 6)

A proposta deste Padrão Comboio lida com limitações para algumas aplicações da busca baseada em disco para a identificação de padrões de objetos se movendo juntos por um período. Jeung et al.[7] apresentam estas limitações como o *problema da perda no Padrão Flock* (*lossy-flock problem*), destacando dois problemas principais:

- a sensibilidade do padrão quanto à escolha do tamanho do disco porque diferentes diâmetros sobre o mesmo conjunto de dados impactam significativamente nos resultados, tanto que a tarefa de encontrar objetos intuitivamente pertencentes ao mesmo grupo torna-se difícil com um disco de tamanho fixo para todos os intervalos de tempo;
- e a questão de que talvez o formato circular não seja apropriado para a identificação desses tipos de padrões, já que grupos de objetos de diferentes natureza podem se mover em formatos diferentes e dinâmicos.

A Figura 9 exemplifica um desses problemas. Na Figura 9(a), é visível que os quatro objetos $\{o_1, o_2, o_3, o_4\}$ estão se movendo em grupo. No entanto, conforme demonstra a Figura 9(b), o tamanho do disco adotado não é suficiente para englobar todos os objetos, não incluindo o_4 como possível membro do $flock$.

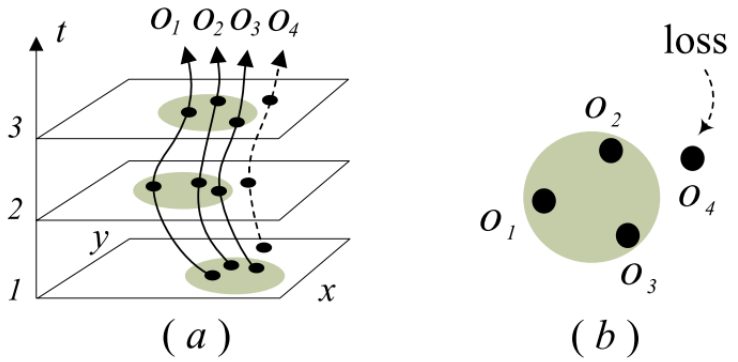


Figura 9 – Problema do *lossy-flock*. (Fonte: 7)

Dessa forma, para evitar as restrições de tamanho e formato dos padrões a serem descobertos Jeung, Shen e Zhou; Jeung et al.[14, 7] propõem utilizar o agrupamento por densidade para a formação de grupos. O Padrão Comboio é formalizado[1] conforme a Definição 10 [7].

Definição 10 *Dados um conjunto de trajetórias $\mathcal{T}$, uma distância limite $\varepsilon > 0$, $\mu > 1$ ($\mu \in \mathbb{N}$) e um tempo de vida $\delta > 1$ ($\delta \in \mathbb{N}$), a consulta de comboio retorna todos os grupos maximais de trajetórias, tal que cada grupo consiste em um conjunto de objetos densamente conectados respeitando ε e μ por pelo menos δ instantes de tempo consecutivos.*

2.4.1 Principais Algoritmos para Detecção do Padrão Comboio

A fim de extrair esses comboios Jeung et al.[7] definem quatro algoritmos offline: CMC (*Coherent Moving Cluster*) e três variações do CuTS (*Convoy Discovery using Trajectory Simplification*)).

O primeiro, CMC, baseia-se na abordagem de *Moving Cluster* (Seção 2.5), similar ao proposto por Kalnis, Mamoulis e Bakiras[43]. Após realizar interpolação linear nas trajetórias com lacunas é aplicado o algoritmo de agrupamento DBSCAN para cada tempo. No segundo momento, os agrupamentos encontrados são fundidos com os do tempo anterior. Nesse ponto, diferentemente do que é feito em [43], a regra mais restritiva a ser atendida, em vez de verificar se há apenas um percentual de itens em comum, analisa se há ao menos μ objetos em comum entre os agrupamentos, visto que no comboio os objetos precisam ser necessariamente os mesmos em toda sua duração. Assim, sendo $c_j^{t_i}$ um agrupamento encontrado no tempo t_i, a condição para identificação de comboios de μ objetos de pelo menos δ instantes de tempo é dado por $|c_j^{t_i} \cap c_j^{t_{i+1}} \cap \ldots \cap c_j^{t_{i+\delta-1}}| \geq \mu$.

[1] Algumas variáveis utilizadas neste texto diferem das variáveis usadas nas referências originais por questões de uniformização.

O segundo tipo de algoritmo proposto é o CuTS e suas variações. A intenção dessa família de algoritmos é a redução do alto custo computacional do CMC, já que este gera posições para todas as lacunas das trajetórias, por meio de interpolação linear, e ainda executa agrupamento custoso em todos os instantes de tempo. Ao contrário do CMC, os algoritmos CuTS realizam a simplificação das trajetórias no primeiro momento, e não a interpolação de suas lacunas. Esses algoritmos, então, realizam muito menos operações de agrupamento sobre essas trajetórias simplificadas, resultando em comboios candidatos. Como último passo, CuTS realiza o processo de agrupamento nas trajetórias originais desses candidatos, para que seja garantido o retorno do resultado correto.

Vale ressaltar que a família de algoritmos CuTS, para a simplificação das trajetórias, utiliza o algoritmo *Douglas-Peucker* (DP) e suas variações. O algoritmo DP requer um parâmetro de tolerância utilizado para descartar os pontos que estão dentro desse limite. Conforme experimentos em [7], o algoritmo mostrou-se eficiente para valores inferiores a ε, porque embora quanto maior o valor para este parâmetro maior poder de simplificação das trajetórias, valores altos deterioram a busca por abrangência no agrupamento (DBSCAN).

2.5 Monitoramento Contínuo de Agrupamentos — *Moving Clusters*

Uma outra linha de abordagem relacionada a padrões de comovimento são os *Moving Clusters*. *Moving Cluster* é definido por um conjunto de objetos (MOs) que se movimentam próximos entre si por um intervalo de tempo. Porém, diferentemente de padrões de comovimento, um *moving cluster* encontrado possui a mesma identificação por toda sua existência embora possa conter modificações completas em sua constituição, ao comparar seu início ao seu fim. Um bom exemplo disso é a migração de animais silvestres de tal forma que esse bando é dinâmico, com a saída e volta de animais a este grupo, inclusive a aparição de outros novos durante o percurso. Assim, esse grupo inicial identificado pode chegar ao fim da migração com animais completamente diferentes, mesmo tendo sido identificado como o mesmo grupo em migração (*moving cluster*) [43].

Kalnis, Mamoulis e Bakiras[43] formalizam a definição de *Moving Cluster*, baseada na premissa de que esse agrupamento se mantém denso — Seção 2.4 — por toda sua duração, mesmo que este termine com MOs completamente distintos em relação ao agrupamento inicial. Desse modo, conforme Definição 11, podendo utilizar o DBSCAN como algoritmo de agrupamento para cada instante de tempo, é considerado um *Moving Cluster* o agrupamento que, em instantes de tempo consecutivos, respeite um limiar mínimo θ (medida de Jaccard) de MOs em comum.

Definição 11 *Seja* $g = <c_1, c_2, \ldots, c_k>$ *a sequência de agrupamentos encontrados tal que para cada* $i\,(1 \leq i < k)$*, o instante de tempo de* c_i *é exatamente o instante de tempo anterior a* c_{i+1}*. Então* g *é um Moving Cluster, respeitando a condição do limiar de* $\theta\,(0 < \theta \leq 1)$*, se* $\frac{|c_i \cap c_{i+1}|}{|c_i \cup c_{i+1}|} \geq \theta, \forall i : 1 \leq i < k$*.*

Como essa abordagem requer a execução de um algoritmo de agrupamento, como o DBSCAN, para todos os pontos em todos instantes de tempo, acaba tendo seu desempenho severamente degradado. A fim de melhorar esse desempenho diversos trabalhos propõem melhorias para a identificação de *Moving Clusters* em especial adotando a estratégia de diminuir ao máximo essa necessidade de se realizar agrupamento a cada instante de tempos [26, 17]. Isso é feito utilizando informações adicionais dos MOs, como velocidade e direção, para que sejam previstos impactos em agrupamentos sendo acompanhados, de tal forma que algoritmos de agrupamento sejam utilizados essencialmente quando necessário e sobre o menor número de MOs possíveis.

3 PADRÕES DE COMOVIMENTO COM FLEXIBILIZAÇÃO DE PARÂMETROS

Este capítulo apresenta alguns dos mais conhecidos padrões de comovimento de objetos móveis e algumas variações do próprio Padrão *flock*, além de salientar suas flexibilizações. Por fim, é demonstrada uma abordagem para reportar os *top-k flocks* e um estudo sobre o impacto da parametrização sobre algoritmos de detecção de *flocks* e métricas de qualidade.

3.1 Padrões com Flexibilização do Parâmetro de Duração

A flexibilização do parâmetro de duração temporal de um padrão de comovimento tem por objetivo detectar padrões mesmo que algumas trajetórias se distanciem temporariamente do grupo ou mesmo por falta de pontos de localização — lacunas nas trajetórias. Este tipo de situação, com lacunas temporais, resultaria no descarte de padrões *Flock* e Comboio, pois exigem a consecutividade espaço-temporal do grupo, ou seja, que os objetos móveis precisam se mover estritamente juntos, limitados por disco ou por densidade, por pelo menos δ instantes de tempo consecutivos. Já os padrões descritos nesta seção propõem flexibilizações da consecutividade espaço-temporal com o intuito de retornar esses grupos que possam ser de interesse para algumas situações.

3.1.1 Padrão *Flock* com Grau de Liberdade

Uma extensão da definição do Padrão *Flock* encontrada na literatura é o Padrão *Flock* em Movimento e com Liberdade (*Freedom Moving Flock Pattern*) Cao, Zhu e Gao[15]. A justificativa para esse novo padrão se fundamenta, basicamente, na necessidade em conseguir extrair $flocks$ de pessoas se movendo juntas. É sabido que pedestres, dadas as exceções, não costumam apresentar trajetórias comportadas, restritas a caminhos bem definidos, como as de carros em ruas, trens, metrôs e até mesmo no deslocamento de bandos de animais como aves, que seguem juntos em um formato bem definido.

A Figura 3 esboça esse problema da perda em um *flock*. Seguindo a Definição 4, com $\mu = 3$ e $\delta = 3$, apenas o *flock* com as trajetórias $\{T_1, T_2, T_3\}$ no intervalo de tempo $I = [t_1, t_3]$ é encontrado. No entanto, embora a pessoa (trajetória) T_7 tenha aparentemente apenas encontrado com o grupo, participando apenas no disco no instante t_3, é visível um outro *flock* — discos tracejados — com trajetórias $\{T_4, T_5, T_6\}$ no intervalo $I = [t_2, t_4]$, considerando uma "liberdade", tal que a pessoal T_6 tenha se afastado do agrupo apenas em um instante de tempo (t_3).

Desse modo, Cao, Zhu e Gao[15] propõem que as trajetórias possam ter uma

liberdade aceitável, ou seja, por até quantos instantes de tempo podem estar desconectados do grupo. Para isso, o grau de liberdade é definido conforme Definição 12.

Definição 12 *Dado um flock f no intervalo de tempo I, o Grau de Liberdade (Degree of Freedom) da trajetória T_i do f, denotado como DOF_i, é dado pela proporção de I^i_{join} para I, conforme Equação 3.1, onde I^i_{join} é o total de tempo em que T_i está contida nos discos de f.*

$$DOF_i = \frac{I^i_{join}}{I} \tag{3.1}$$

Assim, Cao, Zhu e Gao[15], utilizando a Definição 14 de Extensão Espacial $ext(F, I)$ de Wachowicz et al.[8], estabelecem um *Freedom Moving Flock Pattern* de acordo com a Definição 13.

Definição 13 *Dado um conjunto de trajetórias $\mathcal{T}$, um número mínimo de trajetórias $\mu > 1\,(\mu \in \mathbb{N})$, uma distância mínima $\epsilon > 0$ definida sobre uma função de distância d, uma duração mínima $\delta > 1\,(\delta \in \mathbb{N})$ e um grau de liberdade mínimo DOF_{min}, um* ***Padrão de Flock em Movimento e com Liberdade*** *no intervalo de tempo I, denotado como $Flock_{free}(\mu, \epsilon, \delta, DOF_{min})$, reporta todos os flocks $\mathcal{F}$ onde: (1) para cada $f_k \in \mathcal{F}$, o número de trajetórias é maior ou igual a $\mu\,(|f_k| \geq \mu)$, (2) existem δ instantes de tempo consecutivos $(t_j, \ldots, t_{j+\delta-1}) \subset I$ onde há um disco de diâmetro ϵ que cobre boa parte das trajetórias de $f_k^{t_i}$ (flock f_k no tempo t_i), (3) para cada trajetória $T_i \in f_k$, $DOF_i \geq DOF_{min}$, (4) e com extensão espacial $ext(I) \geq \epsilon$.*

O grau de liberdade mínimo de uma trajetória DOF_{min}, então, reflete o tempo mínimo que esta precisa estar presente durante duração do *flock*. Com um $DOF_{min} = 1$, por exemplo, permanece a restrição espaço-temporal, sendo apenas um *Moving Flock* [8]. Wachowicz et al.[8] ao trabalharem com o Padrão *Flock* na identificação de pedestres se movendo juntos apresentam, então, esse conceito de *Moving Flock*. Este estabelece mecanismo para ignorar *flocks* chamados estacionários (*stationary*), ou seja, que não se movimentam o suficiente em um intervalo mínimo de tempo.

Primeiro, portanto, os autores definem o que seria uma distância espacial mínima para diferenciar um grupo estacionário, que pode ser descartado, de um em movimento, conforme Definição 14.

Definição 14 *Dado um flock f encontrado conforme Definição 4, em um intervalo de tempo I, sua extensão espacial (extent spatial), $ext(f, I)$, é definida como $ext(f, I) = max\{l, w\}$, onde l e w são o comprimento e a largura, respectivamente, do retângulo de limite mínimo (MBR) da trajetória de menor extensão do conjunto de subtrajetórias pertencentes ao $flock$.*

Como defendem Wachowicz et al.[8], outra técnica para calcular essa extensão espacial poderia ser escolhida em detrimento do cálculo da MBR. Dessa forma, é possível definir um *Moving Flock* conforme Definição 15.

Definição 15 *Um Moving Flock f_m em um intervalo de tempo $|I| \geq \delta$ é um Flock tal que $ext(f_m, I) \geq \epsilon/2$.*

A Figura 10 apresenta dois exemplos, um *Moving Flock* f_m e um *Stationary Flock* f_s que seria então descartado, considerando $ext(f_m, I) \geq \epsilon/2$. f_m, como pode ser visto, possui sua extensão espacial maior que o raio dos discos, o que não acontece com f_s. Ressalta-se, entretanto, que enquanto Wachowicz et al.[8] configuram o limite da extensão igual ao raio do disco ($\epsilon/2$), Cao, Zhu e Gao[15] utilizam o diâmetro (ϵ) para esse limiar.

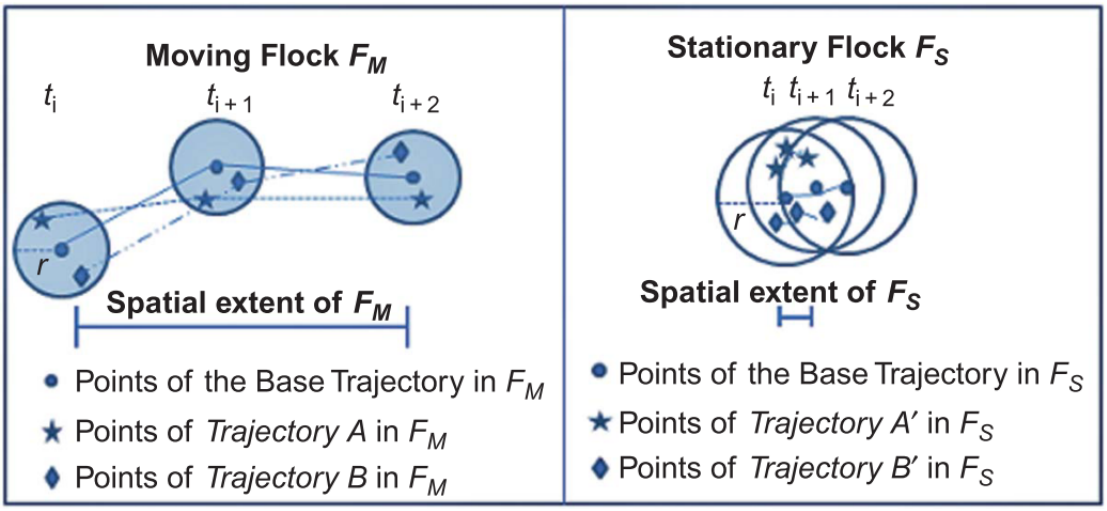


Figura 10 – Exemplos de um *Moving Flock* f_m e um *Stationary Flock* f_s. (Fonte: 8)

3.1.2 Padrão Enxame

Partindo também do entendimento que os padrões de comovimento de objetos móveis, até então, eram restritivos demais, essencialmente em relação à consecutividade temporal dos agrupamentos, Li et al.[9] propõem um novo conceito, o Padrão Enxame (*Swarm Pattern*). Este, por sua vez, defende que esses grupos podem se dispersar no decorrer do tempo — relaxamento temporal —, convergindo novamente para agrupamentos em determinados instantes de tempo. Os autores além de possibilitar encontrar padrões de grupos temporalmente não consecutivos propõem encontrar agrupamentos de formatos conforme domínio do problema, não restringindo a técnica de agrupamento em distância Euclidiana ou por densidade, por exemplo, deixando essa decisão a cargo do usuário.

Seja $O_{DB} = \{o_1, o_2, \ldots, o_n\}$ o conjunto de todos os objetos móveis de um conjunto de dados DB, e $T_{DB} = \{t_1, t_2, \ldots, t_m\}$ o conjunto de todos os instantes de tempo também

de DB. Dado O um subconjunto $O \subseteq O_{DB}$ e T analogamente $T \subseteq T_{DB}$, denota-se $|O|$ e $|T|$ seus tamanhos, o número de objetos e a quantidade de instantes de tempo respectivamente. Pode-se, então, formalmente definir o Padrão Enxame conforme Definição 16.

Definição 16 *Dados min_o e min_t dois limiares mínimos, um par (O, T) é considerado um enxame se todos os objetos de O estão no mesmo agrupamento em todos os instantes de tempo em T, desde que sejam respeitadas as seguintes condições:*

1. *$|O| \geq min_o$: ao menos min_o objetos participantes;*
2. *$|T| \geq min_t$: objetos de O no mesmo agrupamento em pelo menos min_t instantes de tempo;*
3. *e que haja pelo menos um agrupamento contendo todos os objetos de O para cada instante de tempo $t_i \in T$.*

Como essa definição permite a identificação de padrões redundantes, Li et al.[9] propõem um algoritmo off-line, chamado *ObjectGrowth* — implementado no sistema *MoveMine* [25] —, para eficientemente retornar os enxames fechados (*closed swarm*). Visto que esse algoritmo se baseia no conceito de mineração de conjuntos de itens (*itemsets*) frequentes, um enxame fechado segue a mesma ideia de um *itemset* fechado.

Entretanto, o padrão enxame (O, T) é fechado se e somente se for fechado na quantidade de objetos (*object-closed*) e fechado na quantidade de tempos (*time-closed*), conforme a seguir:

- é *object-closed* se fixando T, O não puder ser aumentado ($O' \supsetneq O, \nexists(O', T)$);
- é *time-closed* se fixando O, T não puder ser aumentando ($T' \supsetneq T, \nexists(O, T')$).

A Figura 11 exemplifica, para pelo menos dois objetos em três instantes de tempo, a identificação de enxames, ao mesmo tempo que nenhum comboio, *flock* ou *moving cluster* é identificado. Isso ocorre porque mesmo considerando um alto percentual de objetos em comum nos agrupamentos entre tempos consecutivos, a restrição da consecutividade temporal impede de esses outros padrões sejam identificados. Conforme imagem, dados $O = \{o_1, o_2, o_3, o_4\}$ e $T = \{t_1, t_2, t_3, t_4\}$, os seguintes enxames são identificados: $(\{o_1, o_3, o_4\}, \{t_1, t_3, t_4\})$, $(\{o_1, o_3\}, \{t_1, t_3, t_4\})$, $(\{o_1, o_4\}, \{t_1, t_3, t_4\})$, $(\{o_2, o_3\}, \{t_1, t_2, t_4\})$ e $(\{o_3, o_4\}, \{t_1, t_3, t_4\})$. Utilizando a definição de Enxames Fechados, temos apenas: $(\{o_1, o_3, o_4\}, \{t_1, t_3, t_4\})$ e $(\{o_2, o_3\}, \{t_1, t_2, t_4\})$.

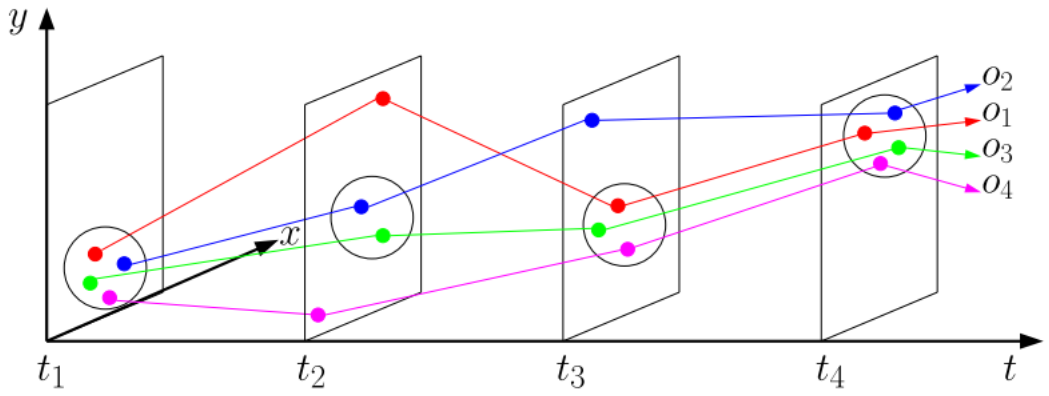


Figura 11 – Exemplo de detecção de enxames e não de outros padrões. (Fonte: 9)

3.1.3 Padrão Pelotão

Outro padrão de comovimento de objetos móveis existente na literatura é o Padrão Pelotão (*Platoon Pattern*). Proposto por Li, Bailey e Kulik[10], esse novo conceito também aborda a questão da restrição temporal muito limitante dos demais padrões: a consecutividade temporal, que, como já abordado, acarreta a perda de padrões importantes. Entretanto, no outro lado desse problema, o Padrão Pelotão busca, também, reduzir a descoberta de padrões ruidosos, não significativos, proporcionados pelo relaxamento excessivo do tempo que acontece no Padrão Enxame — Subseção 3.1.2.

Os impactos da restrição da consecutividade temporal são exemplificados na Figura 12. Em (a) assume-se um grupo de quatro carros se movendo juntos, $\{o_1, o_2, o_3, o_4\}$, que acabam se desconectando no tempo t_3 por causa de um semáforo, juntando-se novamente nos próximos instantes. Supondo um $k = 3$ (consecutividade mínima temporal), nenhum *flock* ou comboio de quatro objetos é identificado em virtude desta quebra temporal. No entanto, conforme será apresentado mais à frente, um pelotão é identificado se aceitar consecutividade temporal local de apenas dois instantes de tempo.

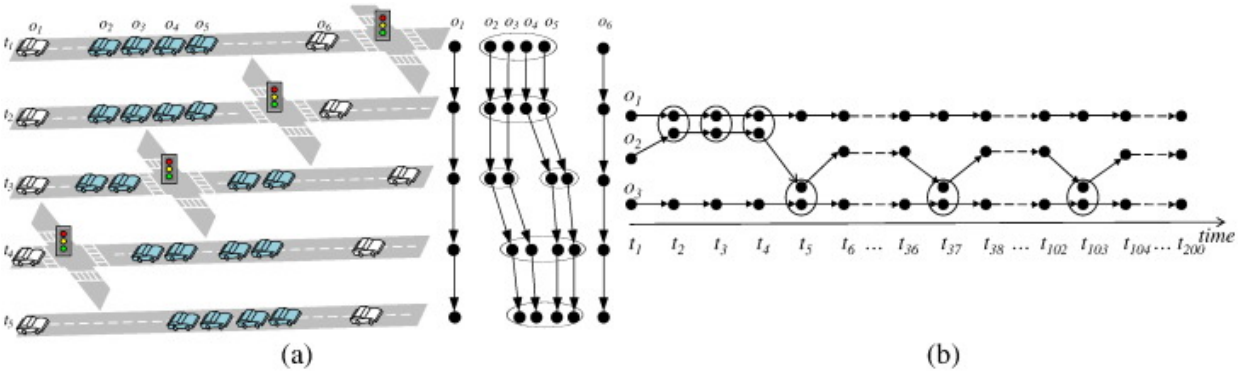


Figura 12 – (a) Exemplo de um Pelotão em detrimento de nenhum *Flock* ou Comboio identificado. (b) Enxame $(\{o_2, o_3\}, \{t_5, t_{37}, t_{103}\})$ identificado, com $k = 3$, mesmo com agrupamentos tão isolados e esparsos no tempo. (Fonte: 10)

Já na Figura 12(b), fica nítido o impacto negativo em um relaxamento total da consecutividade temporal. Para grupos de apenas dois objetos, com um $k = 3$, encontra-se um enxame $(\{o_2, o_3\}, \{t_5, t_{37}, t_{103}\})$. Mas pela distância entre os tempos, certamente trata-se de um grupo de pouca significância no contexto de objetos se movendo juntos. Aumentar k poderia ser uma possibilidade, mas também teria o impacto de restringir a identificação de padrões interessantes, porém, de menores comprimentos. É, portanto, buscando atacar esses impactos que Li, Bailey e Kulik[10] propõem o Padrão Pelotão.

Sejam O_{DB} e T_{DB} os conjuntos de objetos móveis e de instantes de tempo no conjunto de dados DB respectivamente. O um subconjunto de objetos ($O \subseteq O_{DB}$) que se apresentam em agrupamentos nos instantes de tempo T ($T \subseteq T_{DB}$), não necessariamente consecutivos, formam um agrupamento temporal de objetos, andando juntos numa sequência temporal T, denotado por $C = (O : T)$. Dado um limiar mínimo de número de objetos min_o, $C = (O : T)$ é dito *significante* se $|O| \geq min_o$. Dado um número mínimo de instantes de tempo min_t e um número mínimo de instantes de tempo consecutivos min_c, $C = (O : T)$ é *frequente* se $|T| \geq min_t$. C também é dito *localmente consecutivo* se T se decompor em segmentos consecutivos, cada um deles com comprimento de pelo menos min_c. Também, $C = (O : T)$ é *globalmente consecutivo* se todos os seus segmentos tiverem pelo menos min_t de comprimento. Neste último caso, com $min_c = min_t$, a ideia é equivalente à adotada no Padrão Grupo, na Definição 19, de segmentos válidos. Assim, Li, Bailey e Kulik[10] definem esse novo conceito conforme Definição 17.

Definição 17 *Um Padrão Pelotão é um agrupamento temporal de objetos $C = (O : T)$ que é significante, frequente e localmente consecutivo.*

O algoritmo *PlatoonMiner* proposto pelos autores, almejando identificar os resultados mais importantes, sem redundâncias e também buscando desempenho, retorna apenas os Pelotões Fechados (*Closed Platoon*). Para ser um Pelotão Fechado, conforme Definição 18, faz-se necessário ser ao mesmo tempo maximal no número de objetos (*object-maximal*) e maximal no comprimento (*time-maximal*), analogamente aos conceitos *object-closed* e *time-closed*, respectivamente, de um Enxame Fechado — Subseção 3.1.2.

Definição 18 *Um Padrão Pelotão $C = (O : T)$ é fechado se e somente se C for tanto object-maximal e time-maximal.*

Finalizando, mais dois pontos importantes a respeito do Padrão Pelotão apresentado por Li, Bailey e Kulik[10]. Primeiro, ele também é independente da técnica de agrupamento assim como o Padrão Enxame de Li et al.[9], ou seja, pode ser adotado tanto um algoritmo baseado em densidade quanto um baseado em disco na identificação dos agrupamentos. O segundo ponto é que embora uma alternativa com a utilização do

DBSCAN fosse apenas adicionar um processo a mais de refinamento do algoritmo de identificação de Enxames, filtrando apenas aqueles que respeitem a restrição de consecutividade local, os autores demonstraram que isso seria um processo custoso, sendo o *PlatoonMiner* mais eficiente.

3.2 Padrão com Flexibilização do Parâmetro de Tamanho do Grupo

Outro padrão encontrado na literatura interessado em descobrir grupos de usuários espacialmente próximos por um significante montante de tempo é o Padrão Grupo (*Group Pattern*). Este padrão descrito em Wang, Lim e Hwang; Wang, Lim e Hwang[44, 16] difere do Padrão *Flock* basicamente em dois fatores:

- não requer um parâmetro de quantidade mínima de objetos para formar o grupo, como o μ da Definição 4 do Padrão *Flock*, reportando a partir de dois objetos;
- é possível que o grupo se disperse em determinados momentos, mantendo o mesmo grupo, desde que respeitado um cálculo de proporção entre os segmentos válidos (Definição 19) e o comprimento temporal total.

Um padrão de grupo P é encontrado se existir ao menos um segmento válido em um grupo de usuários G, respeitando uma distância máxima max_dis entre esses objetos em uma janela temporal de duração mínima min_dur conforme Definições 19 e 20.

Definição 19 *Dado um grupo G de objetos, uma distância máxima max_dis, uma duração mínima min_dur, uma sequência temporal $[t, t+k]$ é denominado um* ***segmento válido*** *de G se:*

1. *Todos os objetos em G não estão mais distantes entre si que max_dis nos tempos $t, t+1, \ldots, t+k$;*
2. *Alguns objetos em G estão mais distantes que max_dis no tempo $t-1$;*
3. *Alguns objetos em G estão mais distantes que max_dis em $t+k(k+1)$;*
4. *E $(k+1) \geq min\_dur$*

Ou seja, vai existir um segmento válido se os objetos em G permanecerem próximos um dos outros pela duração mínima min_dur, sendo sempre maximal, não existindo outros segmentos contidos neste.

Definição 20 *Dado o conjunto de objetos G, os limiares max_dis e min_dur, um padrão grupo, denotado por P = <G, max_dis, mi_dur>, se G tiver ao menos um segmento válido.*

Esses padrões encontrados, sempre com segmentos dos mesmos k objetos em G, também são conhecidos como *padrões k-grupo*. Na existência de diferentes segmentos em P, os autores validam este como sendo um padrão de grupo válido ou não se for respeitada uma proporção mínima — *weight(P)* Equação 3.3 — entre o comprimento combinado dos segmentos $s_1, \ldots, s_n$ — *weight-count(P)* Equação 3.2 — pela duração total N.

$$\textit{weight-count}(P) = \sum_{i=1}^{n} |s_i| \tag{3.2}$$

$$\textit{weight}(P) = \frac{\textit{weight-count}(P)}{N} = \frac{\sum_{i=1}^{n} |s_i|}{N} \tag{3.3}$$

Como $\textit{weight}(P)$ representa a proporção do quanto os objetos permanecem juntos, quanto maior, mais significativo será o grupo. Assim, o grupo é considerado como um **padrão de grupo válido** se, dado min_wei a proporção mínima, $\textit{weight}(P) \geq min\_wei$. Vale ressaltar que embora Li et al.[17] definam uma Busca de Grupo, trata-se de definições diferentes a aqui apresentada, sendo abordada na Seção 3.3.

3.3 Abordagem On-line para Retornar *Top-k* Grupos em Movimento

Li et al.[17] definem um novo conceito de busca de objetos se movendo juntos, denominado Grupo de Trajetórias, ou simplesmente Consulta de Grupo. Ressalta-se que, como já mencionado na Seção 3.2 — Padrão com Flexibilização do Parâmetro de Tamanho do Grupo —, embora os nomes sejam semelhantes, tratam-se de definições diferentes.

Os autores descrevem como a primeira solução que satisfaz os quatro requisitos defendidos no trabalho, descritos na sequência.

1. **Independência de pontos amostrados**. Diferentes representações da mesma trajetória, ou seja, a utilização de diferentes pontos de localização amostrados do mesmo objeto, não deveria impactar no resultado dos algoritmos. Assim, conforme Definição 21, um algoritmo A satisfaz a independência de pontos amostrados se e somente se $\forall TR_a, TR_b \, (TR_a \equiv TR_b \implies A(TR_a) = A(TR_b))$

2. **Conexão por densidade**. A fim de evitar o problema de perda (*lossy-flock*) discutido por Jeung et al.[7], devido à sensibilidade dos algoritmos baseados em busca

com discos, a proposta de Li et al.[17] utiliza o algoritmo DBSCAN para o agrupamento contínuo como forma de mitigar esse problema.

3. **Processamento on-line**. Suportar processamento on-line enquanto os dados vão chegando.

4. **Aproximação de trajetórias**. E como o processamento on-line de trajetórias brutas requer um processamento considerável, outra importante propriedade é suportar simplificação também on-line destas, buscando acurácia e eficiência.

Definição 21 *Sejam dois conjuntos de pontos de localização amostrados que representam a mesma trajetória, denotado como* $tr_a^{rep} \equiv tr_b^{rep}$. *Duas coleções de conjuntos de pontos de localização amostrados,* TR_0 *e* TR_1, *representando as mesmas trajetórias, denotado como* $TR_0 \equiv TR_1$, *se e somente se* $\forall i \in \{0,1\}\,(\forall tr_a^{rep} \in TR_i\,(\exists tr_b^{rep} \in TR_{1-i}(tr_a^{rep} \equiv tr_b^{rep})))$.

Além de respeitar os quatro requisitos previamente descritos, o algoritmo requer um número $k \geq 1\,(k \in \mathbb{N})$ como entrada para retornar os "*top*" k-grupos mais significativos. Esse ranking é feito pelo algoritmo de acordo com a função de pontuação da Definição 22, baseada na cardinalidade e duração dos grupos processados.

Definição 22 *Dado* C *um conjunto de* N_c *objetos que viajam juntos por um intervalo de tempo* $[t_s, t_e]$. *A pontuação de* C *é dada pela Equação 3.4.*

$$Score(C) = \alpha N_c + (1 - \alpha)\rho \tag{3.4}$$

onde α $(0 \leq \alpha \leq 1)$ é um parâmetro dado pelo usuário e $\rho = (t_e - t_s)$ a duração.

Verifica-se, também pela Equação 3.4, que o parâmetro de peso α adotado permite ao usuário balancear entre o que é mais importante ao padrão sendo minerado: a cardinalidade ou a duração. Com baixos valores de α a duração do grupo torna-se mais importante. Em contrapartida, altos valores privilegiam padrões maiores, de maior cardinalidade.

Outros dois pontos importantes é que a proposta, primeiro para ganho de desempenho, vai podando padrões encontrados que são "dominados" (Definição 23) pelos demais candidatos.

Definição 23 *Dados dois grupos* C_1 *e* C_2, C_1 *domina* C_2, *denotado como* $C_1 \prec C_2$, *se:*

1. C_1 *é um superconjunto de* C_2, *e;*

2. *o intervalo de tempo de* C_1 *contém o de* C_2

Outro ponto, a fim de aumentar a qualidade nos resultados, estabelece que os diferentes padrões encontrados precisam ser mais diversificados entre si — função de similaridade conforme Equação 3.5 — que um dado limite θ fornecido pelo usuário.

$$Sim(C_1, C_2) = \frac{|C_1 \cap C_2|}{|C_1 \cup C_2|} \tag{3.5}$$

Assim, a seguinte regra precisa ser respeitada no conjunto de respostas:

$$\forall C_1, C_2 \in R\,(C_1 \neq C_2 \implies (Sim(C_1, C_2) \leq \theta))$$

Com essas considerações preliminares, a definição de Consulta de Grupo de Li et al.[17] é dada como:

Definição 24 *Dada uma coleção de trajetórias TR, uma distância máxima ε, a cardinalidade mínima m, a duração mínima τ, um inteiro k, e a função de pontuação 3.4, a Consulta de Grupo retorna os k-grupos de objetos, tal que cada grupo:*

1. *é um agrupamento densamente conectado, respeitando ε e m,*
2. *tem uma duração mínima de τ,*
3. *não é dominado por nenhum outro grupo, e*
4. *tem uma pontuação top-k.*

Dada a Definição 24, de um novo padrão de grupo, independente dos pontos amostrados, os autores Li et al.[17] implementaram um algoritmo respeitando-a, com a utilização da abordagem baseada em eventos e de monitoramento contínuo de agrupamentos. Ressalta-se, portanto, que embora tenham definido um novo padrão de comovimento de objetos móveis, sua implementação utiliza a ideia de monitoramento contínuo de agrupamentos (*Moving Clusters* da Seção 2.5).

Sintetizando, o algoritmo funciona da seguinte forma: a cada novo instante de tempo, recebe-se os pontos de localização, que são prontamente processados pelo algoritmo de simplificação de trajetória NOPW (*Normal Opening Window*), depois sendo processados de fato pelo *framework*; o algoritmo on-line, então, vai armazenando os pontos ainda não classificados em grupos (U), a fila de eventos a serem processados ($eventQ$), o histórico dos grupos identificados até então (H), e os *top-k* resultados (R).

Como dito, esse *framework* é baseado em eventos. Estes são divididos em dois grupos: primários e secundários. Os primários são três, denominados *APPEAR*, *DISAPPEAR* e *UPDATE*, captados diretamente da *stream* de dados, que representam, respectivamente,

a aparição de um novo objeto dentre os monitorados; o não mais rastreamento de um objeto; e a chegada de um novo ponto de amostragem de um objeto já sendo monitorado, recebendo sua nova posição e velocidade. Já os eventos secundários, disparados pelos primários, são: *EXIT*, quando um objeto sai de um agrupamento sendo acompanhado; *JOIN*, quando da inserção do objeto em algum agrupamento existente; *EXPIRE*, quando um agrupamento deixa de existir; *MERGE*, quando pelo menos dois agrupamentos se juntam; e *SPLIT*, quando um agrupamento se reparte em dois ou mais agrupamentos.

Os eventos primários, e consequentemente os secundários, são ordenados pelo instante de tempo, adicionados à fila de eventos *eventQ*. Eles são processados apenas quando ocorrem, diferentemente de uma abordagem que espera pontos de localização dos objetos em todos os intervalos de tempo.

Nessa estrutura, dada, por exemplo, o *APPEAR* de novos objetos, verifica-se primeiro se existem nas proximidades desses pelos menos $MinPts$ de algum determinado agrupamento. Caso sim, adiciona-se à *eventQ* um evento *MERGE* do respectivo ponto ao seu agrupamento próximo. Se ainda sobrarem pontos em U, ou seja, não classificados, esses passam pelo algoritmo DBSCAN, respeitando m como $MinPts$ e ε como ε em busca de novos grupos.

Os agrupamentos monitorados são atualizados a cada novo evento referente a objetos neles contidos. Toda vez que acontece um evento que impacta um determinado agrupamento, seus objetos chamados *BORDER* — derivado do conceito de pontos de borda do DBSCAN —, têm seus tempos de saída do agrupamento calculado, adicionando então futuros *EXIT* à *eventQ*. Esses eventos e também quais ainda são os pontos de fronteira e quais são os *CORE* dos agrupamentos, são atualizados a cada processamento no agrupamento.

Sempre que um novo grupo C chegar ao seu fim, é calculado sua pontuação e comparado à pontuação mínima dos *top* k-grupos (R_{score}). Se for maior que o menor dentre os k ($C_{score} < R_{minScore}$), este então entra no grupo R, atualizando a pontuação mínima para a inserção de novos grupos ao resultado ($R_{minScore} = C_{score}$). Dessa forma, os *top* k-grupos de maior pontuação são mantidos em R.

3.4 Abordagens para Tratamento do Parâmetro de Distância

Um ponto importante levantado no trabalho de Wachowicz et al.[8] é justamente a dificuldade em escolher os melhores parâmetros para um conjunto de dados específico, visto que isso depende de diversos fatores do contexto desses dados. Citam, por exemplo, que μ, pelo menos no caso de mineração de $flocks$ de pessoas, depende exclusivamente da vontade do usuário, ou seja, no tamanho do grupo que está querendo ser pesquisado. Já o δ depende se o interesse é em padrões de longa ou de curta duração. Contudo,

como também apontado, o tamanho do disco ϵ é o parâmetro de maior desafio para ser estabelecido, visto que para cada contexto faz-se necessário análises para decidir qual o melhor valor a ser utilizado. Ressalta-se, também, que embora os padrões que utilizem a estratégia de agrupamento por densidade defendam que resolvem a limitação espacial da abordagem de disco, ainda assim requerem que seja predefinido um valor do ε como parâmetro de entrada nos algoritmos.

Como um guia para a escolha de um raio adequado, Wachowicz et al.[8] propuseram uma estratégia baseada na adaptação do DBSCAN, assumindo que um ε seja comparável a um raio, já que o valor de ε separa aquilo que é vizinho do ponto $CORE$ dos pontos ruidosos (*outliers*), distantes. A ideia é, então, calcular a distância do k-ésimo vizinho mais próximo de todos os objetos em todos os instantes de tempo, sendo $k = min\_pts - 1$. Tendo todas essas distâncias, os autores então propuseram ordená-las em ordem não decrescente e plotá-las em um gráfico de linha. Assim, o ponto em que haver um aumento repentino na distância para esse k-ésimo vizinho pode ser considerado um bom valor de raio, de tal modo que a partir desse valor, o raio teria que crescer cada vez mais para alcançar os demais pontos, provavelmente não membros de um *flock*.

3.4.1 Impacto da Parametrização e Métricas de Análise de *Flocks*

Com relação ao problema da parametrização dos algoritmos de agrupamento, o trabalho de Ong et al.[18] realiza uma avaliação empírica dos efeitos dos parâmetros em um algoritmo de identificação de *flocks* ([8]), a fim de delinear uma estimativa de parâmetros orientada a dados. Assim como em [8], Ong et al.[18] ressaltam que as várias definições do *Flock*, como a adotada $Flock(\mu, r, \Delta T, R)$ (sendo r o raio e R a taxa de amostragem), são de grande relevância. No entanto, a utilização desse tipo de algoritmo se torna difícil até mesmo para especialistas do domínio, sendo uma barreira para estes utilizarem tais algoritmos como "caixas-pretas".

A respeito das análises feitas sobre o impacto da variação dos parâmetros, Ong et al.[18] verificaram que a taxa de sincronização dos pontos amostrados R de fato afeta a qualidade dos dados. Valores altos de R podem distorcer as trajetórias, enquanto que pontos amostrados em breves instantes de tempo significam maior custo de processamento. Em suma, maiores valores em R, menos $flocks$ encontrados.

Variações em ΔT apresentam impacto semelhante ao R. Resultados demonstram que quanto maior o intervalo de tempo necessário para formar um *flock*, mais seletiva será a busca. μ, por sua vez, não é um parâmetro crítico, sendo realmente dependente da vontade do usuário a determinação do que constitui um *flock* de tamanho mínimo. Da mesma forma, se o usuário tem o interesse em *flocks* maiores, de menor comprimento, ou nos mais duradouros, porém, de menor granularidade.

Por fim, Ong et al.[18] enfatizam que de fato r, o raio do disco, é o parâmetro mais

crucial, já que seu valor impacta significativamente na quantidade de *flocks* encontrados, assim como nas métricas definidas. E, como forma de estimação de seu valor, a estratégia abordada é a mesma do trabalho [8], com a escolha de um raio onde existir uma mudança repentina no gráfico de linha das distâncias do k-ésimo item.

3.5 Discussão

A mineração de informações em dados espaço-temporais é de grande interesse nos dias atuais. Neste cenário, existem diversos padrões de comovimento que visam formalizar a busca desses grupos de objetos móveis, tentando levar em consideração os comportamentos dinâmicos dos dados reais e, também, propiciar o desenvolvimento de algoritmos eficientes para grandes conjuntos e *streams* de dados. São exemplos desse dinamismo um bando de animais que migram sem um formato predefinido, além de se dispersarem em momentos de riscos; um grupo de pedestres em que uma ou mais pessoas se afastam momentaneamente; e um comboio de carros, dividido temporariamente por semáforos durante seu trajeto.

Após o Padrão *Flock* vários trabalhos surgiram buscando apresentar flexibilizações a este por ser restritivo, principalmente em dois requisitos: exigir um disco de tamanho fixo para acompanhar o grupo e em instantes de tempo consecutivos. Contudo, mesmo trabalhando com algoritmos de agrupamento baseados em densidade como o DBSCAN, conforme identificado na literatura, ainda há a necessidade de se fornecer o parâmetro de restrição espacial (ε) e o próprio diâmetro nas flexibilizações que utilizam a busca por distância/discos. Embora existam propostas de estimativas para encontrar um valor adequado para o parâmetro de restrição espacial (ϵ ou ε), eles ainda permanecem fixos para toda a busca no contexto de dados, não se adequando, muitas vezes, aos comportamentos dinâmicos dos dados reais.

Assim, a alternativa deste trabalho é a utilização de consulta exploratória para identificar a quantidade desejada dos padrões mais importantes, ranqueados de acordo com algum critério pré-definido pelo usuário, e sem que seja necessário especificar um parâmetro não trivial e sensível ao contexto, apresentando definições e algoritmos para essa busca para o Padrão *Flock*. Outra possibilidade apresentada é a detecção dos padrões mais importantes mas desde que não ultrapassem uma restrição, quando de um conhecimento prévio do usuário em relação aos dados/contexto e sua necessidade de consulta. Um exemplo é, dado uma multidão sendo monitorada em uma passeata, o usuário já pré-estabelecer uma restrição para uma busca exploratória, como a de monitorar uma quantidade de grupos que se movam mais juntos, desde que respeitem uma restrição espacial.

Esta abordagem é nova na literatura e visa facilitar a identificação de *flocks* de

maneira uniforme em conjuntos diversos, bem como adaptando-se a variações na distribuição dos objetos móveis no espaço no decorrer do tempo. O próximo capítulo apresenta as variações do Padrão *Flock* e os algoritmos propostos neste trabalho.

4 ALGORITMOS PARA DETECÇÃO ON-LINE DE *FLOCKS* COM PARÂMETRO DE DISTÂNCIA LIVRE

Como já mencionado previamente, definir valores de distância adequados como entrada para algoritmos de mineração de padrões de comovimento requer conhecimento prévio do comportamento dos objetos móveis, que pode variar de acordo com a natureza do MO (pessoas, veículos, animais, eventos climáticos, etc.), com condições de ambiente (malha viária, limitadores de velocidade, semáforos, acidentes, terrenos irregulares, etc), e assim por diante. Essa tarefa é razoavelmente difícil, especialmente para mineração de trajetórias, pois o objetivo é detectar padrões relevantes mas que são desconhecidos e dinâmicos com o passar do tempo. Dessa forma, além da dificuldade em se definir um parâmetro de distância inicial, não é suficiente apenas esse valor fixo para trabalhar com comportamentos dinâmicos de dados reais, demandando até mesmo um ajuste praticamente contínuo.

A contribuição deste trabalho é reduzir este problema a um de monitoramento de um dado número de padrões, identificado segundo condição específica que pode variar com o tempo. A proposta é realizar uma busca exploratória, por meio de uma consulta estilo *top-k* em que são encontrados as k respostas mais relevantes de acordo com um critério de ranqueamento desejado. Essa nova abordagem recebe o nome de **descoberta dos k-padrões de comovimento** e que pode variar conforme o padrão a ser escolhido para a busca, como *flock*, comboio, entre outros; e o parâmetro que define o critério de ranqueamento, como por exemplo a distância e duração de um padrão. A principal vantagem dessa estratégia é deixar de exigir do usuário um parâmetro difícil de se definir e, ao invés disso, apenas o número k de padrões desejados. Uma versão preliminar deste trabalho foi publicado na *21st Conference on Geo-information Science* — AGILE-GIS 2018 [45].

Este capítulo está organizado da seguinte forma. A Seção 4.1 detalha o problema discutido e apresenta os conceitos e novas definições. Na sequência, três algoritmos são apresentados: O Algoritmo *Top-Down* para a Detecção de k_ϵ-*Flocks* na Seção 4.2, Algoritmo *Top-Down* em Janela Deslizante na Seção 4.3 e o Algoritmo *Filtered Top-Down* com Limite Superior na Seção 4.4.

4.1 Definição do Problema

Seguindo a nova abordagem de descoberta dos k-padrões de comovimento, este trabalho foca no Padrão *Flock*, com a distância como condição dinâmica no tempo e, como critério de ranqueamento, a consulta dos *flocks* de tamanho mínimo e com as trajetórias

mais próximas uma das outras, ou seja, com discos de diâmetro mínimo, isso porque presume-se que quanto mais juntos os MOs mais significativos são. Assim, o interesse é fazer o parâmetro de distância um parâmetro livre nessa consulta, ou apenas como um limitador pra filtragem adicional das respostas. Propomos, então, o novo conceito denominado ***k-Flocks* de Diâmetro Mínimo (k_ϵ-*Flocks*)**, que são os k *flocks* em uma janela temporal w tal que o diâmetro do *flock* com o maior diâmetro em k_ϵ-*Flocks* seja o menor possível, ou seja, para qualquer diâmetro menor que esse valor a consulta retorne menos que k *flocks*. Já o Padrão k_ϵ-*Flock* é a consulta dos k_ϵ-*Flocks* em todas as janelas temporais do conjunto de entrada [45]. Todos esses conceitos e definições são formalizados a seguir.

Definição 25 (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 $f_w(\mu, \epsilon') = f_w(\mu, \epsilon)$.*

Definição 26 (*Flock* Mais Extenso) *Dados um conjunto de flocks $\mathcal{F}$, o flock mais extenso nesse conjunto é o flock cujo diâmetro é o maior entre todos os flocks em $\mathcal{F}$. Empates são resolvidos arbitrariamente.*

Definição 27 (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.*

Definição 28 (Padrão k_ϵ-*Flock*) *Um Padrão k_ϵ-Flock(μ, δ) reporta um conjunto de $\mathcal{F}^k$ contendo os k_ϵ-Flocks(μ, w) para cada janela temporal w de comprimento δ válida para o conjunto de trajetórias $\mathcal{T}$, tal que $\mathcal{F}^k = \bigcup_{\forall w} \mathcal{F}_w^k$.*

Definição 29 (*Filtered*-k_ϵ-*Flocks*) *Dados uma janela temporal w, uma distância máxima $\epsilon' > 0$ ($\epsilon' \in \mathbb{R}$), e um número mínimo de trajetórias $\mu > 1$ ($\mu \in \mathbb{N}$), Filtered-k_ϵ-Flocks, representado como Filtered-k_ϵ-Flocks(μ, ϵ', w), é o conjunto $\mathcal{F}_w^{filtered-k}$ contendo k ($k \in \mathbb{N}$) flocks tal que para cada flock $f_w(\mu, \epsilon) \notin \mathcal{F}_w^{filtered-k}$ temos que $\epsilon \geq \epsilon_k$, onde ϵ_k é o menor diâmetro do flock mais extenso em $\mathcal{F}_w^{filtered-k}$, e que $\epsilon_k \leq \epsilon'$.*

Definição 30 (Padrão *Filtered*-k_ϵ-*Flocks*) *Um Padrão Filtered-k_ϵ-Flocks(μ, ϵ', δ) reporta um conjunto de $\mathcal{F}^{filtered-k}$ contendo os Filtered-k_ϵ-Flocks(μ, ϵ', w) para cada janela temporal w de comprimento δ válida para o conjunto de trajetórias $\mathcal{T}$, tal que $\mathcal{F}^{filtered-k} = \bigcup_{\forall w} \mathcal{F}_w^{filtered-k}$.*

A abordagem básica para retornar o Padrão k_ϵ-*Flock* ou o Padrão *Filtered-*k_ϵ*-Flocks* é identificar iterativamente os k_ϵ-*Flocks* ou *Filtered-*k_ϵ*-Flocks* através da estratégia de deslizamento de janela temporal de uma *stream* ou conjunto de dados. Na subseção a seguir são apresentadas alguns conceitos que permitem a criação de um algoritmo *top-down* para a identificação desses padrões;

4.1.1 Propriedades para Identificação de k_ϵ-*Flocks* e *Filtered-*k_ϵ*-Flocks*

A principal propriedade para identificar *flocks* de diâmetro e tamanho mínimo é que um *flock* maior que o tamanho mínimo ($|flock| > \mu$) pode ser dividido em *subflocks* com menos trajetórias. O diâmetro mínimo desses *subflocks* serão menores que o diâmetro do *flock* original. Assim, começando com um único *flock* maximal, contendo todas as trajetórias de uma janela temporal, esse processo de divisão pode ser iterativamente aplicado até convergir aos *flocks* de diâmetro mínimo.

Para apoiar esse premissa a Figura 13 apresenta o número de *flocks* de tamanho maximal encontrados em duas janelas temporais com a redução do diâmetro de busca para dois conjuntos de dados bem conhecidos: pontos de localização de carros no GeoLife e táxis em São Francisco. Na Figura 13a) pode ser observado que o número de *flocks* encontrados aumenta para diâmetros menores até um ponto de saturação e começa a reduzir as respostas para diâmetros menores que esse ponto até que nenhum *flock* seja encontrado. Para a janela temporal do outro conjunto de dados (Figura 13b) o comportamento é similar, no entanto, o número de *flocks* encontrados nem sempre aumenta até o ponto de saturação e nem sempre diminui após ele. Assim, a condição de parada de um algoritmo exploratório como a abordagem *top-down* precisa levar em consideração esse comportamento.

(a) Carros em GeoLife

(b) Táxis em São Francisco

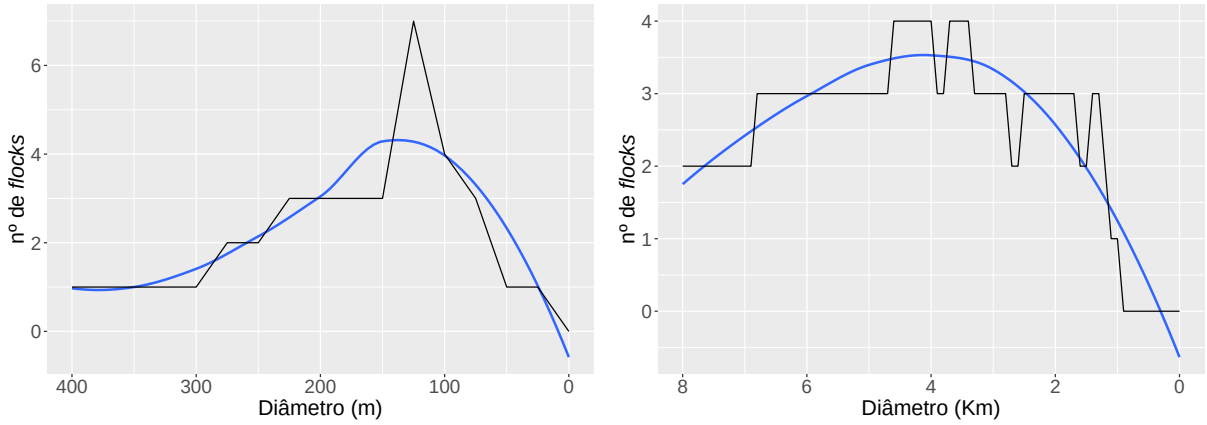


Figura 13 – Comportamento da quantidade de *flocks* de tamanho maximais em uma janela temporal variando ϵ.

Com esse comportamento instável da quantidade de *flocks* maximais em tamanho com a variação do tamanho do diâmetro de busca, uma estratégia simples com chute de diâmetros para identificar uma quantidade específica de resposta torna-se inviável. O Algoritmo 1 apresenta essa ideia. Como entrada a este algoritmo, além dos parâmetros para se achar um *flock* $f_w(\mu, \epsilon)$, possui a quantidade de respostas desejadas k. Assim, ϵ é o chute de um diâmetro inicial aplicado a um algoritmo de mineração de *flocks*, o PSI no caso, sendo os *flocks* retornados armazenados em $\mathcal{F}$ (linha 2). Desse resultado, verifica-se se é a quantidade esperada de respostas (linha 3). Se for, o algoritmo retorna o resultado da busca. Caso contrário, o chute precisa ser refeito com outro diâmetro.

Algoritmo 1: Algoritmo *k-flocks* por Aproximação Iterativa de Diâmetro

Entrada: w: janela temporal contendo o conjunto de trajetórias $\mathcal{T}$
μ: número mínimo de objetos
k: quantidade de *flocks*
ϵ : estimativa inicial de diâmetro
Saída: $\mathcal{F}$: k *flocks*

```
1  repita
2  |  F ← PSI(μ, ε, δ) sobre w_T
3  |  se |F| = k então
4  |  |  retorna F
5  |  senão se |F| < k então
6  |  |  ε ← nova estimativa (maior)
7  |  senão
8  |  |  ε ← nova estimativa (menor)
9  |  |  w_T ← F_T                    // Filtra as trajetórias de w
10 |  fim se
11 até sempre;
```

Essa nova estimativa de chute pode se basear na seguinte premissa: quanto maior o diâmetro de busca, maior a quantidade de trajetórias que podem ser identificadas como juntas, necessitando assim de uma distância maior para mais respostas e menor para menos. Porém, como esses algoritmos encontram *flocks* de tamanho maximal, o experimento da Figura 13 mostra que a redução dessa distância também pode aumentar a quantidade de *flocks* encontrados, pois com diâmetros menores, mais *flocks* menores podem ser identificados. Dessa forma, com a instabilidade da quantidade de respostas encontradas na variação desse parâmetro essa abordagem simplificada de chute iterativo não possui um critério de parada válido para convergir à consulta de k *flocks*.

Para uma abordagem *top-down* exata, o processo de divisão dos *flocks* precisa sempre partir do *flock* mais extenso da iteração. A Figura 14 demonstra a escolha do *flock* mais extenso atual para ser processado. Possuindo dois *flocks* de três trajetórias em uma janela de tempo de três instantes de tempo, um com discos de fundo cinza e outro com discos transparentes e de bordas tracejadas, é possível visualizar que todos

esses discos são de diâmetros diferentes, cada um com o tamanho mínimo para envolver todos os pontos das trajetórias dos seus *flocks* em cada instante de tempo, diferentemente do padrão tradicional (Figura 3) em que todos os discos são de tamanho fixo. Assim, conforme Definição 25, sendo o diâmetro mínimo de um *flock* o maior diâmetro de seus discos (discos vermelhos), e, conforme Definição 26, o *flock* mais extenso $\overline{f_w}$ aquele cujo diâmetro mínimo é o maior dentre todos, consequentemente, o *flock* mais extenso entre os dois desse exemplo é o transparente de bordas tracejadas, já que seu disco em t_2 possui o maior diâmetro.

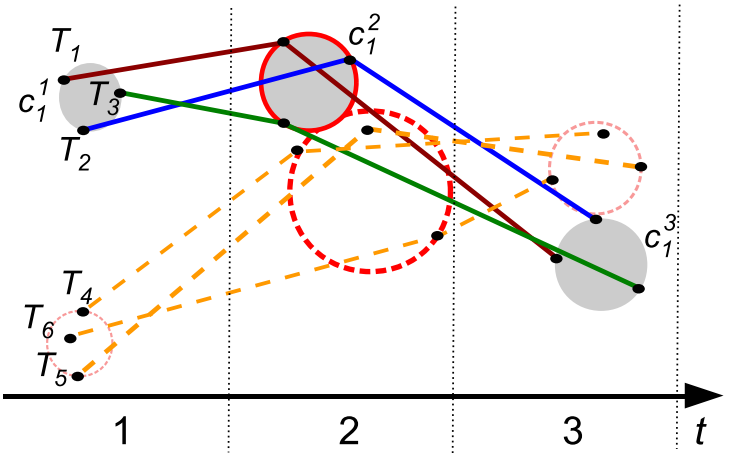


Figura 14 – Diâmetros dos *flocks* em uma janela, com o diâmetro do $\overline{f_w}$ com discos tracejados.

Depois de ter escolhido $\overline{f_w}$ dentre os $flocks$ candidatos da janela, a Figura 15 ilustra como a divisão acontece em duas situações: com dois ou três pontos de borda, os vermelhos. A ideia é descobrir qual trajetória que, se removida de $\overline{f_w}$, dividiria em dois *subflocks* mais extensos. É preciso ressaltar que essa divisão precisa garantir que esses dois *subflocks* mais extensos sejam produzidos para evitar perda de respostas na abordagem *top-down*. Para isso, para cada ponto de borda de $\overline{f_w}$ é criado um *subflock* com todas as demais trajetórias de $\overline{f_w}$, exceto a deste ponto. Na parte esquerda da Figura 15, $\overline{f_w}$ contém apenas dois pontos de borda, então os dois únicos *subflocks* possíveis são f_w^1 e f_w^2. Entretanto, em uma iteração futura f_w^1 se torna o $\overline{f_w}$ atual e precisa ser dividido. Como pode ser observado na parte direita da figura, esse *flock* contém três pontos de borda, criando assim três possíveis *subflocks*. O algoritmo precisa escolher os dois mais extensos (f_w^1 e f_w^2 em verde) e ignorar o menor, f_w^3 em vermelho. Uma propriedade fundamental dessa divisão é que os diâmetros dos *subflocks* são menores que seu *flock* original. Assim, esse processo faz uma redução iterativa do diâmetro que garante alançar os *flocks* de diâmetro mínimo.

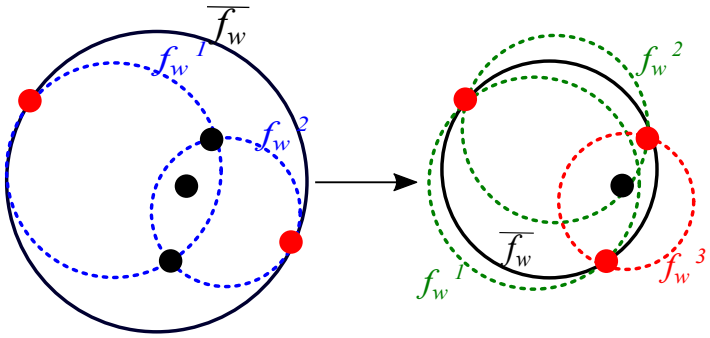


Figura 15 – Exemplo da divisão de dois discos do $\overline{f_w}$ em duas iterações diferentes. No lado esquerdo, $\overline{f_w}$ é dividido apenas nos dois *subflocks* possíveis. No entanto, no lado direito, com três pontos de borda, três *subflocks* são encontrados e o menor (f_w^3) é ignorado.

4.2 Algoritmo *Top-Down* para a Detecção de k_ϵ-*Flocks*

Esta seção detalha o algoritmo proposto pra responder k_ϵ-*Flocks* em uma janela temporal por meio da abordagem *top-down*. Conforme Algoritmo 2, as entradas são a janela temporal w a ser processada, contendo todos os pontos de localização das trajetórias delimitadas ao intervalo da janela; a quantidade mínima de trajetórias μ para determinar um *flock*; e k, como sendo a quantidade desejada de respostas. O algoritmo começa com a inicialização de duas variáveis: $atual\mathcal{F}$ e $\mathcal{F}_w^k$, linhas 1-2, respectivamente. A primeira, responsável por armazenar os *flocks* que serão divididos a cada iteração do algoritmo, é inicializada com o *flock* mais extenso possível em w, ou seja, maximal em seu tamanho e tal que seu diâmetro envolva todos os pontos das trajetórias para todos os instantes de tempo em w. $\mathcal{F}_w^k$, por sua vez, é responsável por armazenar *flocks* de tamanho mínimo durante as iterações do algoritmo, pois, já que não vão mais sofrer divisões, ficam armazenados como possíveis respostas.

Como o algoritmo foi implementado em C++, a estrutura utilizada para ambas as variáveis foi o *container multiset*[1], da biblioteca padrão STL. Com essa estrutura, por exemplo, como os *flocks* mantém-se ordenados por seu diâmetro, possibilitando inclusive *flocks* de diâmetros iguais, a busca para encontrar o mais extenso é feita em tempo constante, já que será sempre o último da estrutura, e a inserção e remoção em complexidade logarítmica.

Antes das iterações, porém, dois casos específicos são tratados. Na linha 3, quando da inexistência de qualquer *flock* na janela e parâmetros fornecidos, um conjunto vazio é retornado. No segundo caso, na linha 6, quando o próprio *flock* mais extenso em w for de tamanho minimal, ele mesmo será o conjunto resposta, contendo esse único *flock* existente.

[1] <http://www.cplusplus.com/reference/set/multiset/multiset/>

Algoritmo 2: *Top-Down*

Entrada: w: janela temporal contendo o conjunto de trajetórias $\mathcal{T}$
μ: número mínimo de trajetórias
k: número requerido de *flocks* de diâmetro mínimo
Saída: $\mathcal{F}_w^k$: k_ϵ-*Flocks*(μ, w)

```
   /* Inicialização                                                    */
1  atualF ← adiciona o flock mais extenso único em w
2  F_w^k ← [ ]
3  se atualF vazio então
4  |  retorna F_w^k                          // sem flocks encontrados
5  fim se
6  se atualF[0] é de tamanho mínimo então
7  |  retorna atualF                  // o mais extenso como flock único
8  fim se
   /* Iterações de refinamento                                         */
9  repita
10 |  f_w ← removeFlockMaisExtenso(atualF)
   |  /* Processa subflocks                                           */
11 |  Divide f_w nos dois subflocks mais extensos, f_w^1 e f_w^2, baseando-se nos
   |    pontos de borda da maior circunferência de f_w
   |  /* Calcula os discos para os subflocks                         */
12 |  f_w^1.ε ← 0;  f_w^2.ε ← 0;
13 |  para cada instante de tempo t_i em w faça
14 |  |  para cada subflock f_w ∈ {f_w^1, f_w^2} faça
15 |  |  |  f_w[c^{t_i}] ← calcula o disco mínimo c^{t_i} de f_w em t_i
16 |  |  |  se  f_w[c^{t_i}].diâmetro > f_w.ε então      // c^{t_i} aumentou f_w.ε?
17 |  |  |  |  f_w.ε ← f_w[c^{t_i}].diâmetro
18 |  |  |  fim se
19 |  |  fim para cada
20 |  fim para cada
21 |  para cada subflock f_w ∈ {f_w^1, f_w^2} faça
22 |  |  se f_w de tamanho mínimo então
23 |  |  |  F_w^k.insere(f_w)                    // f_w como candidato
24 |  |  senão
25 |  |  |  atualF.insere(f_w)              // f_w para futuras iterações
26 |  |  fim se
27 |  fim para cada
28 |  se |F_w^k| > k então                  // tem mais de k candidatos?
29 |  |  F_w^k.descartaMaisExtensos()        // descarta desnecessários
30 |  fim se
31 até existir flock em atualF;
32 retorna F_w^k
```

Para quando não ocorrer os dois casos excepcionais previamente explicados, as linhas 9-31 correspondem ao processo iterativo de refinamento, sempre processando o *flock* mais extenso a cada iteração. Essas iterações ocorrem enquanto existir algum *flock* maior

que o tamanho mínimo, necessitando ainda de divisões para convergir aos seus *subflocks*, ou seja, enquanto existir *flock* em $atual\mathcal{F}$, linha 31. A partir da abordagem exploratória, o algoritmo precisa encontrar todos os *subflocks* de tamanhos mínimos possíveis para então retornar os k_ϵ-*Flocks*, sejam eles realmente k *flocks* ou, quando da inexistência da quantidade requerida, a quantidade máxima de respostas, como por exemplo, no caso da linha 3, quando nenhum *flock* é encontrado, e da linha 6, quando existir apenas um *flock*, mesmo com mais de uma resposta requerida ($k > 1$). A premissa é que enquanto existir *flocks* maiores que o tamanho mínimo, esses mais extensos podem ser divididos em *subflocks* de menor extensão até convergir à resposta.

A cada iteração, o *flock* mais extenso do momento é retirado de $atual\mathcal{F}$ (linha 10) para ser processado. Assim, conforme regra de ordenação dessa estrutura, esse $\overline{f_w}$ será sempre o último. Como $\overline{f_w}$ não será necessariamente de tamanho mínimo, seus *subflocks* precisam ser processados (linhas 11-27).

Para a divisão desse $\overline{f_w}$, dados todos os seus discos, cada um sendo um disco delimitador mínimo envolvendo todos os pontos de localização em seu respectivo instante de tempo ($\overline{f_w}[c^{t_i}],\ t_{inicial} \leq t_i \leq t_{final}$), o algoritmo seleciona dois pontos de localização (linha 11) que estão na borda do maior disco dentre esses discos delimitadores mínimos (Figura 15). O disco delimitador mínimo envolvendo pontos pode ser obtido empregando qualquer algoritmo para encontrar a menor circunferência delimitadora de um conjunto de pontos, problema bem conhecido na matemática. Para esta implementação, por exemplo, adotou-se a biblioteca de algoritmos de geometria computacional CGAL[2]. Vale ressaltar que pode haver mais de dois pontos de borda, normalmente três. Quando esse caso ocorre, os dois pontos escolhidos são aqueles que resultarem nos dois *subflocks* mais extensos dentre os possíveis de $\overline{f_w}$. Assim, dois *subflocks* f_w^1 e f_w^2 são calculados passando por todo o seu comprimento, instantes de tempo também de w, cada um deles contendo todas as trajetórias de $\overline{f_w}$ exceto um dos dois pontos de borda escolhidos.

As linhas 12-20 reconstroem os *subflocks* encontrados. Desse modo, para cada instante de tempo que corresponde aos instantes de w, o novo disco de envolvimento mínimo das trajetórias é recalculado, atribuindo como novo diâmetro minimal do *subflock* o maior dentre esses discos (linha 17). Ressalta-se que o recálculo de disco mínimo é refeito apenas para os discos nos instantes de tempo em que o ponto de borda escolhido para a divisão de $\overline{f_w}$, e sendo removido do *subflock* em questão, esteja também em sua borda. Do contrário, sua remoção do disco não reduzirá seu tamanho.

Ao fim do processamento dos dois *subflocks* provenientes da divisão de $\overline{f_w}$, eles são tratados da seguinte forma. Quando forem de tamanho mínimo, já não podem ser mais divididos, portanto, não serão mais processados em iterações futuras de refinamento. Assim, quando de tamanho mínimo, são inseridos, a princípio, como possíveis respostas,

[2] <https://www.cgal.org/>

linha 23. Por outro lado, quando esses *subflocks* ainda não forem de tamanho mínimo, precisarão passar por mais divisões, sendo, então, recolocados em $atual\mathcal{F}$, linha 25. No entanto, apenas *subflocks* que já não são *subflocks* de algum *flock* em $atual\mathcal{F}$ que podem ser reinseridos para processamento futuro. Essa verificação evita que o algoritmo retorne respostas duplicadas, visto que para esses *subflocks* descartados, serão reencontrados em iterações futuras.

Ao término da iteração, uma verificação é feita pra o algoritmo não ficar mantendo candidatos desnecessários. Nas linhas 28-30 são descartados os candidatos mais extensos de $\mathcal{F}_w^k$ caso já exista mais de k armazenados, pois $\mathcal{F}_w^k$ não poderá conter mais de k *flocks* ao término do algoritmo, visto que se trata do conjunto final de respostas. Por fim, quando não existir mais *flocks* a serem divididos, ou seja, quando $atual\mathcal{F}$ estiver vazio (linha 31), k_ϵ-*Flocks* terá os k, ou quando não existirem, o maior número possível de *flocks*, respondendo, então, k_ϵ-*Flocks* em $\mathcal{F}_w^k$ na linha 32.

4.3 Algoritmo *Top-Down* em Janela Deslizante

O Algoritmo *Top-Down* descrito na Seção 4.2 é capaz de reportar $\mathcal{F}_w^k$, ou seja, k_ϵ-*Flocks* de uma única janela temporal w fornecida. No entanto, para de fato realizar a consulta Padrão k_ϵ-*Flock* em uma *stream* ou conjunto de dados D, o Algoritmo *Top-Down* em Janela Deslizante (Algoritmo 3) é apresentado para encontrar $\mathcal{F}^k$, todos os k_ϵ-*Flocks* de todas as janelas temporais de D. Assim, a entrada do Algoritmo *Top-Down* em Janela Deslizante difere do Algoritmo *Top-Down* pois requer o comprimento (δ) das janelas a serem processadas e a *stream* ou conjunto de dados D, ao contrário de apenas w, já de tamanho fixo.

Algoritmo 3: *Top-Down* em Janela Deslizante

Entrada: D: *stream*/conjunto de dados contendo o conjunto de trajetórias $\mathcal{T}$
μ: número mínimo de trajetórias
k: número requerido de *flocks* de diâmetro mínimo
δ: duração de *flock*
Saída: $\mathcal{F}^k$: k_ϵ-*Flocks* para todas as janelas $w \in D$

```
/* Inicialização                                              */
1 F^k ← []
/* Lê/desliza w                                               */
2 enquanto w = bufferWindow(D, δ) faça
3 |  F^k ← F^k ∪ Top-Down(w, μ, k)          // k_ε-Flocks(μ, w)
4 fim enqto
5 retorna F^k
```

Desse modo, conforme forem chegando os dados de D, como já mencionado, seja pela abordagem off-line com a entrada de todo o conjunto de dados de uma vez, ou on-line, com a *stream* desses dados, o algoritmo realiza o *buffer* de dados, uma janela temporal w

($|w| = \delta$). Após o armazenamento e processamento da primeira w, as próximas iterações apenas realizam o processo de deslizamento temporal da seguinte forma: dada a janela w processada anteriormente, remova-se desta os dados de seu primeiro instante de tempo e adicionam os dados do instante atual ao fim desse *buffer*, construindo a nova janela (linha 1).

Com a janela temporal w, cada iteração apenas invoca o Algoritmo *Top-Down* com μ, δ e w da iteração atual. Essa chamada retorna $\mathcal{F}_w^k$ de w que são adicionados ao conjunto $\mathcal{F}^k$ (linha 3). Dessa forma, quando não mais existir entrada de dados, todos os k_ϵ-*Flocks* de todas as janelas temporais possíveis são retornados em $\mathcal{F}^k$ (linha 5).

4.4 Algoritmo *Filtered Top-Down* com Limite Superior

O Algoritmo *Top-Down* e sua variação para janela deslizante retorna k_ϵ-*Flocks* independentemente do comportamento dos dados em cada momento. Essa busca pode retornar *flocks* com diâmetros grandes quando da dispersão dos MOs nesses momentos. Quando da noção prévia de um especialista, no entanto, mesmo não conhecendo a fundo o conjunto de dados preliminarmente, ele pode optar por estabelecer um critério de poda de respostas, isso é, nesse caso, estabelecer um diâmetro máximo para a busca dos k_ϵ-*Flocks*. Esse valor é denominado Limite Superior Empírico, porque de fato é um limiar superior e empírico porque é um valor proveniente do usuário/especialista por meio de sua experiência.

Algoritmo 4: *Filtered Top-Down*

Entrada: D: *stream*/conjunto de dados contendo o conjunto de trajetórias $\mathcal{T}$
μ: número mínimo de trajetórias
k: número requerido de *flocks* de diâmetro mínimo
ϵ: diâmetro limite superior para a busca
δ: duração de *flock*
Saída: $\mathcal{F}^{filtered-k}$: *Filtered-*k_ϵ*-Flocks* para todas as janelas $w \in D, respeitando\ \delta\ e\ \epsilon$

`/* Inicialização */`
1 $\mathcal{F}^{filtered-k} \leftarrow []$
`/* Lê/desliza` w `*/`
2 **enquanto** $w = bufferWindow(D, \delta)$ **faça**
3 $\quad filteredW \leftarrow PSI(w, \mu, \epsilon, \delta)_\mathcal{T}$ `// Filtra as trajetórias de` w
$\quad$ `/*` k_ϵ`-Flocks`$(\mu, filteredW)$ `*/`
4 $\quad \mathcal{F}^{filtered-k} \leftarrow \mathcal{F}^{filtered-k} \cup$ *Top-Down*$(filteredW, \mu, k)$
5 **fim enqto**
6 **retorna** $\mathcal{F}^{filtered-k}$

O Algoritmo 4 apresenta esta solução (Algoritmo *Filtered Top-Down*). Diferentemente do Algoritmo *Top-Down* em Janela Deslizante, agora há a entrada do parâmetro de

distância ϵ com a função de critério de diâmetro minimal máximo. Assim, a cada iteração, antes de aplicar o Algoritmo *Top-Down* na respectiva janela temporal w, esta é filtrada por um algoritmo de identificação de Padrões *Flock* (linha 3), o PSI no caso, e, joga como entrada do Algoritmo *Top-Down* a janela contendo apenas as trajetórias presentes nos resultados do PSI (linha 4). Assim, ao fim das iterações, apenas os *Filtered-k_ϵ-Flocks* são reportados (linha 6).

5 EXPERIMENTOS E RESULTADOS

Este capítulo apresenta os experimentos para avaliar as contribuições deste trabalho. A Seção 5.1 mostra os conjuntos de dados utilizados e as configurações dos experimentos. A Seção 5.2 discute a variabilidade do diâmetro minimal para encontrar os k_ϵ-*Flocks* nas diferentes janelas temporais e ressalta a dificuldade em estabelecer um diâmetro fixo para essa solução, além dos diferentes números de respostas retornadas. Por fim, a Seção 5.3 apresenta o comportamento dos algoritmos propostos nos conjuntos de dados avaliados em relação ao desempenho.

5.1 Conjuntos de Dados e Configurações

5.1.1 Descrição dos Conjuntos de Dados

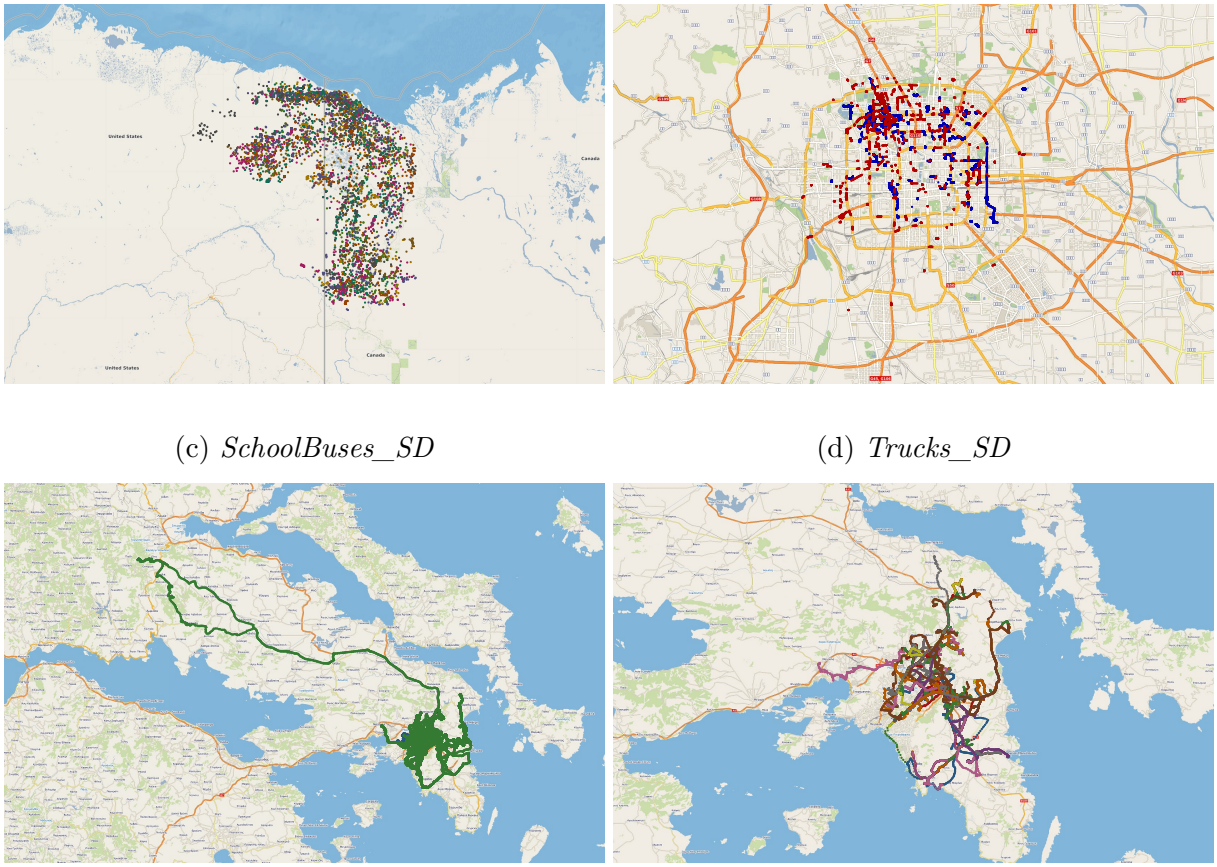
Os experimentos foram realizados em cinco variações de conjuntos de dados reais bem conhecidos na área. O primeiro, *Schoolbuses*[1], consiste em trajetórias de 2 ônibus escolares coletados por alunos na região metropolitana de Atenas, Grécia, por 108 dias distintos. O segundo, *Trucks*[2], é constituído por trajetórias coletadas em 33 dias distintos por 50 caminhões de entrega de concreto para diversas construções também na região metropolitana de Atenas. O terceiro conjunto de dados é o *Caribous*[3] do projeto "*Porcupine Caribou Herd Satellite Collar Project*", possuindo pontos de localização de 44 renas em movimento de migração no noroeste do Canadá. E, por fim, o *GeoLife*[4]. Este conjunto de dados foi coletado por 182 usuários por cinco anos, de 2007 a 2012, contendo 17.621 trajetórias, com taxas de amostragem variadas por serem capturadas com dispositivos diversos. Parte dessas trajetórias possuem a informação do meio de transporte usado, como carro, trem, bicicleta, à pé, dentre outros. Embora boa parte dos pontos de localização tenha sido coletada na região de Pequim, China, há pontos em diversas partes do mundo, inclusive na América do Norte. A Figura 16 apresenta a distribuição espacial de amostras dos conjuntos de dados utilizados nos experimentos.

5.1.2 Pré-processamentos Realizados

Todos os conjuntos de dados, com exceção do *Caribous*, tiveram como pré-processamento a sobreposição de todos os diferentes dias em um único — sufixo "_SD", de "*same day*", mesmo dia em Inglês. Como exemplo, no *SchoolBuses_SD*, cada uma das trajetórias dos dois ônibus em cada dia específico acarreta trajetórias diferentes no mesmo dia com seus

[1] <http://chorochronos.datastories.org/?q=node/6>
[2] <http://chorochronos.datastories.org/?q=node/5>
[3] <www.taiga.net/satellite>
[4] <www.microsoft.com/en-us/research/project/geolife-building-social-networks-using-human-location-history>

(a) *Caribous* (b) *GeoLife_(all/walk)_SD*



(c) *SchoolBuses_SD* (d) *Trucks_SD*

Figura 16 – Regiões e pontos de localização dos quatro conjuntos de dados utilizados nos experimentos. 16a) Região noroeste do Canadá com cada cor representando uma rena do *Caribous*. 16b) Região entre o quinto anel viário de Pequim, sendo as cores azul e vermelha correspondentes aos dados de *GeoLife_walk_SD* e *GeoLife_all_SD*, respectivamente. 16c) Região metropolitana de Atenas, com cada cor representando cada ônibus do *SchoolBuses_SD*. E 16d), também na região metropolitana de Atenas, em que cada cor corresponde a pontos de localização de um dos caminhões do *Trucks_SD*.

respectivos horários, como se fossem ônibus diferentes. Esta estratégia tem como objetivo aumentar a granularidade dos conjuntos de dados, ou seja, aumentar a quantidade de pontos de localização em cada instante de tempo para de fato testar os algoritmos com entradas de dados mais densas e, ao mesmo tempo, com um período satisfatório para identificação dos padrões.

Por outro lado, devido aos mais de 24 milhões de pontos de localização e à alta taxa de amostragem média de 2 segundos do *GeoLife*, adotou-se como estratégia segmentá-lo em dois subconjuntos: *GeoLife_all_SD*, com apenas os primeiros 45 segundos da hora de maior quantidade de pontos — 18h no horário de Pequim —; e *GeoLife_walk_SD* possuindo essa hora completa, mas apenas com os pontos de localização marcados como sendo de pedestres caminhando (*walk*). Esses dois conjuntos de dados também possuem

apenas pontos de localização na região entre o quinto anel viário da cidade, a mais densa em números de pontos coletados pelos mais de cinco anos do projeto *GeoLife*, limitada pelas latitudes 39.75 e 40.03 e longitudes 116.20 e 116.56.

Em relação ao *Caribous*, essa estratégia de condensar todas trajetórias no mesmo dia não foi utilizada, pois o conjunto de dados original não fornece o tempo exato das coletas dos pontos de localização, tornando impossível essa abordagem e também a exatidão de sua taxa de amostragem, em que cada rena possui pontos amostrados apenas após percorridos dezenas ou até mesmo algumas centenas de quilômetros. Outro fator é que essas trajetórias denotam o movimento migratório do bando de renas de tal forma que o grupo não permanece na mesma região com o passar do tempo. Outro dado importante, esse conjunto de dados é o único com granularidade exata pois o conjunto de dados original é justamente as trajetórias sem lacunas de cada rena, contendo um único ponto de localização por instante de tempo, ou seja, são 44 trajetórias com exatamente o mesmo número de pontos.

A Tabela 1 apresenta uma síntese das informações a respeito das variações desses conjuntos de dados após os pré-processamentos descritos. Percebe-se a variedade dos conjuntos gerados, em termos de natureza dos objetos móveis, movimentação em rede viária e fora de rede viária, número de pontos e de trajetórias, taxa de amostragem e granularidade. Desta forma, é um grupo de conjuntos de dados que permite avaliar os algoritmos propostos em diferentes situações.

Tabela 1 – Conjuntos de dados utilizados nos experimentos. (Fonte: próprio autor)

Conjunto de Dados	**Nº de Pontos**	**Nº de Trajetórias**	**Nº de TS**	**Taxa de Amostragem**	**Granularidade**
Caribous	15.796	44	359	-	44
GeoLife_all_SD	8.827	602	23	2s	∼384
GeoLife_walk_SD	106.197	346	1.800	2s	∼59
SchoolBuses_SD	66.096	145	2.007	30s	∼33
Trucks_SD	112.203	276	2.734	30s	∼41

5.1.3 Configurações dos Experimentos

Os algoritmos propostos foram desenvolvidos em C++ e compilados com o GCC v5.4. Os resultados referentes a tempo de execução apresentados neste capítulo são médias de 20 repetições dos experimentos, a fim de reduzir vieses de eventuais interferências externas no tempo de execução. Todos os experimentos deste trabalho, exceto quando explicitamente destacado, quando da variação dos diferentes parâmetros, respeitam os valores definidos conforme a Tabela 2.

Os experimentos deste trabalho foram executados em computador pessoal, com a seguinte configuração: Intel(R) Core(TM) i7-3630QM CPU 2.40GHz, com 8GB de RAM — DDR3 1600MHz —, no sistema operacional Ubuntu 16.04.4 LTS x64. Vale ressaltar

Tabela 2 – Valores dos parâmetros utilizados nos experimentos. (Fonte: próprio autor)

Parâmetro	**Valores avaliados**	**Valor padrão**
$\boldsymbol{\mu}$	3, 5, ..., 19	**5**
$\boldsymbol{k}$	1, 3, ..., 15	**5**
$\boldsymbol{\delta}$	3, 5, ..., 19	**10**

que embora se trate de um processador com oito núcleos, a execução dos algoritmos utiliza sempre apenas um.

5.2 Análise da Variação do Diâmetro Minimal em k_ϵ-*Flocks*

O padrão k_ϵ-*Flocks* tem por objetivo permitir a identificação de *flocks* sem a necessidade da definição de um diâmetro por parte do usuário. Entretanto, se o usuário conhecesse com certa precisão o conjunto de dados, seria factível indicar manualmente um valor que retornasse o número desejado de *flocks*. A pergunta que surge neste contexto é: quão difícil seria fazer isso, de forma a tornar a operação de identificação de k_ϵ-*Flocks* irrelevante?

Como visto no Capítulo 3, trabalhos da literatura indicam a dificuldade em definir adequadamente esse parâmetro para diferentes conjuntos de dados. Neste sentido, esta seção apresenta experimentos que visam ressaltar o quão variado pode ser o diâmetro necessário para se identificar os *flocks* de tamanho mínimo para diferentes domínios de dados e até mesmo apenas em momentos (janelas temporais) diferentes.

5.2.1 Diâmetro Minimal em Diferentes Janelas Temporais

A Figura 17 mostra qual o diâmetro em quilômetros do k-ésimo *flock* de tamanho minimal para os cinco conjuntos de dados. Cada um dos gráficos contém quatorze amostras, sendo que cada uma é a i-ésima janela temporal (w_i) do respectivo conjunto de dados. A escolha deste número de amostras é devido a dois fatores. Primeiro, a grande variação dos diâmetros torna confusa a visualização considerando todas as janelas, assim um número menor de amostras facilita o entendimento. Segundo, esse é o número exato de janelas temporais do conjunto de dados de menor duração, o *GeoLife_all_SD*, padronizando assim os demais gráficos.

A aplicabilidade do Algoritmo *Top-Down* como ferramenta de busca exploratória inicial fica clara na Figura 17 ao confrontarmos as distâncias necessárias para os conjuntos de dados. Enquanto o conjunto (a) *Caribous* requer diâmetros na casa de centenas de quilômetros para encontrar *flocks*, em outros são necessários apenas alguns quilômetros ou até mesmo poucos metros (e.g., *GeoLife_all_SD*). Ressalta-se que esse diâmetro tão grande para o *Caribous* é devido à característica desse conjunto de dados, que conforme previamente explicado, possui uma taxa de amostragem bem irregular para os diferentes

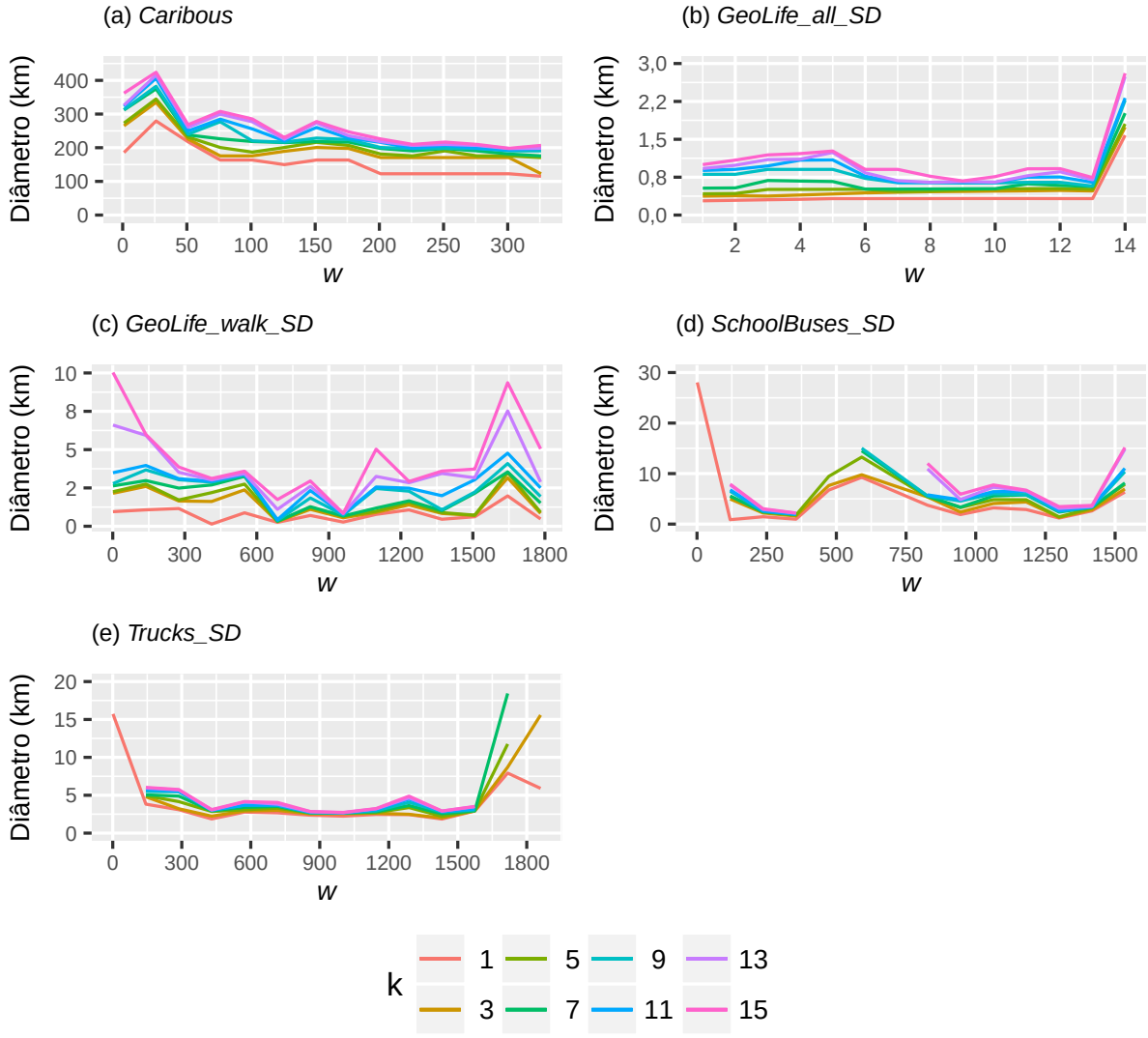


Figura 17 – Diâmetros dos k-ésimos *flocks* em uma amostra de janelas w para cada um dos 5 conjuntos de dados.

animais sendo até mesmo de dezenas e até centenas de quilômetros entre dois pontos amostrados do mesmo animal.

Já em relação à variação de comportamento no próprio domínio, mas em momentos distintos, tomaremos como exemplos os conjuntos de dados *SchoolBuses_SD* e *Trucks_SD*. Analisando o conjunto *SchoolBuses_SD* (Figura 17(d)), é possível visualizar que as viagens ocorrem entre as janelas w_{125} e w_{1400}, no entanto, havendo um período menos intenso por volta da janela w_{370} até a w_{900}. Da mesma forma, no conjunto *Trucks_SD* (Figura 17(e)), nota-se um intervalo entre as janelas w_{150} até w_{1600}, em que os *flocks* são encontrados com diâmetros menores. Trata-se do horário em que os caminhões comumente realizam suas entregas.

Esses resultados mostram que a definição manual do diâmetro para encontrar *flocks* é uma tarefa dependente da natureza do conjunto de dados e também da variação

da distribuição dos objetos no espaço no decorrer do tempo.

5.2.2 Variação no Número de *Flocks* no Fluxo de Dados

Uma análise complementar a respeito da utilidade do Padrão k_ϵ-*Flock* consiste em avaliar a possibilidade de se utilizar um algoritmo de identificação de *flocks* com um diâmetro que retorne, em média, um determinado número de respostas. A Figura 18, compara a quantidade de respostas obtidas pelos padrões k_ϵ-*Flock* (Algoritmo *Top-Down* em Janela Deslizante) e *Flock* convencional (Algoritmo PSI). Como ambos diferem em um parâmetro, k para o *Top-Down* e ϵ para o PSI, a comparação parte do seguinte questionamento: "Com uma média dos diâmetros dos k-ésimos *flocks* em um dado conjunto de dados, quantas respostas o Padrão *Flock* retornaria? Ao menos o número de respostas seria comparável?"

A figura mostra a comparação dos diâmetros do *Top-Down* para k de valores 1, 5 e 9, também para quatorze amostras de janelas temporais. O diâmetro é a mediana do diâmetro minimal do respectivo valor de k para o conjunto de dados a partir da técnica de gráfico de caixas sobre os diâmetros dos k-ésimos *flocks* encontrados no experimento da Figura 17. Adotou-se mediana em vez de média porque a média poderia ser distorcida pelo fato de haver valores discrepantes, ou seja, diâmetros ora muito baixos, ora muito altos.

A Figura 18 mostra que a quantidade de respostas do *Top-Down* é justamente o valor de k, exceto para os casos em que de fato não há tantas respostas, como nas janelas iniciais do (d) *SchoolBuses_SD* e nas finais do (e) *Trucks_SD*. No entanto, para o PSI, não é possível estabelecer nenhuma relação entre as quantidades de respostas obtidas com as do *Top-Down*.

É possível inclusive a identificação de pontos de saturação do diâmetro de distância para o PSI, ou seja, quando aumentar o diâmetro de busca acaba reduzindo a quantidade de resposta em vez de aumentar. Este cenário é bem claro ao verificarmos o gráfico (a) *Caribous* da Figura 18, em que perto da janela w_{200} a linha do maior diâmetro, correspondente à mediana do diâmetro do 9-ésimo *flock* do *Top-Down*, passa a reportar até mesmo menos respostas que o menor diâmetro, para $k = 1$.

Estes resultados mostram que, no decorrer do tempo, uma estratégia de diâmetro fixo produziria um número inconstante de respostas, o que, para algumas situações poderia ser conveniente, mas não acompanharia eventuais dispersões ou aglutinações dos objetos móveis do conjunto de dados.

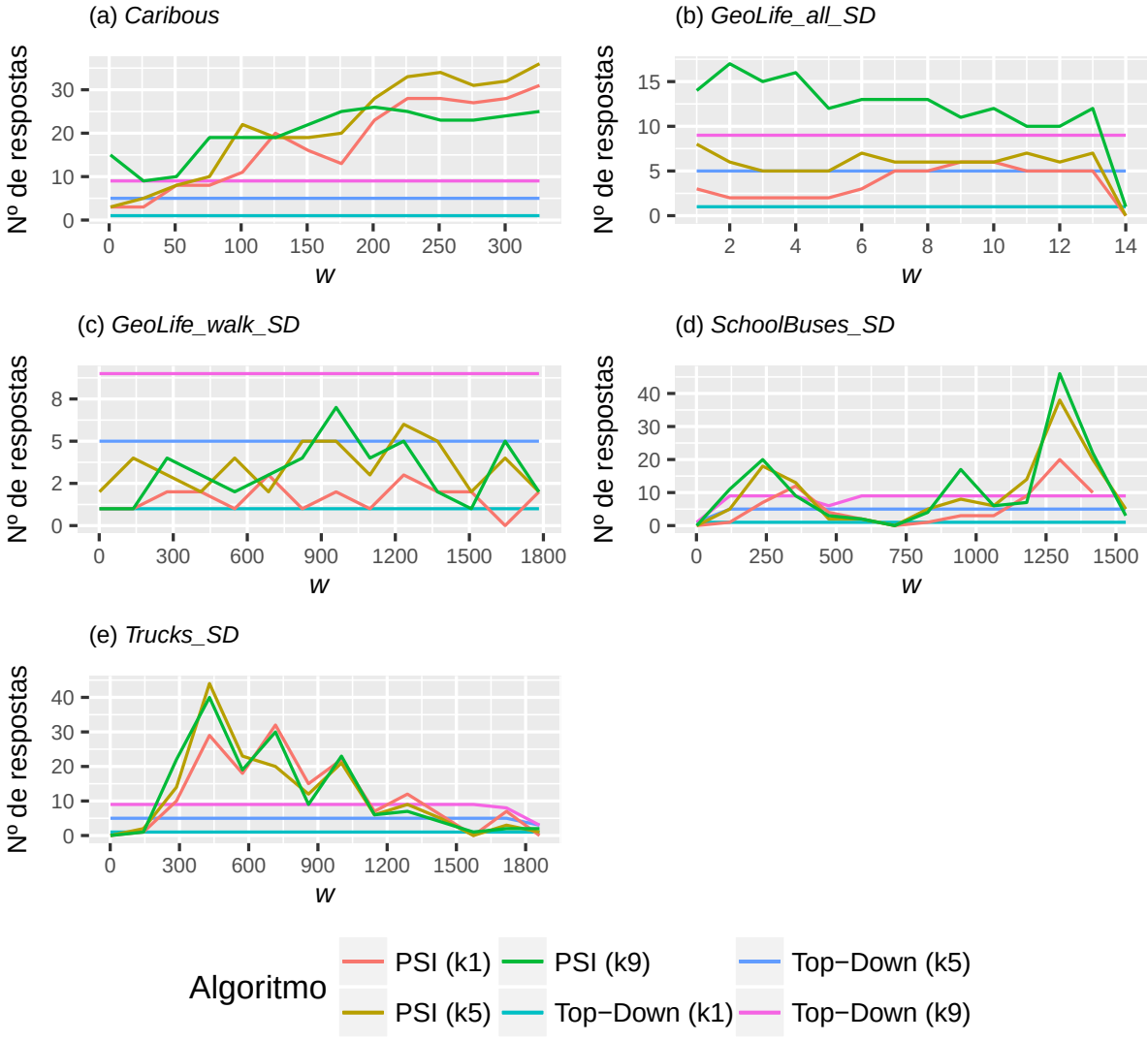


Figura 18 – Comparação da quantidade de respostas obtidas pelo Algoritmo *Top-Down* em Janela Deslizante, com os valores de k 1, 5 e 9; e o Algoritmo PSI, tendo como parâmetro de distância a mediana dos diâmetros dos respectivos k-ésimos *flocks* do *Top-Down* de cada um dos cinco conjuntos de dados.

5.3 Análise de Desempenho

Esta seção apresenta a análise de desempenho do Algoritmo *Top-Down* em Janela Deslizante para a análise exploratória de todo o conjunto de dados; um comparativo de tempo de execução com o Algoritmo PSI com um diâmetro definido pelo usuário; e a análise de desempenho com a abordagem *Filtered Top-Down*, com esse mesmo diâmetro.

5.3.1 Comportamento de Tempo de Execução do *Top-Down* em Janela Deslizante

A Figura 19 demonstra o comportamento do tempo de execução do Algoritmo *Top-Down* ao processar todas as janelas temporais de cada um dos conjuntos de dados,

variando cada um do três parâmetros, ou seja, o Algoritmo *Top-Down* em Janela Deslizante. Quando da variação de um dos parâmetros, os demais são fixados com valores padrões, conforme Tabela 2. Como esperado, o tempo de execução reduz com o aumento de μ, porque quanto maior seu valor, mais restritiva fica a possibilidade de se encontrar *flocks*, reduzindo assim o tempo de processamento.

Em relação ao comprimento da janela temporal (δ), o comportamento do tempo de execução depende essencialmente da quantidade de trajetórias sem lacunas por toda a janela, ou seja, possuindo ponto de localização em todos os instantes de tempo da janela. Desse modo, como pode ser observado na Figura 19 o tempo de execução tende a crescer para janelas pequenas, pois há maior chance de existirem trajetórias completas nesses intervalos, enquanto que para janelas maiores essa chance é gradualmente reduzida. Vale destacar que o crescimento linear especificamente do (c) *Caribus* é esperado justamente por ser o único conjunto de dados a conter todas as trajetórias sem lacunas, possuindo ponto de localização de cada uma das renas para todos os instantes de tempo. Por outro lado, os demais gráficos sugerem um pico com trajetórias completas, seguido de um decrescimento no tempo de execução. Por fim, para a variação do parâmetro k, o tempo de execução permanece estável, pois, praticamente não impacta na operação dominante do algoritmo que é a quantidade de iterações necessárias para convergir às respostas.

A Figura 20 apresenta o diagrama de caixas das quantidades de iterações que o Algoritmo *Top-Down* em Janela Deslizante executa até convergir às respostas para todas as janelas temporais dos cinco conjuntos de dados. Conforme esperado, para os parâmetros mais seletivos, ou seja, aqueles que quanto maiores mais restritas as possibilidades de resposta, μ e δ, o número de iterações tende a decrescer, já que o algoritmo processa um conjunto de trajetórias cada vez mais mais restrito. Esse comportamento não acontece ao *Caribous*, pois, como já mencionado, todas as trajetórias das 44 renas são completas, removendo assim a característica seletiva de δ, visto que não há redução no número de trajetórias com o incremento desse parâmetro.

No entanto, ao confrontarmos a Figura 19 com a Figura 20, o tempo de execução do algoritmo, especialmente quando da variação de δ, nem sempre decresce com a redução do número de iterações. Esse comportamento acontece porque as operações executadas em cada iteração do *Top-Down*, essencialmente na etapa de construção dos *subflocks* a cada divisão tem como constante multiplicativa δ, o comprimento da janela.

5.3.2 Comportamento de Tempo de Execução do *Filtered Top-Down*

Esta seção descreve o comportamento de desempenho do Algoritmo *Filtered Top-Down*. Partindo-se do pressuposto que exista um conhecimento preliminar do domínio dados a serem analisados, no *Filtered Top-Down*, o PSI passa a ser uma ferramenta importante para filtrar candidatos. Assim, além da possibilidade de usá-lo como algoritmo

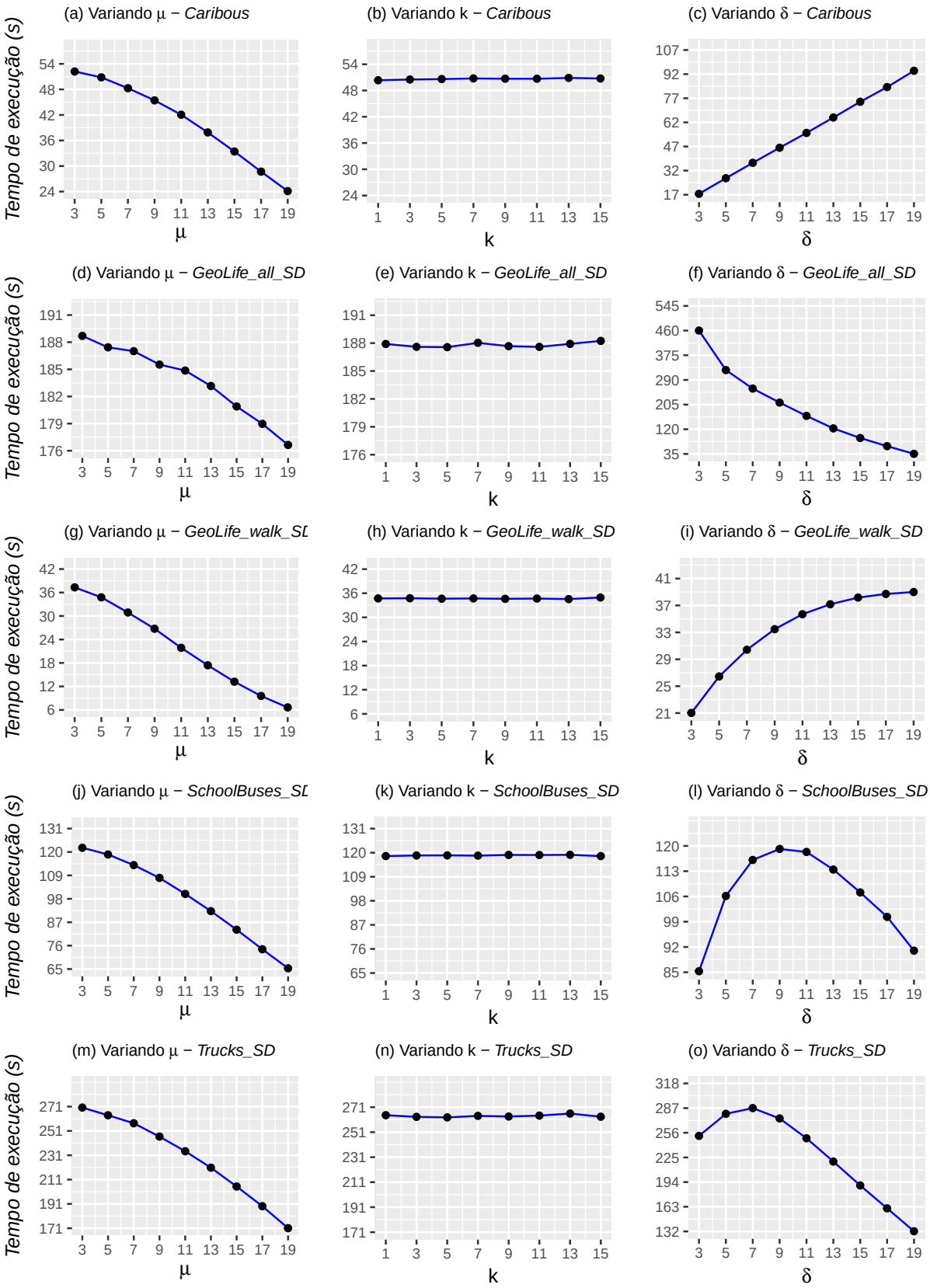


Figura 19 – Comportamento do desempenho do Algoritmo *Top-Down* em Janela Deslizante variando os parâmetros μ, k e δ nos 5 conjuntos de dados.

totalmente exploratório, o usuário pode optar por apenas selecionar os k_ϵ-*Flocks* com até um limite de diâmetro exigido.

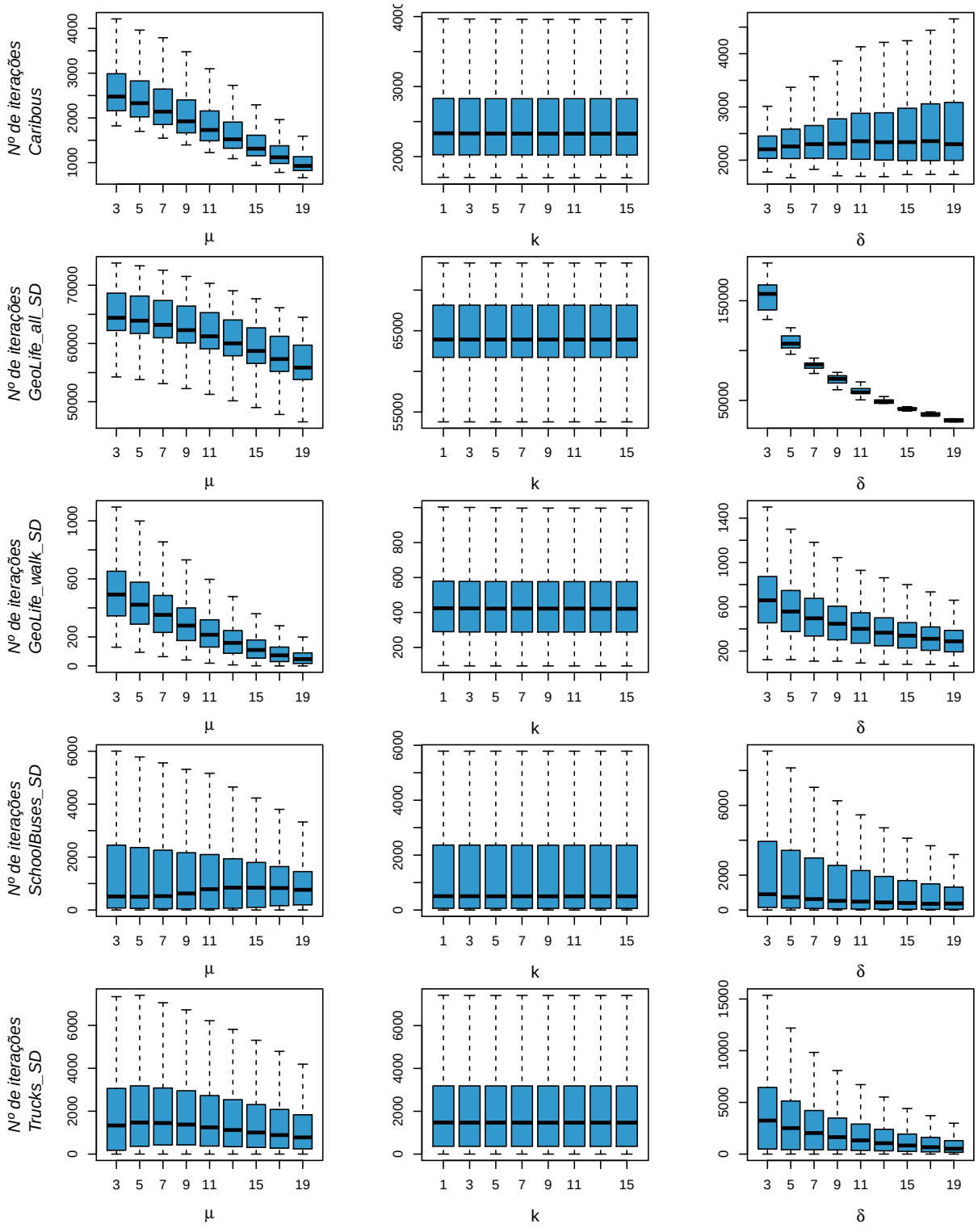


Figura 20 – Diagrama de caixas do número de iterações do Algoritmo *Top-Down* em Janela Deslizante variando os parâmetros μ, k e δ nos 5 conjuntos de dados.

5.3.2.1 Definição do Limite Superior Empírico

O diâmetro fixo para filtragem inicial de *flocks* é denominado nesta seção como Limite Superior Empírico e trata-se de uma distância estabelecida pelo especialista do domínio. No experimento descrito nesta seção, o critério de seleção desse valor é resultado da análise dos diâmetros dos k-ésimos *flocks* nas amostras das janelas dos cinco conjuntos de dados da Figura 17. O diâmetro escolhido refere-se a valores, para os respectivos conjuntos de dados, que consigam encontrar os 15-ésimos *flocks* nas janelas de menor diâmetro minimal. O objetivo do experimento é obter os k_ϵ-*Flocks* com $k = 5$, por isso escolhe-se um diâmetro de um valor maior de k. Desta forma, a probabilidade de que as 5 respostas desejadas estejam contidas nesse diâmetro é razoável.

O limite superior de 7 km para o *SchoolBuses_SD* pois, como visto nas amostras da Figura 17, entre os períodos de viagens mais comuns, o 15-ésimo *flock* pode ser encontrado com este diâmetro. A escolha dos limites superiores reafirma a dificuldade em estabelecer um diâmetro fixo para as diversas janelas temporais de um mesmo conjunto de dados, ou até mesmo para momentos distintos no mesmo conjunto. Por exemplo, o limite superior empírico do *Trucks_SD*, com o diâmetro de 5 km, superior ao 15-ésimo *flocks* de boa parte das amostras visíveis no experimento da Figura 17, é inferior à mediana dos diâmetros para encontrar o primeiro *flock* minimal para todo esse conjunto de dados. Já para o *GeoLife_all_SD*, o valor empírico de 1 km, é muito próximo à mediana do 5-ésimo *flock* desse conjunto. Isso demonstra o quão desafiador e impraticável é para se encontrar o k-ésimo *flock* em diferentes momentos de um mesmo conjunto de dados com um parâmetro fixo. O Algoritmo *Filtered Top-Down* ameniza essa dificuldade, pois realiza o refinamento das respostas filtradas inicialmente, embora em alguns casos retorne menos do que k respostas.

5.3.2.2 Análise dos Custos Internos do *Filtered Top-Down*

O experimento da Figura 21 demonstra como é composto o tempo de execução do Algoritmo *Filtered Top-Down*, isto é, o tempo gasto com o filtro com limite (PSI) e o tempo do refinamento, para identificar os k_ϵ-*Flocks* de tamanho mínimo (*Top-Down*), lembrando que, na verdade, retorna-se $k' \leq k$ *flocks*.

Os gráficos da figura mostram que os custos de cada fase variaram bastante para cada conjunto de dados. Para o conjunto *Caribous*, a filtragem inicial teve um pequeno impacto no custo geral do algoritmo. Já para o *GeoLife_(all/walk)_SD*, ocorreu o contrário, sendo o refinamento a etapa de custo consideravelmente mais baixo. Para os demais conjuntos, ambas as etapas tiveram contribuição significativa para o custo geral do algoritmo.

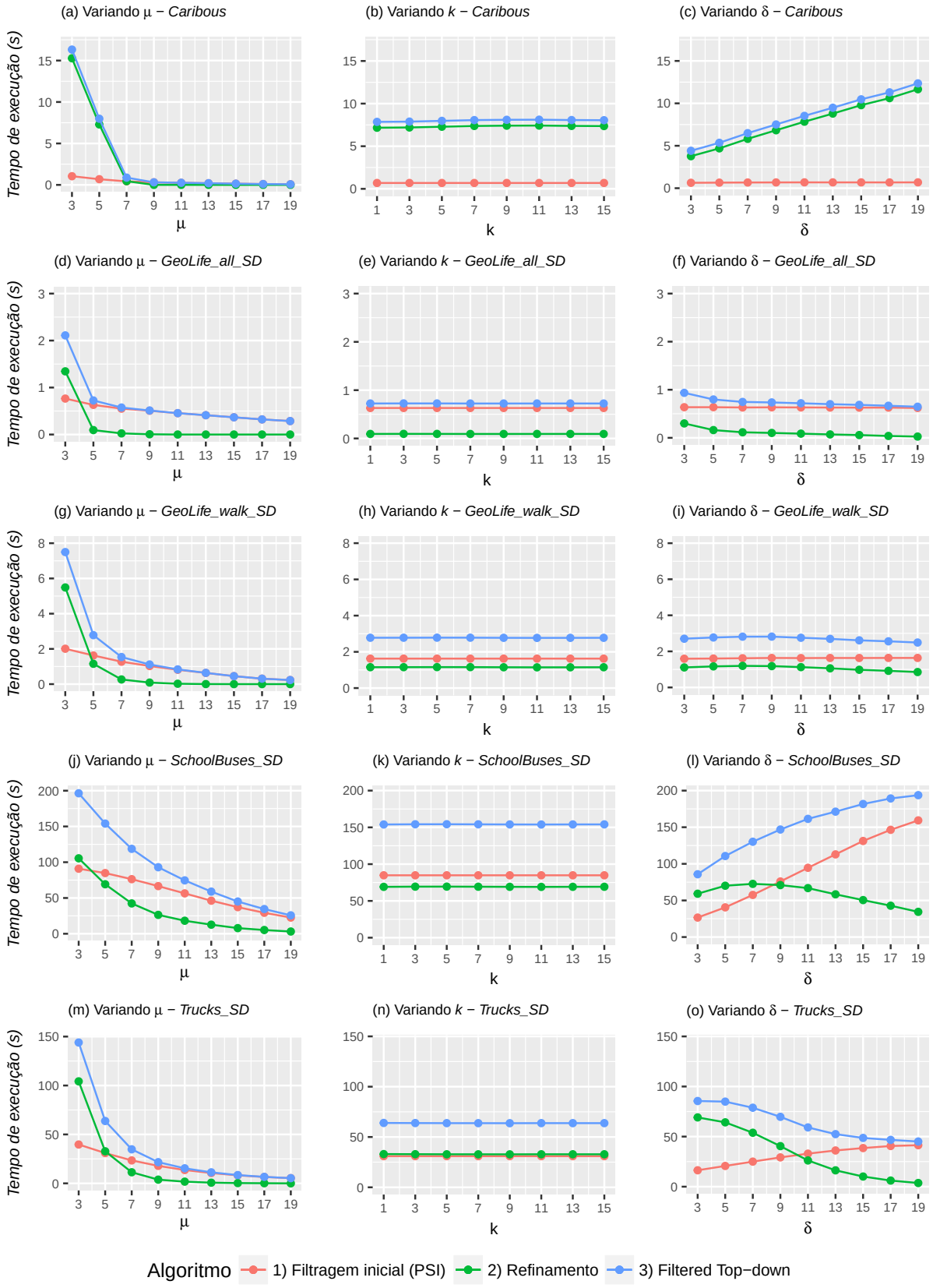


Figura 21 – Composição do tempo de execução do Algoritmo *Filtered Top-Down*.

5.3.3 Análise Comparativa de Desempenho dos Algoritmos

A Figura 22 faz o comparativo de desempenho entre o Algoritmo *Top-Down* em Janela Deslizante em todo o conjunto de dados e o Algoritmo *Filtered Top-Down* com o limite superior empírico. A figura também mostra o tempo de execução do algoritmo PSI, também com o diâmetro de limite superior como entrada, para servir de linha base.

A respeito do comportamento de tempo de execução do algoritmo PSI com o limite superior empírico, assim como no Algoritmo *Top-Down* para a Detecção de k_ϵ-*Flocks*, o parâmetro μ torna a busca mais seletiva, reduzindo assim o número possível de respostas e, consequentemente, o tempo de execução. No entanto, o mesmo não ocorre com o parâmetro δ. Essa disparidade entre ambos os algoritmos para a variação do tamanho da janela é explicada pela forma de processamento dos dados. Enquanto que o Algoritmo *Top-Down* para a Detecção de k_ϵ-*Flocks* sempre processa uma janela completa, ou seja, pontos de localização de δ instantes consecutivos de uma vez, o PSI processa cada instante de tempo por vez, realizando etapas de junção dos candidatos do instante de tempo anterior (Subsubseção 2.3.1.2). Por fim, para a variação do diâmetro ϵ, quanto maior seu valor menos seletiva é a busca, portanto, o PSI acaba manipulando mais candidatos tempo a tempo, piorando seu desempenho.

Com relação ao desempenho relativo entre os algoritmos, naturalmente, o *Filtered Top-Down* tem tempo de execução superior ao PSI, pois utiliza o PSI na etapa de filtragem inicial. Entretanto, o que pode ser observado nesse experimento é que o *Filtered Top-Down* tem um tempo de execução drasticamente menor do que o *Top-Down* em Janela Deslizante, sendo dezenas e até mesmo centenas de vezes mais rápida. Dentre os conjuntos avaliados, a única exceção é o conjunto de dados *SchoolBuses_SD*, em que inclusive o próprio PSI perde para o *Top-Down* exploratório para janelas temporais maiores.

Este resultado é interessante pois abre a possibilidade para o usuário solicitar uma execução consideravelmente mais eficiente durante a busca, ao fornecer uma distância limite. Além disso, os experimentos apresentados neste capítulo corroboram para a utilidade dos padrões k_ϵ-*Flocks* propostos, bem como mostram os comportamentos dos algoritmos desenvolvidos considerando conjuntos de dados variados, que dão uma visão abrangente do desempenho da solução em diferentes situações.

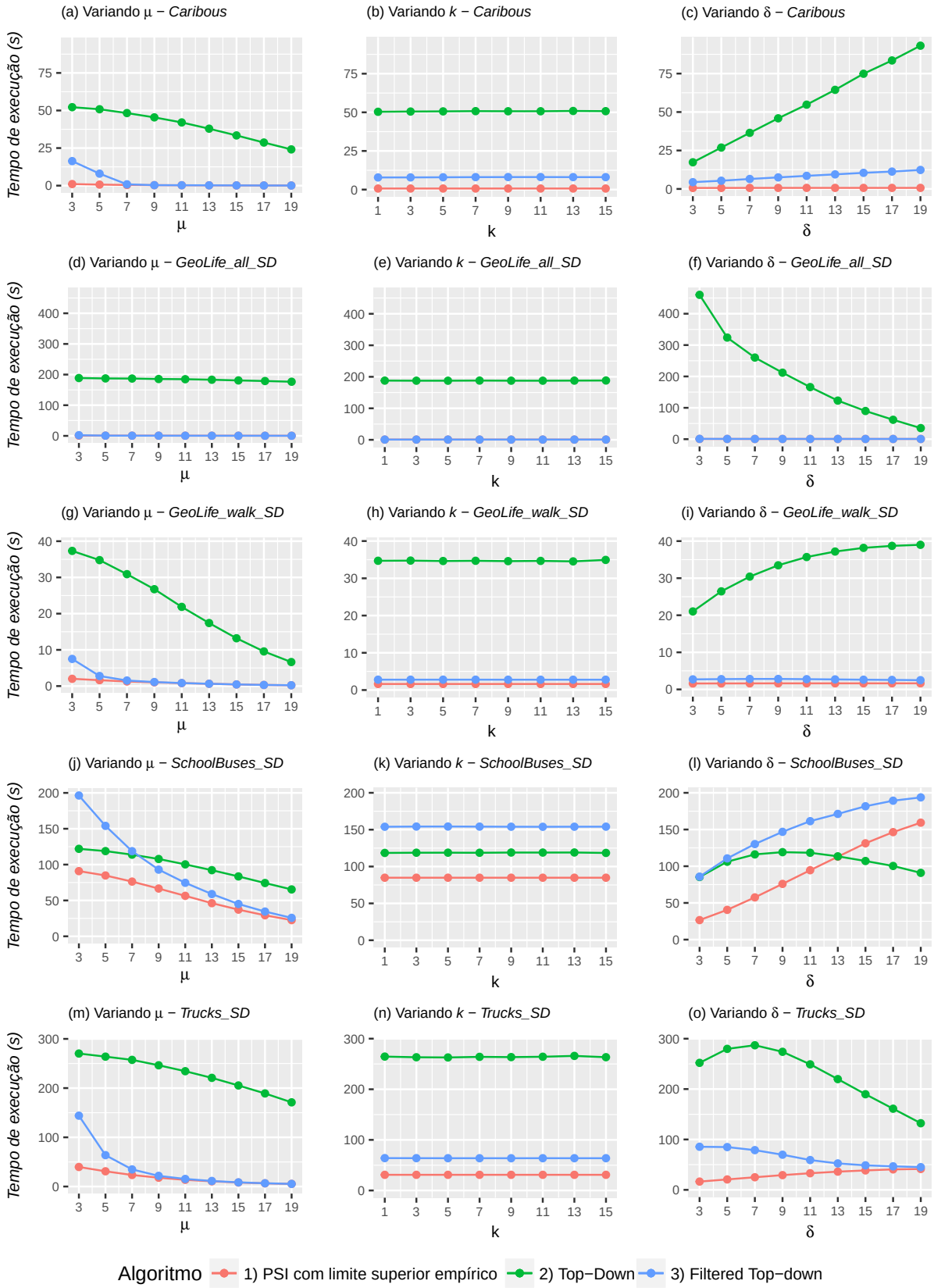


Figura 22 – Comparação dos desempenhos entre os algoritmos PSI com diâmetro limite superior empírico, o *Top-Down* em Janela Deslizante e o *Filtered Top-Down*.

6 CONCLUSÃO

A crescente adoção de tecnologias e serviços baseados em localização vem contribuindo para o aumento de dados espaço-temporais, fonte importante para diversas informações relevantes. Dentre estas, destaca-se a identificação de padrões de comovimento de objetos móveis, ou seja, grupos de objetivos móveis se movendo próximos entre si por um período. Um dos mais conhecidos padrões neste contexto é o Padrão *Flock*, que busca justamente identificar esses grupos em comovimento desde que dado um diâmetro máximo, para cada instante de tempo consecutivo ao longo da duração desse padrão exista um disco com esse diâmetro que consiga envolver todos os objetos móveis.

No entanto, a parametrização dos algoritmos existentes para encontrar esses padrões não é tarefa fácil. Embora estabelecer uma quantidade mínima de objetos para se considerar um grupo e uma duração para se caracterizar um comovimento seja fundamentalmente relacionado ao critério e desejo do usuário, muitas vezes predefinir uma distância fixa de proximidade desse grupo não é trivial nem mesmo para especialistas de domínio. Isso decorre de inúmeros fatores, como o comportamento e liberdade de movimentação de cada tipo de objetivo móvel, fatores técnicos relacionados aos dispositivos de coleta das trajetórias, seja por falhas, ruídos, diferentes taxas de amostragem para cada dispositivo, entre outros. Além disso, especificamente para padrões que utilizam a busca em disco, caso do Padrão *Flock*, acarreta uma limitação no formato espacial que os objetos precisam respeitar ao se moverem.

Desse modo, fixar uma distância para identificar carros em movimento, grupos de pessoas andando nas ruas ou bandos de animais, não é trivial e muitas vezes se torna impossível principalmente quando minerando conjuntos de dados heterogêneos, ou seja, com tipos de objetivos móveis diferentes, como quando analisando tráfego com diferentes tipos de veículos e pedestres. Diversos trabalhos propõe a adoção da estratégia de busca por densidade, como um DBSCAN, em vez da busca por disco. Esses trabalhos defendem que uma busca por densidade resolve o problema da dificuldade em estabelecer um diâmetro para a busca e elimina a limitação do formato espacial em disco. Entretanto, mesmo esse tipo de abordagem requer um raio como entrada para os algoritmos, distância de conectividade entre os objetos, não resolvendo assim a dificuldade dessa parametrização.

Como contribuições este trabalho:

- apresenta um novo conceito de **descoberta de k-padrões de comovimento** (*discovery of k-co-movement patterns*), em que a principal ideia é adotar uma mineração exploratória exata apenas requerendo do usuário um número k de respostas desejadas para a busca, liberando-o de fornecer um parâmetro não trivial;

- define k_ϵ-***Flocks*** que, atuando sobre o parâmetro não trivial do Padrão *Flock*, representa os *k-flocks* de tamanho e diâmetros mínimos para uma dada janela temporal w, tal que dentre entre esses *k-flocks* não exista outro *flock* possível em w que tenha diâmetro menor que o diâmetro do *flock* mais extenso em $\mathcal{F}^k$;
- define o novo **Padrão** k_ϵ-***Flock*** (k_ϵ-*Flock Pattern*) para identificar todos o k_ϵ-*Flocks* de todas as janelas temporais em um conjunto de dados;
- apresenta o **Algoritmo *Top-Down***, exploratório exato para mineração de todos os k_ϵ-*Flocks* em uma janela w e o **Algoritmo *Top-Down* em Janela Deslizante** para *stream* ou conjuento de dados, sem a exigência do parâmetro de distância;
- define ***Filtered-***k_ϵ***-Flocks*** que representa todos os k_ϵ-*Flocks* de uma janela temporal w delimitados por um diâmetro limite;
- define o novo **Padrão *Filtered-***k_ϵ***-Flocks*** para identificar todos os *Filtered-*k_ϵ*-Flocks* de todas as janelas temporais de uma entrada;
- e, apresenta o **Algoritmo *Filtered Top-Down***, adaptação mais eficiente do *Top-Down* em Janela Deslizante, com prévio filtro de candidatos com o PSI, por meio de uma distância limite superior, para identificação de *Filtered-*k_ϵ*-Flocks* de todas as janelas temporais de uma entrada.

Com essas contribuições, o usuário consegue apartir de uma *stream* ou conjunto de dados desconhecido ou heterogêneo, utilizando o *Top-Down*, fazer uma análise exploratória para entendimento inicial do comportamento dos padrões mais significativos existentes, sem exigir um parâmetro de distância fixo predefinido. Ou, já com um conhecimento inicial do domínio de dados é possível, de forma mais eficiente, através do *Filtered Top-Down*, estabelecer um diâmetro máximo para filtrar e identificar apenas os k_ϵ-*Flocks* respeitando o limite dado (*Filtered-*k_ϵ*-Flocks*).

Como extensões a este trabalho, algoritmos similares para demais padrões podem ser implementados, inclusive aos baseados em densidade como o Padrão Comboio. A utilização do conceito de descoberta do k-padrões pode abranger outros parâmetros que não apenas a distância, como por exemplo a duração (números de instantes de tempo consecutivos) do padrão. Por fim, como outro trabalho futuro, a implementação de um algoritmo *Bottom-Up*, almejando otimização em processamento, não com a abordagem de partir do *flock* único mais extenso de uma janela temporal (*top-down*), mas sim partir dos grupos de objetos mais próximos no espaço, utilizando técnicas de expansão de vizinhança, expandindo os candidatos até chegarem ao tamanho mínimo.

REFERÊNCIAS

[1] SPACCAPIETRA, S. et al. A conceptual view on trajectories. *Data and Knowledge Engineering*, v. 65, n. 1, p. 126–146, apr 2008. ISSN 0169023X. Disponível em: <http://linkinghub.elsevier.com/retrieve/pii/S0169023X07002078>.

[2] TANAKA, P. S. *Algoritmos Eficientes para a Detecção On-Line do Padrão Floco em Bancos de Dados Trajetórias* . 85 p. Dissertação (Mestrado) — Universidade Estadual de Londrina, 2016. Disponível em: <http://www.bibliotecadigital.uel.br/document/?code=vtls000206562>.

[3] VIEIRA, M. R.; BAKALOV, P.; TSOTRAS, V. J. On-line discovery of flock patterns in spatio-temporal data. In: *Proceedings of the 17th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems - GIS '09*. New York, New York, USA: ACM Press, 2009. p. 286. ISBN 9781605586496. Disponível em: <http://portal.acm.org/citation.cfm?doid=1653771.1653812>.

[4] TANAKA, P. S.; VIEIRA, M. R.; KASTER, D. S. An Improved Base Algorithm for Online Discovery of Flock Patterns in Trajectories. *JIDM*, v. 7, n. 1, 2016. Disponível em: <https://seer.ufmg.br/index.php/jidm/article/view/1371/2647>.

[5] ROMERO, A. O. C. *Mining moving flock patterns in large spatio-temporal datasets using a frequent pattern mining approach*. 67 p. Tese (Doutorado) — University of Twente, 2011.

[6] YAN, Z. *Semantic Trajectories: Computing and Understanding Mobility Data*. Tese (Doutorado), 2011. Disponível em: <http://infoscience.epfl.ch/record/167178>.

[7] JEUNG, H. et al. Discovery of convoys in trajectory databases. *Proceedings of the VLDB Endowment*, v. 1, n. 1, p. 1068–1080, aug 2008. ISSN 2150-8097. Disponível em: <http://dl.acm.org/citation.cfm?doid=1453856.1453971>.

[8] WACHOWICZ, M. et al. Finding moving flock patterns among pedestrians through collective coherence. *International Journal of Geographical Information Science*, v. 25, n. 11, p. 1849–1864, 2011. ISSN 1365-8816. Disponível em: <http://dx.doi.org/10.1080/13658816.2011.561209>.

[9] LI, Z. et al. Swarm: Mining Relaxed Temporal Moving Object Clusters. *Proceedings of the VLDB Endowment*, v. 3, n. 1-2, p. 723–734, sep 2010. ISSN 21508097. Disponível em: <http://dl.acm.org/citation.cfm?doid=1920841.1920934>.

[10] LI, Y.; BAILEY, J.; KULIK, L. Efficient mining of platoon patterns in trajectory databases. *Data and Knowledge Engineering*, v. 100, n. Yuxuan Li, p. 167–187, 2015. ISSN 0169023X.

[11] HUANG, P.; YUAN, B. Mining massive-scale spatiotemporal trajectories in parallel: A survey. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, v. 9441, p. 41–52, 2015. ISSN 16113349.

[12] SILVA, T. L. C. da; ZEITOUNI, K.; MACEDO, J. A. de. Online Clustering of Trajectory Data Stream. In: *2016 17th IEEE International Conference on Mobile Data Management (MDM)*. IEEE, 2016. p. 112–121. ISBN 978-1-5090-0883-4. ISSN 15516245. Disponível em: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=7517785http://ieeexplore.ieee.org/document/7517785/>.

[13] PARENT, C. et al. Semantic trajectories modeling and analysis. *ACM Computing Surveys*, v. 45, n. 4, p. 42:1–42:32, 2013. ISSN 03600300. Disponível em: <http://dl.acm.org/citation.cfm?doid=2501654.2501656>.

[14] JEUNG, H.; SHEN, H. T.; ZHOU, X. Convoy queries in spatio-temporal databases. *Proceedings - International Conference on Data Engineering*, v. 00, p. 1457–1459, 2008. ISSN 10844627.

[15] 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*. Cham: Springer International Publishing, 2016. v. 9865, p. 310–321. ISBN 978-3-319-45835-9.

[16] WANG, Y.; LIM, E.-P.; HWANG, S.-Y. Efficient mining of group patterns from user movement data. *Data & Knowledge Engineering*, v. 57, n. 3, p. 240–282, jun 2006. ISSN 0169023X. Disponível em: <http://linkinghub.elsevier.com/retrieve/pii/S0169023X05000558>.

[17] LI, X. et al. Effective online group discovery in trajectory databases. *IEEE Transactions on Knowledge and Data Engineering*, v. 25, n. 12, p. 2752–2766, 2013. ISSN 10414347.

[18] ONG, R. et al. Parameter Estimation and Pattern Validation in Flock Mining. In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. [S.l.]: Springer, Cham, 2014. v. 8399 LNAI, p. 3–17. ISBN 9783319084060.

[19] BOGORNY, V. et al. CONSTAnT - A Conceptual Data Model for Semantic Trajectories of Moving Objects. *Transactions in GIS*, v. 18, n. 295179, p. n/a–n/a, 2013. ISSN 13611682. Disponível em: <http://doi.wiley.com/10.1111/tgis.12011>.

[20] ZHENG, Y. U. Trajectory Data Mining : An Overview. *ACM Trans. Intell. Syst. Technol.*, ACM, v. 6, n. 3, p. 1–41, may 2015. ISSN 21576904. Disponível em: <http://dl.acm.org/citation.cfm?id=2743025>.

[21] FENG, Z.; ZHU, Y. A Survey on Trajectory Data Mining: Techniques and Applications. *IEEE Access*, v. 4, p. 2056–2067, 2016. ISSN 2169-3536. Disponível em: <http://ieeexplore.ieee.org/document/7452339/>.

[22] ALVARES, L. O. et al. A model for enriching trajectories with semantic geographical information. *Proceedings of the 15th Annual ACM International Symposium on Advances in Geographic Information Systems (GIS '07)*, n. i, p. 22:1–22:8, 2007. Disponível em: <http://portal.acm.org/citation.cfm?doid=1341012.1341041>.

[23] ZHENG, Y.; ZHOU, X. *Computing with Spatial Trajectories*. New York, NY: Springer New York, 2011. v. 3. e932 p. ISBN 978-1-4614-1628-9. Disponível em: <http://link.springer.com/10.1007/978-1-4614-1629-6>.

[24] LACERDA, T.; FERNANDES, S. Scalable Real-Time Flock Detection. In: *2016 IEEE Global Communications Conference (GLOBECOM)*. IEEE, 2016. p. 1–7. ISBN 978-1-5090-1328-9. Disponível em: <http://ieeexplore.ieee.org/document/7842241/>.

[25] LI, Z. et al. MoveMine: Mining Moving Object Databases. In: *Proceedings of the 2010 international conference on Management of data - SIGMOD '10*. New York, New York, USA: ACM Press, 2010. p. 1203–1206. ISBN 978-1-4503-0032-2. ISSN 07308078. Disponível em: <http://portal.acm.org/citation.cfm?doid=1807167.1807319>.

[26] JENSEN, C. S.; LIN, D.; OOI, B. C. Continuous clustering of moving objects. *IEEE Transactions on Knowledge and Data Engineering*, v. 19, n. 9, p. 1161–1173, sep 2007. ISSN 10414347. Disponível em: <http://ieeexplore.ieee.org/document/4288137/>.

[27] LAUBE, P.; IMFELD, S. Analyzing relative motion within groups of trackable moving point objects. *Geographic information science*, p. 132–144, 2002. ISSN 16113349.

[28] LAUBE, P.; KREVELD, M. van; IMFELD, S. *Finding REMO–detecting relative motion patterns in geospatial lifelines*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004. 201–214 p. ISBN 978-3-540-26772-0. Disponível em: <http://link.springer.com/10.1007/b138045>.

[29] GUDMUNDSSON, J.; KREVELD, M. van; SPECKMANN, B. Efficient detection of motion patterns in spatio-temporal data sets. In: *Proceedings of the 12th annual ACM international workshop on Geographic information systems - GIS '04*. New York, New York, USA: ACM Press, 2004. p. 250. ISBN 1581139799. Disponível em: <http://portal.acm.org/citation.cfm?doid=1032222.1032259>.

[30] GUDMUNDSSON, J.; KREVELD, M. van. Computing longest duration flocks in trajectory data. *Proceedings of the 14th annual ACM international symposium on Advances in geographic information systems*, p. 35–42, 2006.

[31] BENKERT, M. et al. Reporting flock patterns. In: *Annual European Symposium on Algorithms (ESA)*. [s.n.], 2006. v. 6, p. 660–671. Disponível em: <http://link.springer.com/chapter/10.1007/11841036_59>.

[32] BENKERT, M. et al. Reporting flock patterns. *Computational Geometry*, v. 41, n. 3, p. 111–125, nov 2008. ISSN 09257721. Disponível em: <http://www.sciencedirect.com/science/article/pii/S092577210700106X>.

[33] ARIMURA, H.; TAKAGI, T. Finding All Maximal Duration Flock Patterns in High-dimensional Trajectories. *Manuscript, DCS, IST, Hokkaido University, Apr*, 2014.

[34] GENG, X. et al. Enumeration of complete set of flock patterns in trajectories. *Proceedings of the 5th ACM SIGSPATIAL International Workshop on GeoStreaming - IWGS '14*, v. 2014, n. i, p. 53–61, 2014. Disponível em: <http://dl.acm.org/citation.cfm?id=2676552.2676560>.

[35] TANAKA, P. S.; VIEIRA, M. R.; KASTER, D. S. Efficient Algorithms to Discover Flock Patterns in Trajectories. *XVI Brazilian Symposium on Geoinformatics (GEOINFO)*, v. 1, n. XVI, p. 56–67, 2015.

[36] GENG, X.; UNO, T.; ARIMURA, H. Trajectory pattern mining in practice: Algorithms for mining flock patterns from trajectories. *Proceedings of the 5th International Conference on Knowledge Discovery and Information Retrieval (KDIR '13)*, v. 2013, n. September, p. 143–151, 2013. Disponível em: <http://www.scitepress.org/PublicationsDetail.aspx?ID=N0R8w5f5GR0=&t=1>.

[37] TURDUKULOV, U. et al. Visual mining of moving flock patterns in large spatio-temporal data sets using a frequent pattern approach. *International Journal of Geographical Information Science*, v. 28, n. 10, p. 2013–2029, oct 2014. ISSN 1365-8816. Disponível em: <http://www.tandfonline.com/doi/abs/10.1080/13658816.2014.889834>.

[38] PHAN, N.; PONCELET, P.; TEISSEIRE, M. All in One: Mining Multiple Movement Patterns. *International Journal of Information Technology & Decision Making*, v. 15, n. 05, p. 1115–1156, sep 2016. ISSN 0219-6220. Disponível em: <http://www.worldscientific.com/doi/abs/10.1142/S0219622016500280>.

[39] UNO, T. et al. LCM: An Efficient Algorithm for Enumerating Frequent Closed Item Sets. *Fimi*, v. 90, 2003.

[40] UNO, T.; KIYOMI, M.; ARIMURA, H. LCM ver. 2: Efficient Mining Algorithms for Frequent/Closed/Maximal Itemsets. *Workshop on Frequent Itemset Mining*, 2004. Disponível em: <http://www.philippe-fournier-viger.com/spmf/LCM2.pdf>.

[41] UNO, T.; KIYOMI, M.; ARIMURA, H. LCM ver.3: Collaboration of Array, Bitmap and Prefix Tree for Frequent Itemset Mining. In: *Proceedings of the 1st international workshop on open source data mining frequent pattern mining implementations - OSDM '05*. New York, New York, USA: ACM Press, 2005. p. 77–86. ISBN 1595932100. Disponível em: <http://dl.acm.org/citation.cfm?id=1133916{\%}5Cnhttp://portal.acm.org/citation.cfm?doid=1133905.1133916http://portal.acm.org/citation.cfm?doid=1133905.1133>.

[42] ESTER, M. et al. A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. In: *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD-96), Portland, Oregon, USA*. Elsevier, 1996. p. 226–231. Disponível em: <https://ocs.aaai.org/Papers/KDD/1996/KDD96-037.pdf>.

[43] KALNIS, P.; MAMOULIS, N.; BAKIRAS, S. On Discovering Moving Clusters in Spatio-temporal Data. In: *Advances in Spatial and Temporal Databases SE - 21*. [S.l.]: Springer, Berlin, Heidelberg, 2005. v. 3633, p. 364–381. ISBN 978-3-540-28127-6.

[44] WANG, Y.; LIM, E.-P.; HWANG, S.-Y. On Mining Group Patterns of Mobile Users. In: *Database and Expert Systems Applications: 14th International Conference, DEXA 2003, Prague, Czech Republic, September 1-5, 2003. Proceedings*. Prague: Springer, Berlin, Heidelberg, 2003. p. 287–296. ISBN 978-3-540-40806-2. ISSN 03029743.

[45] SANCHES, D. E. et al. A Top-Down Algorithm with Free Distance Parameter for Mining Top-k Flock Patterns. In: *Lecture Notes in Geoinformation and Cartography*. Lund: [s.n.], 2018. part F3, p. 233–249. ISBN 9783319782072. ISSN 18632351. Disponível em: <http://link.springer.com/chapter/10.1007/978-3-319-78208-9_12>.

TRABALHOS PUBLICADOS PELO AUTOR

Trabalhos publicados pelo autor durante o programa.

1. Denis Evangelista Sanches, Luis O. Alvares, Vania Bogorny, Marcos R. Vieira and Daniel S. Kaster, **A Top-Down Algorithm with Free Distance Parameter for Mining Top-k Flock Patterns**, The Annual International Conference on Geographic Information Science (AGILE), 2018, Springer, Cham, p. 233-249, 978-3-319-78208-9 (Qualis CC 2016, B2)

2. Vitor Hugo Bezerra, Denis Evangelista Sanches, Daniel S. Kaster, **Recuperação de Dados por Interpolação para Algoritmos On-Line de Descoberta de Padrões *Flock***, XIII Brazilian Symposium on Information Systems (SBSI), 2017, p. 488-495, 978-85-7669-376-5 (Qualis CC 2016, B2)