ALGORITHMS FOR THE ANALYSIS OF SPATIO-TEMPORAL DATA FROM TEAM SPORTS

A thesis submitted in fulfilment of the requirements for the degree of Doctor of Philosophy in the School of Information Technologies at The University of Sydney

Michael John Horton

January 2018

Abstract

Modern object tracking systems are able to simultaneously record *trajectories*—sequences of time-stamped location points—for large numbers of objects with high frequency and accuracy. The availability of trajectory datasets has resulted in a consequent demand for algorithms and tools to extract information from these data. In this thesis, we present several contributions intended to do this, and in particular, to extract information from trajectories tracking football (soccer) players during matches.

Football player trajectories have particular properties that both facilitate and present challenges for the algorithmic approaches to information extraction. The key property that we look to exploit is that the movement of the players reveals information about their objectives through cooperative and adversarial coordinated behaviour, and this, in turn, reveals the tactics and strategies employed to achieve the objectives. While the approaches presented here naturally deal with the application-specific properties of football player trajectories, they also apply to other domains where objects are tracked, for example behavioural ecology, traffic and urban planning.

The research in this area is at a relatively early stage, and there is currently no consensus on the best approach to a number of key open problems. We present a detailed survey of the algorithmic approaches to mining sports trajectory data, and define a taxonomy for the tasks and problems that have been identified.

Within this taxonomy, we make several individual contributions. We consider the task of automatically classifying passes made during football matches according to their quality, and present a framework that accepts player trajectory data as input and

automatically makes such ratings with high accuracy. We find that the level of agreement of the assigned ratings between the automated classifier and an expert observer is similar to the level of agreement between two experts.

Next, we observe that trajectories are a particular class of a more general data structure of state sequences, and we present a method of summarising sets of state sequences using flow diagrams that are minimal in the number of nodes, and where each state sequence appears as a path in the flow diagram. We prove that an exact algorithm for this problem is computationally intractable except for small inputs, and furthermore show that the exact solution is hard to approximate. As such we present two heuristic algorithms that perform well experimentally, and we also demonstrate the utility of this approach on two use cases on football trajectory data.

We then consider two approaches to clustering of trajectories under the Fréchet distance. First, we investigate the problems of clustering and outlier detection as an integrated task, and present improved heuristic algorithms derived from two distinct integer program formulations of the problem. Both the algorithms are iterative and maintain their state in a set of auxiliary variables, and by monitoring these variables over the iterations of execution of the algorithm, time-series are captured. We claim that these time-series are an inherently two-dimensional view of the clustering and outliers that can be easily visualised and interpreted without suffering from the typical distortions implicit in low-dimensional visualisations of highly- or infinite-dimensional data.

Finally, we investigate the problem of directly clustering trajectories such that each cluster contains an low-complexity exemplar that is representative of the cluster. Each exemplar is a trajectory with a bounded number of location points, and thus is robust to the noise that is typical in many trajectory data sets. We formalise this problem and present an algorithmic framework that decomposes the problem into distinct simplification and clustering tasks. Using previously known algorithms for these tasks, we present a family of approximation algorithms for the trajectory clustering problem where the obtained clustering costs are bounded by a multiplicative factor of the optimal cost.

Statement of Originality

This is to certify that to the best of my knowledge, the content of this thesis is my own work. This thesis has not been submitted for any degree or other purposes.

I certify that the intellectual content of this thesis is the product of my own work and that all the assistance received in preparing this thesis and sources have been acknowledged.

Michael Horton

Acknowledgements

I would like to first acknowledge the financial support that I received from the Commonwealth of Australia under the Australian Postgraduate Award scheme, and from Data61, CSIRO under the NICTA Local Project Award and NICTA Research Project Award schemes. This support was gratefully received and helped to defray the costs of undertaking this research.

I would also like to thank the collaborators I had on various parts of my research—your intuition, knowledge and analytic ability left a deep impression on me. Namely: Boris Aronov, Mark de Berg, Kevin Buchin, Maike Buchin, Anne Driemel, Serge Gaspers, Herman Haverkort, Bernard Mans, Ali Mehrabi, Stefan Rümmele and Stef Sijben.

I had two very enjoyable and educational research visits, and thank Maike Buchin for hosting me at Ruhr-Universität Bochum for a week in 2014 and Mark de Berg for hosting a ten-week visit to TU Eindhoven in 2016.

Finally, I want to acknowledge and thank my supervisors, Joachim Gudmundsson and Sanjay Chawla, for the time, resources, knowledge, belief and the patience that they invested in me as I struggled through the challenges that our research presented. I very much appreciate your contribution to my education, and I could not have achieved a fraction of what I have without your help.

Michael Horton

Sydney, January 17, 2018

List of Publications

This thesis was based on these published works. I was the corresponding author and major contributor for all the works listed.

Chapter 2 is based on this survey paper.

- J. Gudmundsson and M. Horton, "Spatio-temporal analysis of team sports," *ACM Computing Surveys*, vol. 50, no. 2, 22:1–22:34, 2017. DOI: 10.1145/3054132

Chapter 3 was published as a journal paper and a short version as a conference paper.

- S. Chawla *et al.*, "Classification of passes in football matches using spatiotemporal data," *ACM Transactions on Spatial Algorithms and Systems*, vol. 3, no. 2, pp. 1–30, Aug. 2017. DOI: 10.1145/3105576
- M. Horton *et al.*, "Automated classification of passing in football," in *Proceedings of the 19th. Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining (PAKDD '15), Part II*, ser. Lecture Notes in Computer Science, vol. 9078, Springer, May 2015, pp. 319–330. DOI: 10.1007/978-3-319-18032-8_25

Chapter 4 is currently under review as a journal paper and a short version was published in conference proceedings.

- K. Buchin *et al.*, "Compact flow diagrams for state sequences," *Journal of Experimental Algorithmics*, vol. 22, pp. 1–23, Dec. 2017. DOI: 10.1145/3150525

- K. Buchin *et al.*, "Compact flow diagrams for state sequences," in *Proceedings of the 15th. International Symposium on Experimental Algorithms (SEA '16)*, ser. Lecture Notes in Computer Science, vol. 9685, Springer, Jun. 2016, pp. 89–104. DOI: 10.1007/978-3-319-38851-9_7

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

In recent years, advances in object tracking technologies have resulted in a large number of datasets of *trajectories*—sequences of time-stamped location points of a moving object. The availability of this data has motivated an interest in algorithms to process and extract information from the trajectory datasets.

In this thesis, we present several techniques for mining trajectories created by tracking players during football (soccer) matches. These trajectories have distinctive properties that both facilitate and pose challenges to the task of extracting information from them. The trajectories tend to be dense, in the sense that the location of the players are sampled regularly at high frequency and accuracy, and the trajectories are constrained to the football pitch. The trajectories are long running as the players are participating for long periods of the match, and there is always a fixed number of players on the pitch—and hence active trajectories—at all times. Furthermore, the movement of the players has an underlying structure in that team-mates are cooperating, and opponents are responding in an adversarial manner, and this structure is inherent in the resulting trajectories. However, the trajectories may be very "un-smooth", as players change their direction and velocity frequently, and also the trajectories will often cross over themselves and the other trajectories.

Given the distinctive properties of the trajectories, and the public and commercial interest in sports in general, we believe that any contributions in this area will have a

significant impact. With this rationale in mind, this thesis contains a systematic survey of the various research activities that have been undertaken to extract information from player trajectory data, and then presents four contributions that we have made in this area.

1.1 Background

Sports are an important recreational and cultural activity in many societies. They attract participants who compete in, spectate on and derive their livelihood from sports. These stakeholders naturally seek to better understand the nature of sports performance in order to improve performance and enjoyment.

There are several sporting codes that can be classified as *invasion sports* in that they share a common structure: two teams are competing for possession of a ball (or puck) in a constrained playing area, for a given period of time, and each team has simultaneous objectives of scoring by putting the ball into the opposition's goal, while also defending their goal against attacks by the opposition. The team that has scored the greatest number of goals at the end of the allotted time is the winner. Football (soccer), basketball, ice hockey, field hockey, rugby, Australian Rules football, American football, handball, and Lacrosse are all examples of invasion sports.

Aside from the benefits to participants' health and enjoyment, sporting events are significant business, with an estimated revenue of $64 billion in 2009, according to a report by the management consultancy firm ATKearney [58]. Furthermore, invasion sports comprise the majority of this revenue: football 43%; American football 13%; basketball 6% and ice hockey 4%. The value of these sports events is primarily derived from the willingness of fans to watch these events—either live or through television broadcasts.

In all sporting competitions, and particularly at the elite level, participants want to be successful and will seek ways to improve their performance. This can be achieved through improvements in fitness, injury management, training and diet regimes, and

also by improving strategies and techniques to be used during matches. It this last area in which our research is located, in particular the analysis of the events occurring during matches—known as *sports analysis*.

Moreover, television broadcasters have realised that coverage of sporting events can be enhanced by providing analysis of the action, often from ex-professionals using visual tools to describe key instances and sequences during the play. There is also an apparent appetite for statistics from fans, and there are many websites devoted to providing statistical information [70], [157], [177].

The systematic capture and analysis of data on sports play is not a new phenomenon —box-scores for summarising baseball matches have existed since 1860 [129], and manual notation of events in football matches has occurred from the 1950s [167]. The limitations inherent in observing and recording events by hand meant that the subsequent analysis was limited to the major discrete events that occurred during the game. In practice, the utility of this data depended on the particular sport, and sports with a more linear structure in the order that events occur were more amenable to analysis. This was most notable in baseball, where the sequence of events has a linear progression: a designated player pitches the ball to a batter who, in turn, attempts to hit the ball which will subsequently be fielded by another player, who returns the ball to the pitcher. The structure of the game, and the type of data captured, facilitated the development of statistical models that had a significant impact on the game, a phenomenon that was detailed in the book *Moneyball: The Art of Winning an Unfair Game* [134].

However, invasion sports did to benefit from a similar improvement in analytic methods with the availability of manually captured statistics. The fact that these games are inherently less structured contributes to the difficulty in data capture and analysis of these sports. For example, during a football match, there are 22 players on the field, and they are able to move around the entire playing-area. Moreover, significant events are not limited to the player in possession of the ball, and matches are a continuous contest between teams to induce the opposing team into inferior spatial configurations on the field. Thus, the positioning and movement of all players is significant, and this

presents a challenge to manually capture all the significant events. The difficulty in capturing sufficient data also made it impractical to do at scale, for example, a notable early contribution by Intille and Bobick [116] used machine learning models to identify set-plays in American football from trajectories that were manually created from video footage. The input data set contained player trajectories for 11 players from just 29 short plays. Moreover, human observation can be unreliable—experimental results in Franks and Miller [76] showed that the expert observers' recollection of significant match events is as low as $42\,\%$.

The unstructured nature of invasion sports events, combined with the challenges of accurately capturing data, meant that progress in developing sophisticated analysis was limited until the early 2000s. Prior to this period, much of the analysis comprised simple frequency analysis, and was primarily concerned with *what* happened, and did not consider *where*, by *whom* or *why* [143]. The situation changed with the development of automated object tracking systems that were able to capture and record the movement of players and the ball during a match. These systems records the *trajectory* of each player as a sequence of *location points* containing the time-stamp and the location of the player in two or three dimensions, see Section 1.2.

There are several techniques used to capture trajectories. *Optical tracking systems* use fixed cameras to capture the player movement, and the images are subsequently processed to compute the trajectories [25]. There are several commercial vendors who supply tracking services to professional sports teams and leagues [49], [115], [179], [180]. An alternative approach is that that of *device tracking systems* that rely on radio transmitters to infer location, and the devices are attached to the players' clothing or embedded in the ball or puck. These systems can be based on GPS [40] or RFID [178] technology.

The adoption of this technology and the availability to researchers of the resulting data has varied amongst the different sporting codes and is driven by various factors, particularly commercial and technical. There is a cost associated with installing and

maintaining such systems, and while some leagues mandate that all stadium have systems fitted, such as the National Basketball Association (NBA), in others the individual teams will bear the cost, such as the English Premier League, and they view the data as commercially sensitive. Furthermore, the nature of some sports present technical challenges to automated systems, for example, sports such as rugby and American football have frequent collisions present difficulties to optical tracking systems that rely on uniquely identifying each player in match video.

However, as we show in Chapter 2, the availability of large trajectory datasets enabled a multitude of research efforts in a variety of fields with the objective of extracting information about the sports play—collectively known as *spatio-temporal sports analysis*. Today, there are several venues exclusively focused on sports analysis, in particular the *MIT Sloan Sports Analytics Conference* [152] and *SIGKDD 2016 Workshop on Large Scale Sports Analytics* [144]; and there have been special issues of journals on the computational aspects of sports analytics [2], [27], [83].

More generally, the data produced by object tracking technologies has produced research in many other application domains, including traffic and route planning [39], map-making [6], behavioural ecology [5], and urban planning [67]. See the book *Computing with Spatial Trajectories* [200] for a coverage of the general research efforts in trajectory analysis.

Thus, the current interest and conditions offer the opportunity for innovative research contributions to have a meaningful impact in this field. It was in this environment that we conducted the research included in this thesis.

Chapter 2 contains a survey of the research efforts across a number of areas with the objective of extracting meaningful information from player trajectory data. Furthermore, we propose a taxonomy to describe the problems inherent in spatio-temporal sports analysis. In Chapters 3–6 we make four individual contributions to address some of the problems identified within the taxonomy. In these, we develop and apply techniques that all take as input the spatio-temporal data described in Section 1.2 and in each case we experimentally evaluate the techniques using trajectory data from four

football matches.

1.2 Spatio-temporal Sports Data

The research surveyed in Chapter 2 and the contributions in Chapters 3–6 are all based on *spatio-temporal data*, the defining characteristic of which is that it is a sequence of samples containing the time-stamp and location of some phenomena.

The structure of the spatio-temporal data captured across the various sports in this domain have largely coalesced into two distinct types: *object trajectories* that capture the movement of players or the ball; and *event logs* that record the location and time of match events, such as passes, shots or fouls. The two data sets capture different aspects of the activities that occur during play, and although they can be used individually, provide a richer explanation of the game when used in combination. For example, the spatial formation in which a team arranges itself in will be apparent in the set of player trajectories. However, the particular formation used may depend on whether the team is in possession of the ball, which can be determined from the event log. On the other hand, a *shot at goal* event contains the location from where the shot was made, but this may not be sufficient to make a qualitative rating of the shot. Such a rating should consider whether the shooter was closely marked by the opposition defence, and the proximity of attacking team-mates—properties that can be interpolated from the player trajectories.

When working with trajectory data, it is often necessary to have some objective measure of similarity or distance between two trajectories, and several common distance measures that are mentioned in this manuscript are described in Subsection 1.2.5.

1.2.1 Preliminaries

In order to define the trajectories and event logs in context, we need the following. Let $M = \{m_1, \ldots, m_{n_M}\}$ be the set of matches for which data is available. In the sports that we consider, matches are temporally divided into *periods*, e.g. in football

a match consists of two halves, and in basketball a match has four quarters. Let $W = \{w_1, \ldots, w_{n_W}\}$ be an enumeration of the periods in a single match.

We require a global clock that is consistent across trajectories and events, and for this we use a *match clock* $S = \{s_1, \ldots, s_{n_S}\}$. For each period of a match, the match clock is initially 0 and increments by a fixed amount, e.g. for a football match at the beginning of each half the clock will be 0 and increments by $0.1\,\mathrm{s}$ until the half ends after approximately $45\,\mathrm{min}$.

Next, each trajectory is generated by the movement of an object such as a player, a match official or the ball, and events that occur in the event log may involve one or more players, and thus we need to be able to associate the objects that generate trajectories with the objects associated with events. Let U be the set of all objects that have trajectories, and $P = \{p_1, \ldots, p_{n_P}\} \subseteq U$ be the set of all players.

Finally, let V be the set of available event-types, such as *Touch*, *Pass*, *Foul*, etc. A full list of the football event types used in the experimental data sets is described in Subsection 1.2.6 is in the appendix, Section A.1.

1.2.2 Object Trajectories

The primary data generated by object tracking systems are movement traces for each player and for the ball. The location of the player (or ball) is sampled regularly to generate a sequence of *location points*, either in $\mathbb{R}^2$ or $\mathbb{R}^3$ and annotated with a time-stamp. This sequence is commonly called a *trajectory*.

Formally, a trajectory τ in $\mathbb{R}^2$ is defined as a sequence of triples $\langle (s, x, y) \rangle$ such that each triple is a location point denoting that the object $p \in U$ that generated the trajectory was at the planar location $(x, y) \in \mathbb{R}^2$ at time $s \in S$. Geometrically, a trajectory is a polygonal path with vertices (x, y) that are indexed by the time-stamp s. Let $\mathcal{T}$ be the set of all trajectories. A trajectory in $\mathbb{R}^3$ may be defined in a similar manner, and the functions defined subsequently can be derived in the same way for objects in $\mathbb{R}^3$.

We use the notation $\tau[s]$ to reference the location point occurring at time-stamp s

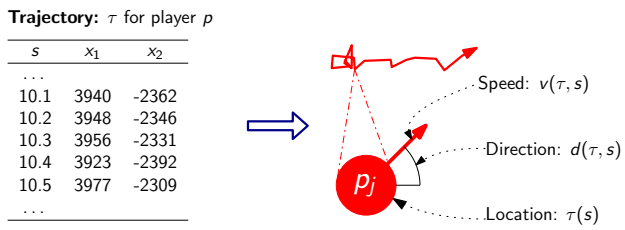


Figure 1.1: Illustration of a trajectory and the functions on location points. Each trajectory is a sequence of location points, and these can be used to extrapolate the basic geometry of a player at a given time-stamp. Similarly, the geometry of events such as the *pass* shown, can be computed from the trajectories of the involved players.

in trajectory τ. Similarly, let $\tau[s_1, s_2]$ be the subsequence of τ between times s_1 and s_2.

Given this representation, some basic attributes of the tracked object can be easily computed, in particular the *location*, *direction* and *speed*. We use the following functions. The location of a player p at time s is $\tau[s]$. Let $d\colon \mathcal{T} \times S \to [-\pi, \pi)$ be a function that, given a trajectory τ for player p and time-stamp s, returns the direction p is facing relative to the positive x-axis. Similarly, $v\colon \mathcal{T} \times S \to \mathbb{R}$ computes the velocity of the player at time $s \in S$ as distance between a location point and its successor, divided by time elapsed between the point and successor. See Figure 1.1 for a simple illustration of a trajectory and the functions that access the geometry of the trajectory.

Typically, player trajectories are *dense*, with sampling rates of 10 to $25\,\mathrm{Hz}$. Furthermore, the trajectories have several distinctive properties that may be exploited for extracting information, or may present challenges for the problem at hand.

The trajectories are *spatially constrained* in the sense that the players spend a majority of their time within the playing area. Moreover, the trajectories are *long running*—a player will typically be active for long periods of the match, particularly in sports such as football where players may be active for the entire match. Moreover, the players will be constantly changing their direction and velocity, and thus the distance between contiguous location points can vary, and the curvature of the trajectory is unbounded. These properties combine to make such trajectories very "messy", and difficult to interpret visually. An example of a trajectory of a player for one half of a football match is shown in Figure 1.2.

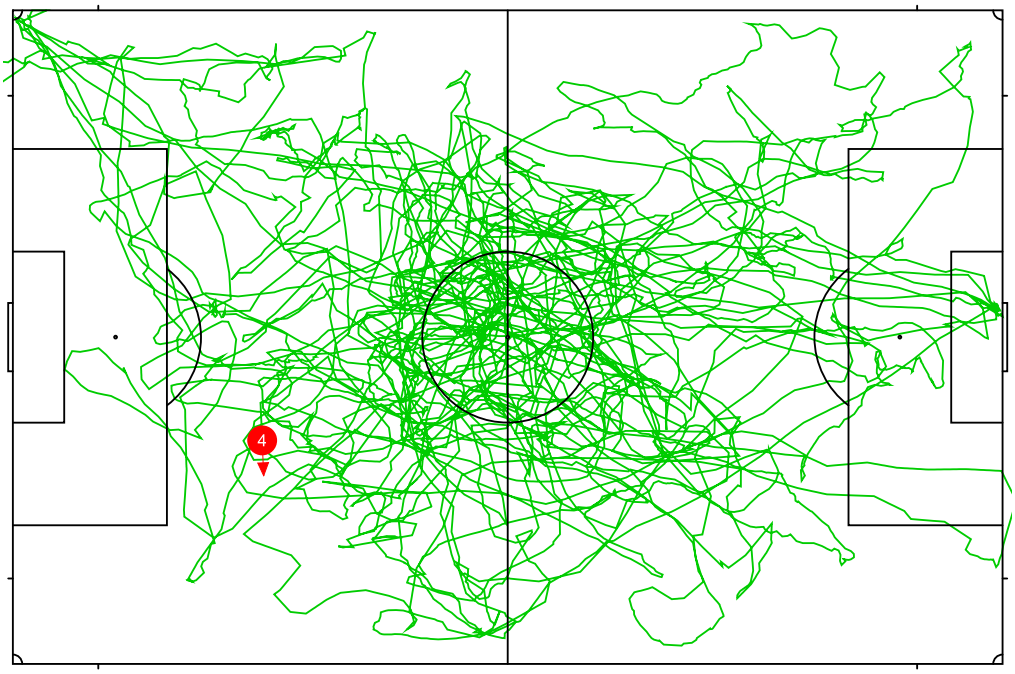


Figure 1.2: Example player trajectory during one half of a football match. The trajectory is not smooth, and crosses itself frequently, and this makes it difficult to visually extract any meaning from it.

1.2.3 Event Logs

Event logs are a sequence of significant events that occur during a match. Events can be broadly categorised as *player events* such as passes and shots; and *technical events* that are typically instigated by the referee or umpire, for example fouls, time-outs, and start/end of period, see Figure 1.3.

An event log ε is a sequence of triples $\langle(s, v, R)\rangle$ where each triple is an event where s is the time-stamp of the event, v is the event type and $R \subset P$ is a set of players who are involved in the event, and let $\mathcal{E}$ be the universe of all possible event logs. Similarly to trajectories, let $\varepsilon[s]$ be the event occurring at time s and let $\varepsilon[s_1, s_2]$ be the subsequence of events between s_1 and s_2.

The event logs provide details about the discrete events that occur during a match, and also allow the duration of the match to be segmented into intervals of different types. For example, a match could be subdivided into intervals when the play is active and when it is stopped for fouls, injuries, etc. The intervals of active play could be

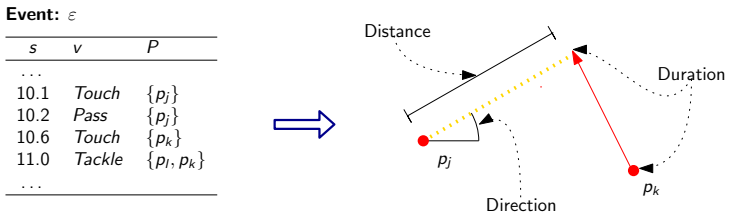


Figure 1.3: Illustration of an event log ε. An event $e_k \in \varepsilon$ has a time-stamp, event type and a list of zero or more associated players. This diagram on the right shows a *Pass* event, and illustrates that some properties of the event can be derived from the event and its successor event.

further subdivided into periods when a single team is in control of the ball, or when a single player is in possession. The start and end points of these intervals can be identified by specific events and possibly the player associated with the event.

Event logs are qualitatively different from the player trajectories in that they are not dense—samples are only captured when an event occurs—however they can be semantically richer.

1.2.4 Mappings

In order to identify the trajectories and event log that apply to a given match period, we require the following mapping functions. Let $T\colon M \times W \times P \to \mathcal{T}$ be a mapping that, for a given match, period and player the mapping will return the player's trajectory. Next, let $E\colon M \times W \to \mathcal{E}$ be the mapping from a given match and period to the corresponding event log. Let $\mathcal{M}_{\mathcal{T}}$ and $\mathcal{M}_{\mathcal{E}}$ be the set of trajectory and event mappings, respectively.

For convenience, we use the notation $T_{m,w,p} \equiv t(m, w, p)$ for the trajectory generated by player p during period w of match m. Similarly, let $E_{m,w} \equiv e(m, w)$ be the event log for period w of match m. It is also convenient to access the trajectories for a given match period, so let $T_{m,w} = \{t(m, w, p) \mid p \in P\} \setminus \{\varnothing\}$ be the set of trajectories for period w of match m.

1.2.5 Distance Measures

A fundamental requirement of many techniques for extracting information from trajectories is to be able to measure the similarity between trajectories, and a common approach is to compute the *distance* between two trajectories, as a measure of the (inverse) similarity.

There have been several measures proposed. The *Hausdorff distance* [103] is perhaps the most obvious measure, which computes, for two trajectories τ and τ', the largest distance over all points $p \in \tau$ to its closest point $q \in \tau'$:

$$d_H(\tau, \tau') = \max \left\{ \sup_{p \in \tau} \inf_{q \in \tau'} |p - q|, \sup_{q \in \tau'} \inf_{p \in \tau} |q - p| \right\}$$

where, $|\cdot|$ is the Euclidean distance. The Hausdorff distance can be efficiently computed, however it suffers from the drawback that the mapping of points from τ to τ' is not a continuous 1–1 mapping and can obtain small distances for apparently dissimilar curves [7].

A more appropriate measure the distance between trajectories is the Fréchet distance [78]. The Fréchet distance between two trajectories τ and τ' is defined as:

$$d_F(\tau, \tau') = \inf_{\mu} \max_{p \in \tau} |p - \mu(p)|.$$

The function $\mu \colon \tau \to \tau'$ is a bicontinuous function that maps each point $p \in \tau$ to a corresponding point in τ'. The start of τ is mapped under μ to the start of τ' and as p progresses along τ, $\mu(p)$ also progresses along τ'.

The exact Fréchet distance can be computed in $\mathcal{O}(n^2 \log n)$ where n is the number of location points in the longer of τ and τ', using the algorithm by Alt and Godau [7]. Buchin *et al.* [34] have recently improved the running time to $O(n^2(\log \log n)^2)$. The running time is somewhat prohibitive, particularly for trajectories with a large number of location points, and thus the discrete Fréchet distance is often used as a substitute.

The *discrete Fréchet distance* [68] computes the distances only between location

Table 1.1: Summary statistics for the matches used in experiments in Chapters 3–5.

Match Date	Home Team	Away Team	Period	# Trajs.	Mean # Loc. Pts	# Events
10-01-2009	Arsenal	Bolton Wanderers	first half	23	26,930	1563
10-01-2009	Arsenal	Bolton Wanderers	second half	26	24,374	1207
11-02-2008	Arsenal	Blackburn Rovers	first half	22	27,891	1413
11-02-2008	Arsenal	Blackburn Rovers	second half	25	25,172	1647
15-11-2008	Arsenal	Aston Villa	first half	22	27,603	1269
15-11-2008	Arsenal	Aston Villa	second half	25	25,960	1386
19-04-2008	Arsenal	Reading	first half	22	27,790	1262
19-04-2008	Arsenal	Reading	second half	28	22,631	1298
Total				193	208,354	11,045

points on τ and τ', and it provides a good approximation of the Fréchet distance when the trajectories are dense. It can be computed in $\mathcal{O}(n^2)$ time using a dynamic programming algorithm. The discrete Fréchet distance is used in Chapters 5 and 6 to cluster trajectories.

1.2.6 Experimental Data set

Chapters 3–5 include experimental results, all of which are based on trajectory and event data from four football matches. This data, provided by STATS LLC [180], is of four matches played by Arsenal Football Club at the Emirates Stadium in the English Premier League season in 2008.

The data set contains trajectories for every player that participates in each half of each of the four matches, and an event log for each match. The trajectories are uniformly sampled at $10\,\mathrm{Hz}$ and have a resolution of $10\,\mathrm{cm}$. A selection of summary statistics on the input data sets is included in Table 1.1.

The data set contains additional meta-data about the matches, teams and players, detailed in Appendix A.

1.3 Organisation and Contributions

This thesis contains five chapters which each contain a discrete research contribution.

Chapter 2 contains a systematic literature review of research efforts that have been undertaken using spatio-temporal data from team sports, as is described in Section 1.2,

with the objective of providing tools and data structures for analysis. We observe that research in this field is at a relatively early stage, and that for many problems there is not yet a consensus on the best methods to solve them. We define a taxonomy to classify the types of problems and the techniques that have been proposed to solve them. The subsequent chapters in this thesis present four contributions to address problems identified within this taxonomy.

In Chapter 3 we consider the problem of designing a classifier to rate passes made in football matches with a label of *Good*, *OK* or *Bad*. We started from the thesis that much of the information required to make the pass ratings is available in the trajectory signal. Our intuition was that using complex data structures derived from computational geometry would enable domain football knowledge to be included in the model by computing metric variables in a principled and efficient manner. We designed a model that computes a vector of predictor variables for each pass made, and uses machine learning techniques to determine a classification function that can accurately rate passes based only on the predictor variable vector. Experimental results showed that the learned classification functions can rate passes with $90.2\,\%$ accuracy. The agreement between the classifier ratings and the ratings made by a human observer is comparable to the agreement between the ratings made by human observers, and suggests significantly higher accuracy is unlikely to be achieved. Furthermore, we show that the predictor variables computed using methods from computational geometry are among the most important to the learned classifiers.

Chapter 4 introduces the concept of using a flow diagram to compactly represent the segmentation of a large number of state sequences according to a set of criteria. We argue that this flow diagram representation is an effective way to summarise segmentations of a large number of state sequences. In essence, the objective is to generate a flow diagram with a minimum number of nodes that models a segmentation of the states in the input sequences. For a small number of state sequences we present efficient algorithms to compute a minimal flow diagram. For a large number of state sequences we show that it is unlikely that efficient algorithms exist. Specifically, the

problem is $W[1]$-hard if the number of state sequences is taken as a parameter. We introduce several heuristics for this problem. We argue about the usefulness of the flow diagram by applying the algorithms to two problems in sports analysis, and evaluate the performance of our algorithms on a football data set and synthetic data.

In Chapter 5 we present a novel visualisation technique for exemplar-based clustering and outlier detection. We examine two well-known heuristic algorithms for this problem, each of which is based on an integer program formulation of the facility location with outliers problem: a Lagrangian heuristic and an affinity propagation algorithm. Although these formulations and the derived algorithms are quite different, they both seek to optimise a set of *dual variables* using gradient-based methods. By capturing the values of these dual variables at each iteration of the algorithm, a collection of time-series are produced for each variable. These time-series are inherently two-dimensional and thus can be easily visualised, and the visualisations are useful for exploring both the operation of the algorithms and the structure of the input data that is clustered. We demonstrate their utility of the visualisations by identifying an issue with the design of an existing Lagrangian heuristic algorithm for this task, and we modified the algorithm so that it obtained improved numerical stability and objective function value. We evaluate the new formulations of these algorithms experimentally on synthetic data, and then consider a case-study when trying to cluster trajectories under the Fréchet distance. There are difficulties in visualising trajectory clusters in $\mathbb{R}^2$, and we show how the proposed dual variable visualisation are a useful tool for gaining understanding of the computed clusters.

Chapter 6 also looks at the problem of clustering trajectories under the Fréchet distance, however in this case from a theoretical perspective. In particular, we are interested in *exemplar-based clustering* where each cluster has an associated exemplar—a trajectory that is in some way representative of the cluster. Typically, the exemplar is a trajectory from the input that belongs to the cluster, however this approach has the disadvantage that if the input is noisy, then the obtained exemplars will also be noisy. As an alternative, we investigate the (k, ℓ)-CENTER problem that partitions the input

trajectories into k clusters, and where each exemplar is a trajectory of bounded number of location points ℓ, and this encourages the exemplars to be smooth. We present a framework that decomposes the problem into sub-problems of curve simplification and k-center clustering, which have both been well studied. Using existing algorithms for the sub-problems we are able to obtain a family of approximation algorithms for the (k, ℓ)-CENTER problem that bound the obtained clustering cost by a multiplicative factor of the optimal cost.

Chapter 2

Spatio-temporal Sports Analysis: A Survey

The availability of large spatio-temporal datasets has motivated research into algorithms and techniques to process and extract information from the data about the underlying objectives, strategies and tactics of the players that created them. This chapter contains a comprehensive and systematic survey of these research efforts. To date, the majority of data sets available for research are sourced from football and basketball, and the research we surveyed reflects this, see Figure 2.1.

The research included in this survey has come from a variety of communities, including machine learning, network science, geographic information science, computational geometry, computer vision, complex systems science, statistics and sports science. There has been a consequent diversity of methods and models used in the research, and our intention in writing this survey was to provide an overview and framework on the research efforts to date.

We used the following criteria to demarcate the types of research considered for this survey:

1. We consider **team-based invasion sports**.

2. The model used in the research has **spatio-temporal data** as its primary input.

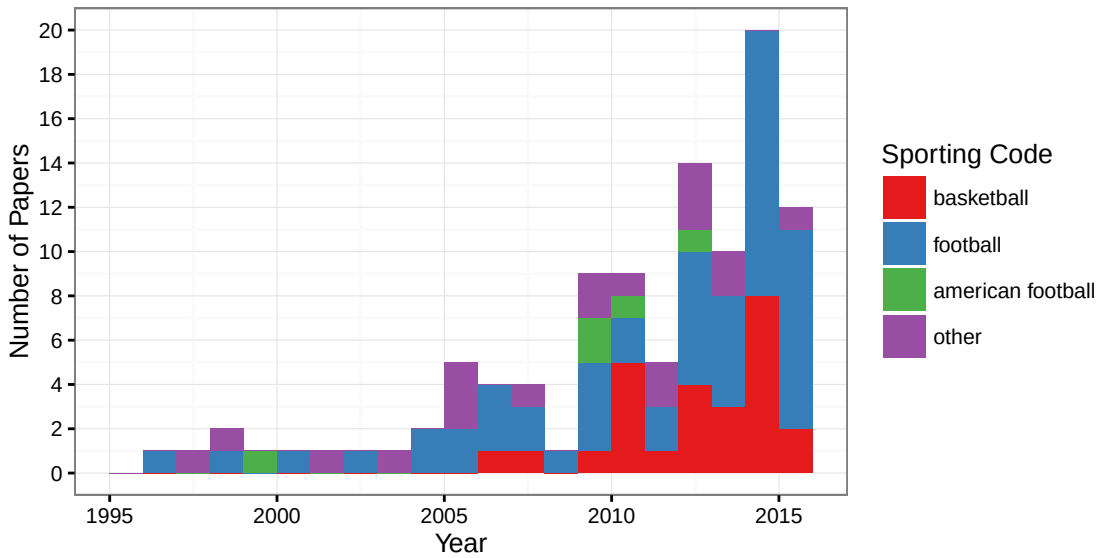


Figure 2.1: Spatial sports research papers cited in this survey, by year, 1995–October 2015, divided by sporting code. There has been a significant increase in papers published in this area as data has become available for researchers, particularly in football and basketball.

3. The model performs some **non-trivial computation** on the spatio-temporal data. In other words, a novel algorithmic approach is presented or applied.

This chapter contains the following sections, summarised in Figure 2.2. The models and techniques surveyed are all based on the object trajectories and event logs from matches, detailed in Section 1.2. Section 2.1 describes approaches that have been used to subdivide the playing area into regions that have a particular property. The playing area may be discretized into a fixed subdivision and the occurrences of some phenomena counted, for instance, a player occupying a particular region or a shot at goal occurring from that region, producing an *intensity map* of the playing area. Subdivisions of the playing area that are based on areas dominated by particular players has also been used in several papers.

In Section 2.2, we survey approaches that represent temporal sequences of events as *networks* and apply network-theoretic measures to them. For example, sequences

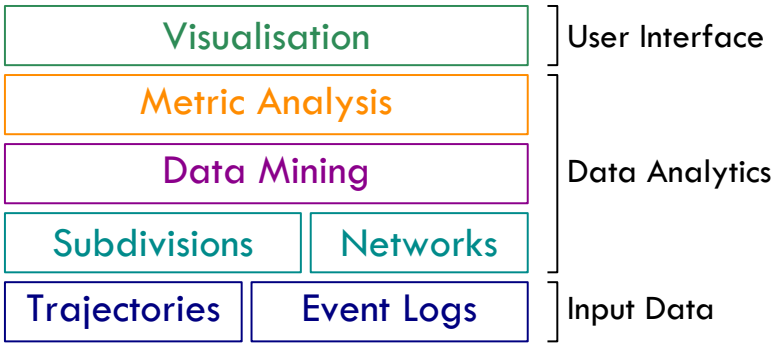


Figure 2.2: Summary of the major approaches surveyed, each approach corresponding to a section of this chapter. The techniques described in a particular layer of this diagram can be used as input to a technique described in a higher layer.

of passes between players can be represented as a network with players as the vertices, edges between vertices where a pass occurred between the associated players, edge weights denoting the frequency of passes between pairs of players Using this *passing network* various network measures can be computed to quantify the passing performance.

Section 2.3 is a task-oriented survey of the approaches to uncover information inherent in the spatio-temporal data using *data mining* techniques. Furthermore, several papers define metrics to measure the performance of players and teams, and these are discussed in Section 2.4. Finally, we detail the research into *visualisation* techniques to succinctly present metrics of sports performance in Section 2.5.

Table 2.1 summarises the techniques surveyed in this chapter and indicates the particular sports that they have been applied to. In Section 2.6, we discuss the factors may be considered when applying these techniques to other sports.

2.1 Playing Area Subdivision

The player trajectories and event logs, described in Section 1.2, are both low-level representations, and can be challenging to work with. One way to deal with this issue is to discretize the playing area into regions and assign the location points contained

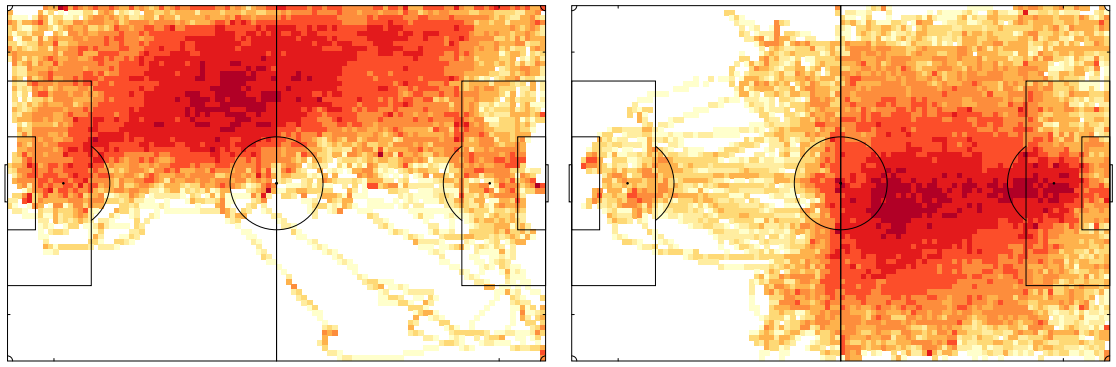
Table 2.1: Summary of the approaches and techniques described in this survey and the sports that they have been applied to.

	Football	Basketball	Field Hockey	Ice Hockey	American Football	Handball
2.1. Playing Area Subdivision						
2.1.1. Intensity Matrices and Maps	⋆	⋆				
2.1.2. Low-rank Factor Matrices		⋆				
2.1.3. Movement Models and Dominant Regions	⋆	⋆				
2.2. Network Techniques for Team Performance Analysis						
2.2.1. Centrality	⋆	⋆				
2.2.2. Clustering Coefficients	⋆	⋆				
2.2.3. Density and Heterogeneity	⋆					
2.2.4. Entropy, Topological Depth, Price-of-Anarchy and Power Law Distributions		⋆				
2.3. Data Mining						
2.3.1. Applying Labels to Events	⋆	⋆				
2.3.2. Predicting Future Event Types and Locations	⋆	⋆				
2.3.3. Identifying Formations	⋆	⋆	⋆			
2.3.4. Identifying Plays and Tactical Group Movement	⋆				⋆	
2.3.5. Temporally Segmenting the Game	⋆	⋆				⋆
2.4. Performance Metrics						
2.4.1. Offensive Performance	⋆	⋆				
2.4.2. Defensive Performance		⋆				
2.5. Visualisation	⋆	⋆		⋆		

in the trajectory or event log to a discretized region. The frequency—or intensity—of events occurring in each region is a spatial summary of the underlying process, alternatively, the playing area may be subdivided into regions such that each region is dominated in some sense by a single player, for example by the player being able to reach all points in the region before any other player. There are a variety of techniques for producing playing area subdivisions that have been used in the research surveyed here, and are summarised in this section.

2.1.1 Intensity Matrices and Maps

Spatial data from team sports have the useful property that they are constrained to a relatively small and symmetric playing area—the pitch, field or court. The playing area may be subdivided into regions and events occurring in each region can be counted to produce an intensity matrix, and can be visualised with an intensity map, see Figure 2.3. This is a common preprocessing step for many of the techniques described in subsequent sections.

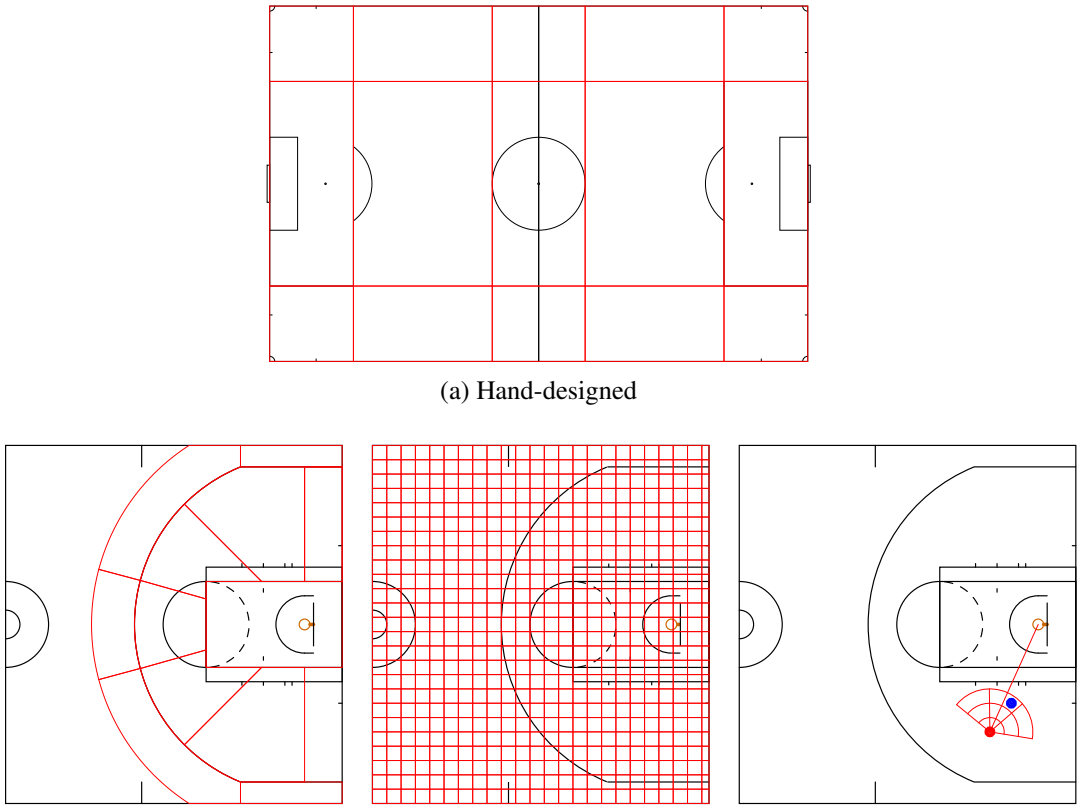


(a) Left-back (b) Striker

Figure 2.3: Example intensity maps showing areas of the football pitch that the player's occupy. The player trajectories have been oriented such that the play is from left to right. (a) The left-back is positioned on the left of the field, but is responsible for taking attacking corner-kicks from the right. (b) The striker predominantly stays forward of the half-way line, however will retreat to help defend corner-kicks.

When designing a spatial discretization, the number and shape of the induced regions can vary. A common approach is to subdivide the playing area into rectangles of equal size [16], [22], [41], [74], [139], [154], [175], for example see Figure 2.4(c). However, the behaviour of players may not vary smoothly in some areas. For example: around the three-point line on the basketball court, a player's propensity to shoot varies abruptly; or the willingness of a football defender to attempt a tackle will change depending on whether or not they are inside the penalty box. The playing area may be subdivided to respect such predefined assumptions of the player's behaviour. Camerino *et al.* [37] subdivides the football playing-area into areas that are aligned with the penalty box, see Figure 2.4(a), and interactions occurring in each region were counted. Similarly, Maheswaran *et al.* [147] and Goldsberry and Weiss [90] define subdivisions of the basketball half-court that conforms with the three-point line and is informed by intuition of shooting behaviour, see Figure 2.4(b).

Transforming the playing area into polar space and inducing the subdivision in that space is an approach used in several papers. This approach reflects the fact that player behaviour may be similar for locations that are equidistant from the goal or basket. Using the basket as the origin, polar-space subdivisions were used by Reich *et al.* [168]



(a) Hand-designed

(b) Hand-designed (c) Cartesian grid (d) Polar grid

Figure 2.4: Examples of subdivisions used to discretize locations: (a), (b) hand-designed subdivision reflecting expert knowledge of game-play in basketball [147] and football [37]; (c) subdivision of court into unit-squares [41]; (d) polar subdivision where origin is centred on ball-carrier and grid is aligned with the basket [198].

and by Maheswaran *et al.* [146]. Yue *et al.* [198] used a polar-space subdivision to discretize the position of the players marking an attacking player. Under this scheme, the location of the attacking player was used as the origin, and the polar space aligned such that the direction of the basket is at $0°$, see Figure 2.4(d).

Given a subdivision of the playing area, counting the number of events by each player in each region induces a discrete spatial distribution of players' locations during the match. This can be represented as an $\mathbb{R}^{n \times v}_{\geq 0}$ intensity matrix containing the counts $\mathbf{X}$ for n players in each of the v regions of the subdivision. The event $\mathbf{X}$ may be the number of visits by a player to the region, e.g. Maheswaran *et al.* [147] used the location points from player trajectories to determine whether a cell was visited.

Bialkowski *et al.* [17] used event data such as passes and touches made by football players to determine the regions a player had visited.

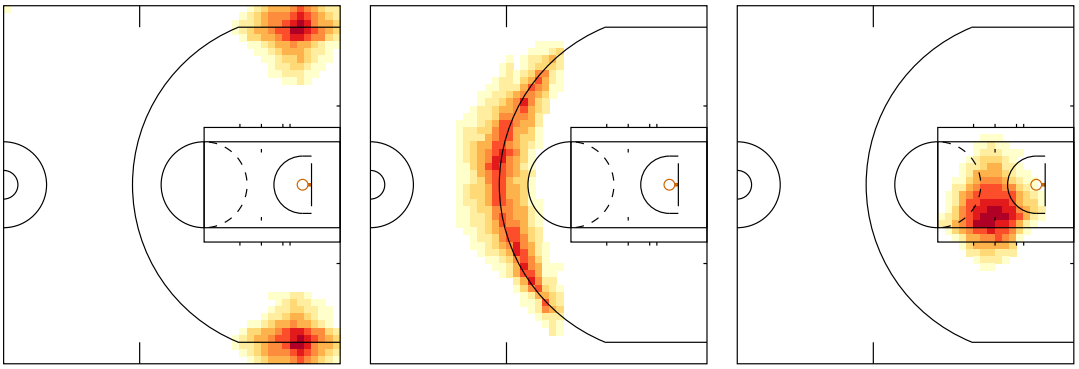
The number of passes or shots at goal that occur in each region may also be counted. For example, many papers counted shots made in each region of a subdivision of a basketball court [74], [90], [146], [168], [175]. Similarly, Borrie *et al.* [22], Camerino *et al.* [37], Narizuka *et al.* [154], and Cervone *et al.* [41] counted the number of passes made in each region of a subdivision of the playing area.

2.1.2 Low-rank Factor Matrices

Matrix factorization can be applied to intensity matrices described in Subsection 2.1.1, to produce a compact, low-rank representation. This approach has been used in several papers to model shooting behaviour in basketball [41], [74], [198]. The insight that motivates this technique is that similar types of players tend to shoot from similar locations, and so each player's shooting style can be modelled as a combination of a few distinct *types*, where each *type* maps to a coherent area of the court that the players are likely to shoot from.

The input is an intensity matrix $\mathbf{X} \in \mathbb{R}_{\geq 0}^{n \times v}$. Two new matrices $\mathbf{W} \in \mathbb{R}_{\geq 0}^{n \times k}$ and $\mathbf{B} \in \mathbb{R}_{\geq 0}^{k \times v}$ are computed such that $\mathbf{WB} \approx \mathbf{X}$ and $k \ll n, v$. The k spatial bases in $\mathbf{B}$ represent areas of similar shooting intensity, and the n players' shooting habits are modelled as a linear combination of the spatial bases. The factorization is computed from $\mathbf{X}$ by minimizing some distance measure between $\mathbf{X}$ and $\mathbf{WB}$, under the constraint that $\mathbf{W}$ and $\mathbf{B}$ are non-negative. The non-negativity constraint, along with the choice of distance function encourages sparsity in the learned matrices. This leads to intuitive results: each spatial basis corresponds to a small number of regions of the halfcourt; and the shooting style of each player is modelled as the mixture of a small number of bases, see Figure 2.5 for examples of learned spatial bases.

Miller *et al.* [150] used non-negative matrix factorization to represent shooting locations in basketball. They observe that the shooting intensity should vary smoothly



(a) Corner three-point (b) Top-of-key three-point (c) Right low-post

Figure 2.5: Examples of spatial bases induced by using non-negative matrix factorization. Each basis represents an intensity map of where a subset of players tend to shoot from. Shown are three spatial basis intensity maps that represent defined shooting locations.

over the court space, and thus fit a Log-Gaussian Cox Process to infer a smooth intensity surface over the intensity matrix, which is then factorized.

Yue *et al.* [198] used non-negative matrix factorization to model several event types: shooting; passing and receiving. They include a spatial regularization term in the distance function used when computing the matrix factorization, and claim that spatial regularization can be seen as a frequentist analog of the Bayesian Log-Gaussian Cox process used by Miller *et al.* [150].

Cervone *et al.* [41] also used non-negative matrix factorization to find a basis representing player roles, based on their occupancy in areas of the court. Players who are similar to a given player were identified as those who are closest in this basis, and this was used to compute a similarity matrix between players.

2.1.3 Movement Models and Dominant Regions

A team's ability to control space is considered a key factor in the team's performance, and was one of the first research areas in which computational techniques were developed. Intuitively a player dominates an area if he can reach every point in that area before anyone else (see Definition 2.1). An early algorithmic attempt to develop a computational tool for this type of analysis was presented by Taki *et al.* [184], which defined the *Minimum Moving Time Pattern*—subsequently renamed the *Motion Model*—and the *Dominant Region*.

2.1.3.1 Motion Model

The motion model presented by Taki *et al.* [184] is simple and intuitive: it is a linear interpolation of the acceleration model. It assumes that potential acceleration is the same in all directions when the player is standing still or moving very slowly. As speed increases it becomes more difficult to accelerate in the direction of the movement. However, their model did not account for deceleration and hence is only accurate over short distances.

Fujimura and Sugihara [84] presented a more realistic motion model, in particular they incorporated a resistive force that decrease the acceleration. The maximum speed of a player is bounded, and based on this assumption, Fujimura and Sugihara [84] formulated the following equation of motion:

$$m\frac{d}{dt}v = F - kv, \tag{2.1}$$

where m is the mass, F is the maximum driving force, k is the resistive coefficient, and v is the velocity. The solution of the equation is:

$$v = \frac{F}{k} - (\frac{F}{k} - v_0) \cdot \exp(-\frac{k}{m}t),$$

where v_0 is the velocity at time $t = 0$. If the maximum speed $v_{\max} = F/k$ and the

magnitude of the resistance $\alpha = k/m$ are known, then the motion model is fixed. To obtain α and $v_{\max}$, Fujimura and Sugihara [84] studied players' movement on video and empirically estimated α to be 1.3 and $v_{\max}$ as 7.8m/s. This is then generalised to two dimensions as follows:

$$m\frac{d}{dt}\mathbf{v} = \mathbf{F} - k\mathbf{v}.$$

Solving the equation we get that all the points reachable by a player, starting at position x_0 with velocity $\mathbf{v_0}$, can reach point x within time t form the circular region centred at

$$x_0 + \frac{1 - e^{-\alpha t}}{\alpha} \cdot \mathbf{v_0} \quad \text{with radius} \quad \mathbf{v}_{\max} \cdot \frac{1 - e^{-\alpha t}}{\alpha}.$$

They compared this model empirically and observed that the model yields a good approximation of actual human movement, but they stated that a detailed analysis is a topic for future research.

A different model was used in a recent paper by Cervone *et al.* [41] with the aim to predict player movement in basketball. They present what they call a micro-transition model. The micro-transition model describes the player movement during a single possession of the ball. Separate models are then used for defense and attack. Let the location of an attacking player ℓ at time t be $(x^\ell(t), y^\ell(t))$. Next they model the movement in the x and y coordinates at time $(t + \varepsilon)$ using

$$x^\ell(t + \varepsilon) = x^\ell(t) + \alpha_x^\ell[x^\ell(t) - x^\ell(t - \varepsilon)] + \eta_x^\ell(t), \tag{2.2}$$

and analogously for $y^\ell(t+\varepsilon)$. This expression derives from a Taylor series expansion of the function for determining the ball-carrier's position such that $\alpha_x^\ell[x^\ell(t) - x^\ell(t-\varepsilon)] \approx \varepsilon x^\ell(t)$, and $\eta_x^\ell(t)$ represents the contribution of higher order derivatives modelling accelerations and jerks. When a player receives the ball outside the three-point line, the most common movement is to accelerate towards the basket. On the other hand, a player will decelerate when closer to the basket. Players will also accelerate away from the boundary of the court as they approach it. To capture this behaviour the authors

suggest mapping a player's location to the additive term $\eta_x^\ell(t)$ in (2.2). The position of the five defenders are easier to model, conditioned on the evolution of the attack's positions, see Cervone *et al.* [41] for details.

Next we consider how the motion models have been used to develop other tools.

2.1.3.2 Dominant Regions

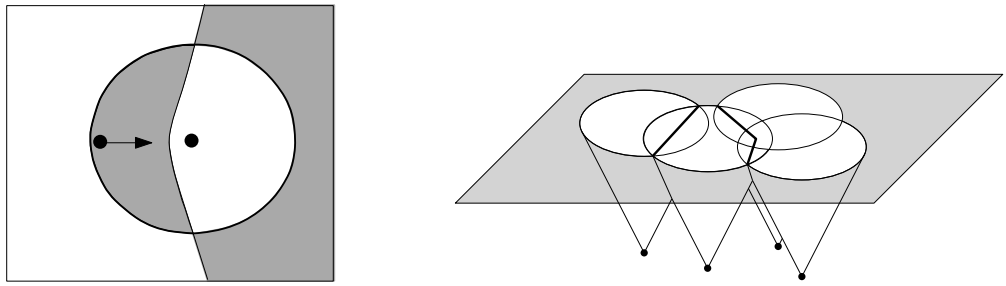
The original paper by Taki *et al.* [184] defined the dominant region as:

Definition 2.1. The *dominant region* of a player p is the region of the playing area where p can arrive before any other player.

Consequently the subdivision induced by the dominant regions for all players will partition the playing area into cells. In a very simple model where acceleration is not considered, the dominant region is equivalent to the Voronoi region and the subdivision can be efficiently computed [73]. However, for more elaborate motion models, such as the ones described in Section 2.1.3.1, the distance function is more complex. For some motion models the dominant region may not be a connected area [183], an example is shown in Figure 2.6(a). A standard approach used to compute the subdivision for a complex distance function is to compute the intersection of surfaces in three dimensions, as shown in Figure 2.6(b). However, this is a complex task and time-consuming for non-trivial motion models. Instead approximation algorithms have been considered in the literature.

Taki and Hasegawa [182], [183] implemented algorithms to compute dominant regions, albeit using a simple motion model. Instead of computing the exact subdivision they considered the 640×480 pixels that at that time formed a computer screen and for each pixel they computed the player that could reach that pixel first, hence, visualizing the dominant regions. The same algorithm for computing the dominant region was used by Fujimura and Sugihara [84], although they used a more realistic motion model, see Section 2.1.3.1.

However, the above algorithms were shown to be slow in practice, for example preliminary experiments by Nakanishi *et al.* [153] stated that the computation requires

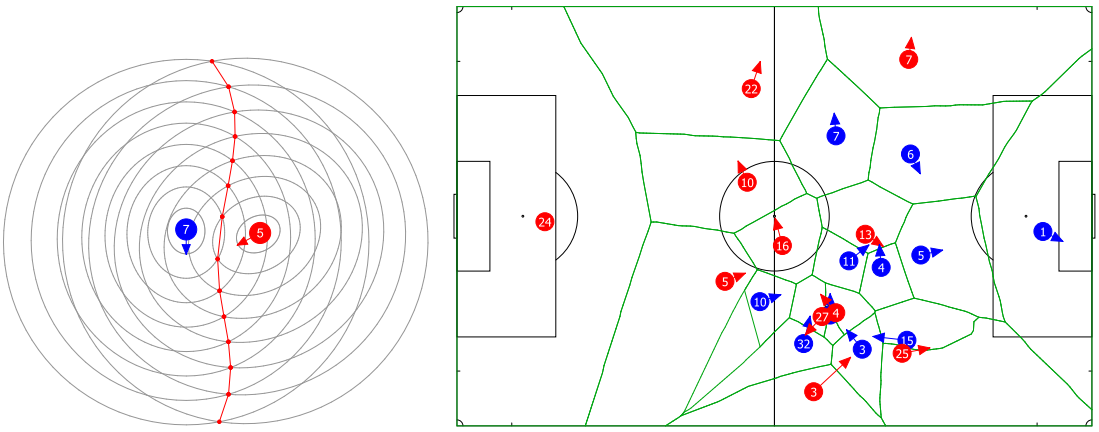


(a) (b)

Figure 2.6: (a) Showing the dominant region for two players. The left player is moving to the right with high speed and the right player is standing still. Using the motion models discussed in subsubsection 2.1.3.1 the resulting dominant region for a single player might not be connected. (b) A standard approach used in computational geometry to subdivide the plane is to compute the projection of the intersection of surfaces in three dimensions onto the plane.

$10\,\mathrm{s}$ to $40\,\mathrm{s}$ for a 610×420 grid. To achieve the real-time computation required for application in the RoboCup robot football tournament [123], the authors proposed an alternative approach. Instead of computing the time required for every player to get to every point, Nakanishi *et al.* [153] used a so-called *reachable polygonal region* (RPR). The RPR of a player p given time t is the region that p can reach within time t. An advantage with using the RPR for computing dominant regions is that more complex motion models can be used by simply drawing the RPR for different values of t. They presented the following high-level algorithm. Given a sequence of time-steps t_i, $1 \leq i \leq k$ compute the RPRs for each player and each time-step. The algorithm then iterates through the sequence of time-steps and for each pair of players, the *partial dominant regions* are constructed from the RPRs. The partial dominant regions are then combined with the dominant regions computed in the previous time-step to form new dominant regions. Assuming that the RPR is a convex area for any p and any t, Nakanishi *et al.* claim a factor of 1000 improvement in computation time at the cost of roughly a 10% drop in accuracy.

Gudmundsson and Wolle [96] used RPRs induced from real trajectory data. They also presented an algorithm for constructing an approximate dominant region subdivision, which is superficially similar to the algorithm by [153]. However, instead of



(a) (b)

Figure 2.7: (a) An approximate bisector between two players using the intersection points of the RPRs. (b) An example of the approximate dominant region subdivision by Gudmundsson and Wolle [96].

computing partial dominant regions for each pair of players at each time-step, an approximate bisector is constructed for every pair of players. An example of an approximate bisector between two players is shown in Figure 2.7(a), and in Figure 2.7(b) the final subdivision generated by the algorithm in Gudmundsson and Wolle [96] is depicted.

A closer study of a player's dominant region was performed by Fonseca *et al.* [72] in an attempt to describe the spatial interaction between players. They considered two variables denoting the smallest distance between two teammates and the size of the dominant region. They observed that the individual dominant regions seem to be larger for the attacking team. They also found that for the defending team the two measures were more irregular which indicates that their movement was more unpredictable that the movement of the attacking team.

According to the authors, the player and team dominant regions detect certain match events such as "when the ball is received by an attacker inside the defensive structure, revealing behavioural patterns that may be used to explain the performance outcome."

Ueda *et al.* [187] compared the team-area and the dominant region (within the

team-area) during offensive and defensive phases. The *team area* is defined as the smallest enclosing orthogonal box containing all the field players of the defending team. The results seem to show that there exists a correlation between the ratio of the dominant region to team area, and the performance of the team's offence and defence. Dominant regions of successful attacks were thinner than those for unsuccessful attacks that broke down with a turnover event located near the centre of the playing area. The conclusion was that the dominant region is closely connected to the offensive performance, hence, perhaps it is possible to evaluate the performance of a group of players using the dominant region.

Open 2.1. *The function modelling player motion used in dominant region computations has often been simple for reasons of tractability or convenience. Factors such as the physiological constraints of the players and* a priori *momentum have been ignored. A motion function that faithfully models player movement and is tractable for computation is an open problem.*

2.1.3.3 Further Applications

The dominant region is a fundamental structure that has been shown to support several other interesting measures, and are discussed next.

1. **(Weighted) Area of team dominant region.** Taki *et al.* [184] defined a *team dominant region* as the union of dominant regions of all the players in the team. Variations in the size of the team dominant region was initially regarded by [184] as a strong indication on the performance of the team. However, Fujimura and Sugihara [84] argued that the size of a dominant region does not capture the contribution of a player. Instead they proposed using a *weighted* dominant region, by either weighting with respect to the distance to the goal, or with respect to the distance to the ball. They argued that both these approaches better model the contribution of a player compared to simply using the size of the dominant region. However, no further analysis was performed.

Fujimura and Sugihara [84] also suggested that the weighted area of dominant regions can be used to evaluate attacking teamwork: tracking the weighted dominant region ("defensive power") over time for the defender marking each attacker will indicate the attacker's contribution to the team.

2. **Passing evaluation.** A player's *passable area* is the region of the playing area where the player can potentially receive a pass. The size and the shape of the passable area depends on the motion model, and the positions of the ball and the other players. Clearly this is also closely related to the notion of dominant region.

   Definition 2.2. [96] A player p is open for a pass if there is some direction and (reasonable) speed that the ball can be passed, such that p can intercept the ball before all other players.

   Taki and Hasegawa [183] further classified a pass as "successful" if the first player that can receive the pass is a player from the same team. This model was extended and implemented by Fujimura and Sugihara [84], as follows. They empirically developed a motion model for the ball, following formula (2.1) in Section 2.1.3.1. They then defined the *receivable pass variation* (RPV) for each player to be the number of passes the player can receive among a set of sampled passes. They experimentally sampled $54{,}000$ passes by discretizing $[0, 2\pi)$ into 360 unit directions and speeds between 1 and $150\,\mathrm{km\,h^{-1}}$ into 150 units.

   Gudmundsson and Wolle [96] also used a discretization approach, but viewed the problem slightly differently. Given the positions, speeds and direction of motion of the players, they approximated who is open for a pass for each discretized ball speed. For each fixed passing speed they built RPRs for each player and the ball over a set of discrete time-steps. Then an approximate bisector is computed between the ball and the player. Combining the approximate bisectors for all the players, a piecewise linear function f is generated over the domain $[0, 2\pi)$. The segments of the bisectors that lie on the lower envelope of f map to intervals on

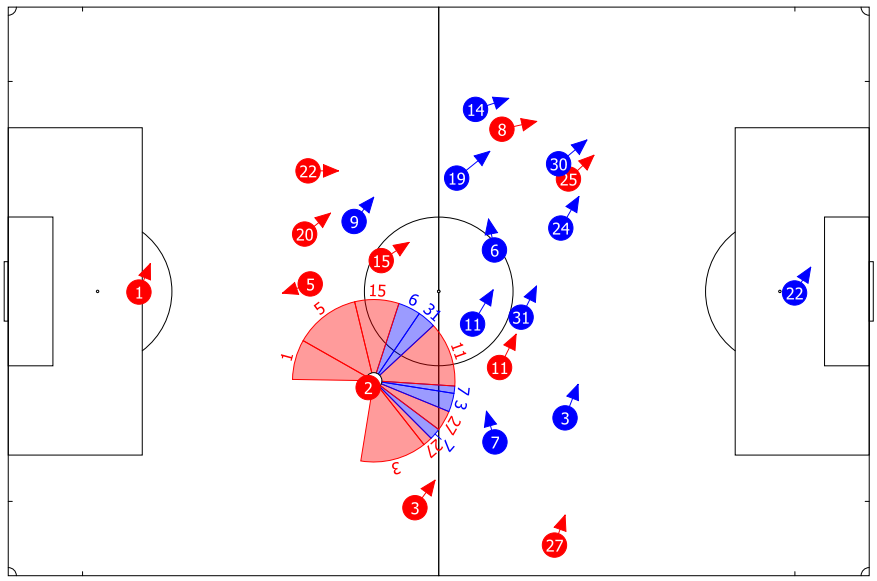


Figure 2.8: Available receivers of a pass by player Red 2 where velocity of the ball is $20\,\mathrm{m\,s^{-1}}$. Each sector represents an interval on $[0, 2\pi)$ that indicates which player may receive the pass. Players may receive a pass made at more than one interval, for example Blue 7.

the domain where the player associated with the bisector is open for a pass. An example of the output is shown in Figure 2.8 for a fixed ball speed.

Open 2.2. *The existing models for determining whether a player is open to receive a pass only consider passes made along the shortest path between passer and receiver and where the ball is moving at constant velocity. The development of more realistic model that allows for aerial passes, effects of ball-spin, and variable velocities is an interesting research question.*

3. **Spatial Pressure.** An important tactical measure is the amount of spatial pressure the team exerts on the opposition. Typically when a team believes that the opponent is weak at retaining possession of the ball, then a high pressure tactic is used. Taki *et al.* [184] defined spatial pressure for a player p as:

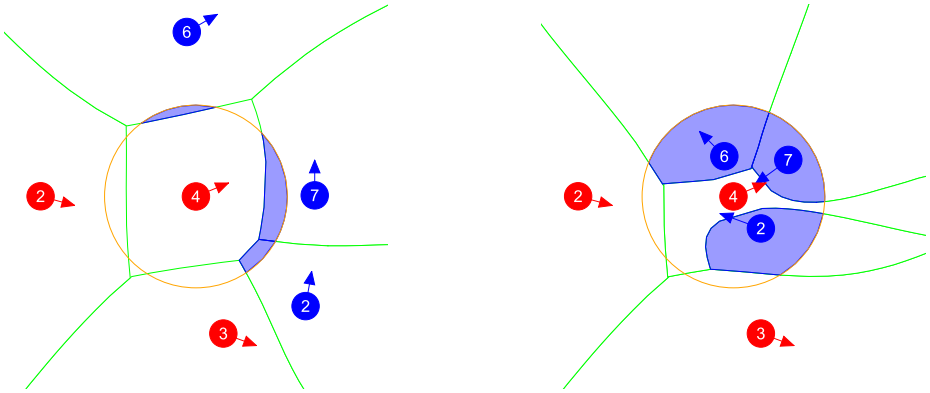
$$m \cdot (1 - P) + (1 - m) \cdot (1 - d/D),$$

where, for a fixed radius r, P denotes the fraction of the disk of radius r with center at p that lies within the dominant region of opposing players, d is the distance between p and the ball, D is the maximal distance between p and any point on the playing area, and m is a preset weight. This definition was also used by Chawla *et al.* [45] (see also Section 3.3). See Figure 2.9 for two examples of spatial pressure.

Open 2.3. *The definition of spatial pressure in Taki* et al. *[184] is simple and does not model effects such as the direction the player is facing or the direction of pressuring opponents, both of which would intuitively be factors that ought to be considered. Can a model that incorporates these factors be devised and experimentally tested?*

4. **Rebounding.** Traditionally a player's rebounding performance has been measured as the average number of rebounds per game. Maheswaran *et al.* [147] presented a model to quantify the potential to rebound unsuccessful shots in basketball in more detail. Simplified, the model considers three phases. The first phase is the *position* of the players when the shot is taken. From the time that the ball is released until it hits the rim, the players will try to move into a better position – the *crash* phase. After the crash phase the players have the chance to make the rebound. The proficiency of a player in rebounding is the measured by the *conversion*—the third phase.

   Both the positioning phase and the crash phase make use of the dominant region (Voronoi diagram) to value the position of the player, i.e., they compute a "real estate" value of the dominant region of each player both when the shot is made, and when the shot hits the rim. These values, together with the conversion, are combined into a *rebounding* value.



(a) (b)

Figure 2.9: Comparing the pressure that the encircled player is under in the two pictures shows that the encircled player in the right figure is under much more pressure.

2.2 Network Techniques for Team Performance Analysis

Understanding the interaction between players is one of the more important and complex problems in sports science. Player interaction can give insight into a team's playing style, or be used to assess the importance of individual players to the team. Capturing the interactions between individuals is a central goal of social network analysis [192] and techniques developed in this discipline have been applied to the problem of modelling player interactions.

An early attempt to use networks for sports analysis was in an entertaining study by Gould and Gatrell [92] where they explore all passes made in the 1977 FA Cup Final between Liverpool and Manchester United. They studied the simplicial complexes of the passing network and made several interesting observations, including that the Liverpool team had two "quite disconnected" subsystems and that Kevin Keegan was "the linchpin of Liverpool". However, their analysis, while innovative, did not attract much attention.

In the last decade numerous papers have appeared that apply social network analysis to team sports. Two types of networks have dominated the research literature to

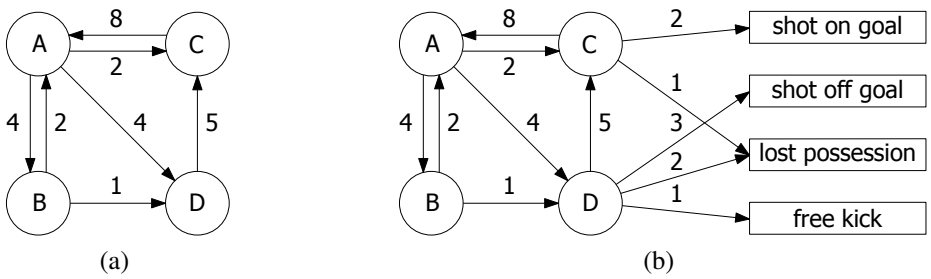


Figure 2.10: (a) A passing network modelling four players $\{A, B, C, D\}$ and the passes between the players. (b) A transition network is a passing network extended with outcomes. For example, twice player C made a shot on goal and once the player lost possession.

date: *passing networks* and *transition networks*.

Passing networks have been most frequently studied type in the research field. To the best of our knowledge, they were first introduced by Passos *et al.* [159]. A passing network is a graph $G = (V, E)$ where each player is modelled as a vertex and two vertices v_1 and v_2 in V have a directed edge $e = (v_1, v_2)$ from v_1 to v_2 with integer weight $w(e)$ such that the player represented by vertex v_1 has made $w(e)$ successful passes to the player represented by vertex v_2. A small example of a passing graph is shown in Figure 2.10a. Passing networks can be constructed directly from *event logs*, defined in Section 1.2. A temporal sequence of passes made in a match is encoded as a path within the passing network. A passing network that is extended with outcomes, as illustrated in Figure 2.10b, is then referred to as a *transition network*.

Many properties of passing networks have been studied, among them *density*, *heterogeneity*, *entropy*, and *Nash equilibria*. However, the most studied measurement is *centrality*. We begin by considering centrality and its variants, and then we briefly consider some of the other measures discussed in the literature.

2.2.1 Centrality

Centrality measures were introduced in an attempt to determine the key nodes in a network, for example, to identify the most popular persons in a social network or

super-spreaders of a disease [155]. In team sports the aim of using centrality measurements is generally to identify key players, or to estimate the interactivity between team members. For an excellent survey on network centrality see Borgatti [21].

2.2.1.1 Degree centrality

The simplest centrality measure is the *degree centrality*, which is the number of edges incident to a vertex. For directed networks one usually distinguish between the in-degree and the out-degree centrality. In sports analysis the out-degree centrality is simply referred to as *centrality* while the in-degree centrality is usually called the *prestige* of a player. Some papers do consider both centrality and prestige, see for example Clemente *et al.* [56], but most of the literature has focused on centrality.

Fewell *et al.* [69] considered a transition graph induced from basketball games where the vertices represented the five traditional player positions (point guard, shooting guard, small forward, power forward, and center), possession origins and possession outcomes. The centrality was computed on the transition graph, split into two outcomes: "shots" and "others". The measure was computed on 32 basketball games and prior knowledge about the importance of players to the teams involved was compared to the centrality values of the players. They used degree centrality to compare teams that heavily rely on key players with teams with a more even distribution between their team members. Unfortunately, the data was not definitive since the overall centrality rankings did not show a strong relation to the teams performance.

Grund [93] used degree centrality together with Freeman centralization [80]. The idea by Freeman was to consider the relative centrality of the most important node in the network. That is, how central is the most central node compared to the centrality of the other nodes in the network. The Freeman centrality is measured as the sum of the differences between the node with the highest degree centrality and all other nodes; divided by a value depending only on the size of the network [80]. They used an extensive set of $283{,}259$ passes from 760 English Premier League football matches for their experiments. From a team performance perspective Grund [93] set out to

answer two hypotheses: (i) increased interaction between players leads to increased team performance; and (ii) increased interaction centralization leads to decreased team performance. The latter is strongly connected to centrality and Grund [93] went on to show that a high level of centralization decreases team performance.

In a series of recent papers, Clemente *et al.* [51], [52], [55], [56] argue that centrality may recognise how football players collaborate, and also the nature and strength of their collaboration. For example, central midfielders and central defenders usually show higher degree centrality then other players. Some exceptions were shown in Clemente *et al.* [54] where the left and right defenders also obtained very high degree centrality. In general goal-keepers and forwards have the lowest centrality measure.

2.2.1.2 Betweenness Centrality

The betweeness centrality of a node is the number of times it lies on the shortest path between two other nodes in the network. Originally it was introduced by Freeman [79] in an attempt to estimate "a human's potential control of communication in a social network".

Peña and Touchette [160] claimed that the betweenness centrality measures how the ball-flow between football players depends on a particular player and as such provides a measure of the impact of the "removal" of that player from the game, either by being sent off or by being isolated by the opponents. They also argued that, from a tactical point of view, a team should aim to have a balanced betweenness score for all players.

A centrality measure closely related to the betweenness centrality is flow centrality. The flow centrality is measured by the proportion of the entire flow between two vertices that occur on paths of which a given vertex is a part.

Duch *et al.* [66] considered flow centrality for transition networks where the weight of an edge from a player v_1 to a player v_2 is equal to the fraction of passes initiated by v_1 to reach v_2. Similarly, the shooting accuracy for a player (the weight of the edge from the player to the vertex "shots on goal") is the fraction of shots made by

the player that end up on goal. They then studied the flow centrality over all paths that results in a shot in football. They also take the defensive performance into account by having each player initiate a number of flow paths which is comparable to the number of times the player wins possession of the ball. The *match performance* of the player is then the normalised value of the logarithm of this combined value. They argue that this gives an estimate of the contribution of a single player and also of the whole team. The team's match performance value is the mean of the individual player values. Using these values, both for teams and individual players, Duch *et al.* [66] analysed 20 games from the football 2008 UEFA European Cup. They claim that their measurements provide "sensible results that are in agreement with the subjective views of analysts and spectators", in other words, the better paid players tend to contribute more to the team's performance.

2.2.1.3 Closeness Centrality

The standard distance metric used in a network is the length (weight or cost) of the shortest path between pairs of nodes. The *closeness centrality* of a node is defined as the inverse of the *farness* of the node, which is the sum of its distance to all other nodes in the network [11]. Peña and Touchette [160] argued that the closeness score is an estimate of how easy it is to get the ball to a specific player, i.e., a high closeness score indicates a well-connected player within the team. They made a detailed study using the 2010 FIFA World Cup passing data. The overall conclusion they reached was that there is a high correlation between high scores in closeness centrality, *PageRank* and clustering (see below), which supports the general perception of the players performance reported in the media at the time of the tournament.

2.2.1.4 Eigenvector Centrality and *PageRank*

The general idea of Eigenvector centrality and PageRank is that the importance of a node depends, not only on its degree, but also on the importance of its neighbours. Cotta *et al.* [60] used the eigenvector centrality calculated with the power iteration

model by Mises and Pollaczek-Geiringer [151]. The measure aims to identify which player has the highest probability to be in possession of the ball after a sequence of passes. They also motivated their measure by a thorough analysis of three games from the 2010 FIFA World Cup, where they argued the correlation between the eigenvector centrality score and the team's performance.

A variant of the eigenvector centrality measure is *PageRank*, which was one of the algorithms used by Google Search to rank web-pages [28]. The passing graph is represented as an adjacency matrix A where each entry A_{ji} is the number of passes from player j to player i. In football terms, the *PageRank* centrality index for player i is defined as:

$$x_i = p \sum_{j \neq i} \frac{A_{ji}}{L_j^{out}} \cdot x_j + q,$$

where $L_j^{out} = \sum_k A_{jk}$ is the total number of passes made by player j, p is the parameter representing the probability that a player will decide to give the ball away rather than keep it and shoot, and q is a 'free' popularity assigned to each player. Note that the *PageRank* score of a player is dependant on the scores of the player's team mates. Peña and Touchette [160] argue that the *PageRank* measure gives each player a value that is approximately the likelihood that the player will be in possession of the ball after a fixed number of passes. Using data from the 2010 FIFA World Cup, they computed the *PageRank* for the players in the top 16 teams, but focused their discussion on the players in the top four teams: Spain, Germany, Uruguay and the Netherlands. They showed that the *PageRank* of players in the Dutch and Uruguay teams were more evenly distributed than players from Spain and Germany. This indicates that no player in those teams has a predominant role in the passing scheme while Xavi Hernandez (Spain) and Bastian Schweinsteiger (Germany) were particularly central to their teams.

2.2.2 Clustering Coefficients

A clustering coefficient is a measure of the degree of which nodes in a network are inclined to cluster together. In the sport science literature both the *global* and the

local clustering coefficients have been applied. The idea of studying the global cluster coefficient of the players in a team is that it reflects the cooperation between players, that is, the higher coefficient for a player the higher is his cooperation with the other members of the team [51], [69], [160]. Fewell *et al.* [69] also argued that a high global clustering coefficient indicates that attacking decisions are taken by several players, and thus increases the number of possible attacking paths that have to be assessed by defences in basketball. Peña and Touchette [160] showed, using the 2010 FIFA World Cup passing data, that Spain, Germany and the Netherlands consistently had very high clustering scores when compared to Uruguay, suggesting that they were extremely well connected teams, in the sense that almost all players contribute.

Cotta *et al.* [60] considered three games involving Spain from the 2010 FIFA World Cup and used the local clustering coefficient as a player coefficient. They studied how the coefficient changed during the games, and argued for a correlation between the number of passes made by Spain and the local clustering coefficient. They claimed that Spain's clustering coefficient remains high over time, "indicating the elaborate style of the Spanish team".

It should be noted that it is not completely clear that there is a strong connection between the clustering coefficient and the team performance. For example, Peña and Touchette [160] stated that in their study they did not get any reasonable results and "will postpone the study of this problem for future work."

Open 2.4. *Various centrality and clustering measures have been proposed to accurately represent some aspect of player or team performance. A systematic study reviewing all such measures against predefined criteria, and on a large data set would be a useful contribution to the field.*

2.2.3 Density and Heterogeneity

In general it is believed that stronger collaboration between football players (i.e. more passes) will make the team stronger—known as the *density-performance hypothesis* [10]. Therefore a widely-assessed measure of networks is density, which is traditionally calculated as the number of edges divided by the total number of possible edges. This is the density measure used by Clemente et al. in a series of recent papers [52], [53], [55], [56]. For weighted graphs the measurement becomes slightly more complex. Grund [93] defined the *intensity* of a team as the sum of the weighted degrees over all players divided by the total time the team have possession of the ball, i.e., possession-weighted passes per minute.

Related to the density is *passing heterogeneity*, which Cintia *et al.* [50] defined as the standard deviation of the vertex degree for each player in the network. High heterogeneity of a passing network means that the team tends to coalesce into sub-communities, and that there is a low level of cooperation between players [52]. One interesting observation made by Clemente *et al.* [52] was that the density usually went down in the 2nd half while the heterogeneity went up.

Open 2.5. *The density-performance hypothesis suggests an interesting metric of team performance. Can this hypothesis be tested?*

2.2.4 Entropy, Topological Depth, Price-of-Anarchy and Power Law Distributions

As described above, Fewell *et al.* [69] considered an extended transition graph for basketball games, where they also calculated *player entropy*. Shannon entropy [173] was used to estimate the uncertainty of a ball transition. The *team entropy* is the aggregated player entropies, which can be computed in many different ways. Fewell *et al.* [69] argue that from the perspective of the opposing team the real uncertainty is the number of options, and computed the team entropy from the transition matrix describing ball movement probabilities across the five standard player positions and

the two outcomes.

Skinner [176] showed that passing networks have two interesting properties. They identified a correspondence between a basketball transition network and a traffic network, and used insights from the latter to make suppositions about the former. They posited that there may be a difference between the Nash equilibrium of a transition network and the community optimum—the *Price of Anarchy*. In other words, for the best outcome one should not always select the highest-percentage shot. A similar observation was made in Fewell *et al.* [69] who noted that the low flow centrality of the most utilised position (point guard) seems to indicate that the contribution of key players can be negatively affected by the point guard controlling the ball more often than other players. Related to the same concept, Skinner [176] suggested that removing a key player from a match—and hence the transition network—may actually *improve* the team performance, a phenomena known as the *Braess' paradox* in network analysis [26].

2.3 Data Mining

The representations and structures described in Sections 1.2 and 2.2 are informative in isolation, but may also be the input for more complex algorithmic and probabilistic analysis of team sports. In this section, we present a task-oriented survey of the techniques that have been applied, and outline the motivations for these tasks.

2.3.1 Applying Labels to Events

Sports analysts are able to make judgments about events and situations that occur in a match, and apply qualitative or quantitative attributes to that event, for example, to rate the riskiness of an attempted shot on goal, or the quality of a pass. Event labels such as these can be used to measure player and team performance, and are currently obtained manually by video analysis. Algorithmic approaches to automatically produce such labels may improve the efficiency of the process.

Chawla *et al.* [45] presented a classifier that determines the quality of passes made in football matches by applying a label of *good*, *OK* or *bad* to each pass made, and were able to obtain an accuracy rate of $85.8\,\%$. The classifier uses features that are derived from the spatial state of the match when the pass occurs, including features derived from the dominant region described in Subsection 2.1.3, which were found to be important features to the classifier.

In research by Beetz *et al.* [13], the approach was to cluster passes, and to then induce a decision tree on each cluster where the passes were labelled as belonging to the cluster or not. The feature predicates, learned as splitting rules, in the tree could then be combined to provide a description of the important attributes of a given pass.

Bialkowski *et al.* [16] used the formation descriptors computed with the algorithm presented in [18] (see Subsection 2.3.3) to examine whether formations could accurately predict the identity of a team. In the model, a linear discriminant analysis classifier was trained on features describing the team formation, and the learned model was able to obtain an accuracy of $67.23\,\%$ when predicting a team from a league of 20 teams.

In Maheswaran *et al.* [146] the authors perform an analysis of various aspects of the rebound in basketball to produce a rebound model. The rebound is decomposed into three components: the location of the shot attempt; the location where the rebound is taken; and the height of the ball when the rebound is taken. Using features derived from this model, a binary classifier was trained to predict whether a missed shot would be successfully rebounded by the offensive team. The model was evaluated and obtained an accuracy rate of $75\,\%$ in experiments on held-out test data.

Tora *et al.* [186] employs a deep learning approach to detecting puck events in ice hockey games, such as shot, pass, loose puck recovery, dump-in and dump-out. The model uses a *long short-term memory* (LSTM) recurrent neural network [109] to predict, given an input sequence including player locations, whether any of the puck events occurred at each time-step. The model is trained using an *imitation learning* algorithm, and was found to outperform existing baseline methods for this task in an

experimental evaluation.

2.3.2 Predicting Future Event Types and Locations

The ability to predict how play will unfold given the current game-state has been researched extensively, particularly in the computer vision community. This has an application in automated camera control, where the camera filming a match must automatically control its pitch, tilt and zoom. The framing of the scene should ideally contain not just the current action, but the movement of players who can be expected to be involved in future action, and the location of where such future action is likely to occur.

Kim *et al.* [122] considered the problem of modelling the evolution of football play from the trajectories of the players, such that the location of the ball at a point in the near future could be predicted. Player trajectories were used to compute a dense motion field over the entire playing area, and points of convergence within the motion field identified. The authors suggest that these points of convergence indicate areas where the ball can be expected to move to with high probability, and the experiments described in the paper demonstrate this with several examples.

Yue *et al.* [198] construct a model to predict whether a basketball player will shoot, pass to one of four teammates, or retain possession. The action a player takes is modelled using a multi-class conditional random field. The input features to the classifier include latent factors representing player locations which are computed using non-negative matrix factorization—see Subsection 2.1.1—and the experimental results show that these features improve the predictive performance of the classifier.

Wei *et al.* [193] constructed a model to make short-term predictions of which football player will be in possession of the ball after a given interval. They propose a model—augmented-Hidden Conditional Random Fields (aHCRF)—that combines local observation features with the hidden layer states to make the final prediction of the player who possess the ball. The experimental results show that they are able to design a model that can predict which player will be in possession of the ball after $2\,\mathrm{s}$ with

$99.25\,\%$ accuracy.

Predicting the location of players over longer durations is a challenging problem for several reasons. The future location of a player is partially dependent on the locations of the other players up to, and including, the time-stamp of the required prediction. This suggests that the policies that define the movement of all players must be learned jointly. Le *et al.* [133] defines a model based on a LSTM recurrent neural network that is trained using imitation learning to estimate the movement policies of players. This model is applied to *ghosting* in Le *et al.* [132]. Ghosting is a visualisation technique where the player's locations are displayed markers on the playing area, and for each player a "ghost" marker is also displayed that shows the player's "ideal" location according to the learned policy.

Furthermore, the movement that players make is the manifestation of competing long- and short-term priorities. Research by Zheng *et al.* [199] proposes a hierarchical policy model, instantiated as a recurrent neural network, to learn policies that balance short-term micro actions with longer-term macro goals. This model was applied to a basketball setting to predict long-term player trajectories, that were realistic in the short term—e.g. passing, dribbling—and achieved longer-term goals—e.g. moving to a scoring position.

Short-term prediction of player movement is also used in *camera planning* where the goal is to automatically control a camera's tilt, pan and zoom in order to include all the relevant action within the frame while simultaneously ensuring that the movement of the camera is smooth. Chen *et al.* [47] uses a recurrent decision tree framework for this problem, where the input is the noisy location points of players and the desired output is a smooth sequence of camera movements that induce frames containing the all the relevant players.

2.3.3 Identifying Formations

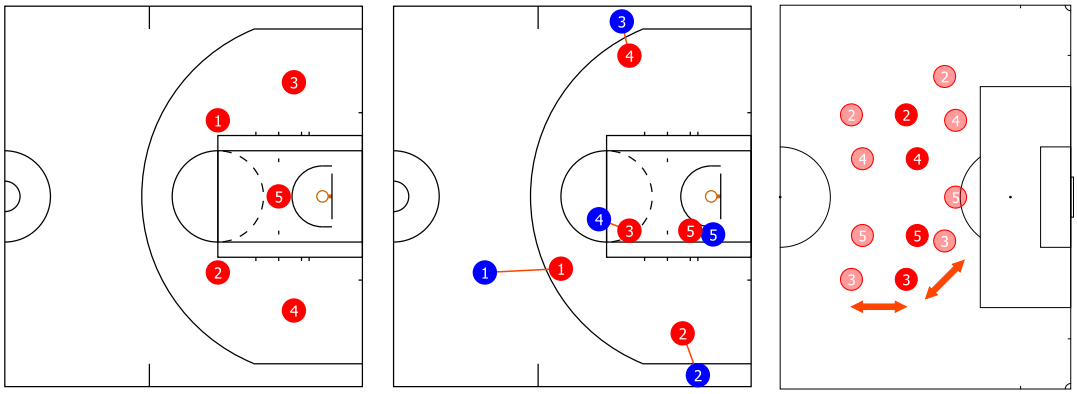
Sports teams use pre-devised spatial formations as a tactic to achieve a particular objective. The ability to automatically detect such formations is of interest to sports

analysts and coaches. For example, a coach would be interested in understanding the proportion of time that a team maintains an agreed formation, and also when the team is compelled by the circumstances of the match to alter its formation. Moreover, when preparing for a future opponent, an understanding of the formation used, and periods where the formation changes would be of interest.

A formation is a positioning of players, relative to the location of other objects, such as the playing-area boundaries or goal/basket, the players' team-mates, or the opposition players. Formations may be spatially anchored, for example a zone defence in basketball where players position themselves in a particular location on the playing area, see Figure 2.11(a). On the other hand, a formation may vary spatially, but maintain a stable relative orientation between the players in the formation. For example, the defensive players in a football team will position themselves in a straight line across the playing-area, and this line will move as a group around the playing-area, depending on the phase of play, Figure 2.11(c). Finally, a different type of formation is a *man marking* defence, where defending players will align themselves relative to the attacking players that they are marking, Figure 2.11(b). In this scenario, the locations of defenders may vary considerably, relative to their teammates or to the boundaries of the playing area.

Moreover, the players that fulfil particular roles within a formation may switch, either explicitly through substitutions or dynamically where players may swap roles for tactical reasons. The following approaches have been used to determine formations from the low-level trajectory signal.

Lucey *et al.* [140] investigated the assignment of players to roles in field hockey, where teams use a formation of three lines of players arrayed across the field. At any time t there is a one-to-one assignment of players to roles, however this assignment may vary from time-step to time-step. This problem is mathematically equivalent to permuting the player ordering $\mathbf{p}_t^\tau$ using a permutation matrix $\mathbf{x}_t^\tau$ which assigns the players to roles $r_t^\tau = \mathbf{x}_t^\tau \mathbf{p}_t^\tau$. The optimal permutation matrix $\mathbf{x}_t^\tau$ should minimise the total Euclidean distance between the reference location of each role and the location



(a) Zone Defence (b) Man-marking Defence (c) Back-four Defence

Figure 2.11: Examples of typical formations used in basketball and football. (a) The zone defence is spatially anchored to the dimensions of the court and the players positioning is invariant to the phase of play. (b) Defenders who are man-marking will align themselves relative to their opposing player, typically between the attacker and the basket. (c) The back-four formation in football maintains the alignment of players in the formation, but will move forward and laterally, depending on the phase of play.

of the player assigned to the role, and can be computed in closed form using the Hungarian algorithm [128].

Wei *et al.* [194] used this approach as a preprocessing step on trajectory data from football matches, and the computed role locations were subsequently used to temporally segment the matches into game phases. Lucey *et al.* [141] applied role assignment to basketball players in sequences leading up to three-point shots. They analysed close to 20,000 such shots and found that role-swaps involving particular pairs of players in the moments preceding a three-point shot have a significant impact on the probability of the shooter being *open*—at least 6 feet away from the nearest marker—when the shot is made.

Furthermore, Bialkowski *et al.* [18] observed that the role assignment algorithm presented by Lucey *et al.* [140] required a predefined prototype formation to which the players are assigned. They consider the problem of simultaneously detecting the reference location of each role in the formation, and assigning players to the formation, using an expectation maximization approach [61]. The initial role reference locations

are determined as the mean position of each player. The algorithm then uses the Hungarian algorithm to update the role assignment for each player at each time-step, and then the role reference locations are recomputed according to the role assignment. The new locations are used as input for the next iteration, and process is repeated until convergence.

The learned formations for each team and match were then clustered into six formations, and the authors claim that the clustered formations were consistent with expert knowledge of formations used by football teams. This was validated experimentally by comparing the computed formation with a formation label assigned by an expert, and an accuracy of $75.33\,\%$ was obtained.

In a subsequent paper, Bialkowski *et al.* [17] investigated differences in team strategies when playing home or away, by using formations learned with the role assignment algorithm. By computing the mean position when teams are playing at home from when they are playing away, they observed that teams defend more deeply when away from home, in other words they set their formation closer to the goal they are defending.

A qualitatively different formation is for players to align themselves with the positions of the opposition players, such as *man-marking* defense used in basketball, see Figure 2.11(b). Franks *et al.* [74] defined a model to determine which defender is marking each attacker. For a given offensive player at a given time, the mean location of the defender is modelled as a convex combination of three locations: the position of the attacker, the location of the ball and the location of the hoop. The location of a defender, given the observed location of the marked attacker, is modelled as a Gaussian distribution about a mean location. The matching between defenders and the attacker that they are marking over a sequence of time-steps is modelled using a Hidden Markov Model, ensuring that the marking assignments are temporally smoothed.

2.3.4 Identifying Plays and Tactical Group Movement

Predefined *plays* are used in many team sports to achieve some specific objective. American football uses highly structured plays where the entire team has a role and their movement is highly choreographed. On the other hand, plays may also be employed in less structured sports such as football and basketball when the opportunity arises, such as the *pick and roll* in basketball or the *one-two* or *wall pass* in football. Furthermore, teammates who are familiar with each other's playing style may develop ad-hoc productive interactions that are used repeatedly, a simple example of which is a sequence of passes between a small group of players. Identification of plays is a time-consuming task that is typically carried out by a video analyst, and thus a system to perform the task automatically would be useful.

An early attempt in this direction attempted to recognise predefined plays in American football [116]. They model a play as a choreographed sequence of movements by attacking players, each trying to achieve a local goal, and in combination achieve a group goal for the play. The approach taken was to encode predefined tactical plays using a temporal structure description language that described a local goal in terms of a sequence of actions carried out by an individual player. These local goals were identified in the input trajectories using a Bayesian belief network. A second belief network then identified whether a global goal had been achieved based on the detected local goals—signifying that the play has occurred.

Two papers by Li *et al.* investigated the problem of identifying group motion, in particular the type of offensive plays in American football. Li *et al.* [136] presented the Discriminative Temporal Interaction Network (DTIM) framework to characterise group motion patterns. The DTIM is a temporal interaction matrix that quantifies the interaction between objects at two given points in time. For each predefined group motion pattern—a play—a multi-modal density was learned using a properly defined Riemannian metric, and a MAP classifier was then used to identify the most likely play for a given input set of trajectories. The experiments demonstrated that the model was

able to accurately classify sets of trajectories into five predefined plays, and outperformed several other common classifiers for the task. This model has the advantage of not requiring an *a priori* definition of each player's movement in the play, as required in Intille and Bobick [116].

Li and Chellappa [135] considered group motion segmentation, where a set of unlabelled input trajectories are segmented into the subset that participated in the group motion, and those that did not. The problem was motivated by the example of segmenting a set of trajectories into the set belonging to the offensive team (who participated in the play) and the defensive team (who did not). The group motion is modelled as a dynamic process driven by a *spatio-temporal driving force*—a densely distributed motion field over the playing area. The driving force is modelled as a 3×3 matrix $F(t_0, t_f, x, y)$ such that $X(t_f) = F(t_0, t_f, x, y)X(t_0)$. Thus, an object located at $X(t_0)$ at time t_0 will be driven to $X(t_f)$ at time t_f. Using Lie group theory [170], a Lie algebraic representation f of F is determined with the property that the space of all fs is linear, and thus tractable statistical models can be induced from f. A Gaussian mixture model was used to learn a fixed number of driving forces at each time-step, which was then used to segment the trajectories.

There has been number of diverse efforts to identify commonly occurring sequences of passes in football matches. In Borrie *et al.* [22], the playing-area is subdivided into zones and sequences of passes are identified by the zones that they start and terminate in. A possession can thus be represented by a string of codes representing each pass by source and target zone, and with an elapsed time between them. They introduce *T-pattern* analysis which is used to compute possessions where the same sequence of passes are made with consistent time intervals between each pass, and frequently occurring patterns could thus be identified. Camerino *et al.* [37] also used T-pattern analysis on pass strings, however the location of passes was computed relative to the formation of the team in possession, e.g. between the defense and midfield, or in front of the attacking line.

An algorithm to detect frequently occurring sequences of passes was presented in

Gudmundsson and Wolle [96]. A suffix tree [195] was used as a data structure D to store sequences of passes between individual players. A query (τ, o) can then be made against D that returns all permutations of τ players such that the ball is passed from a player p_1 to p_τ, via players $p_2, \ldots, p_{\tau-1}$ at least o times, and thus determine commonly used passing combinations between players.

Van Haaren *et al.* [188] considered the problem of finding patterns in offensive football strategies. The approach taken was to use inductive logic programming to learn a series of clauses describing the pass interactions between players during a possession sequence that concludes with a shot on goal. The passes were characterised by their location within the playing-area, and a hierarchical model was defined to aggregate zones of the playing-area into larger regions. The result is a set of rules, expressed in first-order predicate logic, describing the frequently-occurring interaction sequences.

Research by Wang *et al.* [191] also aimed to detect frequent sequences of passing. They claim that the task of identifying tactics from pass sequences is analogous to identifying topics from a document corpus, and present the Team Tactic Topic Model (T^3M) based on Latent Dirichlet Allocation [19]. Passes are represented as a tuple containing an order-pair of the passer and receiver, and a pair of coordinates representing the location where the pass was received. The T^3M is an unsupervised approach for learning common tactics, and the learned tactics are coherent with respect to the location where they occur, and the players involved.

As discussed in Subsection 2.3.2, the problem of predicting the future movement of players must account for the structure implicit in the tactical organisation of teammates and the opposing players. Le *et al.* [133] presents a model that learns individual movement policies for players using imitation learning. The model is defined with an explicit entropy regularization term in the objective function that incorporates the policies of other players. The individual learned policies are thus consistent as a group and can be viewed in aggregate as a joint policy.

2.3.5 Temporally Segmenting the Game

Segmenting a match into phases based on a particular set of criteria is a common task in sports analysis, as it facilitates the retrieval of important phases for further analysis. The following paragraphs describe approaches that have been applied this problem for various types of criteria.

Hervieu *et al.* [106], [107] present a framework for labelling phases within a handball match from a set of predefined labels representing common attacking and defensive tactics. The model is based on a hierarchical parallel semi-Markov model (HPaSMM) and is intended to model the temporal causalities implicit in player trajectories. In other words, modelling the fact that one player's movement may cause another player to subsequently alter their movement. The upper level of the hierarchical model is a semi-Markov model with a state for each of the defined phase labels, and within each state the lower level is a parallel hidden Markov model for each trajectory. The duration of time spent in each upper level state is modelled using a Gaussian mixture model. In the experiments, the model was applied to a small data set of handball match trajectories from the 2006 Olympics Games final, and resulted in accuracy of $92\,\%$ accuracy on each time-step, compared to the ground truth provided by an expert analyst. The model exactly predicted the sequence of states, and the misclassifications were all the result of time-lags when transitioning from one state to the subsequent state.

Perše *et al.* [162] investigated segmentation of basketball matches. A framework with two components was used, the first segmented the match duration into sequences of offensive, defensive or time-out phases. The second component identified basic activities in the sequence by matching to a library of predefined activities, and the sequences of activities were then matched with predefined templates that encoded known basketball plays.

Wei *et al.* [194] considered the problem of automatically segmenting football matches into distinct game phases that were classified according to a two-level hierarchy, using a decision forest classifier. At the top level, phases were classified as being *in-play* or

a *stoppage*. *In-play* phases were separated into highlights or non-highlights; and *stoppages* were classified by the reason for the stoppage: *out for corner*, *out for throw-in*, *foul* or *goal*. The classified sequences were subsequently clustered to find a team's most probable method of scoring and of conceding goals.

In a pair of papers by Bourbousson *et al.* [23], [24], the spatial dynamics in basketball was examined using relative-phase analysis. In Bourbousson *et al.* [23], the spatial relation between dyads of an attacking player and their marker were analysed. In Bourbousson *et al.* [24], the pairwise relation between the centroid of each team was used, along with a *stretch index* that measured the aggregate distance betweens players and their team's centroid. A Hilbert transformation was used to compute the relative phase in the x and y direction of the pairs of metrics. Experimental results showed a strong in-phase relation between the various pairs of metrics in the matches analysed, suggesting individual players and also teams move synchronously. The authors suggest that the spatial relations between the pairs are consistent with their prior knowledge of basketball tactics.

Frencken *et al.* [81] performed a similar analysis of four-a-side football matches. They used the centroid and the convex hull induced by the positions of the players in a team to compute metrics, for example the distance in the x and y direction of the centroid, and the surface area of the convex hull. The synchronized measurements for the two teams were modelled as coupled oscillators, using the HKB-model [101]. Their hypothesis was that the measurements would exhibit in-phase and anti-phase coupling sequences, and that the anti-phase sequences would denote game-phases of interest. In particular, the authors claim that there is a strong linear relationship between the x-direction of the centroid of the two teams, and that phases where the centroid's x-directions cross are indicative of unstable situations that are conducive to scoring opportunities. They note that such a crossing occurs in the build up to goals in about half the examples.

Open 2.6. *Coaches and analysts are often interested in how the* intensity *of a match varies over time, as periods of high intensity tend to present more opportunities and*

threats. It is an interesting open problem to determine if it is possible to compute a measure of intensity from spatio-temporal data, and thus be able to determine high-intensity periods.

2.4 Performance Metrics

Determining the contribution of the offensive and defensive components of team play has been extensively researched, particularly in the case of basketball which has several useful properties in this regard. For example, a basketball match can be easily segmented into a sequence of *possessions*—teams average around 92 possessions per game [127]—most of which end in a shot at goal, which may or may not be successful. This segmentation naturally supports a variety of offensive and defensive metrics [127], however the metrics are not spatially informed, and intuitively, spatial factors are significant when quantifying both offensive and defensive performance. In this section we survey a number of research papers that use spatio-temporal data from basketball matches to produce enhanced performance metrics.

2.4.1 Offensive Performance

Shooting effectiveness is the likelihood that a shot made will be successful, and *effective field goal percentage* (EFG) is a de-facto metric for offensive play in basketball [127]. However, as Chang *et al.* [43] observe, this metric confounds the efficiency of the shooter with the difficulty of the shot. Intuitively, spatial factors such as the location where a shot was attempted from, and the proximity of defenders to the shooter would have an impact of the difficulty of the shot. This insight has been the basis of several efforts to produce metrics that provide a more nuanced picture of a player or team's shooting efficiency.

Early work in this area by Reich *et al.* [168] used shot chart data (a list of shots attempted, detailing the location, time, shooter and outcome of each shot). The paper contained an in-depth analysis of the shooting performance of a single player—Sam

Cassell of the Minnesota Timberwolves—over the entire 2003/2004 season. A vector of boolean-valued predictor variables was computed for each shot, and linear models fitted for shot frequency, shot location and shot efficiency. By fitting models on subsets of the predictor variables, the authors analysed the factors that were important in predicting shot frequency, location and efficiency.

Miller *et al.* [150] investigated shooting efficiency by using vectors computed with non-negative matrix factorization to represent spatially distinct shot-types, see Subsection 2.1.2. The shooting factors were used to estimate spatial shooting efficiency surfaces for individual players. The efficiency surfaces could then be used to compute the probability of a player making a shot conditioned on the location of the shot attempt, resulting in a spatially-varying shooting efficiency model for each individual player.

Cervone *et al.* [41] present *expected possession value* (EPV), a continuous measure of the expected points that will result from the current possession. EPV is thus analogous to a "stock ticker" that provides a valuation of the possession at any point in time during the possession. The overall framework consists of a macro-transition model that deals with game-state events such as passes, shots and turnovers, and micro-transition model that describes player movement within a phase when a single player is in possession of the ball. Probability distributions, conditioned on the spatial layout of all players and the ball, are learned for the micro- and macro-transition models. The spatial effects are modelled using non-negative matrix factorization to provide a compact representation that the authors claim has the attributes of being computationally tractable with good predictive performance. The micro- and macro-transition models are combined in a Markov chain, and from this the expected value of the end-state—scoring 0, 2 or 3 points—can be determined at any time during the possession. Experimental results in the paper show how the EPV metric can support a number of analyses, such as *EPV-Added* which compares an individual player's offensive value with that of a league-average player receiving the ball in the same situation; or *Shot Satisfaction* which quantifies the satisfaction (or regret) of a player's decision to shoot,

rather than taking an alternative option such as passing to a teammate.

Chang *et al.* [43] introduces another spatially-informed measure of shooting quality in basketball: *Effective Shot Quality* (ESQ). This metric measures the value of a shot, were it to be taken by the league-average player. ESQ is computed using a learned least-squares regression function whose input includes spatial factors such as the location of the shot attempt, and the proximity of defenders to the shooter. Furthermore, the authors introduce EFG+, which is calculated by subtracting ESQ from EFG. EFG+ is thus an estimate of how well a player shoots relative to expectation, given the spatial conditions under which the shot was taken.

A further metric, *Spatial Shooting Effectiveness*, was presented by Shortridge *et al.* [175]. Using a subdivision of the court, an empirical Bayesian scoring rate estimator was fitted using the neighbourhood of regions to the shot location. The spatial shooting effectiveness was computed for each player in each region of the subdivision, and is the difference between the points-per-shot achieved by the player in the region and the expected points-per-shot from the estimator. In other words, it is the difference between a player's expected and actual shooting efficiency, and thus measures how effective a player is at shooting, relative to the league-average player.

Lucey *et al.* [142] considered shooting efficiency in football. They make a similar observation that the location where a shot is taken significantly impacts the likelihood of successfully scoring a goal. The proposed model uses logistic regression to estimate the probability of a shot succeeding—the *Expected Goal Value* (EGV). The input features are based on the proximity of defenders to the shooter and to the path the ball would take to reach the goal; the location of the shooter relative to the lines of players in the defending team's formation; and the location where the shot was taken from. The model is empirically analysed in several ways. The number of attempted and successful shots for an entire season is computed for each team in a professional league, and compared to the expected number of goals that the model predicts, given the chances. The results are generally consistent, and the authors are able to explain away the main outliers. Furthermore, matches where the winning team has fewer shots

at goal are considered individually, and the expected goals under the model are computed. This is shown to be a better predictor of the actual outcome, suggesting that the winning team was able to produce fewer—but better—quality chances.

2.4.2 Defensive Performance

Measures of defensive performance have traditionally been based on summary statistics of *interventions* such as blocks and rebounds in basketball [127] and tackles and clearances in football. However, Goldsberry and Weiss [90] observed that, in basketball the defensive effectiveness ought to consider factors such as the spatial dominance by the defence of areas with high rates of shooting success; the ability of the defence to prevent a shot from even being attempted; and secondary effects in the case of an unsuccessful shot, such as being able to win possession or being well-positioned to defend the subsequent phase.

In order to provide a finer-grained insight into defensive performance, Goldsberry and Weiss [90] presented *spatial splits* that decompose shooting frequency and efficiency into a triple consisting of close-range, mid-range and 3-point-range values. The offensive half-court was subdivided into three regions, and the shot frequency and efficiency were computed separately for shots originating in each region. These offensive metrics were then used to produce defensive metrics for the opposing team by comparing the relative changes in the splits for shots that an individual player was defending to the splits for the league-average defender.

An alternate approach to assessing the impact of defenders on shooting frequency and efficiency was taken by Franks *et al.* [74], [75]. They proposed a model that quantifies the effectiveness of man-to-man defense in different regions of the court. The proposed framework includes a model that determines who's marking whom by assigning each defender to an attacker. For each attacker, the canonical position for the defender is computed, based on the relative spatial location of the attacker, the ball and the basket. A hidden Markov model is used to compute the likelihood of an assignment of defenders to attackers over the course of a possession, trained using the

expectation maximization algorithm [61]. A second component of the model learned spatially coherent shooting type bases using non-negative matrix factorization on a shooting intensity surface fitted using a log-Gaussian Cox process. By combining the assignment of markers and the shot type bases, the authors were able to investigate the extent to which defenders inhibit (or encourage) shot attempts in different regions of the court, and the degree to which the efficiency of the shooter is affected by the identity of the marker.

Another aspect of defensive performance concerns the actions when a shot is unsuccessful, and both the defence and attack will attempt to *rebound* the shot to gain possession. This was investigated by Maheswaran *et al.* [147] where they deconstructed the rebound into three components: *positioning*; *hustle* and *conversion*, described in subsubsection 2.1.3.2. Linear regression was used to compute metrics for player's *hustle* and *conversion*, and experimental results showed that the top-ranked players on these metrics were consistent with expert consensus of top-performing players.

On the other hand, Wiens *et al.* [196] performs a statistical evaluation of the options that players in the offensive team have when a shot is made in basketball. Players near the basket can either *crash the boards*—move closer to the basket in anticipation of making a rebound—or retreat in order to maximise the time to position themselves defensively for the opposition's subsequent attack. The model used as factors the players' distance to the basket, and proximity of defenders to each attacking player. The experimental results suggested that teams tended to retreat more than they should, and thus a more aggressive strategy could improve a team's chances of success.

The analysis of defense in football would appear to be a qualitatively different proposition, in particular because scoring chances are much less frequent. To our knowledge, similar types of analysis to those presented above in relation to basketball have not been attempted for football.

Open 2.7. *There has been significant research into producing spatially-informed metrics for player and team performance in basketball, however there has been little research in other sports, particularly football. It is an open research question whether*

similar spatially-informed sports metrics could be developed for football.

2.5 Visualisation

To communicate the information extracted from the spatio-temporal data, visualization tools are required. For real-time data the most common approach is so-called *live covers*. This is usually a website that comprise of a text panel that lists high-level updates of the key events in the game in almost real time, and several graphics showing basic information about the teams and the game. Live covers are provided by leagues (e.g. NHL, NBA and Bundesliga), media (e.g. ESPN) and even football clubs (e.g. Liverpool and Paris Saint-Germain). For visualizing aggregated information the most common approach is to use heat maps. Heat maps are simple to generate, are intuitive, and can be used to visualize various types of data. Typical examples in the literature are, visualizing the spread and range of a shooter (basketball) in an attempt to discover the best shooters in the NBA [89] and visualizing the shot distance (ice hockey) using radial heat maps [164]. Two recent attempts to provide more extensive visual analytics systems have been made by Perin *et al.* [161] and Janetzko *et al.* [181] .

Perin *et al.* [161] developed a system for visual exploration of phases in football. The main interface is a timeline and *small multiples* providing an overview of the game. A *small multiple* is a group of similar graphs or charts designed to simplify comparisons between them. The interface also allows the user to select and further examine the *phases* of the game. A phase is a sequence of actions by one team bounded by the actions in which they first win, and then finally lose possession. A selected phase can be displayed and the information regarding a phase is aggregated into a sequence of views, where each view only focus on a specific action (e.g. a long ball or a corner). The views are then connected to show a whole phase, using various visualization techniques such as a passing network, a time line and sidebars for various detailed information.

In two papers Janetzko *et al.* [118], [181] present a visual analysis system for interactive recognition of football patterns and situations. Their system tightly couple the visualization system with data mining techniques. The system includes a range of visualization techniques (e.g., parallel coordinates and scalable bar charts) to show the ranking of features over time and plots the change of game play situations, attempting to support the analyst to interpret complex game situations. Using classifiers they automatically detected the most common situations and introduced semantically-meaningful features for these. The exploration system also allows the user to specify features for a specific situations and then perform a similarity search for similar situations.

Open 2.8. *The area of visual interfaces to support team sports analytics is a developing area of research. Two crucial gaps are large user studies with the aim to (1) explore the analytical questions that experts need support for, and (2) which types of visual analytical techniques can be understood by experts?*

2.6 Applicability of Approaches to Other Sports

The techniques and tools described in this survey are often motivated by a particular sport. This begs the question as to how easily the techniques could be applied to other team-based invasion sports. For the most part, our view is that the techniques are generally applicable to most other invasion sports, however the peculiar rules and dynamics of each sport will impact on the specifics of the implementation. In this section, we provide a general discussion on some of the considerations inherent in applying techniques to various invasion sports.

Broadly speaking, the spatio-temporal analytic techniques described in this survey are designed to identify structure within spatio-temporal data, in particular persistent formations and patterns. These are typically the manifestations of the tactical or strategic behaviour of the players, and such behaviour is common to all invasion sports. This objective, combined with other common features of all invasion sports suggest that the

techniques can be broadly applied. Examples of the common features are: two opposing teams alternating between defence and attack; constrained playing area and match duration; the concept of possession of the ball/puck as a means of scoring a goal; and the physical and physiological constraints of nature and the players.

On the other hand, there are a qualitative differences between various invasion sports that can impact how these techniques can be applied. Perhaps the most significant difference between sports in this context, is the temporal segmentation of the match into *plays* that occur between stoppages, and the frequency and variety of the events that occur within each *play*. At one extreme are sports such as American football, where each *play* is short, a single team is typically in possession for the duration of the *play*, and a small number of events can occur, such as passes or tackles. This can be contrasted with sports such as football, where a *play* may continue for an arbitrarily long time, possession may alternate between the two teams multiple times, and multiple events can occur. However, in both American football and football, a *play* will often end without a scoring opportunity, which is in contrast to basketball, where scoring events are more frequent.

These qualitative differences between *plays* impacts on many of the analytic techniques described herein. For example, in sports like football, hockey and rugby where the possession of the ball can alternate multiple times within a *play*, spatial dominance of the entire playing area, described in Section 2.1, is perhaps more significant than a sport such as basketball where the defending team will often cede the *back-court* and only seek to dominate the half-court below the basket they are defending, or even just the area around the ball-carrier. Similarly, the network techniques described in Section 2.2 rely on sequences of passing events within a *play*, and will be less applicable to sports where such sequences do not exist, like American football.

The research surveyed in Section 2.3 typically apply standard data-mining techniques to problems in the sporting domain and use spatio-temporal data as the primary input data. The objectives of these works, such as event classification and prediction, formation and group movement identification, and temporal segmentation would be

applicable to any invasion sport. This appears to be an effective strategy, and we expect continued interest in this approach.

A final consideration is the frequency in which particular events occur. Many of the techniques are predicated on the occurrence of events, most notably scoring events, and the number of times which such events can occur varies between different sports. Thus, a technique that applies to basketball—where fifty or more scoring events occur per match—may be applicable in theory to football—where two or three scoring events per match is normal—however in practice a significantly larger number of matches must be available to effectively apply the technique.

2.7 Conclusion

The proliferation of optical and device tracking systems in the stadia of teams in professional leagues in recent years have produced a large volume of player and ball trajectory data, and this has subsequently enabled research efforts across a number of communities to produce algorithms and tools to analyse this data. To date, a diversity of techniques have been brought to bear on a number of problems described in this survey, however there is little consensus on the key research questions or the techniques to use to address them. Thus, this survey sought to define a taxonomy of the types of problems that have been considered, and to detail the research efforts aimed at addressing these problems. The following chapters describe several specific contributions that address specific problems within this taxonomy.

Chapter 3

Classification of Passes in Football Matches Using Spatio-temporal Data

In this chapter, we present a supervised machine learning framework that automatically labels passes made during football matches with a rating of the quality of the pass. A pass is an event where the player in possession of the ball kicks the ball to a teammate with the intention that the receiver will be able to control the ball and therefore retain possession for the team. Passing the ball is a core skill, and the quality and impact of the passes made by a player are a significant factor in evaluating the player's performance. Passes can be manually rated by an informed observer, such as a coach or analyst, however this is a time-consuming task. Here, we present a framework for automatically rating passes made during matches. This framework assigns an ordinal rating to each pass made during a football match using a simple scale, e.g. *Good*, *OK* and *Bad*.

The facility to automatically rate passes provides a number of benefits for the analysis of football matches. Pass rating is a useful metric in player evaluation, and can be used to track the development of players, identify potential new recruits and analyse the strengths and weaknesses of future opponents. Furthermore, given that a large fraction of passes made are of average quality—in the experiments carried out for this research, $83\,\%$ of passes were rated as *OK* by observers—the ability of identify only

the particularly good and poor passes would improve the efficiency of tasks such as match video analysis.

State-of-the-art object tracking systems can capture the movement of players in football matches and produce spatio-temporal trajectory data on the players with high accuracy and frequency. Our intuition is that much of the information required to evaluate a pass is inherent in the movement of the players and the ball. The challenge is then to take the low-level trajectory signal as input and design a framework to produce the high-level pass rating.

The framework presented in this chapter uses data structures and algorithms from computational geometry to compute complex predictor variables, and then apply machine learning classification algorithms to produce an ordinal measure of passing performance. The results show that this combination is promising, and may be applied to problems in domains other than football.

The remainder of the chapter is structured as follows. In Section 3.2 we formalise the problem and the framework. The feature functions that compute the predictor variables used for the pass classification are described in Section 3.3 and the process for obtaining labelled examples in Section 3.4. The learning algorithms used are described in Section 3.5. The experimental setup and results are reported in Section 3.6, and an analysis of the results is provided in Section 3.7. A summary of the current state and future directions for research in this area is contained in Section 3.8.

3.1 Related Work

The work described in this chapter clearly fits into the research surveyed in Chapter 2, however we believe that the approach of using complex geometric structures to compute predictor variables for learning algorithms is novel. In this section, we point out several contributions that are directly related to this research.

Taki and Hasegawa [183] define a geometric subdivision called a *dominant region*, similar to a Voronoi region [14], that subdivides the playing-area into cells "owned"

by players such that the player can reach all areas of their cells before any other player. The dominant region considers the direction and velocity of the players. This concept is further developed by Fujimura and Sugihara [84] and Nakanishi *et al.* [153] who define efficient approximation models for calculating the dominant region.

Gudmundsson and Wolle [95] consider a related problem of computing the passing options that the player in possession has, based on the areas of the playing-area that are reachable by the other players. Kang *et al.* [120] uses a simplified motion model to compute a similar subdivision in order to define a set of performance measures for assessing player's passing and receiving performance.

Recent efforts using machine learning to extract information from spatial data are detailed in Section 2.3. Perhaps the most similar are those by Beetz *et al.* [13] who use decision tree learning to identify attributes of passes, and that of Maheswaran *et al.* [146] who used a classification algorithm on geometric input features to predict the outcome of a rebound in basketball.

Several recent application-focused papers have investigated passing as a measure of performance. Power *et al.* [165] provides an analysis of impact of passing on team performance. A logistic regression classifier was trained to obtain two binary response variables: *risk* and *reward*. The model used hand-crafted predictor variables, and the ground-truth response variables were obtained from the context and outcome of the possession in which the pass occurred. The classifier was then applied to all passes made during a season to facilitate player and team performance analysis. The same model was also used as a component of the analysis of the performance of Leicester City's premiership-winning season of 2015–2016 in Ruiz *et al.* [171].

3.2 Preliminaries

We formally define the problem and the solution framework in this section. Given input of the player movement trajectories and a list of the events during a match as defined in Section 1.2, then we wish to apply a label from an ordered set, such as

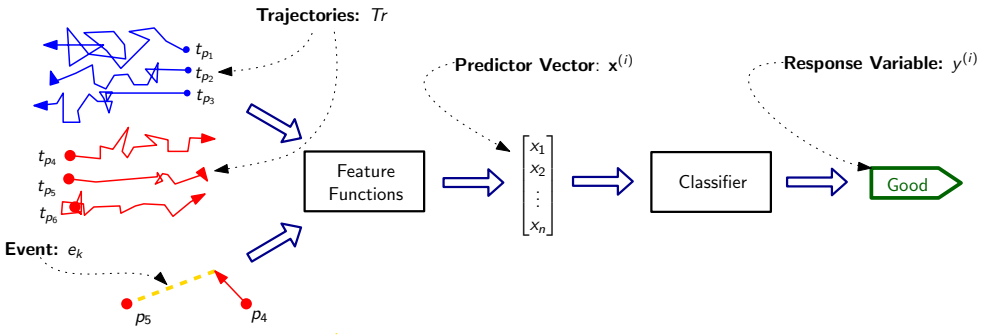


Figure 3.1: The high-level framework for pass classification. The first component, described in Subsection 3.2.1, produces a vector $\mathbf{x}$ for each pass from the raw input. The second component is the classification function h.

$\{Good, OK, Bad\}$, to each pass made. The overall objective is to find a classification function that, for each pass made, outputs a label for the pass.

The approach that we selected was to use supervised machine learning algorithms to learn a classification function to produce the ratings. These algorithms are designed to learn the required classification function from a set of training examples where the required output is already known: the *ground-truth* labels. In this case the ground-truth is a rating for each pass that has been made by human observers. The learning algorithm accepts the training examples and corresponding ground-truth labels and determines a classification function whose output labels are consistent with the ground-truth labels.

The input to both the learning algorithm and resulting classification function is a vector of numeric values that ideally contains sufficient information to accurately rate each pass event. However, the raw input trajectory and event data is not suitable as input into the classification function, and thus a preprocessing step is required to compute a *predictor vector* from the input for each pass event. Thus, the proposed framework has two components, and a high-level digram is provided in Figure 3.1.

There are many supervised machine learning algorithms designed to learn such classification functions, and each algorithm may have parameters known as *hyper-parameters*. By varying the choice of learning algorithm and hyper-parameters, many

candidate classification functions can be computed, and an objective method for comparing them is required. We use an *evaluation function* that can score the performance of the classification function on a set of test examples where the ground-truth labels are available. In this way, different classification functions can be compared and the best performing candidate can be identified.

In this section, we outline the structure of the framework by formalising the inputs and output of each component. First, we describe how the input trajectory and event data is preprocessed into vectors of predictor variables that are input to the classification function. Next, we describe the evaluation functions used to compare different classification functions. Finally, using the notation in the preceding paragraphs, we present a formal statement of the problem.

3.2.1 Predictor Variables

The raw trajectory and event data is used in a preprocessing step to compute a *predictor vector* for each pass made. The values contained in this vector are intended to contain sufficient discriminative information so that it is possible to train a classifier to make a good prediction of the quality of the pass. Each component of the predictor vector is a *predictor variable* and is computed using a *feature function*. The predictor variable for a pass can (in principle) depend upon all the previous and future events of the match and period in which the pass occurs. Similarly, a predictor variable can depend on the current location of all players, and possibly their previous and future locations.

Using the notation defined in Section 1.2, let ϕ_j be the feature function that produces the j-th predictor variable for a *Pass* event:

$$\phi_j : M \times W \times S \times \mathcal{T} \times \mathcal{E} \rightarrow \mathbb{R}$$

Feature function $\phi_j(m, w, s, T_{m,w}, E_{m,w})$ computes a predictor variable for the *Pass* event that occurs at time-stamp s of period w of match m, using the set of trajectories $T_{m,w}$ and event log $E_{m,w}$ for match m and period w.

It is convenient to work with a vector of predictor variables for the i-th pass event:

$$\mathbf{x}^{(i)} = \begin{bmatrix} \phi_1(m, w, s, T_{m,w}, E_{m,w}) \\ \phi_2(m, w, s, T_{m,w}, E_{m,w}) \\ \vdots \\ \phi_n(m, w, s, T_{m,w}, E_{m,w}) \end{bmatrix}$$
$$\mathbf{x}^{(i)} \in \mathbb{R}^n$$

Finally, in order to train the learning algorithms, example labels are required for each class as response variables. The response variable is drawn from an ordered set e.g. $Y = \{Good, OK, Bad\}$. Combining the response variable vectors with the corresponding predictor variables gives a matrix of training examples.

$$\mathbf{X} = \begin{bmatrix} —(\mathbf{x}^{(1)})^\top— \\ —(\mathbf{x}^{(2)})^\top— \\ \vdots \\ —(\mathbf{x}^{(m)})^\top— \end{bmatrix} \qquad \mathbf{y} = \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(m)} \end{bmatrix}$$
$$\mathbf{X} \in \mathbb{R}^{m \times n} \qquad \mathbf{y} \in Y^m$$

Section 3.3 gives a detailed discussion of the feature functions used in the framework.

3.2.2 Learning Algorithm and Classification Function

Given a training set of pass events $\mathbf{X}$ and labels $\mathbf{y}$, the learning algorithm is run to determine a parameterisation θ that characterises a classification function

$$h_\theta : \mathbb{R}^n \mapsto Y.$$

The parameterisation θ should be optimal in the sense that the corresponding h_θ produces output labels that are close to the ground-truth. Each algorithm has an associated cost function $J(\mathbf{X}, \mathbf{y}, \theta)$, and will select the parameterisation, $\hat{\theta} \in \mathbb{R}^m$ such

that

$$\hat{\theta} = \underset{\theta}{\operatorname{argmin}}\, J(\mathbf{X}, \mathbf{y}, \theta).$$

This parameterisation characterizes the classifier function $h_{\hat{\theta}}(\mathbf{x})$ that will predict the response variable $y^{(i)}$, given the input vector $\mathbf{x}^{(i)}$.

Section 3.5 details the learning algorithms used in the experiments.

3.2.3 Evaluation Functions

Each learning algorithm and hyper-parameterisation run on a training set of data will produce a classification function h_θ, and we can thus compute a set of candidate classification functions H. Given such a set H, we need a method to compare how well each function performs the task of predicting the ground-truth label for each pass, and for this we define an evaluation function.

Consider a vector of m ground truth labels $\mathbf{y}$ for a given input $\mathbf{X}$ and the labels $\hat{\mathbf{y}}$ computed by $h_{\hat{\theta}}$ on the same input $\mathbf{X}$. An evaluation function $\rho : Y^m \times Y^m \mapsto \mathbb{R}$ compares vectors to compute a measure of the similarity between the vectors. An evaluation function ρ can thus be used to compare classification functions to find the function that induces the optimal score for the evaluation function within the set.

3.2.4 Problem Statement

Using the framework and notation described above, we define a function $h : M \times W \times S \times \mathcal{T} \times \mathcal{E} \times \mathcal{M}_\mathcal{T} \times \mathcal{M}_\mathcal{E} \mapsto Y$ for the overall task. Thus, given a time-stamp, period and match and the full input data, h produces a label y for the pass. We wish to find the "best" h within the set of functions that we search, and thus we are now able to formally define the decision problem.

PROBLEM PASS RATING

Instance: The set of player trajectories $\mathcal{T}$, the set of match events $\mathcal{E}$, the trajectory and event mapping sets $\mathcal{M}_{\mathcal{T}}$ and $\mathcal{M}_{\mathcal{E}}$, a labelling $\mathbf{y}$ of ground-truth ratings for each pass event, an evaluation function ρ, and a threshold value t.

Question: Is there a function $\hat{h}$ that, given an instance as input, computes a rating for each pass $\hat{\mathbf{y}}$, such that $\rho(\mathbf{y}, \hat{\mathbf{y}}) > t$?

3.3 Predictor Variables

The objective of the feature engineering task is to extract information from the spatio-temporal match data so that the classifiers are able to make accurate inferences about the quality of the passes made in a match. To place this in context, consider how an informed observer of a football match would make an assessment about the quality of a pass.

At a basic level, the observer would consider the fundamentals of the pass, such as the distance and speed of the pass and whether the intended recipient of the pass was able to control the ball. These are the basic geometric aspects of the pass, but even at this level, the observer is required to make some inferences, such as who the intended recipient of the pass is. They would also likely consider the context of the match state when the pass was made. For example, was the passer under pressure from opposition players? To make such assessments, the observer would consider the positions of the players and the speed and direction that they were moving in, and the observer would make assumptions about whether the defending players are physically able to influence the pass by pressuring the passer or intercepting the pass. The observer thus has a mental model of the physiological capabilities of the players, and will consider this in their estimation of the quality of a pass made.

At a higher level, passes are not made simply to move the ball from one player to a teammate, but also to improve the tactical or strategic position of their team. Passes can be made to improve the position of the ball, typically by trying to move the ball

closer to the opponent's goal in order to have an opportunity to score. Passes may also be made to improve the match state by moving the ball from a congested area of the playing-area to a location where the team in possession has a numerical or positional advantage. Meanwhile, the opposition will be actively trying to reduce the options of the player in possession to make passes. Thus, the match observer would need to consider the tactical and strategic objectives of the passer, and thus would have an understanding of the tactics and strategies employed by the player and team, and apply them to their estimate [197]. Likewise, the observer would consider the defensive team and their strategies and tactics.

A football match can also be viewed as a sequence of events occurring at particular times. The event-type that we are concerned with, *Pass*, can thus be viewed as part of a sequence. This sequence can be subdivided in various ways, for example by unbroken sub-sequences of events where a single player or team is in possession of the ball, or by a sub-sequence of events that occur between stoppages in play such as fouls, goals or injuries. When assessing the quality of a pass, the observer may consider the context of the pass in the sequence of events.

Finally, the observer may also consider the opportunity cost of the pass. By making the pass, the player forsook the other options available, such as passing to other players, dribbling or shooting.

For the trained observer, synthesising all this disparate information and making a prediction is a mental exercise that can be done in a matter of seconds. The problem described in this chapter is to replicate this in a computational process.

3.3.1 Feature Functions

Feature functions are used to compute the predictor variables that are input to the classifier. The feature function $\phi_j(m, w, s, T_{m,w}, E_{m,w})$ outputs the j-th predictor variable for the i-th pass event. The predictor variables are divided into the following categories in a manner consistent with our analysis of the types of information discussed above. The full list of features is provided in the Appendix.

Basic Geometric Predictor Variables

The basic geometric predictor variables are derived from the basic orientation of the players and ball within the playing-area. The feature functions for these predictor variables implement simple geometric operations such as determining angles between points, measuring Euclidean distances, and calculating velocity of objects over a time interval. Typical examples of basic features would be: the distance of a pass; the velocity of the pass; and the angle of the pass relative to the direction the passer is facing.

Sequential Predictor Variables

Sequential predictor variables are constructed from the event sequence data. Currently three types of sequences are modelled: player possession, where a single player is in possession; team possession, where players from the same team are in uninterrupted possession; and play possession, where events between stoppages are grouped in a sequence. Examples of these predictor variables are the ordinal position of the event in a sequence, the duration of the sequence, and the event that is the final outcome of the current sequence.

Physiological Predictor Variables

Physiological predictor variables incorporate some aspect of the physiological capabilities of the players, generally how quickly they can reach a given point. Inherent in these predictor variables is a motion model that simulates the physical capabilities of the players. This is discussed in detail in Subsection 3.3.2.

Strategic Predictor Variables

Strategic predictor variables are designed to provide some information about strategic elements of a football match. The approach taken was to design predictor variables based on the dominant region structure proposed in [183], see Subsection 3.3.3.

3.3.2 Player Motion Model

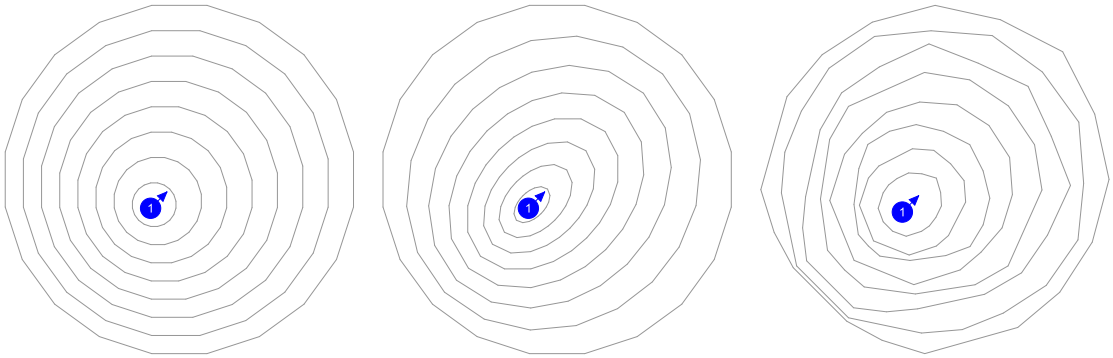
The physiological and strategic features used in the model are based to some degree on an estimate of how quickly a player currently travelling at a given speed and direction can reach a given point. In particular, if a player can reach a point before any other player, then that player is said to dominate the point. This notion is the basis for the physiological and strategic predictor variables we have defined in our model. In order to determine the time required to reach a given point, a motion model of the player is required.

The motion model is defined as a function g for a player p that takes as arguments: the coordinates of a point $(x, y) \in \mathbb{R}^2$, the trajectory τ_p of player p, and the time-step s at which to determine the distance. The function returns the time $t \in \mathbb{R}_{\geq 0}$ it would take for the player to reach the point. As this time is dependent on the existing direction and velocity of the player, these factors are extrapolated from the player's trajectory at time-step s.

$$g(x, y, \tau_p, s) = t \tag{3.1}$$

Gudmundsson and Wolle [95] propose three simple motion models to approximate the *reachable region* for a player, that is, the region a player can reach in a given time. These models discretize the reachable region by introducing a fixed set of time-steps T where the boundary of the reachable region is computed. Thus, for each $\mathrm{t} \in \mathrm{T}$, the motion model contains a closed boundary curve that surrounds the initial starting point of the player. Furthermore, each curve is approximated by an n-sided polygon. The three motion models considered are based on a circular boundary, an elliptical boundary and a boundary constructed by sampling from the trajectory sequences, see Figure 3.2.

When computing the reachable region, the time that it would take for a player positioned at a certain point to reach another point is determined by a number of factors. The maximum achievable velocity of the player is a factor, as is the speed and direction



(a) Circle (b) Ellipse (c) Data-driven

Figure 3.2: Reachable region boundaries produced using different motion models. The grey polygons surrounding the player are the boundary of the area that the player can reach in a given time, and are based on the speed and direction that the player is moving.

the player is heading.

The motion models are used in several of the feature functions, in particular those based on the dominant region, below. We evaluated all three motion models in [95], however since the variance between the models is small, particularly for small distances such as on a football field, we consider only the ellipse motion model here, see Figure 3.2.

For more details about the motion models, we refer the interested reader to [95].

3.3.3 The Dominant Region

Taki and Hasegawa [183] presents the dominant region as a dynamic area of influence that a player in a football match can exert dominance over, where dominance is defined as being the regions of the playing-area that the player is able to reach before any other player. We propose to use the dominant region as a measure to approximate the strategic position of a team at a given point in time, and to subsequently construct feature functions based on it. Furthermore, we use the dominant region to also construct predictor variables that model the pressure exerted on the player in possession of the ball by opposition players in close proximity.

Intuitively, these appear to be useful predictors for the task of rating passes. The passing player wants to put the ball at a point where the intended recipient can reach it first, and this is, by definition, in the receiving player's dominant region. Thus, the proportion of the playing-area that the team in possession dominates is a factor in the passing options of the passing player. Similarly, the size of the dominant region surrounding the passing and receiving players provide information about the pressure the player is under.

The playing-area can be partitioned into dominant regions, each dominated by a particular player. It is thus conceptually similar to the Voronoi region [14], the difference being the function that determines the region that a particular point belongs to: the function for a Voronoi region is usually the Euclidean distance; and for the dominant region is the time it takes for a player to reach a given point.

The dominant region is defined in [183] by the following equation for a player p_i at time-step s, where g is defined in Equation (3.1).

$$D(p_i, s) = \{(x, y) \in \mathbb{R}^2 \mid g(x, y, \tau_{p_i}, s) \leq g(x, y, \tau_{p_j}, s) \text{ for } j \neq i, \tau_{p_i}, \tau_{p_j} \in \mathcal{T}, s \in S\} \tag{3.2}$$

The subdivision of the dominant regions for all players will thus partition the playing-area into cells. An approximate version can be seen in Figure 3.3. However, $(x, y) \in \mathbb{R}^2$ is continuous, and there is currently no algorithm available to efficiently compute this continuous function. For simple distance functions, such as Euclidean distance, the dominant region can be computed efficiently [73]. However, in our setting, the distance function is more complex. In fact, the dominant region of a player may not even be a single connected region [183]. Computing the intersection of surfaces in three dimensions, as required in Equation (3.2) is non-trivial and time-consuming. As such, we use a discrete algorithm to compute an approximation of the dominant region, presented in Subsection 3.3.4. Taki and Hasegawa [183] and Nakanishi et al. [153] both present approximation algorithms where (x, y) is approximated by a discrete grid $Y \subset \mathbb{R}^2$, and the dominant regions are thus computed for all points

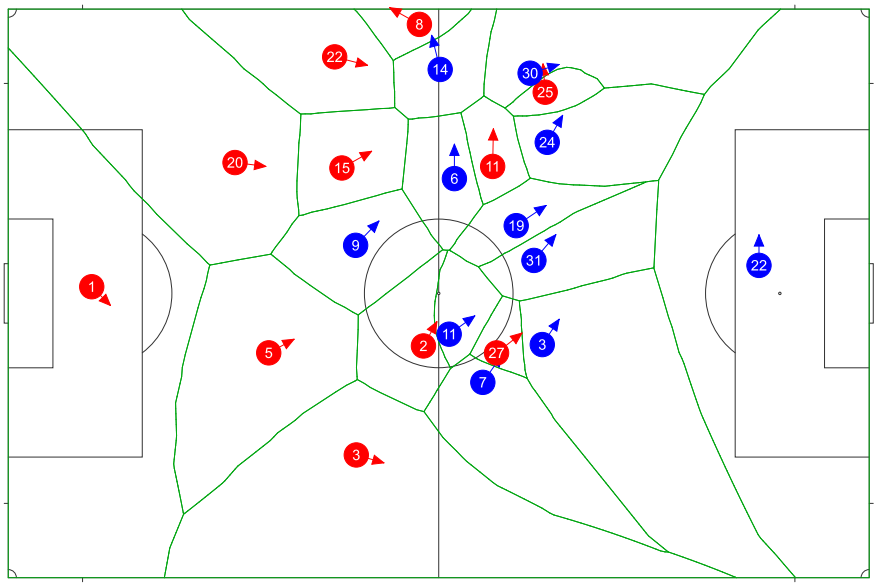


Figure 3.3: Dominant regions constructed using ellipse motion model. The playing field is subdivided into regions surrounding each player. For each player, the surrounding region is the area of the playing field that the player can reach before any other player. Note how the player's speed and direction can induce concave boundaries, such as the boundary around "Red-25", top-right.

in Y.

Ten of the feature functions constructed are based on the dominant region and the motion model. Examples of these feature functions are the dominant region when the player in possession passes; the net change in dominant region area between when a pass is made and when it is received; and the pressure the passer is under, defined by the area of the passer's dominant region. A brief description of the feature functions is provided as an appendix. The intuition is that the computed predictor variables will be important to the classifier, as they provide domain-specific information about the physiology and strategy of the players; and this information would not otherwise be available to the classifier. We investigate the importance assigned to these features by the classification algorithms in the experiments, and provide an analysis of the results in Subsection 3.7.2.

3.3.4 Discrete Algorithm to Approximate Dominant Region

This section describes a discrete algorithm for efficiently computing an approximation of the dominant cell arrangement of the playing-area at a given point in time. The algorithm, outlined in Algorithm 1, has three steps. First, for every pairwise combination of players, the intersection points are determined between the reachable region polygons for each time-step. In the second step, the intersection points are used to produce a reachable boundary between each pair of players. This is done using a modified version of Kruskal's minimum spanning tree algorithm [125], constrained so that the degree of every vertex is at most two, and thus the output is a set of one or more disconnected paths that span the intersection points, see Figure 3.4. The third step constructs the smallest enclosing polygon around each player from the boundaries, and this is the player's dominant region.

Algorithm 1 Discrete algorithm to compute the approximate dominant region at a given time s.

Require: $P \leftarrow \{p_o, \cdots, p_n\}$ ▷ Players
Require: $\mathrm{T} \leftarrow \{\mathrm{t}, \cdots, \mathrm{t}_{max}\}$ ▷ Time-steps for boundaries
1: $M \leftarrow \emptyset$ ▷ Set of MSPs
2: $D \leftarrow \emptyset$ ▷ Set of dominant regions
3: **for all** pairs of players $(p_i, p_j) \in P \times P$ **do**
4: $I \leftarrow \emptyset$ ▷ The set of intersection points
5: **for all** time-steps $\mathrm{t} \in \mathrm{T}$ **do**
6: $x \leftarrow$ INTERSECTIONPOINTS(p_i, p_j, t) ▷ Compute intersection points
7: $I \leftarrow I \cup \{x\}$
8: **end for**
9: $m \leftarrow$ MSP(I) ▷ Compute "minimum spanning path" boundary
10: $M \leftarrow M \cup \{m\}$
11: **end for**
12: **for all** players $p \in P$ **do**
13: $d \leftarrow$ DOMINANTREGION(p, M) ▷ Compute dominant region for p as smallest enclosing polygon of boundaries
14: $D \leftarrow D \cup \{d\}$
15: **end for**
16: **return** D ▷ The set of dominant regions for all players

In the following paragraphs we detail the three steps of the algorithm. The first step is to compute the intersection points between the reachable regions of each pair of

players. The intersection points at time s are determined using a line segment intersection algorithm [14]. Each intersection point v_j has a time-step attribute $\mathrm{t}(v_j)$ denoting the time-step of the reachable region polygon that it was constructed from. In most cases, there will be zero or two intersection points between the polygons however degenerate cases exist where there are one, or three or more intersection points [110]. The intersection points $V = \{v_0, v_1, \ldots, v_m\}$ between the pair of players for all time-steps are collected. A graph $G = (V, E)$ is constructed using V as vertices and adding edges to E between two points if either: the time-step of the two vertices are adjacent; or the two vertices have the same time-step and this time-step is the minimum of all intersection vertices:

$$E = \{(v_i, v_j) \mid (\mathrm{t}(v_j) = \mathrm{t}(v_i) + 1) \ \vee (\mathrm{t}(v_i) = \mathrm{t}(v_j) = \min(\{\mathrm{t}(v_k) \mid k \in V\}))\}.$$

The second step in the algorithm takes the graph G as input and computes the reachable boundary between each pair of players. G contains edges between each intersection point for consecutive time-steps. Typically each vertex in V will have degree four, with edges to two vertices whose time-step is immediately prior to the time-step of the current vertex, and two edges to the vertices whose time-step is immediately subsequent. The objective of this step is to prune the graph so that each vertex in the graph has degree no greater than two, and that the edges retained will be the shortest edges. The pruned graph will thus be a path. This is performed using a modified version of Kruskal's algorithm [125]. The algorithm is modified so that an edge (v_i, v_j) is added to the output tree only if $\deg(v_i) < 2$ and $\deg(v_j) < 2$. This modification means that the algorithm will return "spanning paths", i.e. the set of paths from the input graph that span the connected components of the graph. In most cases there will be a single path, see Figure 3.4. If there are two or more paths, select the path that intersects the bisector between the sites of the two players. The graphs for each pair of players are collected into a single graph $G_{\mathrm{MSP}} = (V_{\mathrm{MSP}}, E_{\mathrm{MSP}})$. The edges in this graph are the line segments that form the reachable boundaries of each player.

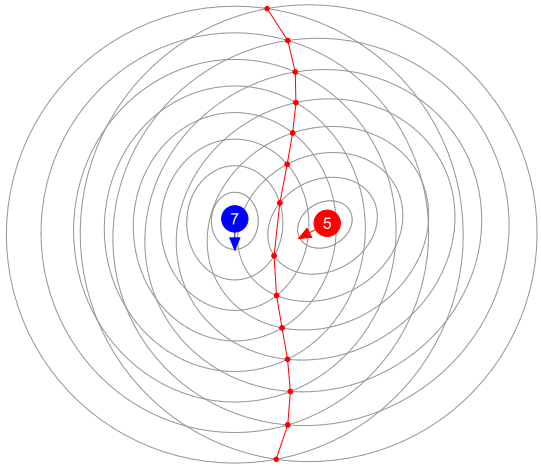


Figure 3.4: The intersection of time-step polygons defines the boundary between player's dominant regions, computed using the ellipse motion model. The boundary contains the points that both players are able to reach at the same time.

The final step of the algorithm is to compute the dominant region for each player by collecting the line segments that comprise the boundaries the player has with all other players, and then determining the polygon that encloses the site of the player. The algorithm shoots a ray in an arbitrary direction from the site of the player and locates the closest intersecting line segment. The algorithm then collects the closest line segments by repeatedly "turning left" and walking to the next intersection with another line segment. This loop stops when the original intersection point is again encountered again. As such, this algorithm walks around the innermost polygon surrounding the player's site, and this polygon defines the player's dominant region. The dominant regions of all players induce a subdivision of the playing-area, as required, see Figure 3.3.

The time complexity of the discrete algorithm to compute the approximate dominant regions is $O(n^2)$, where n is the number of line segments on the boundary of all of the m reachable regions used in the approximation. The following is a sketch of the computational complexity of the approximation algorithm. The time and space complexity of the first two steps of the algorithm is $O(n \log n)$. However, the complexity includes a large constant of $\binom{22}{2}$ as these steps must be carried out pairwise for all 22 players. The worst-case complexity of the third step (determine the enclosing

polygon) is quadratic in the number of line segments in the player motion model. For each line on the polygon boundary, we need to compute the closest intersecting line segment which is done by computing the intersection point for the line segment with every other line segment, which has linear complexity. The number of lines on the polygon boundary is at most n thus yielding the quadratic complexity for this step, and for the overall algorithm. However, in practice this value is much smaller than n. In particular, the polygons of reachable regions will only intersect at zero or two points in most cases, rather than some fraction of n, hence resulting in a total running time of $O(n \log n)$ for most practical instances.

3.4 Label Data

The supervised machine learning algorithms used in the experiments learn from training examples that have been labelled with the ground-truth values; in this case a rating of each pass' quality. This section describes the process used to capture and validate the label data.

3.4.1 Labelling Process

The labels used to train the classifiers were created by two human observers watching video footage of the matches. Each observer separately watched video clips of the passes and assigned a rating to each pass using a six-point Likert scale consisting of ratings *Very Good*, *Good*, *Marginally Good*, *Marginally Bad*, *Bad* and *Very Bad*. The experiments, described in Section 3.6 were carried out with two labelling schemes: the first using the full six-point scale; and the second using a three-point scale where the ratings were aggregated into the labels: *Good*, *OK* and *Bad*.

The video clip for each pass included two seconds of footage preceding and following the pass event. The intention was to provide sufficient context to rate the pass, and to ensure that each pass was rated in isolation, but not to include longer-term considerations such as the eventual outcome of a sequence of possession.

The candidate passes were selected using events labelled as *Pass* in the Event Log, see Section 1.2, and includes both successful and unsuccessful passes (i.e. where the ball did not reach the intended receiver). This approach clearly adds an additional layer of subjectivity in which movements of the ball are labelled as passes, as the player's intention is unknown in terms of whether the movement was indeed a pass and also who the intended recipient was. In contrast, the approach taken in Power *et al.* [165], and Ruiz *et al.* [171] was to automatically label the passes using a function on the context that the pass was made on. Such an approach would eliminate the subjective nature of the labelling process described here, however would introduce a different source of error in that the computed labels may not adequately capture passing player's intention. An analysis of the correlation between the two labelling methods would be interesting to consider in future work.

When evaluating the pass, the rater was asked to consider the following questions when determining the rating to apply.

- What is the technical quality of the pass?
- Did the pass reach the intended receiver?
- Was the pass made with appropriate speed and accuracy?
- Was the receiver able to control the ball without having to change direction or speed?
- Did the pass improve the field position of the team?
- Was the pass risky or lucky in any way?
- Did the pass change the relative pressure of the player in possession? i.e. was the passer under more or less pressure from the opposition than the receiver?
- Was the pass the best option that the player had? For example, would the player have been better to make an alternative pass, dribble the ball, or shoot?

3.4.2 Process Validation

The classification process was designed using principles and techniques described by Lincoln and Guba [137]. The intention was to produce a robust set of labels created using a consistent process.

Prolonged engagement

Each observer viewed video clips of over 2,900 passes, consistent with Lincoln and Guba's technique of persistent observation. The intention was to ensure that the observers would identify characteristics that were most relevant to the quality of a pass.

Member checking

After two matches had been labelled, the difference between the labels assigned by the observers was calculated. The passes where the distance between the two ratings was two or more were then selected. The observers then viewed the footage of these passes together and discussed their reasoning. The purpose of this was to explore the characteristics of a pass that impacted the classifications, and was intended to ensure a consensus on the significant characteristics.

Triangulation

The labelling task was carried out by two observers so that the labels used in training the classifier did not rely on a single source. The two labels for each pass were compared, and where there was not agreement, the label nearest the middle of the scale was selected. The rationale for this decision was to produce a conservative set of labels.

3.4.3 Analysis of Classification Results

The ratings made by the observers are subjective, and it is natural to expect some disagreement between raters in their labels. Cohen's kappa [57] is a heuristic measure of agreement between raters that takes into account the possibility that ratings can

agree by chance, with a value of 0 denoting chance agreement and 1 denoting perfect agreement. Formally, Cohen's kappa is defined as:

$$\kappa = \frac{p_o - p_e}{1 - p_e}$$

Here, p_o is the fraction of observations where the raters were in agreement, and p_e is the fraction of observations that could be expected to agree by chance.

The ratings made by the two observers yielded a Cohen's kappa of 0.393 on the six-point scale which indicates a moderate level of agreement. The value of the kappa for the three-point scale was 0.697, which indicates a high level of agreement, see kappa between Annotator 1 and Annotator 2 in Table 3.4. Although, Cohen's kappa does not have a statistical significance, it suggests that the observers produced similar outcomes.

3.5 Learning Algorithms

The pass rating task is a classification problem and we evaluate several supervised machine learning algorithms for this task. The learning algorithm computes a classification function $h_\theta(\mathbf{x}^{(i)})$ that can predict the ground truth variable $y^{(i)}$ for a given input vector of predictor variables, $\mathbf{x}^{(i)}$. The algorithm is trained on the labelled example data with the objective of learning a parameterisation $\hat{\theta}$ such that the prediction error measured by the function J is minimised.

The nature of the pass classification problem, and properties of the data informed the choice of learning algorithms that were selected. The distribution of example data used in our experiments is unbalanced amongst the classes. The majority of examples were clustered towards the middle of the scale, see Tables 3.1(b) and 3.1(a), and thus learning algorithms that are robust to class imbalance were selected.

The rating labels that we wish to predict for each pass have a natural ordering, and thus the classification problem can be considered as an ordinal classification problem, or using a weaker formulation as a multinomial classification problem where the

ordering of the labels is ignored.

We examined two support vector machine (SVM) classifiers—c-SVC and ε-SVR; the RUSBoost classifier; the multinomial logistic regression (MLR) with three different cost functions; and the ordered logistic regression classifier.

The intention was to perform the experiments using diverse types of learning algorithms: the logistic regression classifiers attempt to model the probability distribution of the labels, given the input predictor variables [148], [149]; SVM is a maximum margin classifier [189] and RUSBoost is an ensemble method that utilises sampling and boosting of weak classifiers [172].

Moreover, the SVM algorithms accept a per-class weight vector as a hyperparameter for dealing with imbalanced classes, and RUSBoost is designed to mitigate the effect of class imbalance, and thus were considered to be appropriate choices for the experiments.

Two of the chosen classifiers are ordinal: ordered logistic regression and the ε-SVC variety support vector machine. The remaining classifiers are multinomial classifiers.

For multinomial logistic regression, the cost function J is computed as the sum of an empirical risk term and a regularization term. We evaluated three models learned using different empirical risk functions. The function (3.3) is the risk function that minimises the error between the observed values and the predicted values calculated using the maximum likelihood estimation of θ. The arithmetic (3.4) and quadratic (3.5) risk functions are intended to perform better under class imbalance conditions by computing the per-class risk [138]. The arithmetic risk takes the sum of the per-class error, whereas the quadratic risk uses the root of the sum of the squared errors for each class.

$$f_{\theta_j}(\mathbf{x}^{(i)}) = \log \frac{e^{\theta_j^\top \mathbf{x}^{(i)}}}{\sum_{l=1}^{k} e^{\theta_l^\top \mathbf{x}^{(i)}}}$$

$$R_{\mathrm{L}}(\mathbf{X}, \mathbf{y}, \theta) = -\frac{1}{m}\left[\sum_{i=1}^{m}\sum_{j=1}^{k} \mathbb{I}[\{y^{(i)} = j\}] f_{\theta_j}(\mathbf{x}^{(i)})\right] \tag{3.3}$$

$$R_{\mathrm{A}}(\mathbf{X}, \mathbf{y}, \theta) = -\frac{1}{k}\sum_{j=1}^{k}\left[\frac{1}{m_j}\sum_{i=1}^{m} \mathbb{I}[\{y^{(i)} = j\}] f_{\theta_j}(\mathbf{x}^{(i)})\right] \tag{3.4}$$

$$R_{\mathrm{Q}}(\mathbf{X}, \mathbf{y}, \theta) = -\sqrt{\frac{1}{k}\sum_{j=1}^{k}\left[\frac{1}{m_j}\sum_{i=1}^{m} \mathbb{I}[\{y^{(i)} = j\}] f_{\theta_j}(\mathbf{x}^{(i)})\right]^2} \tag{3.5}$$

$$J(\mathbf{X}, \mathbf{y}, \theta) = R(\mathbf{X}, \mathbf{y}, \theta) + \lambda \|\theta\|_p \tag{3.6}$$

$$\hat{\theta} = \operatorname*{argmin}_{\theta} J(\mathbf{X}, \mathbf{y}, \theta) \tag{3.7}$$

Here, the function $\mathbb{I}[\cdot]$ is the indicator function that accepts a logical proposition as argument and returns 1 if true, and 0 if false. The cost function J contains a regularization term, and we evaluate regularization terms based on the ℓ_1- and ℓ_2-norms in the experiments, i.e. $p \in \{1, 2\}$ in Equation (3.6). Moreover, the ℓ_1-norm will induce a sparse parameterisation of θ [119], and we investigate the predictor variables whose corresponding value in θ is non-zero as a measure of the importance of the predictor variable.

3.6 Experiments

The experiments were designed to address the following questions:

1. Is it possible to find a classification function and a set of predictor variables to accurately predict the quality of a pass?

2. To what extent did the predictor variables computed using algorithms and data structures from computational geometry contribute to the performance of the classifier?

Table 3.1: Class frequencies for (a) three- and (b) six-class labelling schemes

(a) three-class

Class	Rel. Freq.	Count
Good	0.067	195
OK	0.830	2,433
Bad	0.104	304

(b) six-class

Class	Rel. Freq.	Count
Very Good	0.008	24
Good	0.058	171
Marginally Good	0.624	1,829
Marginally Bad	0.206	604
Bad	0.088	257
Very Bad	0.016	47

3.6.1 Setup

The objective of the experiments was to learn an optimal vector of parameters $\hat{\theta}$ that yield a classification function $h_{\hat{\theta}}(x)$ such that the classification function makes correct predictions on unseen examples.

The setup for the experiments was as follows. The training sample $\mathbf{X}$ and corresponding labels $\mathbf{y}$ were randomly permuted and then split into a training set of $80\,\%$ of the examples, $\mathbf{X}_{\text{TRAIN}}$ and $\mathbf{y}_{\text{TRAIN}}$, and a test set of the remaining $20\,\%$, $\mathbf{X}_{\text{TEST}}$ and $\mathbf{y}_{\text{TEST}}$. Each predictor vector $\mathbf{x}^{(i)}$ in the training examples had 114 components, 10 of which were derived from the dominant region arrangement.

A set of candidate classifier models was defined by performing a grid search over a range of hyper-parameters to the classifier algorithm.

Each model was then trained on the training set using tenfold cross validation, and then evaluated by using the model to make predictions $\mathbf{y}_{\text{PRED}}$ on the test set predictor vectors $\mathbf{X}_{\text{TEST}}$. The predicted labels $\mathbf{y}_{\text{PRED}}$ were compared with the ground truth tables $\mathbf{y}_{\text{TEST}}$ and the model evaluation metrics were computed using the following functions: accuracy, precision, recall, F_1-score, mean absolute error (MAE$^{\mu}$), macro-averaged mean absolute error (MAE^{M}), mean squared error (MSE$^{\mu}$), macro-averaged mean squared error (MSE^{M}). The functions for precision, recall and F_1-score were also macro-averaged, i.e. the metrics were calculated on a per-class basis, and a simple mean of the per-class values was used as the metric. The MAE and MSE evaluation functions are ordinal in that they compute the distance in the ordered set Y between y and $\hat{y}$, and will penalise misclassified predicted values that are a further from the

Table 3.2: Summary evaluation metric results for experiments using (a) three- and (b) six-class labelling schemes. The greatest obtained value for each column is shown in bold.

(a) three-class

Classifier	Accuracy	Precision	Recall	F_1-score	MAE$^\mu$	MAE^M	MSE$^\mu$	MSE^M
MLR	**0.902**	0.812	0.717	0.748	**0.101**	0.893	**0.107**	0.980
MLR-Arith	0.898	0.847	0.802	0.737	0.105	0.670	0.113	0.820
MLR-Quad	0.895	0.847	0.802	0.732	0.109	0.670	0.117	0.820
c-SVC	0.898	**0.882**	0.818	0.741	0.106	0.609	0.111	0.728
ϵ-SVR	0.898	0.854	0.686	0.746	0.105	0.983	0.109	1.070
RUSBoost	0.838	0.676	**0.864**	**0.759**	0.169	**0.469**	0.182	**0.606**
Ordered Logit	0.888	0.741	0.690	0.714	0.116	0.971	0.122	1.054

(b) six-class

Classifier	Accuracy	Precision	Recall	F_1-score	MAE$^\mu$	MAE^M	MSE$^\mu$	MSE^M
MLR	**0.704**	0.581	0.407	**0.461**	**0.326**	4.041	0.399	5.248
MLR-Arith	0.697	0.541	0.426	0.442	0.337	3.924	0.412	**4.773**
MLR-Quad	0.684	0.483	0.426	0.444	0.354	3.924	0.443	4.782
c-SVC	0.693	0.630	0.454	0.437	0.334	**3.809**	0.394	4.834
ϵ-SVR	0.695	**0.639**	0.394	0.447	0.335	4.287	**0.388**	5.571
RUSBoost	0.572	0.360	**0.484**	0.402	0.575	3.811	0.908	5.772
Ordered Logit	0.693	0.576	0.377	0.454	0.339	4.602	0.409	6.421

ground-truth value [9].

Each experiment was repeated ten times with a different random split between the training and test sets, and the final evaluation metrics were calculated as the mean over the ten iterations.

3.6.2 Results

The results of the three-class experiments are summarised in Table 3.2(a) and the six-class results in Table 3.2(b). For the first four evaluation metrics, the values are in the range $[0, 1]$ and a larger score indicates better performance. The MAE metrics compute results in the range $[0, |Y| - 1]$ and the MSE metrics compute results in the range $[0, (|Y| - 1)^2]$. For both MSE and MAE, smaller values indicate better performance.

The evaluation metrics for both the three- and six-class experiments show that the classifiers produce broadly similar results, with the exception of RUSBoost. The best

RUSBoost classifier significantly outperformed all the other classifiers under the recall metric and underperformed when using the accuracy and precision metrics. Conversely, the other classifiers produced better precision than recall scores.

Interestingly, the ordinal classifiers—ordered logistic regression and ε-SVR—did not outperform on the ordinal evaluation metrics. On the other hand, the ordinal classifiers had similar scores for the other evaluation metrics, with the exception of recall, where they performed worse.

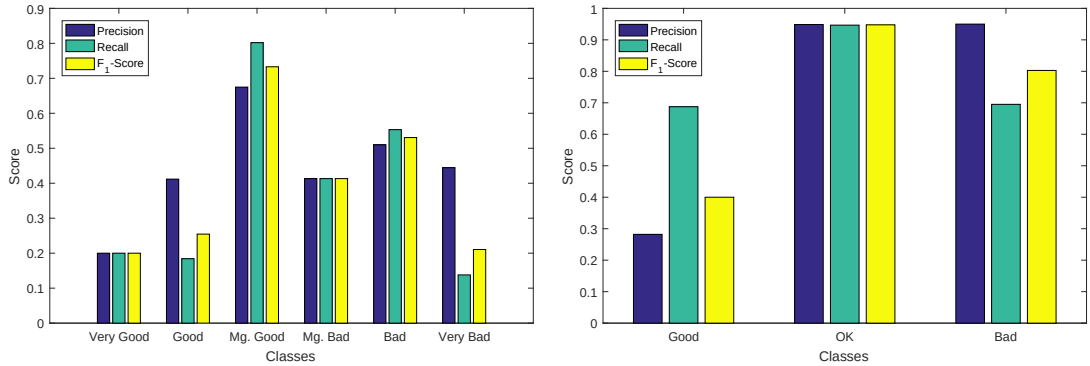
3.7 Analysis

In this section, we analyse the results of the experiments to validate the approach taken and identify any limitations that may have impacted the results. The observations in this section are predominantly based on the experiments carried out using the three-class label scheme. The analysis generally applies to the experiments using the six-class scheme as well, but classification task in this case is more difficult.

3.7.1 Classifier Performance

The experimental results in Table 3.2(a) show that, in the three-class case, it is possible to learn a classifier that performs well against all the evaluation metrics used. In the six-class case the classification task is more difficult, and the metrics obtained reflect this, see Table 3.2(b).

The performance of the classifier across the classes was variable. Figure 3.5 shows the metric values on a per-class basis using the three- and six-class label schemes. Classes with larger numbers of examples tend to result in higher metrics, which is to be expected. An exception is the *Marginally Bad* class in the six-class setup, which scores poorly even though $16.9\,\%$ of the examples are labelled with this class. An examination of the misclassification errors for these examples showed that there were many misclassification between the *Marginally Good* and *Marginally Bad* classes, suggesting that the classifier is unable to discriminate between these cases.



(a) Six-class label scheme (b) Three-class label scheme

Figure 3.5: Per-class obtained values of the evaluation metrics. The performance of the classifier was variable over the classes: as can be expected, classes with larger numbers of training examples performed better.

Furthermore, we examined the misclassification error rate on the training examples, and on the unseen test examples. There was no significant difference, e.g. for the MNR classifier, the training and generalization error rates were both $29.6\,\%$. In the three-class case, the error rates on the training and test examples were $9.9\,\%$ and $10.1\,\%$, respectively.

This implies that there was sufficient training data to learn classifiers that generalized well to unseen data, however the relatively poor performance of the classifiers on classes with fewer examples suggests that more examples would improve the performance of the classifiers.

3.7.2 Predictor Variable Importance

The predictor variables used in these experiments were computed using feature functions of varying complexity. This begs the question whether the effort involved in implementing and computing complex feature functions resulted in improved performance. We selected the 10 predictor variables that were computed using the dominant region described in Subsection 3.3.3.

The task of assessing and ranking the importance of predictors in a given classification task has been the subject of much research [100]. The ability to assess the

importance a classifier assigns to predictors is desirable for two related, but contradictory, reasons. First, it can provide an insight into the problem domain. For example, in case of the pass rating problem described in this chapter, the importance that a classifier assigns to a predictor may provide insight into the elements of a successful pass, which could be used by a coach to improve team performance. Conversely, if the classifier assigns high importance to predictors that are known to significant in the underlying problem domain, then this will provide assurance that the model is behaving as intended.

However, eliciting predictor importance from a classifier is not a straightforward proposition. The importance of a given predictor may not derive simply from its individual contribution, but also from the dependency between sets of predictors [156]. Moreover, while some classification algorithms, such as RUSBoost, compute measures of predictor importance, many do not, for example SVM. Standalone tests of predictor importance have also been proposed, see [100] for a survey.

We used the following tests to assess the importance of the predictors based on the dominant region.

Pearson's correlation coefficient

The correlation coefficient was computed between each individual predictor and the class labels. The predictors were then ranked by the absolute coefficient.

Single-predictor classifier

Multinomial logistic regression classifiers were learned and evaluated using a single predictor. Tenfold cross-validation was used and the resulting misclassification rate was collected for each predictor. The predictors were then ranked by the misclassification rate.

RUSBoost predictor importance

The importance of predictors to the RUSBoost classifier can be calculated by determining the risk for instances of the predictor in the ensemble relative to the sum of the risks of the instance's child vertices. The importance scores can thus be used to rank predictors.

ℓ_1-Regularized multinomial logistic regression

ℓ_1-regularization produces sparse coefficients for MLR classifiers [119]. Using the classifier parameterization that produced the highest accuracy in our experiments, we examined the predictors that had non-zero coefficients in the learned parameters. This test does not rank the predictors but examines the subset of 40 predictors that were selected for this classifier.

Table 3.3 summarises the results of the test for the dominant region-based predictors. The receiver pressure predictor is highly ranked by all three ranking tests, yet is not included by the ℓ_1-regularized MNR classifier. This can be explained by the fact that the receiver pressure predictor is a linear combination of two other predictors: the passer pressure and the passer/receiver pressure net change, both of which are included by the MNR classifier, and thus the receiver pressure information can be recovered by the classifier. Moreover, the receiver pressure and the receiver dominant region predictors are highly correlated ($93.0\,\%$) which is another possible reason why both were not retained by the MLR classifier.

The correlation and single-feature classifiers both assign high rank to the receiver dominant region and pressure predictors and to the reachable angle predictor, which suggests that these predictors are significant in isolation. However, these tests cannot capture the importance of the dependencies between predictors. The predictor importances assigned by the RUSBoost classifier and the predictors retained by the ℓ_1-regularized MLR classifier may provide some insight. In particular, the RUSBoost classifier rates all 10 predictors in the top 38. In the case of the MLR classifier, 5 of

Table 3.3: Dominant region-based predictor importance. These predictors were ranked based on their scores for: correlation with the label; misclassification rate on a single feature MLR classifier; and feature importance assigned using RUSBoost. The final column indicates the predictors with non-zero coefficients in the best-performing ℓ_1-regularized MLR classifier.

Feature	Correlation	1-Feature[a]	RUSBoost	ℓ_1-Regularized
Dominant Region - Passer	90	16	17	0
Dominant Region - Receiver	18	13	4	1
Dominant Region Net Change - Passer	110	16	14	1
Dominant Region Net Change - Receiver	27	108	38	1
Dominant Region - Team	60	16	23	0
Dominant Region Net Change - Team	30	108	24	0
Pressure - Passer	53	16	35	1
Pressure - Receiver	6	1	5	0
Pressure Net Change - Passer-Receiver	113	16	37	1
Reachable Angle	19	16	43	0

[a] Only the first 15 single-feature classifiers performed better than the "majority-class classifier", i.e. selecting the "OK" label for all test examples. 91 of the 114 learned single-feature classifiers were majority-class classifiers.

the 10 predictors are non-zero, from a total of 40 non-zero coefficients.

This analysis suggests that the dominant region-based predictors are important to the learned classifiers. Moreover, as discussed in Section 3.3, the purpose of these predictors is to capture some of the physiological and strategic aspects of the match state that would not otherwise be available to the classifier, and the fact that they are important to the classifier provides a level of assurance of the validity of the proposed model.

3.7.3 Inter-Rater Agreement

In Subsection 3.4.3 we discussed the fact that it is reasonable to expect that the observers would rate passes differently, and we applied Cohen's kappa [57] as a heuristic to evaluate the inter-rater agreement between the observers. Here we extend that analysis to the responses produced by the classifiers in the experiments. We computed Cohen's kappa in a pairwise manner on all labellings, see Table 3.4. The objective was to examine whether the inter-rater agreement between an observer and a classifier was significantly different from that between two observers.

The classifiers have a similar level of agreement with a given human observer to

Table 3.4: Cohen's Kappa showing inter-rater agreement between human observers and learned classifiers using the three-class labels. Note that the level of agreement for Annotator 2 is similar for all other raters.

	Annotator 1	Annotator 2	MLR	MLR-Arith	MLR-Quad	c-SVC	ϵ-SVR	RUSBoost	Ordered Logit
Annotator 1	-	0.697	0.595	0.602	0.591	0.579	0.559	0.598	0.570
Annotator 2	0.697	-	0.626	0.631	0.628	0.611	0.596	0.628	0.607
MLR	0.595	0.626	-	0.933	0.896	0.926	0.909	0.620	0.894
MLR-Arith	0.602	0.631	0.933	-	0.940	0.929	0.903	0.643	0.902
MLR-Quad	0.591	0.628	0.896	0.940	-	0.899	0.874	0.651	0.877
c-SVC	0.579	0.611	0.926	0.929	0.899	-	0.937	0.608	0.907
ϵ-SVR	0.559	0.596	0.909	0.903	0.874	0.937	-	0.585	0.888
RUSBoost	0.598	0.628	0.620	0.643	0.651	0.608	0.585	-	0.603
Ordered Logit	0.570	0.607	0.894	0.902	0.877	0.907	0.888	0.603	-

the agreement between the two observers. The inter-rater agreement between the best classifier and each of the two human observers is 0.602 and 0.631, which are comparable to the agreement between the observers of 0.697. The pass classification task is essentially subjective, and a level of disagreement between raters is to be expected. That the classifiers can produce a similar level of agreement with an observer suggests that the performance of the classifiers may be as good as can be expected for this task.

The agreement between the classifiers and Annotator 2 is consistently higher than with Annotator 1, which is interesting given that the voted labels used to train the classifiers were based on both observers. This suggests that Annotator 2's was more conservative in their labelling decisions, as the tie-breaking strategy used to selecting the voted label was to use the label closer to the middle of the Likert scale. A comparison of the agreement between the observers and the voted labels bears this out, as Annotator 2's label agrees with the voted label in $96.2\,\%$ of cases, whereas for Annotator 1's label is the same in $92.4\,\%$ of cases.

3.7.4 Limitations of Experimental Setup

A desirable property of the feature functions described in Section 3.3 is to capture sufficient information to train a classifier that has good predictive performance. However, this objective is constrained by several aspects of the experimental setup that was used. In the following section we provide an analysis of the constraints of the experimental

setup, what the potential impact is, and also whether any mitigation might be possible.

Sample Bias

The trajectory data available is limited to four matches. All of these matches are home matches for Arsenal Football Club. Arsenal were a strong team in the 2007/08 season, finishing third. They were unbeaten at home, playing 19 games, winning 14 and drawing 5. The opposition teams in the four matches were Aston Villa (finished 6th in season), Blackburn Rovers (7th), Bolton Wanderers (16th) and Reading (18th). Given that teams will often vary their tactics based on whether they are playing home or away, and also in terms of the relative strength of the opposition, there is the possibility of bias. This effect could be mitigated by repeating the experiments on a larger number of matches, involving a larger number of teams, and hosted at different stadia.

Source Data

The data used to produce the predictor variables is limited to trajectory data for the players and event sequence. The learning algorithm must train the classifier using only this information. However, the observers who labelled the training data may consider a number of other aspects when making their rating, for example the aesthetics of the pass, their prior belief about the player making the pass, or the apparent intensity of the current state of the match.

Video Framing

The labelling made by the observers was performed by viewing televised video footage of the match. This footage does not display the entire playing field, and thus the observer cannot take into account the state of players not in the video frame. The classifier, however, does not know what parts of the playing field were visible to the observer, and thus cannot discriminate based on this. This situation could be improved by obtaining wide-angle video footage of the match that included the entire playing-area in the picture frame.

Facing direction of players

The source data used to construct the predictor variables only provides the location of each player. The orientation of the player is not available, and thus must be extrapolated. Two plausible extrapolations were used in designing the feature functions: that the player is facing in the direction of motion; or that the player is facing the ball. However, there are clearly situations where a player may face a direction other than these two, particularly at low speeds, where a player may move backwards or diagonally from the direction they are facing.

Ball trajectory

The trajectory of the ball is not provided, and thus is extrapolated by using the event data to determine when a player touches the ball. In between such events, the location and speed of the ball is interpolated using a simple linear model. This is clearly an approximation, as the ball may not travel in a straight line, for example if it is kicked in the air. Moreover, the velocity of the ball will not remain constant between events as is the case in the model.

3.8 Conclusion

In this chapter we present a model that is able to learn a classifier to rate the quality of passes made during a football match with an accuracy of up to $90.2\,\%$. We compared the ratings made by the classifier with those made by human observers, and found that the level of agreement between the machine classifier and an observer is similar in magnitude to the level of agreement between two observers.

The model uses feature functions based on algorithms and data structures from computational geometry, in particular the dominant region [183]. This structure is intended to provide information about the strategic and physiological state of the match, however it is costly to compute. We evaluate the importance to the classifier of the predictor variables based on the dominant region, and find them to be important to the

classifiers, suggesting that the cost of computation is worthwhile in this case.

The experimental results suggest that the model described in this chapter is effective in solving the problem of automatically classifying passes made in football matches. The framework and experiments also provide a foundation for further investigation, and several areas became apparent during this research.

The problem investigated was to assign a single rating to each pass, however it is apparent that the overall quality of a pass convolves several factors. These factors include how well the player executed the pass, the difficulty of executing the pass given the situation, the riskiness of the pass, and the strategic value of the pass. Rating each pass according to several distinct, but possibly dependent, criteria is a natural extension from the pass rating problem, and may provide deeper insights into determining the quality of passes.

Similarly, the model currently rates passes that have been made. An interesting question would be whether it is possible to rate passes that were not made, and from this be able to gain an insight into whether the player in possession made the best passing decision, given the conditions under which the pass was made.

There already exists some work in this direction. The problem of determining what players are available to receive a pass at a given point in time has been investigated by Gudmundsson and Wolle [96]. Furthermore, research by Yue *et al.* [198] attempts to construct a model to predict whether a basketball player should shoot, pass or retain possession in a given situation.

Another interesting extension to this model would be to consider sequences of passes during a possession, and attempt to rate the passes in the sequence jointly. Intuitively, the quality of the previous passes made in a possession sequence will have some impact on current pass, and thus this information could be exploited to improve the overall quality of the ratings assigned.

Chapter 4

Summarising State Sequences with Flow Diagrams

The player trajectories and event logs described in Section 1.2 are examples of data that track movement or activity as a sequence of states. Similar datasets are captured using object tracking sensors in other application domains such as traffic analysis [39] and behavioural ecology [5]. A natural question to ask is how can large sets of activity sequences be represented in a model that is both compact and reveals the underlying structure within the activities. That is, a model that summarizes the activity sequences in such a way that it can be easily interpreted by domain experts. To this end, we introduce the concept of representing the "flow" of activities in a compact way using a flow diagram, and we argue that this model is helpful to detect and visualize patterns in large sets of state sequences.

To make this idea concrete, consider the following simple example that tracks three people and their activities during a single day. The activities of each person over a day are modelled as a state sequence, and the set of state sequences $\mathcal{T} = \{\tau_1, \tau_2, \tau_3\}$ are shown in Figure 4.1(a). As input we are also given a set of criteria $\mathcal{C} = \{C_1, \ldots, C_k\}$, in Figure 4.1(b). Each criterion is a Boolean function on a single subsequence—or a set of subsequences—of states. In the example, the criterion $C_1 =$ "eating" is true for Person 1 at time intervals 7–8am and 7–9pm, but false for all other time intervals. If a

criterion C is true for a set of subsequences, we say the subsequences *fulfil* C. Thus, a set of criterion $\mathcal{C}$ partitions a state sequence into subsequences, called *segments*, where each segment fulfils a criterion $C \in \mathcal{C}$. A *segmentation* of $\mathcal{T}$ according to $\mathcal{C}$ is a partition of each sequence in $\mathcal{T}$ into segments, each fulfilled by a particular criterion in $\mathcal{C}$. A segmentation can thus be represented as the corresponding sequence of criteria. To continue the above example, the segments of $\mathcal{T}$ according to the set $\mathcal{C}$ are shown in Figure 4.1(c).

The aim is to efficiently summarize the segmentations of all sequences in a compact representation; that is, to build a flow diagram $\mathcal{F}$, that has a small number of nodes, including a start state s and end state t, such that for each state sequence τ_i, $1 \leq i \leq m$, there exists a segmentation according to $\mathcal{C}$ which appears as an s–t path in $\mathcal{F}$. One possible flow diagram for the example input $(\mathcal{T}, \mathcal{C})$ is shown in Figure 4.1(d), and from this flow diagram one can observe that the three people either eat (C_1) or exercise (C_3) before commuting (C_2), followed by working or studying (C_4), and finally they eat (C_1) and possibly shop (C_6). Each interval in which a person carries out an activity is a segment of the overall segmentation. We believe that the obtained flow network is considerably easier to interpret than the input. The flow diagram for $\mathcal{T}$ according to $\mathcal{C}$ can be validated by going through a segmentation of each object while following a path in $\mathcal{F}$ from s to t. For example, for Person 1 the s–t path $s \rightarrow C_1 \rightarrow C_2 \rightarrow C_4 \rightarrow C_1 \rightarrow t$ is a valid segmentation, and is marked in red in Figure 4.1(d).

We are not aware of any (efficient) approach to extract flow patterns in large sets of state sequences. Clustering would be a natural alternative but we do not know of any simple and useful way to cluster state sequences. One possible approach would be to simply cluster each individual state in the set of sequences, but then one would lose important information linked to the fact that we are dealing with sequences. Our approach combines both the sequential nature of the data and the aggregation based on similar parts, and we believe this is a natural "sparse" visualisation of the output that can be interpreted by a domain expert and that it highlights frequent patterns that are hard to extract computationally. Even the small example above shows that there can be

States	Person 1	Person 2	Person 3
07:00–08:00	breakfast	gym	breakfast
08:00–09:00	cycle	drive	cycle
09:00–17:00	work	work	work
17:00–19:00	study	dinner	shop
19:00–21:00	dinner	shop	dinner

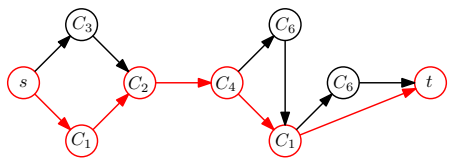
(a)

Criteria		Satisfying States
C_1	Eating	breakfast, dinner
C_2	Commuting	cycle, drive
C_3	Exercising	gym, cycle
C_4	Working or studying	working, studying
C_5	Working for at least 4 hours	working, studying
C_6	Shopping	shopping
C_7	At least two people eating simultaneously	eating

(b)

States	Person 1	Person 2	Person 3
07:00–08:00	$[C_1, C_7]$	$[C_3]$	$[C_1, C_7]$
08:00–09:00	$[C_2, C_3]$	$[C_2]$	$[C_2, C_3]$
09:00–17:00	$[C_4, C_5]$	$[C_4, C_5]$	$[C_4, C_5]$
17:00–19:00	$[C_4]$	$[C_1]$	$[C_6]$
19:00–21:00	$[C_1, C_7]$	$[C_6]$	$[C_1, C_7]$

(c)



(d)

Figure 4.1: The input is (a) a set $\mathcal{T} = \{\tau_1, \ldots, \tau_m\}$ of sequences of states and (b) a set of criteria $\mathcal{C} = \{C_1, \ldots, , C_k\}$. (c) The criteria partition the states into a segmentation. (d) One possible flow diagram representing a valid segmentation of $\mathcal{T}$ according to $\mathcal{C}$, with the s–t path for Person 1 (highlighted in red).

considerable space savings by representing a set of state sequences as a flow diagram. It is important to note that is not a lossless representation and therefore comes at a cost. The flow diagram contains a valid segmentation for each input state sequence, however, the particular segmentation corresponding to an individual state sequence is not apparent. Furthermore, a state sequence may have many valid segmentations, and only one is guaranteed to be represented as an s–t path in the flow diagram. On the other hand, not all s–t paths in $\mathcal{F}$ necessarily represent a valid segmentation of one of the state sequences in $\mathcal{T}$. However, as we will argue in Section 4.5, paths representing many segments in the obtained flow diagrams show interesting patterns, and we give two examples. First we consider segmenting the morphology of formations of a defensive line of football players during a match, and find that the obtained flow diagram provides an intuitive summary of these formations. The second example uses state sequences to model the attacking actions of the team in control of the ball during

a football match. The summary given by the induced flow diagram on the state sequences highlights the differences in attacking tactics between teams playing at home and teams playing away from home.

The remainder of this chapter is organized as follows. Section 4.2 formally defines the problem and states the properties of the criteria that we use to improve the runtime of the algorithms. Theorems proving the hardness of the problem are stated in Section 4.3. In Section 4.4 we present algorithms for the FLOW DIAGRAM problem using criteria with the properties described above. Moreover, to obtain flow diagrams for larger groups of state sequences we propose two heuristics for the problem. In Section 4.5, the exact and heuristic algorithms are evaluated experimentally in order to determine their practical performance characteristics, and also to assess the utility of the output flow diagrams.

4.1 Related work

To the best of our knowledge the concept of compactly representing sequences of states as flow diagrams has not been considered before. The only related work of which we are aware comes from the area of trajectory analysis, and from machine learning using Markov models. Spatial trajectories are a special class of state sequences where a trajectory describes the movement of an object through space over time and the states are the location points. Furthermore, the states may also include additional information such as direction, speed, and temperature.

For a single trajectory a common way to obtain a compact representation is *simplification* [38]. Trajectory simplification aims to determine a subset of the location points that represents the trajectory well by minimising the maximum distance between the input and simplified trajectories. If the focus is on characteristics other than the location, then *segmentation* [5], [8], [29] may be used to partition a trajectory into a small number of sub-trajectories, where each subtrajectory is homogeneous with respect to some given characteristic. This allows a trajectory to be compactly represented as a

sequence of characteristics.

For multiple trajectories distinct techniques apply. A large set of trajectories may contain groups of similar trajectories where trajectories from different groups are dissimilar, and hence *clustering* may be used. Clustering on complete trajectories may not uncover information about interesting parts of the trajectories; so clustering on sub-trajectories may be required [35], [102]. A set of trajectories that forms different groups over time may be captured by a *grouping structure* [30], and approaches such as this also focus on location over time.

For the special case of spatial trajectories, a flow diagram can be illustrated by a simple example: trajectories of migrating geese, see [36]. The individual trajectories can be segmented into phases of activities such as directed flight, foraging and stop overs. This results in a flow diagram containing a path for the segmentation of each trajectory. More complex criteria can be imagined that depend on a group of geese, or frequent visits to the same area, resulting in complex state sequences that are impractical to analyze without computational tools.

Segmenting a single sequence can also be viewed from a stochastic learning perspective as a *structured prediction* problem. For each sequence state, a distribution is modelled to assign a probability that the state fulfils each criteria. The problem can thus be represented as a *hidden Markov model*, see [185] for a detailed discussion. Typically, the objective of such models is to maximise the likelihood of an assignment of criteria to the state sequence, and thus does not attempt to find an assignment that is minimal in the number of segments.

Bonchi *et al.* [20] proposes a Markov model that uses a hierarchical taxonomy of criteria that uses the concept of *lumpability*, where sequences of states can be aggregated within the taxonomy to produce a partition of the state sequence that is compact and also maintains the characteristics of the input sequence. A number of algorithms are defined, in particular one that minimises the complexity of the partition, which is analogous to minimising the number of segments.

4.2 Preliminaries

4.2.1 Problem Definition

A *flow diagram* is a node-labelled directed acyclic graph (DAG) with a distinguished source node s and sink node t, and where all other nodes are labelled with a criterion. Given a set $\mathcal{T}$ of state sequences and a set $\mathcal{C}$ of criteria, the goal is to construct a flow diagram with a minimum number of nodes, such that a valid segmentation of each sequence of states in $\mathcal{T}$ is represented—that is, included as an s–t path—in the flow diagram. Furthermore (when criteria depend on multiple state sequences, e.g. C_7 in Figure 4.1) we require that the segmentations represented in the flow diagram are consistent, i.e. the segmentations can be jointly realized. The FLOW DIAGRAM problem thus requires the segmentations of each sequence of states and the minimal flow diagram of the segmentations to be computed, and can be formally stated as:

PROBLEM FLOW DIAGRAM (FD)

Instance: A set of sequences of states $\mathcal{T} = \{\tau_1, \ldots, \tau_m\}$, each of length at most n, a set of criteria $\mathcal{C} = \{C_1, \ldots, C_k\}$ and an integer $\lambda > 2$.

Question: Is there a flow diagram $\mathcal{F}$ with $\leq \lambda$ nodes, such that for each $\tau_i \in \mathcal{T}$, there exists a segmentation according to $\mathcal{C}$ which appears as an s–t path in $\mathcal{F}$?

4.2.2 Properties of Criteria

The efficiency of the algorithms depend on properties of the criteria on which the segmentations are based. Here we consider three cases:

- general criteria without restrictions,
- monotone decreasing and independent criteria, and
- monotone decreasing and dependent criteria.

These properties are illustrated using the example in Figure 4.1.

A criterion C is *monotone decreasing* [29] for a given sequence of states τ that fulfil C, if all subsequences of τ also fulfil C. For example, if criterion C_4—working and studying—is fulfilled by a sequence τ then any subsequence τ' of τ will also fulfil C_4. In contrast, criterion C_5—working for at least 4 hours—is not monotone decreasing.

A criterion C is *independent* if verifying whether a subsequence τ' of a sequence $\tau_i \in \mathcal{T}$ fulfils C can be achieved without reference to other sequences $\tau_j \in \mathcal{T}, i \neq j$. Conversely, C is *dependent* if verifying that a subsequence τ' of τ_i requires reference to other state sequences in $\mathcal{T}$. In the above example C_4—working or studying—is an example of an independent criterion while C_7—at least two people eating simultaneously—is a dependent criterion since it requires that at least two objects to satisfy the criterion simultaneously.

4.3 Hardness Results

In this section, the hardness and inapproximation results are stated, and the necessary proofs and reductions provided.

Theorem 4.1. *The FD problem is NP-hard. This even holds when only two criteria are used or when the length of every state sequence is* 2*. For any* $0 < c < 1/4$*, the FD problem cannot be approximated within factor of* $c \log m$ *in polynomial time unless* $NP \subset DTIME(m^{\text{polylog}\, m})$.

Furthermore, for bounded number of state sequences m the running times of our algorithms are somewhat high, and we can show that there are good reasons for this. The algorithms will only run in polynomial time if m is constant, and this is essentially the best we can hope for as the problem is $W[1]$-hard. Unless $W[1] = FPT$, this rules out the existence of algorithms with time complexity of $O(f(m) \cdot (nk)^c)$ for some constant c and any computable function $f(m)$, where m, n and k are the number of state sequences, the length of the state sequences and the number of criteria, respectively.

Theorem 4.2. *The FD problem parameterized in the number of state sequences is* $W[1]$*-hard even when the number of criteria is constant.*

To obtain the stated results we will perform two reductions: from the SHORTEST COMMON SUPERSEQUENCE problem; and from the SET COVER problem.

4.3.1 Reduction from SHORTEST COMMON SUPERSEQUENCE

PROBLEM SHORTEST COMMON SUPERSEQUENCE (SCS)

Instance: A set of strings $R = \{r_1, r_2, \ldots, r_k\}$ over an alphabet Σ, a positive integer λ.

Question: Does there exist a string $s \subset \Sigma^*$ of length at most λ, that is a supersequence of each string in R?

The SCS problem has been extensively studied over the last 30 years (see [77] and references therein). Several hardness results are known, we will use the following two.

Lemma 4.3 (Pietrzak [163])**.** *The SCS problem parameterized in the number of strings is $W[1]$-hard even when the alphabet has constant size.*

Lemma 4.4 (Räihä and Ukkonen [166])**.** *The SCS problem over a binary alphabet is NP-complete.*

The SCS problem can be reduced to the FD problem as follows. Given an instance $I = (R = \{r_1, \ldots, r_m\}, \Sigma = \{c_1, \ldots, c_k\})$ of SCS construct an instance of FD as follows. Each character c_l corresponds to a criterion C_l. Each string r_i corresponds to a state sequence T_i, where $T_i[j] = c_{r_i[j]}$. Thus, each state of T_i fulfils exactly one criterion C_j.

An algorithm for the FD problem will, given an instance, output a flow diagram $\mathcal{F}$ of size ℓ. Given $\mathcal{F}$, one can compute a topologically sorted linear sequence b of vertices in $\mathcal{F}$, as shown in Figure 4.2(a). The linear sequence b has $\ell - 2$ vertices (omitting the start and end state of $\mathcal{F}$) and it is a supersequence of each string in R. It follows that the size of $\mathcal{F}$ is λ if the number of characters in the SCS of I has length $\lambda - 2$. Note that $\mathcal{F}$ contains a linear sequence of vertices (after topological sort) that correspond to a supersequence and a set of directed edges. Consequently, a solution

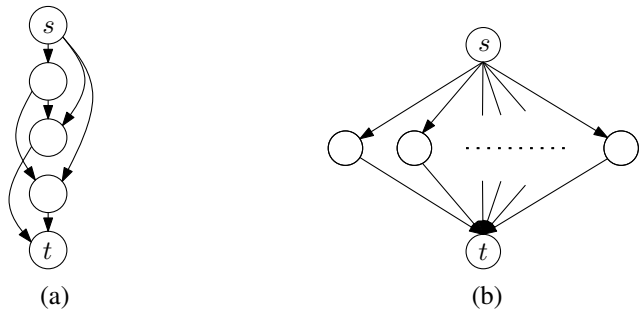


Figure 4.2: Examples of flow diagrams produced by the reductions: (a) From SHORTEST COMMON SUPERSEQUENCE. (b) From SET COVER.

for the FD problem can easily be transformed to a solution for the SCS problem but not vice versa.

From the above reduction, together with Lemmas 4.3-4.4, we obtain Theorem 4.2 and the following lemma.

Lemma 4.5. *The FD problem is NP-hard even for two criteria.*

4.3.2 Reduction from SET COVER

PROBLEM SET COVER (SC)

Instance: A set of elements $E = \{e_1, e_2, \ldots, e_m\}$, a set of n subsets of E, $S = \{S_1, S_2, \ldots, S_n\}$, and a positive integer λ.

Question: Does there exist set of λ items in S whose union equals E?

SET COVER is well known to be NP-hard, and also hard to approximate:

Lemma 4.6 (Lund and Yannakakis [145])**.** *For any $0 < c < 1/4$, the SC problem cannot be approximated within factor of $c \log m$ in polynomial time unless $NP \subset DTIME(m^{polylogm})$.*

We prove that the FD problem is equivalent to the SC problem using the following reduction. Given an instance $I = (E = \{e_1, e_2, \ldots, e_m\}, S = \{S_1, S_2, \ldots, S_n\})$ of SC, an instance of FD can be constructed using the following steps. Each item e_i corresponds to a state sequence T_i of length one. Each subset S_j corresponds to a criterion C_j. If a S_j contains e_i then the whole state sequence T_i fulfils criterion C_j.

An algorithm for FD given the constructed instance will output a flow diagram $\mathcal{F}$ of size ℓ, depicted in Figure 4.2(b). Given $\mathcal{F}$ the interior vertices of $\mathcal{F}$ corresponds to a set of subsets in S whose union is E. The diagram $\mathcal{F}$ has ℓ vertices if and only there is $\ell - 2$ subsets in S that forms a set covering of E.

From the above reduction, we obtain Theorem 4.1 and Lemmas 4.5 and 4.6.

4.4 Algorithms

In this section, we present algorithms that compute a flow diagram of minimal vertex size, representing a set of m state sequences of length n for a set of k criteria. First, we present an algorithm for the general case, followed by more efficient algorithms for the case of monotone increasing and independent criteria, the case of monotone increasing and dependent criteria, and finally two heuristic algorithms.

4.4.1 General criteria

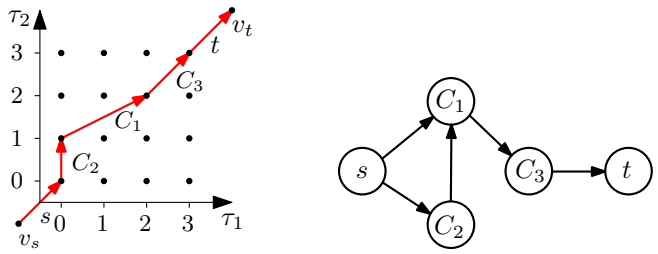
The first algorithm is for general criteria, where we do not seek to exploit any of the properties described in Subsection 4.2.2. Recall that an interior node v in the flow diagram represents a criterion C_j that is fulfilled by a contiguous segment in one or more of the state sequences. Let $\tau[i, j]$, $i \leq j$, denote the subsequence of τ beginning at the $i+1$-th state of τ and ending at the j-th state, where $\tau[i, i]$ is an empty sequence.

The *prefix graph* G is constructed as follows. G contains an $(n+1)^m$ grid of vertices, where a vertex with coordinates $(x_1, \ldots, x_m)$, $0 \leq x_1, \ldots, x_m \leq n$, represents the set $(\tau_1[0, x_1], \ldots, \tau_m[0, x_m])$ of m prefix subsequences of $\mathcal{T}$. An edge exists between two vertices $v = (x_1, \ldots, x_m)$ and $v' = (x'_1, \ldots, x'_m)$, labeled by some criterion C_j, if and only if, for every i, $1 \leq i \leq m$, one of the following two conditions is fulfilled:

- one or more $\tau_i[x_i + 1, x'_i]$ jointly fulfil C_j, or
- $x_i = x'_i$.

	τ_1	τ_2
1	$[C_1]$	$[C_2]$
2	$[C_1]$	$[C_1, C_2]$
3	$[C_3]$	$[C_3]$

(a)



(b) (c)

Figure 4.3: (a) A segmentation of $\mathcal{T} = \{\tau_1, \tau_2\}$ according to $\mathcal{C} = \{C_1, C_2, C_3\}$. (b) The prefix graph G of the segmentation, showing only five edges that comprise a shortest path in G. (c) The resulting flow diagram generated from the highlighted path in the prefix graph.

As an example, consider the edge between $(x_1, x_2) = (0, 0)$ and $(x'_1, x'_2) = (0, 1)$ in Figure 4.3(b). Here $x_1 = x'_1$ and $\tau_2[x_2 + 1, x'_2]$ fulfils C_2.

Finally, define two additional vertices and edges in G, that are "outside" of the grid. Vertex $v_s = (-1, \ldots, -1)$ with an outgoing edge to $(0, \ldots, 0)$ labelled with the "criterion" s; and $v_t = (n+1, \ldots, n+1)$, which has an incoming edge labelled t from $(n, \ldots, n)$. This completes the construction of the prefix graph G.

Now, a path in G from v_s to a vertex v represents a valid segmentation of some prefix of each state sequence, and induces a (partial) flow diagram that describes these segmentations in the following way:

- every edge of the v_s–v path induces a node in the flow diagram, labeled by the criterion that the corresponding segments fulfil, i.e. the label of the edge in G.
- directed edges terminating at the new node are added to the flow diagram originating at each node that represents a criterion fulfilled by at least one segment in the segmentation that immediately precedes the segments corresponding to the new node.

The directed edges terminating at a given target node can be directly read from the prefix graph. Consider the case where a node $u \neq s$ is added to the flow diagram, and is either labelled with a criterion C_j or is the sink node t. The node u corresponds to an edge (v, v') in the prefix graph G, labelled with the same criterion C_j or is the

distinguished edge labelled t. Edges must be added to $\mathcal{F}$ connecting each node representing a criterion that precedes C_j in the segmentation. The edges are manifest in G as the sub-path of the v_s–v path starting at $v'' = (x''_1, \ldots, x''_m)$ such that $\tau_i[x''_i + 1, x_i]$ is fulfilled by a single segment in the segmentation for all $i = 1, \ldots m$.

A geometric interpretation of this construction is that the v_s–v path in G is backtracked until the vertex v'' is reached, such that the line-segment $\overline{v''v}$ is not axis-parallel in any of the m dimensions. Then, the set of edges in the v''–v path in G corresponds to a set of nodes in $\mathcal{F}$ and a directed edge from each of these nodes to u is added into $\mathcal{F}$. For example, in Figure 4.3, the node u labelled C_1 in $\mathcal{F}$ has two incoming edges from nodes labelled s and C_2 respectively. The nodes from where these edges originated were identified in G by backtracking along the highlighted path to a vertex that describes a line segment with $(1, 0)$ that is not axis parallel, in this case v_s. For each edge in G traversed in the backtracking, an edge is added into $\mathcal{F}$ from the corresponding node to the target u.

This construction ensures that the flow diagram represents a valid segmentation and that each node represents at least one segment. Clearly, the length of a path (counted as the number of edges) in a prefix graph G equals the number of nodes of the corresponding flow diagram $\mathcal{F}$. Thus, we are able to compute an optimal flow diagram $\mathcal{F}$ by finding a shortest v_s–v_t path in G.

Lemma 4.7. *A smallest flow diagram for a given set of state sequences is represented by a shortest v_s–v_t path in G.*

Proof. We show that every v_s–v_t path P in G represents a valid flow diagram $\mathcal{F}$, with the path length equal to the flow diagram's cost, and vice versa. Thus, a shortest path induces a minimal valid flow diagram for the given state sequences.

Let $P := (v_s =: v_0, v_1, \ldots, v_\ell := v_t)$ be a v_s–v_t path of length ℓ in G. As described in the prequel, every v_s–v_t path in G represents a valid flow diagram, and every vertex visited by the path contributes exactly one node to the flow diagram. Thus, P represents a valid flow diagram with exactly ℓ nodes.

For the other direction, let $\mathcal{F}$ be a valid flow diagram of a set of state sequences $\{T_1, \ldots, T_m\}$, each of length n. That is, there exist segmentations $\{\mathcal{S}_1, \ldots, \mathcal{S}_m\}$ of the state sequences such that each segmentation is represented in $\mathcal{F}$ in the following way: assume the nodes of $\mathcal{F}$ are $\{s =: f_0, f_1, \ldots, f_\ell := t\}$ according to some topological ordering. Let $\mathcal{S}_j$ consist of the segments $s_{j,1}, \ldots, s_{j,\sigma_j}$, where σ_j is the number of segments in $\mathcal{S}_j$. Then there exists a path $(s, f_{j,1}, \ldots, f_{j,\sigma_j}, t)$ in $\mathcal{F}$ such that each segment $s_{j,i}$ fulfils the criterion $C(f_{j,i})$ associated with $f_{j,i}$.

Let $b_{j,i}$ be the index in T_j at which $s_{j,i}$ ends, for $1 \leq i \leq \sigma_j$, and let $b_{j,0} := 0$. Since $\mathcal{S}_j$ is a segmentation of T_j, then $b_{j,\sigma_j} = n$.

Let $\mathcal{F}_\lambda$ be the subdiagram of $\mathcal{F}$ induced by $(f_1, \ldots, f_\lambda)$, for $1 \leq \lambda \leq \ell$. We define $x_{j,\lambda} := \max\{b_{j,i} \mid f_{j,i} \in \{f_1, \ldots, f_\lambda\}\}$. We show inductively that for each $\lambda \in \{1, 2, \ldots, \ell\}$, G contains a path from v_s to the vertex $v_\lambda := (x_{1,\lambda}, \ldots, x_{m,\lambda})$ with length λ, i.e. the number of nodes in $\mathcal{F}_\lambda$.

Base case Note that $v_0 = v_s$ and v_s has a single outgoing edge—by construction—to v_1, and thus there is a path of length $\lambda = 1$ from v_s to v_1.

Induction step The node $f_{\lambda+1}$ represents the segments

$$\{T_j[x_{j,\lambda} + 1, x_{j,\lambda+1}] \mid 1 \leq j \leq m \wedge x_{j,\lambda} \neq x_{j,\lambda+1}\}.$$

Since the flow diagram is valid, these segments fulfil the criterion $C(f_{\lambda+1})$, and thus G contains an edge from v_λ to $v_{\lambda+1}$. Since a path from v_s to v_λ of length λ exists by the induction hypothesis, there is a path from v_s to $v_{\lambda+1}$ of length $\lambda+1$.

For every state sequence T_j, there exists an index $\varphi_j \in \{1, \ldots, \ell - 1\}$ such that $x_{j,\lambda} = n$ for all $\lambda \geq \varphi_j$. Thus, $v_{\ell-1} = (n, n, \ldots, n)$ and G contains an edge from $v_{\ell-1}$ to $v_\ell = v_t$. So, there is a path from v_s to v_t of length ℓ. □

Furthermore, it is often the case, and indeed likely, that there are multiple shortest s–t paths within G. This suggests that different strategies can be used for choosing the shortest path to use to compute the flow diagram, and moreover, the choice of strategy

may impact the properties of the flow diagram. The shortest path may be chosen arbitrarily, or using an objective function—for example, the path that maximises the minimum number of sequences that is advanced by each edge in the path. An analysis of the strategies for selecting the shortest path is provided in Subsection 4.5.3.

We now consider the complexity of the algorithm. Recall that G has $(n+1)^m$ vertices. Each vertex has $O(k(n+1)^m)$ outgoing edges, thus, G has $O(k(n+1)^{2m})$ edges in total. To decide if an edge is present in G, check if the nonempty segments the edge represents fulfil the criterion. Thus, we need to perform $O(k(n+1)^{2m})$ of these checks. There are m segments of length at most n, and we assume the cost for checking this is $T(m,n)$. Thus, the cost of constructing G is $O(k(n+1)^{2m} \cdot T(m,n))$, and finding the shortest path requires $O(k(n+1)^{2m})$ time, yielding the following theorem.

Theorem 4.8. *The algorithm described above computes a smallest flow diagram for a set of m state sequences, each of length at most n, and k criteria in $O((n+1)^{2m}k \cdot T(m,n))$ time, where $T(m,n)$ is the time required to check if a set of m subsequences of length at most n fulfils a criterion.*

4.4.2 Monotone decreasing and independent criteria

If all criteria are monotone decreasing and independent, we can use ideas similar to those presented in [29] to avoid constructing the full graph. From a given vertex with coordinates $(x_1, \ldots, x_m)$, and given a criterion C_j, we can greedily move as far as possible along the sequences while C_j is fulfilled, since the monotonicity guarantees that this never leads to a solution that is worse than one that represents shorter segments. For a given criterion C_j, we can compute for each τ_i independently the maximum x'_i such that $\tau_i[x_i+1, x'_i]$ fulfils C_j. This results in coordinates $(x'_1, \ldots, x'_m)$ for another vertex, which is the optimal next vertex using C_j. By considering all criteria we obtain k new vertices. However, unlike the case with a single state sequence, there is not necessarily one vertex that is better than all others (i.e. furthest ending position), since there is no total ordering on the vertices. Instead, we must consider all vertices that are not *dominated* by another vertex, where a vertex p dominates a vertex p' if each

coordinate of p is at least as large as the corresponding coordinate of p', and at least one of p's coordinates is larger.

Let V_i be the set of vertices of G that are reachable from v_s in exactly i steps, and define $M(V) := \{v \in V \mid \text{no vertex } u \in V \text{ dominates } v\}$ to be the set of *maximal vertices* of a vertex set V. Then a shortest v_s–v_t path through G can be computed by iteratively computing $M(V_i)$ for increasing i, until a value of i is found for which $v_t \in M(V_i)$. Observe that $|M(V)| = O((n+1)^{m-1})$ for any set V of vertices in the graph. Also note that $V_0 = M(V_0) = v_s$.

Lemma 4.9. *For each $i \in \{1, \ldots, \ell\}$, every vertex in $M(V_i)$ is reachable in one step from a vertex in $M(V_{i-1})$, where ℓ is the length of the v_s–v_t path.*

Proof. Assume there exists a vertex $v \in M(V_i)$ that has no edge from a vertex in $M(V_{i-1})$. Since $v \in M(V_i)$, v is also contained in V_i, and thus its distance from v_s is i. Thus, there must be a vertex v' at distance $i-1$ from v_s, i.e. $v' \in V_{i-1}$, that has an edge to v representing a criterion C_j. By assumption, v' is not contained in $M(V_{i-1})$, and thus there is a vertex $v'' \in M(V_{i-1})$ that dominates v'. But then, by the monotonicity of C_j, there must be a vertex reachable from v'' that is identical to v or dominates v. Both cases lead to a contradiction. □

$M(V_i)$ is computed by computing the farthest reachable vertex for each $v \in M(V_{i-1})$ and each criterion C_j, thus yielding a set D_i of $O(k(n+1)^{m-1})$ vertices. This set contains $M(V_i)$ by Lemma 4.9, so we now need to remove all vertices that are dominated by some other vertex in the set to obtain $M(V_i)$.

We find $M(V_i)$ using a copy of G. Each vertex may be marked as being in D_i or dominated by a vertex in D_i. We process the vertices of D_i in arbitrary order. For a vertex v, if it is not yet marked as either being in D_i or dominated by a vertex in D_i, we mark it as being in D_i. When a vertex is newly marked, we mark its $\leq m$ immediate neighbours dominated by it as being dominated. After processing all vertices, the grid is scanned for the vertices still marked as being in D_i. These vertices are exactly $M(V_i)$.

When computing $M(V_i)$, $O(k(n+1)^{m-1})$ vertices need to be considered, and the maximum distance from v_s to v_t is $m(n+1)$, so the algorithm considers $O(mk(n+1)^m)$ vertices. We improve this bound by a factor m using the following:

Lemma 4.10. *The total size of D_i for all $0 \leq i \leq \ell - 1$, is $O(k(n+1)^m)$.*

Proof. If a vertex v appears in $M(V_i)$ for some $i \in \{0, \ldots, \ell-1\}$, it generates vertices for D_{i+1} that dominate v, and thus $v \notin M(V_{i+j})$ for any $j > 0$. So, each of the n^m vertices appears in at most one $M(V_i)$ and generates k candidate vertices for D_{i+1} (not all unique). Hence the total size of all D_i is $O(kn^m)$. □

Using this result, we compute all $M(V_i)$ in $O((k+m)(n+1)^m)$ time, since $O(k(n+1)^m)$ vertices are marked directly, and each of the $(n+1)^m$ vertices is checked at most m times when a direct successor is marked. One copy of the grid can be reused for each $M(V_i)$, since each vertex of D_{i+1} dominates at least one vertex of $M(V_i)$ and is thus not yet marked while processing D_j for any $j \leq i$.

Since the criteria are independent, the farthest reachable point for a given starting point and criterion can be precomputed for each state sequence separately. Using the monotonicity we can traverse each state sequence once per criterion and thus need to test only $O(nmk)$ times whether a subsequence fulfils a criterion.

Theorem 4.11. *The algorithm described in this section computes a smallest flow diagram for m state sequences of length n with k independent and monotone decreasing criteria in $O(mnk \cdot T(1,n) + (k+m)(n+1)^m)$ time, where $T(1,n)$ is the time required to check if a subsequence of length at most n fulfils a criterion.*

4.4.3 Monotone decreasing and dependent criteria

For monotone decreasing and dependent criteria, we can use a similar approach to that described above, however, for a given start vertex v and criterion C, there is not a single vertex v' that dominates all vertices reachable from v using this criterion. Instead there may be $\Theta((n+1)^{m-1})$ maximal reachable vertices from v for criterion C. The

maximal vertices can be found by testing $O((n+1)^{m-1})$ vertices on or near the upper envelope of the reachable vertices in $O((n+1)^{m-1} \cdot T(m,n))$ time. Using a similar reasoning as in Lemma 4.10, we can show that the total size of all D_i ($0 \leq i \leq \ell - 1$) is $O(k(n+1)^{2m-1})$, which gives:

Theorem 4.12. *The algorithm from the previous section computes a smallest flow diagram for m state sequences of length n with k monotone decreasing criteria in $O(k(n+1)^{2m-1} \cdot T(m,n) + m(n+1)^m)$ time, where $T(m,n)$ is the time required to check if a set of m subsequences of length at most n fulfils a criterion.*

4.4.4 Heuristics

The hardness results presented in Section 4.3 indicate that it is unlikely that the performance of the algorithms will be acceptable in practical situations, other than for very small inputs. As such, we investigated heuristics that may produce usable results and can be computed in reasonable time.

We considered two heuristic algorithms. These are based on the observation that by limiting the number of outgoing edges in the prefix graph from the vertices in V_i—i.e. the set of vertices reachable from v_s in i steps—to a fixed size then the complexity of the algorithm can be controlled. Given that every s–t path in a prefix graph represents a valid segmentation of $\mathcal{T}$, any path chosen in the prefix graph will induce a valid, though not necessarily optimal flow diagram.

For some vertex $v \in G$, let $E = \{(v, v') \mid v \in V_{i-1}\}$ be the candidate outgoing edges each satisfying some $C \in \mathcal{C}$. Each such edge $(v, v') \in E, v = (x_1, \ldots, x_m)$ represents a set of subsequences fulfilling C in a candidate group segmentation of $\mathcal{T}$:

$$\mathcal{S}_{v,v'} = \{T_j[x_j + 1, x'_j] \mid 1 \leq j \leq m \wedge x_j \neq x'_j\}.$$

This implies two obvious metrics for the "value" of the edge (v, v'). First, $|S_{v,v'}|$ is the number of state sequences whose prefixes are advanced by adding (v, v') into G. Similarly, the total number of states that are included in the corresponding prefixes by

adding (v, v') to G can be computed by:

$$f(\mathcal{S}_{v,v'}) := \sum_{T_j[x_j+1,x'_j] \in \mathcal{S}_{v,v'}} x'_j - x_j - 1$$

Given $v \in V_{i-1}$, we consider two strategies for selecting the edges in (v, v') to add into G and thus determine $V_i \ni v'$:

Sequence heuristic For each candidate edge (v, v'), compute the number of state sequences that the corresponding segment advances, i.e. $|\mathcal{S}_{v,v'}|$, and select the q edges yielding the largest sets of state sequences.

State heuristic For each candidate edge (v, v') determine the number of states that are advanced by computing $f(\mathcal{S}_{v,v'})$, and select the q edges yielding the largest such values.

In our experiments we use $q = 1$ since any larger value would immediately give an exponential worst-case running time. Using $q = 1$, the worst case is for each edge in the shortest path in G to advance a single state in a single state sequence, and at each vertex along the path in G all k criteria must be evaluated to decide on the next edge, so k operations must be performed at a cost of $T(m, n)$. This implies a running time of $O(kmn \cdot T(m, n))$.

4.5 Experiments

The objectives of the experiments were twofold: to empirically investigate the performance of the algorithms on inputs of varying sizes and structure; and to determine whether compact and useful flow diagrams could be produced in real application scenarios. We implemented the algorithms for general criteria, monotone decreasing and independent criteria, state and sequence heuristics, described in Section 4.4, using the Python programming language.

For the first objective, the algorithms were run on generated data sets of varying sizes to investigate the impact of different parameterisations on the computation time required to produce the flow diagram and the complexity of the induced flow diagram. Furthermore, we were interested in whether flow diagrams could recover the underlying structure in a given input. To this end, we induced flow diagrams from input state sequences that were perturbations of a small number of seed sequences, and investigated whether the structure of the seed sequences was identifiable in the induced flow diagrams.

For the second objective, we considered the application of flow diagrams to two practical problems in football analysis in order to evaluate their usefulness.

4.5.1 Performance Testing

In the second experiment, we used a state sequence generator that outputs synthetic state sequences with assigned criteria, and tested the performance of the algorithms on inputs of varying sizes.

The segmentations were generated using Markov-Chain Monte-Carlo sampling. Nodes representing the criteria set of size k were arranged in a ring and a Markov chain constructed, such that each node had a transition probability of 0.7 to remain at the node, 0.1 to move to the adjacent node, and 0.05 to move to the node two places away. Segmentations were computed by sampling the Markov chain starting at a random node. Thus, simulated data sets were generated with arbitrary size m, state sequence length n, criteria set size k.

We performed two tests on the generated segmentations. In the first, experiments were run on the four algorithms described in Section 4.4 with varying configurations of m, n and k to investigate the impact of the input size on the algorithm's performance. Two evaluation metrics were defined: we used the time to compute the prefix graph to evaluate runtime performance of all algorithms; and the complexity in the number of nodes of the output flow diagram was used to evaluate the sub-optimality of the heuristic algorithms. To evaluate the sub-optimality, we compared the complexity of the

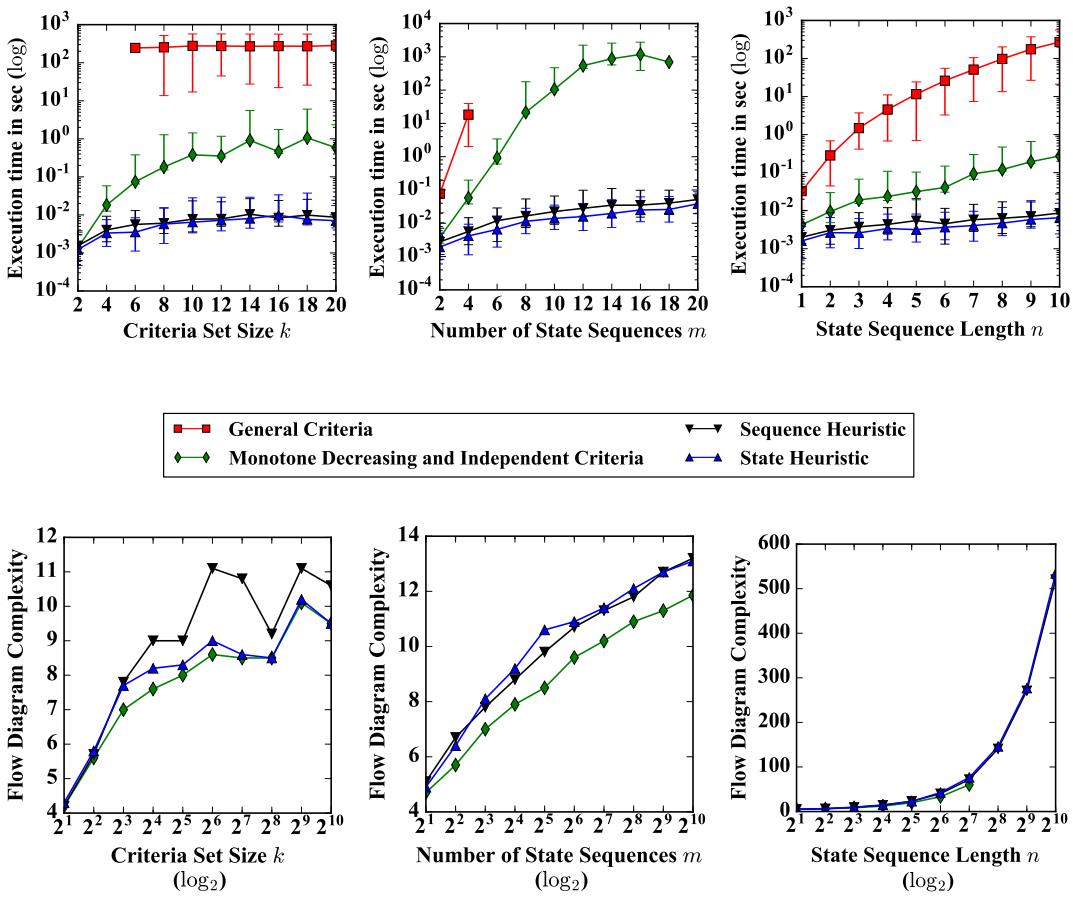


Figure 4.4: Runtime statistics for generating flow diagram *(top)*, and total complexity in the number of nodes of flow diagrams produced *(bottom)*. Parameter values of $m = 4$, $n = 4$ and $k = 10$ were used, except for the parameter appearing on the x-axis of each chart. The data points are the mean value and the error bars delimit the range of values over the five trials run for each input size.

output flow diagram produced by the two heuristic algorithms with the baseline complexity of the flow diagram produced by the exact algorithm for monotone increasing and independent criteria.

Each experiment was repeated ten times with different randomly-generated inputs for each trial, and the results presented are the mean values of the metrics over the trials. Limits were set such that the algorithm was terminated if the CPU time exceeded 1 hour, or the memory required exceeded 8GB.

The results of the first test showed empirically that the exact algorithms have

time and storage complexity consistent with the theoretical worst-case bounds, Figure 4.4 *(top)*.

For the second test, we investigated the complexity of the flow diagram induced by inputs of varying parameterisations when using the heuristic algorithms. The objective was to examine how close the complexity was to the optimal complexity produced using an exact algorithm. The criteria applied to the input state sequences were monotone decreasing and independent, and thus the corresponding algorithm was used to produce the baseline. Figure 4.4 *(bottom)* summarises the results for varying input parameterisations. The complexity of the flow diagrams produced by the two heuristic algorithms are broadly similar, and increase at worst linearly as the input size increases. Moreover, while the complexity is not optimal it appears to remain within a constant factor of the optimal, suggesting that the heuristic algorithms could produce usable flow diagrams for inputs where the exact algorithms are not tractable.

4.5.2 Perturbation Testing

The intuition that underpins the idea using flow diagrams to compactly represent state sequences is that there is some underlying structure with respect to the criteria in the state sequences, and that can be exploited. Within a set of m state sequences there are $g \ll m$ underlying processes that generate the sequences, and for each process, the sequences that are generated are similar but not identical. For example, in subsubsection 4.5.4.2 we consider attacking play sequences from football matches. The states in the sequences are actions taken by players from one team while that team is solely in possession of the ball. The sequences are segmented using criteria based on how the ball is moved around the playing-area. The team that generated the sequences will have a number of different tactics that they use to attempt to score a goal, and we propose that these tactics will manifest themselves as a series of such ball movements. The sequences may be noisy—additional ball movements may occur—but the underlying attack type should be apparent to the informed observer.

A flow diagram that is induced by a sequence such as this should ideally have g

paths with "heavy" weight edges that encode the sequences generated from the underlying process. The noise inherent in the sequences should be characterised by nodes with low degree, and can be removed from the flow diagram to produce a more compact graph, but without reducing the information inherent in the graph.

In order to test this insight, we conducted experiments on generated sequences that could be grouped by underlying structure. We used the state sequence generator described in Subsection 4.5.1 to construct *seed* sequences. The g seed sequences were then replicated by randomly selecting a seed sequence m times to produce the m input sequences, and the replicated sequences were perturbed by modifying the states that each state fulfilled. A randomly chosen fraction of the segments were perturbed by applying one of the following operations:

Changed criterion The criterion applied to a segment was replaced with another randomly-chosen valid criterion.

Extended segment The segment was extended such that it covered an adjacent state to its original starting or ending state.

Added segment A new segment was added to the segmentation.

Remove segment A segment was removed from the segmentation.

Thus, two sets of state sequences were produced: the set of g seed sequences to simulate the underlying processes in the state sequences; and the full set of m seed and perturbed sequences to simulate the noisy manifestation of the underlying processes. We then computed flow diagrams on both sets.

The full flow diagram will clearly contain more nodes than the seed flow diagram. We wanted to investigate whether the seed flow diagram appears as a sub-diagram in the full flow diagram. Furthermore, as the seed diagram will, in general, not be known, we investigated techniques that could be used to prune the full flow diagram such that the output would resemble the seed flow diagram. To this end, we adopted a simple strategy to determine whether the seed flow diagram could be recovered from

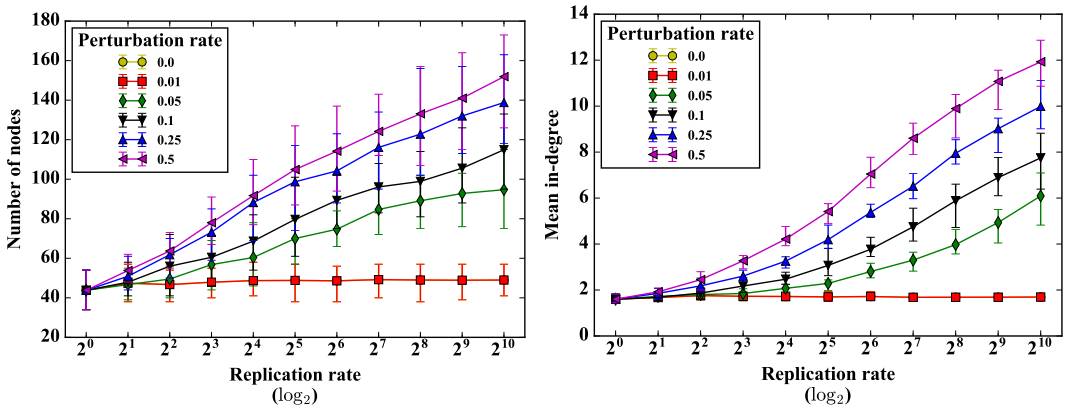


Figure 4.5: Mean complexity and node in-degree for flow diagrams for perturbed input state sequences. A seed set of 10 state sequences were generated, and test sets were derived by replicating the seed sequences and applying perturbation operations to the states. Error bars show minimum, maximum and mean results from repeating experiment using 10 randomly generated inputs.

full flow diagram. Edges were deleted from the full flow diagram if the associated "flow" of state sequences across the edge was below a given threshold. Nodes—and their incident edges—were then removed if the node's in-degree or out-degree was zero. This process was repeated until all nodes had non-zero in- and out-degree, and produced a *pruned flow diagram*.

The experiments were set up as follows. For each trial, a set of $g = 10$ seed trajectories of length $n = 20$ using an alphabet of size $k = 20$ were generated. Test sets of trajectories were generated for combinations of replication rate and perturbation rate and used as input to the state heuristic algorithm to produce a flow diagram. Ten trials were carried out for each combination, and the minimum, mean and maximum complexity and mean in-degree of the resulting flow diagrams were determined. The results are summarised in Figure 4.5.

The results suggest that the flow diagrams are able to compactly represent the underlying state. As the perturbation rate and replication rate increase, the complexity of the induced flow diagrams increases, however the increase appears to be linear even as the replication rate increases exponentially. Even in the worst case in our experiments, when the input were the seed sequences that had been replicated 1024 times and $50\,\%$

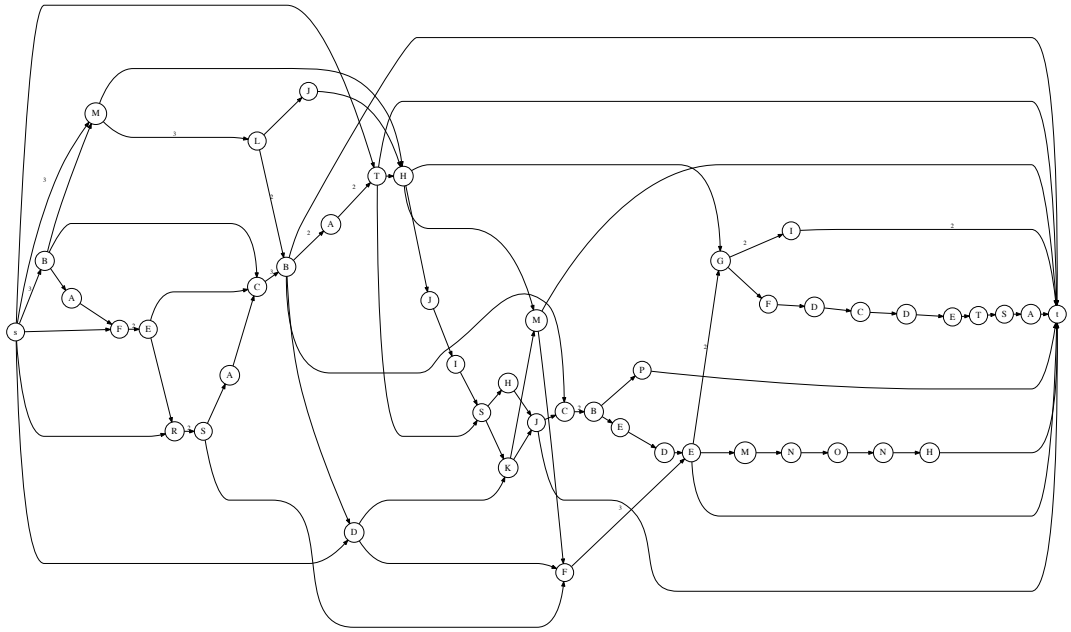
of the nodes had been perturbed, the average complexity of the resulting flow diagram was 153 compared to 43 for the seed flow diagram, i.e. only ≈ 3.5 times larger than the seed.

The pruned flow diagrams were also inspected to determine their similarity to the corresponding seed flow diagram. For each of the flow diagrams computed in the experiments, pruned flow diagrams were constructed and the complexity and mean in-degree were captured. Clearly, increasing the replication factor and perturbation rate of the input state sequences would result in flow diagrams of higher complexity. By choosing a pruning threshold that removes edges with a low number of associated sequences, a flow diagram of similar complexity and in-degree distribution could be found, see Figure 4.6. This pruning strategy was subsequently used to produce compact flow diagrams for the analysis in subsubsection 4.5.4.1.

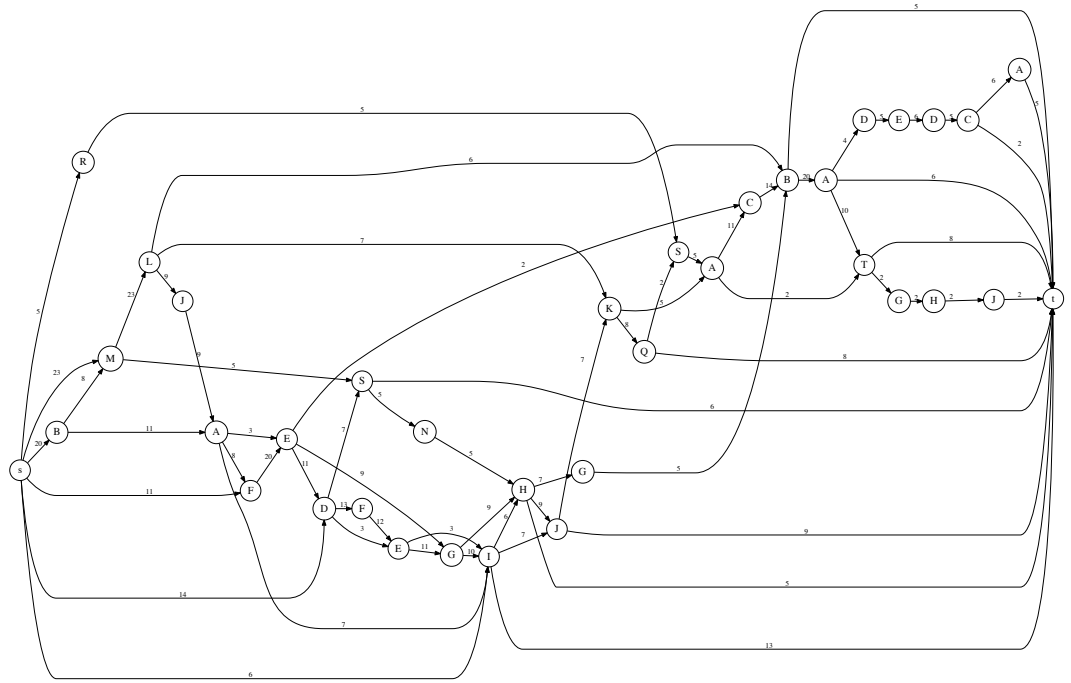
4.5.3 Shortest Path Selection

The exact algorithms described in Section 4.4 find a shortest s–t path in the prefix graph to induce the output flow diagram $\mathcal{F}$. There are typically many distinct candidate shortest paths in the prefix graph, and the choice of the path used to induce the flow diagram can influence various properties of the induced flow diagram. One option is to determine the shortest path by defining an objective function to rank the candidates and select the shortest path that is a maximiser on this function. To make this precise, let $\mathcal{P}$ be the set of shortest s–t paths in G. Define an objective function $f : P \to \mathbb{R}_{\geq 0}$ to score each shortest path $P \in \mathcal{P}$, and select the optimal path $P^* = \arg\max_{P \in \mathcal{P}} f(P)$.

For example, each edge in a shortest path P must advance one or more of the input sequences, and we can annotate the edges with a count of the number of sequences advanced. Let $f(P)$ be the minimum edge count of advanced sequences in P, and choose P^* as the path that maximizes the objective function. Each edge induces a corresponding node in $\mathcal{F}$ and thus by choosing P^* using this objective, $\mathcal{F}$ has the property that the smallest capacity of all the nodes in $\mathcal{F}$ is maximised. This may be desirable if one wishes to eliminate nodes with small capacity.



(a)



(b)

Figure 4.6: Flow diagrams computed from same input of 10 state sequences. (a) shows the flow diagram computed on the seed sequences, and (b) shows the flow diagram computed on an input where the seed sequences had been replicated 8 times and 10 % of the states had been perturbed. Much of the structure of the seed flow diagram has been preserved in the pruned flow diagram, suggesting that the underlying structure of the input has been recovered.

Table 4.1: Summary of the number of shortest paths (SPs) in a prefix graph constructed with parameters $k = 10$, $m = 5$ and $n = 10$ using the exact algorithm for general input. The objective function values are calculated on the number of sequences satisfied by the criteria associated with each edge in the path. There are large numbers of shortest paths within the graph, and also paths that optimise a given objective function, however the induced flow diagram in all cases has the same complexity in both number of edges and nodes.

Obj. Fnc.	# SPs	# Optimal SPs	Obj. Fnc. Value	# FD Nodes	# FD Edges
Min	1,731,423	322,028	2.00	16	26
25-th Percentile	1,731,423	146,346	2.25	16	26
Median	1,731,423	146,346	3.50	16	26
75-th Percentile	1,731,423	146,346	4.75	16	26
Mean	1,731,423	146,346	3.50	16	26

On the other hand, using an objective function of median or mean sequence count may produce a flow diagram with different characteristics. Analogously to the greedy strategy used in the heuristic algorithms, $f(P)$ may be calculated using either the number of sequences, or the total number of states that are advanced by the criteria associated with each edge in the path. In this section we describe the experiments undertaken to explore what impact the choice of objective function has on the resulting flow diagrams.

For a given G, there is often a large number of candidate shortest paths $\mathcal{P}$ in the prefix graph. Furthermore, for a given $f(P)$, there may also be many shortest paths that optimise $f(P)$, see Table 4.1. We analysed the shortest paths and induced flow diagrams, and found two phenomena that may explain why there are many such shortest paths.

First, two or more distinct shortest paths in G may induce flow diagrams that are isomorphic and have the same criteria applied to each node. As a simple example, consider an input of two sequences τ_1 and τ_2, each consisting of a single state, and each sequence is segmented with a single segment fulfilling criteria C_1 and C_2 respectively. The computed prefix graph will contain two shortest paths, however both will induce an identical flow diagram, see Figure 4.7.

Secondly, the prefix graph is a multi-graph where any two incident vertices may be linked by multiple edges, each labelled with a different criteria. Thus, if two vertices u

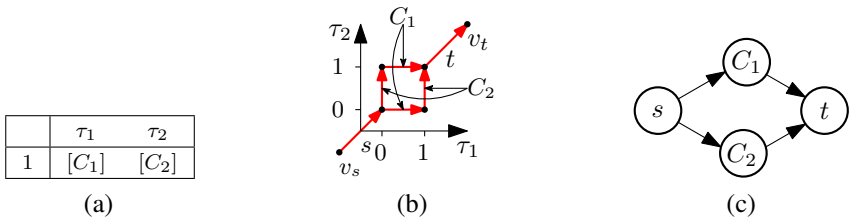


Figure 4.7: Even simple inputs can result in multiple shortest paths in the prefix graph that, in turn, induce identical flow diagrams.

and v are incident and are connected by multiple edges, then if there exists a shortest path using one of the multi-edges, then there will be a shortest path for each of the multi-edges. Each set of multi-edges connecting two vertices in the prefix graph will thus induce a set of flow diagrams that are isomorphic, other than that the node that corresponds to the multi-edge will be labelled with a different criteria.

Clearly, the existence of many isomorphic flow diagrams, some with nodes labelled with identical criteria, is not ideal. On the other hand, the set of candidate flow diagrams that are produced from a given input may contain insights into the selection of the best candidate.

4.5.4 Tactical Analysis in Football

Sports teams will apply tactics to improve their performance, and computational methods to detect, analyse and represent tactics have been the subject of several recent research efforts [17], [96], [140], [188], [191], [194]. See also the recent survey [94]. Two manifestations of team tactics are in the persistent and repeated occurrence of spatial formations of players, and in *plays* — a coordinated sequence of actions by players during an interval where a single team retains possession of the ball. We posited that flow diagrams would be a useful tool for compactly representing both these manifestations. Next, we describe the two approaches used in this section.

The input for the experiments is a database containing player trajectory and match event data from four home matches of the Arsenal Football Club from the $2007/08$ season, provided by [180]. For each player and match, there is a trajectory comprising

Table 4.2: Summary of the performance of the algorithms for the two football experiments. The columns are the runtime of the algorithm, the size of the input number of sequences (n), the total number of segments applied to the sequences, and the number of nodes and edges in the output flow diagram.

	Runtime (s)	# Sequences	# Segments	# States	# Nodes	# Edges
Formation	0.08	153	1492	1238	32	98
Formation (Pruned)	–	–	–	–	14	38
Possession: Home	0.04	66	866	1157	21	38
Possession: Visiting	0.01	38	358	483	16	28

a sequence of time-stamped location points in the plane, sampled at $10\,\mathrm{Hz}$ and accurate to $10\,\mathrm{cm}$. The origin of the coordinate system coincides with the centre point of the playing-area and the longer side of the playing-area is parallel to the x-axis — i.e. the playing-area is oriented so the goals are to the left and right. In addition, for each match, there is a log of all the match events, comprising the type, time-stamp and location of each event.

This data is used to generate two distinct inputs for the flow diagram algorithms, and the resulting flow diagrams are evaluated. Table 4.2 summarises the runtime performance and complexity of the inputs and outputs.

4.5.4.1 Defensive Formations

The spatial formations of players in football matches are known to characterize a team's tactics [16], and a compact representation of how formations change over time would be a useful tool for analysis. We investigated whether a flow diagram could provide such a compact representation of the defensive formation of a team, specifically to show how the formation evolves during a phase of play. In our match database, all the teams use a formation of four defensive players who orient themselves along a line across the playing-area. Broadly speaking, the ideal is for the formation to be "flat", i.e. the players are positioned along a line parallel to the y-axis. However the defenders will react to changes in circumstances, for example in response to opposition attacks, possibly causing the formation to deform. We constructed the following flow diagram to analyse the defensive formations used in the football matches in our database.

For each match in the database, the trajectories of the four defensive players were re-sampled at one-second intervals to extract the point-locations of the four defenders. The samples were partitioned into sequences $\mathcal{T} = \{\tau_1, \ldots, \tau_m\}$ corresponding to intervals such that a single team was in possession of the ball. Let $\tau_i[j]$ be the j-th state in the i-th state sequence. Each $\tau_i[j] = (p_1, p_2, p_3, p_4)$, where p_i is the location of a player in the plane, such that the locations are ordered by their y-coordinate: $y(p_i) \leq y(p_{i+1}) : i \in \{1, 2, 3\}$.

The criteria used to summarise the formations were derived from those presented by Kim et al. [121]. The angles between pairs of adjacent players (along the defensive line) were used to compute the formation criteria, see Figure 4.8. The scheme in Kim et al. was extended to allow multiple criteria to be applied where the angle between pairs of players is close to the threshold between intervals. The reason for this was to facilitate compact results by allowing for smoothing of small variations in contiguous states.

The criteria C applied to each state is a triple (x_1, x_2, x_3), computed as follows. Given two player positions p and q as points in the plane such that $y(p) \leq y(q)$, let p' be an arbitrary point on the interior of the half-line from p in the direction of the positive y-axis, and let $\angle p'pq$ be the angle induced by these points, and thus denotes the angle between the two player's positions relative to the goal-line. Let $R(-1) = [-90°, -10°)$, $R(0) = (-30°, +30°)$, and $R(1) = (+10°, +90°]$ be three angular ranges. Thus, $\mathcal{C} = \big\{(x_1, x_2, x_3) : x_1, x_2, x_3 \in \{-1, 0, 1\}\big\}$ is the set of 27 available criteria.

Each state sequence $\tau_i \in \mathcal{T}$ is segmented according to the criteria set $\mathcal{C}$. A given state $\tau_i[j] = (p_1, p_2, p_3, p_4)$ may satisfy the criteria (and thus have the formation) (x_1, x_2, x_3) if $\angle p'_i p_i p_{i+1} \in R(x_i)$ for all $i \in \{1, 2, 3\}$.

The state heuristic algorithm was run on an input of 153 state sequences derived from the morphology of the defensive formations of a single team in four matches when the opposing team is attacking, and the possession ends with a shot at goal or a tackle or foul within $30\,\mathrm{m}$ of goal. The induced flow diagram was then pruned by

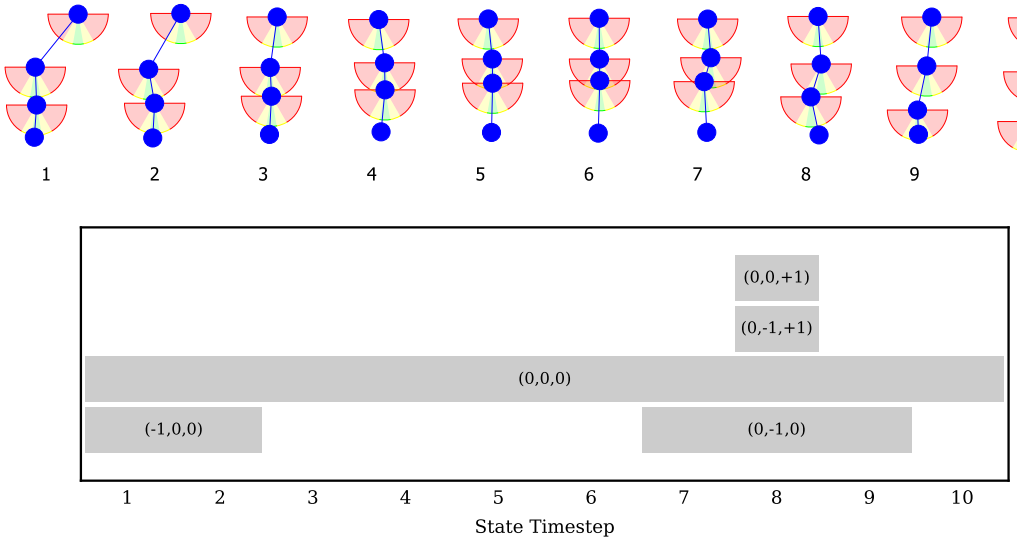


Figure 4.8: Segmentation of a single state sequence τ_i. The formation state sequence is used to compute the segmentation representation, where segments corresponding to criteria span the state sequence *(bottom)*. The "gauges" near each edge show the intervals for the criteria, and edges in the yellow interval will result the formation fulfilling two criteria.

removing low-volume edges associated with only a single edge, see Figure 4.9.

The resulting flow diagram produces a compact summary of the team's defensive strategy, and several observations can readily be made. The defenders are able to maintain a stable (0,0,0) formation—a *flat back-four*—for many of the possessions. In 57 possessions ($37.3\,\%$), the defense maintained this formation for the duration. Furthermore, in $14\,(9.2\,\%)$ other cases, the defenders began with a flat back-four, and then changed to another formation before returning to the original formation. The flat back-four is generally known to be an effective formation for defenses to use when attempting an offside trap—a defensive tactic that has the objective of trapping an opposition player in an offside position, and thus winning a free-kick. The flow diagram makes it clear that this team prefers a flat back-four and thus may favour the offside trap. An insight such as this would appear be useful to a coach when analysing the tactics of an opponent in an upcoming match.

The flow diagram also suggests that the defenders are able to maintain a stable

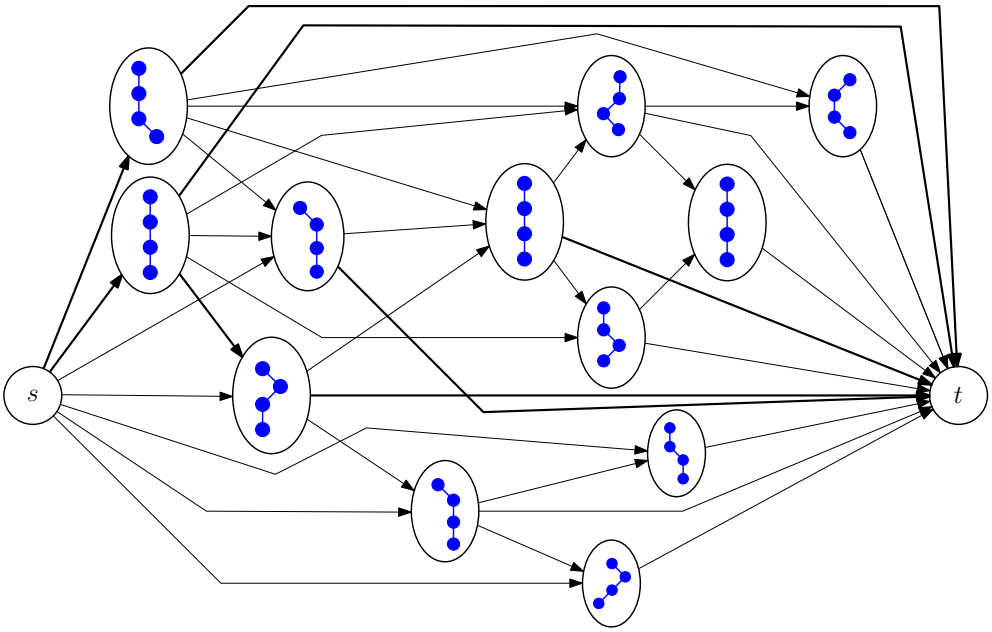


Figure 4.9: Flow diagram for formation morphologies of 153 defensive possessions. The flow diagram was pruned to remove edges associated with only a single state sequence. The size of the nodes and weights of the edges are sized to reflect the "flow" of sequences passing through them.

formation. The s–t paths within the graph are short, with a maximum path length of 5 states. This suggests that the team is well-organised defensively, and able to maintain a preferred formation persistently, in the face various opposition attacks.

4.5.4.2 Attacking Plays

In this second experiment, we used a different formulation to produce flow diagrams to summarise phases of attack. During a match, the team in possession of the ball regularly attempts to reach a position where they can take a shot at goal, and will use a variety of tactics to achieve such a position. For example, teams can vary the intensity of an attack by pushing forward, moving laterally, making long passes, or retreating and regrouping. We modelled attacking possessions as state sequences, segmented according to criteria representing the attacking intensity and tactics employed, and computed flow diagrams for the possessions. In particular, we were interested in determining whether differences in tactics employed by teams when playing at home

Table 4.3: The criteria applied to the attacking sequences, including the symbols used in the flow diagrams to represent each movement type.

C_1		*Backward movement:* $v_x(\tau_i[j]) < 1$ — a sub-sequence of passes or touches that move in a defensive direction.
C_2		*Lateral movement:* $-5 < v_x(\tau_i[j]) < 5$ — passes or touches that move in a lateral direction.
C_3		*Forward movement:* $-1 < v_x(\tau_i[j]) < 12$ — passes or touches that move in an attacking direction, at a velocity in the range achievable by a player sprinting, i.e. approximately $12m/s$.
C_4		*Fast forward movement:* $8 < v_x(\tau_i[j])$ — passes or touches moving in an attacking direction at a velocity generally in excess of maximum player velocity.
C_5		*Long ball:* $30 < d_x(\tau_i[j])$ — a single pass travelling $30m$ in the attacking direction.
C_6		*Cross-field ball:* $20 < d_y(\tau_i[j]) \wedge \angle\tau_i[j] \in [-10, 10] \cup [170, 190]$ — a single pass travelling $20m$ in the cross-field direction with an angle within $10°$ of the y-axis.
C_7		*Shot resulting in goal:* a successful shot resulting in a goal.
C_8		*Shot not resulting in goal:* a shot that does not produce a goal.

or away, see [17], are apparent in the flow diagrams.

We focus on *ball events*, where a player touches the ball, e.g. passes, touches, dribbles, headers, and shots at goal. The event sequence for each match was partitioned into sequences $\mathcal{T} = \{\tau_1, \ldots, \tau_m\}$ such that each τ_i is an event sequence where a single team was in possession, and $\mathcal{T}$ includes only the sequences that end with a shot at goal. Let $\tau_i[j]$ be a tuple (p, t, e) where p is the location point in the plane where an event of type $e \in \{touch, pass, dribble, header, shot, clearance\}$ occurred at time t. We are interested in the movement of the ball between an event state $\tau_i[j]$ and the next event state $\tau_i[j+1]$, in particular, let $d_x(\tau_i[j])$ (resp. $d_y(\tau_i[j])$) be the distance in the x-direction (resp. y-direction) between state $\tau_i[j]$ and the next state. Similarly, let $v_x(\tau_i[j])$ (resp. $v_y(\tau_i[j])$) be the velocity of the ball in the x-direction (resp. y-direction) between $\tau_i[j]$ and its successor state. Let $\angle\tau_i[j]$ be the angle defined by the location of $\tau_i[j]$, $\tau_i[j+1]$ and a point on the interior of the half-line from the location of $\tau_i[j]$ in the positive y-direction.

Criteria were defined to characterise the movement of the ball — relative to the goal the team is attacking — between event states in the possession sequence. The definitions of the criteria $\mathcal{C} = \{C_1, \ldots, C_8\}$ are presented in Table 4.3.

For a football analyst, the first four criteria are simple movements, and may not be

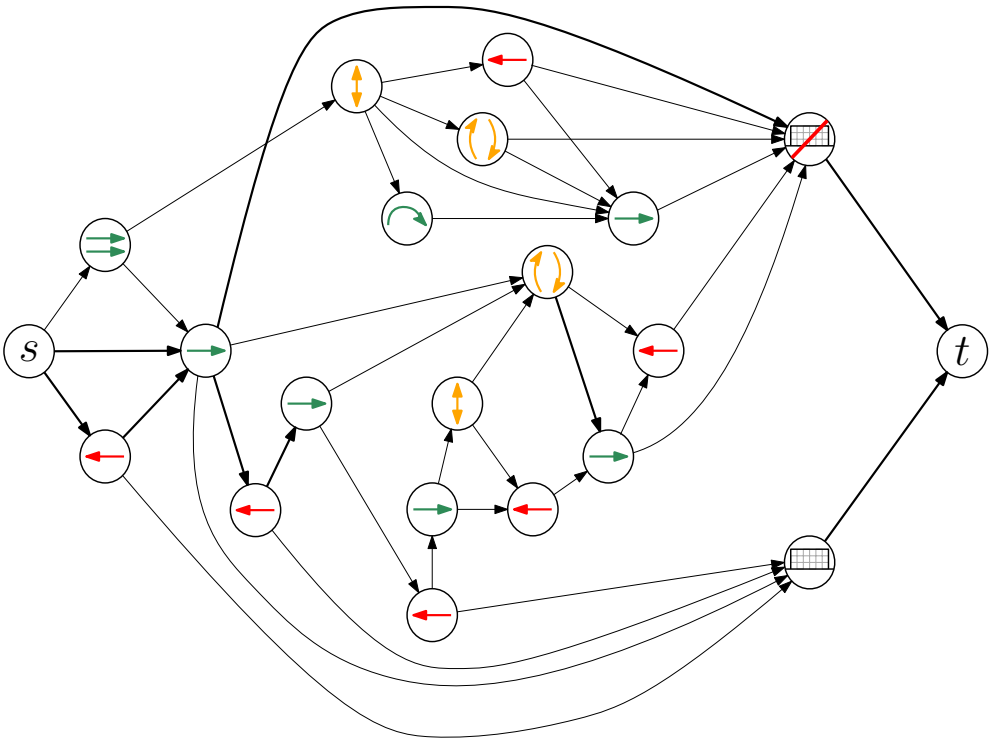


Figure 4.10: Flow diagram for the home team. The edges with thicker stroke weight are supported by at least ten attack sequences.

particularly interesting. The last four events are likely to be more significant: the long ball and cross-field ball change the locus of attack; and the shot criteria represent the objective of an attack.

The possession state sequences for the home and visiting teams were segmented according to the criteria and the state heuristic algorithm was used to compute the flow diagrams. The home-team input consisted of 66 sequences covered by a total of 866 segments, and resulted in a flow diagram with 21 nodes and 38 edges, see Figure 4.10. Similarly, the visiting-team input consisted of 39 state sequences covered by 358 segments and the output flow diagram complexity was 16 nodes and 28 edges, as shown in Figure 4.11.

At first glance, the differences between these flow diagrams may be difficult to appreciate, however closer inspection reveals several interesting observations. The s–t paths in the home-team flow diagram tend to be longer than those in the visiting team's, suggesting that the home team tends to retain possession of the ball for longer,

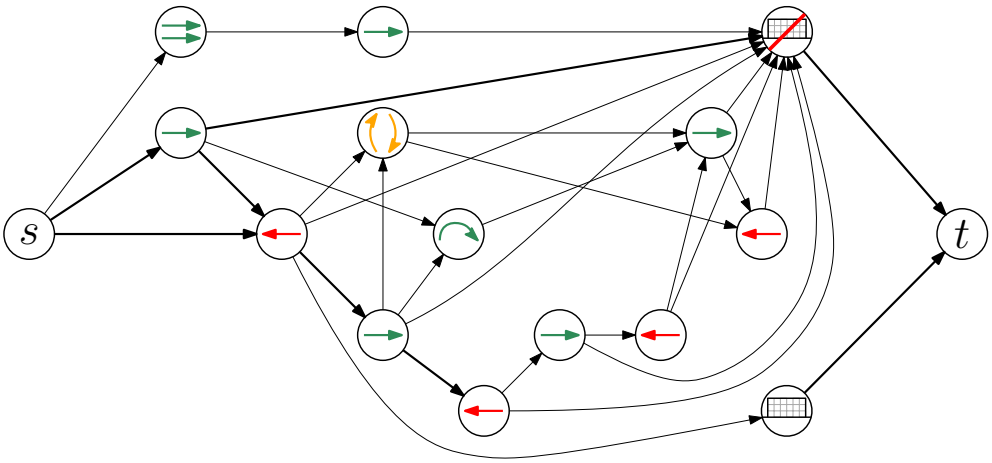


Figure 4.11: Flow diagram for the visiting team. The edges with thicker stroke weight are supported by at least ten attack sequences.

and varies the intensity of attack more often. Moreover, the nodes for cross-field passes and long-ball passes to occur earlier in the s–t paths in the visiting team's flow diagram. These are both useful tactics as they alter the locus of attack, however they also carry a higher risk. This suggests that the home team is more confident in its ability to maintain possession for long attack possessions, and will only resort to such risky tactics later in a possession. Furthermore, the tactics used by the team in possession are also impacted by the defensive tactics. As Bialkowski et al. [17] found, visiting teams tend to set their defence deeper, i.e. closer to the goal they are defending. When the visiting team is in possession, there is thus likely to be more space behind the home team's defensive line, and the long ball may appear to be a more appealing tactic. The observations made from these are consistent with our basic understanding of football tactics, and suggest that the flow diagrams are interpretable in this application domain.

4.6 Conclusion

We introduced flow diagrams as a compact representation of a large number of state sequences. We argued that this representation gives an intuitive summary allowing the user to detect patterns among large sets of state sequences, and gave several algorithms depending on the properties of the segmentation criteria. These algorithms only run

in polynomial time if the number of state sequences m is constant, which is the best we can hope for given the problem is $W[1]$-hard when m is a parameter. As a result we considered two heuristic algorithms capable of processing large data sets in reasonable time, however we were unable to give an approximation bound. We tested the algorithms experimentally to assess the utility of the flow diagram representation in a sports analysis context, and also analysed the performance of the algorithms of inputs of varying parameterisations.

Chapter 5

Integrated Clustering and Outlier Detection

As data set sizes continue to explode there is an urgent need to be able to process, browse, explore and navigate large, highly- or infinitely-dimensional data sets. Many visualization techniques are based on some form of dimensionality reduction and are plagued with distortion errors leading to ambiguity and misinterpretation. In this paper we make a radical departure and use a principled approach based on optimization duality to visualize and browse the data space in an unsupervised fashion.

In the specific case of trajectory analysis, there is a natural visualisation of the trajectories in their original Euclidean space, for example in the plane, however it is often difficult to discern much more than the general distribution of the trajectories in the space. For all but the most simple inputs, the trajectories will often intersect with themselves and other trajectories, and it is difficult to identify properties such as the shape, length and orientation. This issue is particularly acute for football player trajectories, but as we will see, can occur in other applications.

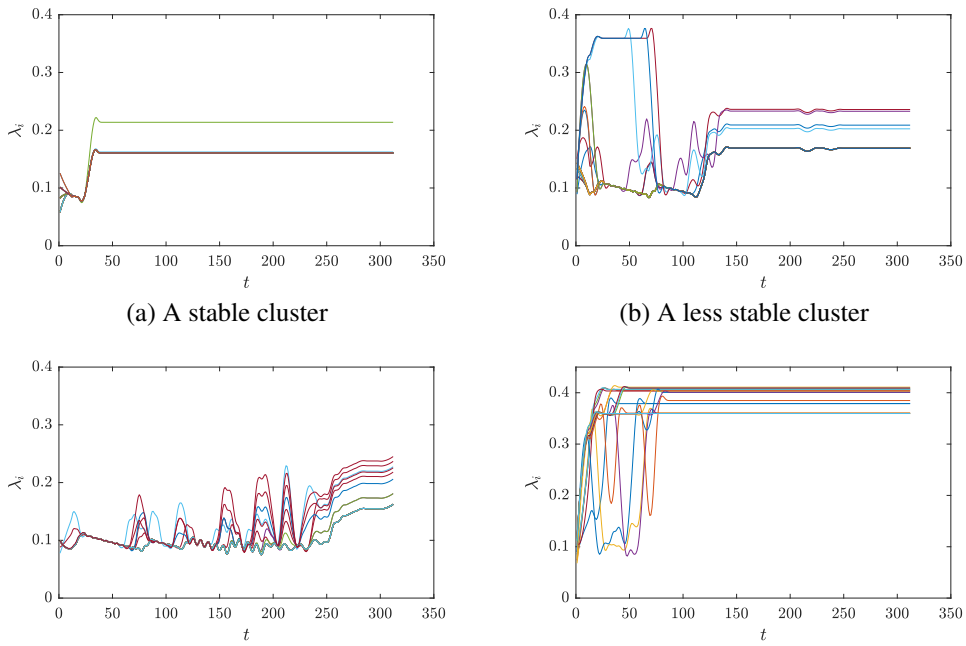
The two prominent unsupervised learning tasks of clustering and outlier detection have traditionally been considered separately. However, there is a growing body of work that considers them in unison. In fact the clustering and outlier detection tasks have a "dual" relationship—a data point located deep inside a cluster is a non-outlier.

Conversely, the outliers are, by definition, those points that are not part of a cluster.

We model the problem as the facility location with outliers problem (FLO). The input is a set of *data points* $\mathcal{C}$, an *establishment cost* f_j for setting data point j as an *exemplar*, a *connection cost* d_{ij} for connecting data point i to exemplar j, and an *outlier allowance* ℓ that permits exactly ℓ data points to be treated as outliers. The problem is to find a set of exemplars $\mathcal{E} \subseteq \mathcal{C}$, a set of ℓ data points that are outliers $\mathcal{O} \in \mathcal{C}$, and an assignment of the remaining $n - \ell$ data points connecting each point to an exemplar, such that the sum of the establishment costs of the exemplars and the connection costs of the connected data points is minimised.

This problem is known to be NP-hard [97], and furthermore cannot be approximated with in factor of 1.822 under the Euclidean metric or $3 - \epsilon$ under a general metric, unless P = NP [44]. We consider two distinct techniques for efficiently finding a good approximate solution the **FLO** problem that approach the problem from significantly different perspectives: the Lagrangian heuristic algorithm (LH) and the affinity propagation algorithm (AP). Both the techniques model the problem as an integer program, but with significantly different formulations, and from these formulations heuristic algorithms are derived. The LH algorithm uses a Lagrangian relaxation of the **FLO** problem which can be solved using the subgradient algorithm [15]. The AP algorithm is a *message passing* algorithm [126] based on a formulation of the problem as a probabilistic graphical model. Both algorithms are iterative, simple to implement, and will often efficiently find a feasible solution. Although neither algorithms offers an optimality guarantee, or even a guarantee of finding a feasible solution, they have been shown to work well in practice, finding near optimal solutions in a small number of iterations.

Both formulations define a set of *auxiliary variables* which is used to guide the algorithm towards a good solution: the Lagrangian *dual variables* for the LH algorithm and the *message variables* for the AP algorithm. By monitoring these auxiliary variables over the iterations of the algorithms, a time-series is generated. These time-series produce an implicitly two-dimensional representation of data regardless of the



(a) A stable cluster

(b) A less stable cluster

(c) An unstable cluster

(d) The outliers

Figure 5.1: The data represented in the Lagrangian space. All figures show how the Lagrangian multiplier changes as the algorithm iterates towards termination.

original dimensionality of the data. We claim that representation is an effective means of visualizing the cluster properties of the input data, and can furthermore be used to monitor performance of the heuristic algorithms.

Example Figure 5.1 shows the sequence of Lagrangian multipliers (λ) for data points in three clusters (a)–(c) and a set of outliers (d). Several interesting observations can be made by visual inspection: the cluster in (a) stabilizes by the 50-th iteration and thus is likely to be more compact, while the cluster in (c) is less stable and the Lagrangian multiplier values are oscillating until the end of the algorithm execution. The cluster in (b) has intermediate stability relative to (a) and (c). Finally, the fact that the cluster in (c) does not stabilize until around 320 iterations, long after the remaining clusters and the outliers have stabilized suggests that the data point chosen as exemplar was oscillating between two or more candidate data points.

The remainder of this chapter is structured as follows. In Section 5.1 we review

the related work to this problem. The problem is formally defined and the formulations for the Lagrangian heuristic and affinity propagation algorithm are presented in Section 5.2. For the Lagrangian heuristic algorithm we propose a modification to the algorithm in Ott *et al.* [158], which we argue is an improvement. Furthermore, we present a new affinity propagation formulation for the **FLO** problem, that requires fewer variables and is more interpretable in comparison to the previous known formulation in [158]. We then define the visualisation method on the auxiliary variables for both the LH and AP algorithms.

In Section 5.3 we report on the process used to find the improved LH algorithm as a case study to demonstrate the utility of the proposed visualisation method for understanding the execution properties of the algorithm. We used the dual visualisation method presented in Section 5.2 to uncover an issue with the original formulation of the LH algorithm, and to validate that the improved algorithm is free from this issue. Finally, Section 5.4 contains the results of experiments to evaluate the presented algorithms on synthetic and real datasets in order to illustrate their effectiveness. In particular, we segment trajectories made by attacking players during a football match, and then cluster under the flow diagram. The auxiliary variable visualisation technique is used to show several interesting observations about the obtained clusters and outliers.

5.1 Related Work

Clustering and outlier detection have often been treated as distinct unsupervised learning tasks [4], [42]. The trend to treat them in a combined fashion is relatively recent but is growing in interest [46], [87], [99], [158].

Some early work in this area was made to design an approximation algorithm for the **FLO** problem, with a guarantee on the optimality of the solution. Jain and Vazirani [117] used a primal-dual approach to formulate a 3-approximation for the metric facility location problem, and this algorithm was modified by Charikar *et al.* [44] to solve

the **FLO** problem. While the approach was of theoretical interest, the algorithm was not practical for many inputs, and furthermore the optimality bound was quite large, particularly in comparison to the results that were obtainable by heuristics.

The method of using Lagrangian relaxations on integer problems was first used by Held and Karp [104], [105] to devise an algorithm for the travelling salesman problem. The approach was then applied to many combinatoric optimization problems—including the facility location problem—to identify lower bounds on the objective function value and to devise efficient algorithms, see Fisher [71] for a survey. There have been several algorithms proposed to solve the relaxed version of the facility location problem [12], [59]. The approach taken in this paper is based on the *subgradient algorithm* [15, chap. 4], which uses the subgradient method to optimise the piece-wise linear concave Lagrangian dual problem [174].

A fundamentally different approach to the facility location problem was presented by Frey and Dueck [82], by formulating the problem as a *factor model* and defining an energy function to optimise on the factors [124, chap. 11]. A solution to the problem can then be efficiently computed by using the *sum-product* algorithm [126]. This model was subsequently refined in [88], [131].

From an optimization perspective our work is most similar to that in Ott *et al.* [158] for the LH algorithm and Lazic *et al.* [131] for the AP algorithm. However there are key differences: (i) our emphasis is on using the auxiliary variables to create a dual space, and to map and browse the data in that space; (ii) at a technical level we solve the intermediate problem for the LH algorithm in a significantly different fashion; and (iii) we extend the factor model used by Lazic *et al.* to support outliers.

5.2 Method

We model the joint clustering and outlier detection task as a facility location problem with outliers. Given an instance of the problem as:

- $\mathcal{C} = \{1, \ldots, n\}$ data points,

- a vector $\mathbf{f}$ where f_j is the exemplar establishment costs of data point j,
- a matrix $\mathbf{d}$ such that d_{ij} is the connection cost of assigning point i to exemplar j, and
- a parameter $\ell \in \mathbb{Z}_{\geq 0}$ that is number of outliers allowed.

We wish to find a subset $\mathcal{E} \subseteq \mathcal{C}$ of points designated as exemplars, an assignment of $n - \ell$ data points to exactly one exemplar from $\mathcal{E}$ with the remaining ℓ data points $\mathcal{O} \in \mathcal{C}$ assigned as outliers, such that the assignment minimises the total establishment cost of the exemplars and connection cost incurred by connecting the non-outlier points to their nearest exemplar.

The *primal problem* can be formalized as an integer program, defined as

$$\mathbf{FLO} \equiv \min_{\mathbf{x},\mathbf{o}} \sum_j f_j x_{jj} + \sum_i \sum_j d_{ij} x_{ij} \tag{5.1a}$$

$$\text{subject to} \quad x_{ij} \leq x_{jj} \tag{5.1b}$$

$$o_i + \sum_j x_{ij} = 1 \tag{5.1c}$$

$$\sum_i o_i = \ell \tag{5.1d}$$

$$x_{ij}, o_i \in \{0, 1\} \tag{5.1e}$$

and denote the obtained minimal objective function value as Z_p.

Given an assignment to $\mathbf{x}$ and $\mathbf{o}$ that are a minimiser for the problem, we can recover the cluster and outlier assignment as follows. The selected exemplars lie on the diagonal of $\mathbf{x}$ and can be recovered as $\{j \mid x_{jj} = 1\}$. The assignment of points to exemplars is recovered as $\{(i, j) \mid x_{ij} = 1\}$ and the points selected as outliers are $\{i \mid o_i = 1\}$.

The constraints enforce a feasible solution:

- points can only be assigned to valid exemplars (5.1b);

- every point must be assigned to exactly one other point or be declared an outlier (5.1c);
- exactly ℓ outliers have to be selected (5.1d);
- only integer solutions are allowed (5.1e).

This formulation describes the facility location problem [63] with the addition of outlier selection.

Furthermore, for a given assignment to $\mathbf{x}$ and $\mathbf{o}$, h_p computes the objective function value:

$$h_p(\mathbf{x}, \mathbf{y}) = \sum_j f_j x_{jj} + \sum_i \sum_j d_{ij} x_{ij}.$$

In the sequel we formalise the proposed Lagrangian heuristic and affinity propagation algorithms that solve the $\mathbf{FLO}$ problem.

5.2.1 Lagrangian Heuristic Algorithm

The LH algorithm is a simple iterative algorithm for finding an approximate solution to the $\mathbf{FLO}$ problem. One or more of the constraints in the primal problem are relaxed to obtain a simpler problem, which is parameterized by two sets of variables: the original primal variables and the dual variables: the *Lagrangian multipliers* created during the relaxation. The primal and relaxed problems exhibit weak duality in the sense that for a fixed assignment to the primal variables the assignment of dual variables that is a maximiser of the relaxed problem is a lower bound on the objective in the primal problem, and vice versa. This suggests an approach to finding a good solution to the primal problem by alternately solving the relaxed problem in the primal and dual variables until such time as the gap between the primal and dual objectives are small. The algorithm presented here is a version of the subgradient algorithm and uses this approach.

The Relaxed Problem The first step is to relax one or more of the "tough" constraints of the **FLO** problem by removing the constraint and adding a new term to the objective with an associated coefficient, known as the Lagrangian multiplier $\lambda \in \mathbb{R}^n$. The Lagrangian multiplier λ can be considered as the penalties incurred for the violation of each of the relaxed constraints.

For the **FLO** problem, we relax the constraint (5.1c): $o_i + \sum_j x_{ij} = 1$ for each $i \in \mathcal{C}$ and associate a Lagrange multiplier λ_i with the objective term to obtain the *relaxed problem* $\mathbf{FLO}(\lambda)$

$$\mathbf{FLO}(\lambda) \equiv \min_{\mathbf{x},\mathbf{o}} \sum_j f_j x_{jj} + \sum_i \sum_j d_{ij} x_{ij} + \sum_i (1 - o_i - \sum_j x_{ij})\lambda_i \tag{5.2}$$

$$\begin{aligned} \text{subject to} \quad & x_{ij} \leq x_{jj} \\ & \sum_i o_i = \ell \\ & x_{ij}, o_i \in \{0,1\} \quad \forall i, j \end{aligned}$$

This is a minimisation problem in the primal variables $\mathbf{x}$ and $\mathbf{o}$, for a given λ. We could also look to maximise the dual variables λ for $\mathbf{FLO}(\lambda)$ to find the tightest lower bound on the relaxed objective, thus giving the *dual problem* $\mathbf{FLO}_D$

$$\mathbf{FLO}_D \equiv \max_{\lambda} \mathbf{FLO}(\lambda) \tag{5.3}$$

$$\text{subject to} \quad \lambda_i \text{ free} \quad \forall i$$

and denote Z_d as the obtained optimal objective function value for the problem. This is an unconstrained optimisation problem on a piecewise linear objective function [15, chap. 4] and this can be solved using the *subgradient method* [174], a numerical optimisation technique. The dual problem implies an assignment of the primal variables, and although it does not directly help us find an optimal assignment of the primal variables, we can use the technique as part of the heuristic *subgradient algorithm*, described below.

Interpretation of Lagrangian Multipliers We use the following interpretation of the $\mathbf{FLO}(\lambda)$ problem to reason about the algorithm, and we will show that this interpretation implies an algorithm that out-performs the existing algorithm presented in Ott *et al.* [158]. This interpretation closely follows a similar argument that Jain and Vazirani [117] used in designing their primal-dual algorithm for facility location.

Intuitively, λ_i can be thought of as a *budget* that point i is willing to "expend" to pay to connect to an exemplar j. However, j can only be an exemplar if it is connected to by a set of points who have sufficient surplus budget—after paying the connection cost—to cover the establishment cost of j. Thus, a point i can expend its budget λ_i with an exemplar j in two ways: first by paying the connection cost d_{ij} to j; and secondly by contributing to the establishment cost f_j of j. Furthermore, ℓ points can be chosen as outliers, and thus not pay to connect to any exemplar. Intuitively, a point i would be an outlier if it has insufficient budget to connect to its nearest exemplar.

Under this interpretation, the relaxed problem (5.2) is parameterised with the budgets λ that each point is willing to pay, and the objective is to choose the exemplars that nearby points are willing to pay to connect to, and cover their share of the establishment cost, such that the total expenditure is minimised. Taking (5.2) and arranging the terms of the objective function, we have

$$h_d(\mathbf{x}, \mathbf{o}, \lambda) = \sum_i (1 - o_i)\lambda_i + \sum_j \left[f_j x_{jj} - \sum_i (\lambda_i - d_{ij}) x_{ij} \right] \tag{5.4}$$

where h_d is the objective function value obtained for a given assignment of the primal and dual variables. This formulation gives an insight into the structure of the problem. It would appear, at face value, that selecting outliers as the points with the largest λ_i and thus minimising the first term in (5.4) would be a good strategy for finding an optimal assignment, however there is also the *opportunity cost* that must be paid for the remaining points to be connected to exemplars, and thus a more nuanced approach

is necessary. To see this, let

$$\mu_j = f_j - \sum_i \max\left[\lambda_i - d_{ij}, 0\right].$$

The value of μ_j for a given λ is a measure of whether there are a sufficient number of data points who have budget to connect to j, and pay the total establishment cost of j. Thus if $\mu_j < 0$ then j is a good candidate to be an exemplar. It should be clear, however, that if outliers are selected where λ_i is relatively large, any contribution that point i is making to a μ_j will be removed. There is thus a conflict in that a "large" value of λ_i which implies that i is an outlier, **and** that i is willing to pay the connection and a share of the establishment cost of an exemplar. Resolving this conflict is the main insight behind the improved LH algorithm we present here, and is manifested in the strategy for solving the relaxed problem by inspection, below.

Next, given the budget λ_i that data point i is willing to expend to connect to an exemplar, let $\mathcal{E} = \{j \mid \mu_j < 0\}$ be the subset of $\mathcal{C}$ that are "paid for" as exemplars. Let

$$\delta_i = \lambda_i - \min_{j \in \mathcal{E}} d_{ij}.$$

This value represents the total surplus (or deficit) budget that data point i has to contribute to its nearest exemplar. If $\delta_i \geq 0$ then point i can connect to its nearest exemplar and possibly contribute to the establishment cost of the exemplar, and if $\delta_i < 0$ then this indicates that i is an outlier, as it is unwilling to pay even the connection cost to its closest exemplar.

Under this interpretation, it would appear that the ideal quantities for dual variables would satisfy the following conditions:

Proposition 5.1. *For a primal feasible assignment of* $\mathbf{x}$ *and* $\mathbf{o}$*, the following properties of the dual variables* λ*,* μ *and* δ *are desirable:*

1. *Each outlier* i *has insufficient budget to connect to the nearest exemplar, i.e.* $\delta_i < 0$.

2. *For each exemplar j, the total budget of all points that connect to j covers the total connection cost and the establishment cost, with no surplus, i.e. $\mu_j = 0$.*

3. *For each point i connected to an exemplar j, i has sufficient budget to cover its connection cost, and makes a non-negative contribution to the establishment cost of j, i.e. $\delta_i \geq 0$.*

Although it is not possible to provide any guarantee on optimality, there is good reason to suggest that an assignment satisfying these conditions is a good assignment. As we will show, the strategy we use for selecting outliers, combined with these conditions implies some useful properties for an obtained solution. Furthermore, we find experimentally that these conditions are indeed met in most cases—at least up to a small error—see Figure 5.5 and Subsection 5.4.2.

The values λ_i, μ_j and δ_i each give a perspective into the state of the execution of the algorithm for each iteration, and we use these to construct time-series to monitor the system, see Subsection 5.2.3.

The Subgradient Algorithm The LH algorithm presented here is a specific instance of the subgradient algorithm. This algorithm seeks to obtain an assignment of the primal variables to obtain a primal objective function value that is close to the optimal value. Starting from an initial assignment of the dual variables, the algorithm alternately finds an assignment of the primal variables that approximately optimises the relaxed objective and then an assignment of the dual variables that improves the dual objective.

The approach taken for finding the primal and dual assignments are distinct. To find an assignment of the primal variables, we *solve by inspection* to obtain, in time polynomial in n, an assignment of $\mathbf{x}$ and $\mathbf{o}$, that is feasible in the relaxed problem, and induces a good—though not provably optimal—objective function value. There is a trade-off, where optimality is sacrificed for performance, and we describe the procedure for finding this assignment in the sequel.

The dual assignment is found by computing a *subgradient* s_i of λ_i for each i, and then performing one step of the subgradient method [174], which is a gradient ascent algorithm that finds the optimal value of a non-differentiable concave function—such as $\mathbf{FLO}(\lambda)$. The algorithm works by computing a subgradient of the objective function by updating λ in the direction of the subgradient s by a certain step-length θ, see Algorithm 2. The step-length may change at each iteration, and the update rule for the dual variables in iteration t is

$$\lambda_i^{t+1} = \max(\lambda_i^t + \theta^t s_i, 0).$$

Algorithm 2 The Lagrangian Heuristic

```
procedure SUBGRADIENTALGORITHM(d, f, ℓ, α, θ)
    λ^0 ← 0, t ← 1                                  ▷ Initialise
    Ẑ_p ← ∞, Ẑ_d ← −∞                               ▷ Objective estimates
    while |Ẑ_p − Ẑ_d| > ε and t ≤ t_limit do
        x, o, Ẑ_p, Ẑ_d ← SOLVERELAXED(λ^t, d, f, ℓ)   ▷ Solve by inspection
        for all i ∈ C do                            ▷ Update subgradient
            s_i ← 1 − o_i − Σ_j x_ij
        end for
        λ^t ← max[λ^{t−1} + θs, 0]                  ▷ Update λ
        θ ← αθ                                      ▷ Update step-length
        t ← t + 1
    end while
    return x, o
end procedure
```

Under certain technical conditions on the step-length at each iteration, the subgradient method is guaranteed to converge. One such set of conditions is that the step-length is non-summable and diminishing:

$$\lim_{t\to\infty} \theta_t = 0, \quad \sum_{t=1}^{\infty} \theta_t = \infty, \quad \theta_t \geq 0.$$

We use the following step-length rule, which satisfies the conditions. Given an initial step value θ_0 and a decay parameter $\alpha \in [0, 1)$, we compute the step-length θ^t for

iteration t of the algorithm as

$$\theta^t = \theta^{t-1}\alpha.$$

It is important to note that this convergence guarantee for the subgradient method does not imply a similar guarantee for the subgradient algorithm, where the primal assignment that obtains $\mathbf{FLO}(\lambda)$ is not necessarily optimal. However, the algorithm has been shown to work well in practice on a variety of problems [15, chap. 4], and is numerically stable. As we will see in Section 5.4, the algorithm works well for this problem too.

Computing the Subgradient One possible subgradient s_i of λ_i for $\mathbf{FLO}(\lambda)$ is the derivative of $\mathbf{FLO}(\lambda)$ with respect to λ_i:

$$s_i = 1 - o_i - \sum_j x_{ij}$$

For each data point i, the subgradient s_i is thus one less the number of times i is assigned—either as an outlier or to an exemplar. For an assignment that is feasible in the primal problem, we require each i to be assigned exactly once, i.e. $s_i = 0$. Similarly, if i is *under-assigned* then $s_i = 1$, and if i is *over-assigned* the $s_i < 0$. During the execution of the algorithm, the value of λ_i for the next iteration is $\lambda_i + \theta s_i$, and an under-assignment will increase λ_i, an over-assignment will decrease it, and a feasible assignment will leave it unchanged. Thus, the number of times that a point i is assigned when solving the relaxed problem signals the direction in which λ_i should be updated.

Solving the Relaxed Problem by Inspection The key step that allows the Lagrangian heuristic algorithm to find near-optimal solutions efficiently is that the relaxed problem may be solved by inspection in time polynomial in n. The goal of this step is to find, for a fixed assignment of λ, an assignment of the primal variables $\mathbf{x}$ and $\mathbf{o}$ that is feasible in the relaxed problem, and yields a value of the objective function h_d that is

near optimal.

The assignment is found by performing the following three steps:

1. Select exemplars;

2. Select outliers;

3. Assign remaining points to exemplars;

As we observed above, choosing the outliers and exemplar assignments separately may not give a good solution. This approach was used in Ott *et al.* [158], and we present a simple modification this strategy for outlier selection, that we show yields empirically better performance.

Exemplars are selected when the candidate point is paid for, let $\mathcal{E} = \{j \mid \mu_j < 0\}$ be the set of exemplars. Let

$$\chi_i = \lambda_i - \delta_i = \min\left[\lambda_i, \min_{j \in \mathcal{E}} d_{ij}\right],$$

i.e. χ_i is the lesser of the value of λ_i or the distance from point i to the nearest paid-for exemplar $j \in \mathcal{E}$. The outliers are selected as the ℓ points that obtain the ℓ largest values of χ: let $u_1, \ldots, u_n \to \mathcal{C}$ be an ordering of $\mathcal{C}$ such that $\chi_{u_i} \geq \chi_{u_{i'}} \forall i < i'$, and $\mathcal{O} = \{u_1, \ldots, u_\ell\}$ be the set of outliers. This implies that a point i will not be selected as an outlier if it is relatively close to an exemplar, regardless of the value of λ_i. Finally, a point i is connected to an exemplar $j \in \mathcal{E}$ when i is a non-outlier contributing to the establishment cost of j, and let $\mathcal{A} = \{(i, j) \mid i \in \mathcal{C} \setminus \mathcal{O}, j \in \mathcal{E}, \delta_{ij} > 0\}$ be the set of pairs defining the connections.

A high-level algorithm description for solving the relaxed problem by inspection is given in Algorithm 3.

The strategy for choosing outliers, exemplars and connections between points and exemplars may not appear obvious, but the following observation suggests that this strategy may be sound.

Algorithm 3 Solve $\mathbf{FLO}(\lambda)$ by inspection

```
procedure SOLVERELAXED(λ, d, f, ℓ)
    ℰ ← 𝒪
    for all j ∈ 𝒞 do                                   ▷ Determine exemplars
        μ_j ← f_j − Σ_i [max(λ_i − d_ij, 0)]
        if μ_j < 0 then
            ℰ ← ℰ ∪ {j}
        end if
    end for
    for all i ∈ 𝒞 do                                   ▷ Determine outliers
        δ_i ← λ_i − min_{j∈ℰ} d_ij
        χ_i ← λ_i − δ_i
    end for
    u_1, ..., u_n ← ordering of 𝒞 according to χ_i
    𝒪 ← {u_1, ..., u_ℓ}                                ▷ Outliers are ℓ points with largest χ_i
    𝒜 ← {(i, j) | i ∈ 𝒞 \ 𝒪, j ∈ ℰ, δ_ij ≥ 0}          ▷ Determine connections

    x ← 0, o ← 0
    for all (i, j) ∈ 𝒜 do                              ▷ Set connections
        x_ij ← 1
    end for
    for all j ∈ ℰ do                                   ▷ Set exemplars
        x_jj ← 1
    end for
    for all i ∈ 𝒪 do                                   ▷ Set outliers
        o_i ← 1
    end for

    Ẑ_p = Σ_j f_j x_jj + Σ_i Σ_j d_ij x_ij             ▷ Estimated primal objective value
    Ẑ_d = Ẑ_p + Σ_i (1 − o_i − Σ_j x_ij) λ_i           ▷ Estimated Lagrangian relaxed objective value
    return x, o, Ẑ_p, Ẑ_d
end procedure
```

Observation 5.1. For $\mathcal{E}$, $\mathcal{A}$ and $\mathcal{O}$ that is an optimising assignment for the **FLO** problem, the distance from any outlier $i \in \mathcal{O}$ to its nearest exemplar will not be less than the distance of any connected data point $i' \in \mathcal{C} \setminus \mathcal{O}$ to its nearest exemplar.

Clearly, this is a necessary condition for an optimal assignment—otherwise simply replacing i with i' as an outlier, and connecting i to its nearest exemplar would result in an other feasible assignment with a smaller primal objective function, contradicting the optimality assertion.

We claim that in situations where the LH algorithm finds a feasible assignment to the primal problem, then this condition will hold. For any connected data point i' we have, by construction, that

$$\delta'_i \geq 0 \iff \min_{j \in \mathcal{E}} d_{i'j} \leq \lambda'_i$$

And since, for i' and any outlier i

$$\begin{aligned} & \chi_{i'} \leq \chi_i \\ \iff & \min\left[\lambda_{i'}, \min_{j \in \mathcal{E}} d_{i'j}\right] \leq \min\left[\lambda_i, \min_{j \in \mathcal{E}} d_{ij}\right] \\ \iff & \min_{j \in \mathcal{E}} d_{i'j} \leq \min_{j \in \mathcal{E}} d_{ij} \end{aligned}$$

Furthermore, recall that u_ℓ is the ℓ-th outlier selected. We also have that, for all non-outliers $\min_{j \in \mathcal{E}} d_{i'j} < \lambda_{u_\ell}$ and thus λ_{u_ℓ} is an upper bound on the connection cost of all the connected data points.

5.2.2 Affinity Propagation

The affinity propagation algorithm is another technique for solving the discrete optimization problems and which approaches the problem from a completely different perspective. The AP algorithm models the problem as a type of probabilistic graphical model known as a *factor model*, which represents a Gibbs distribution of the likelihood of a variable assignment. Performing *maximum a posteriori* (MAP) inference

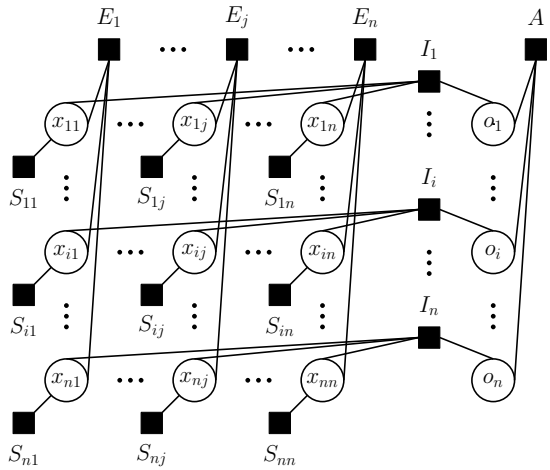


Figure 5.2: Graphical model for **FLO** problem. The variable nodes are drawn as circles and the factor nodes as filled squares. Each factor function is parameterised by one or more variables, and the corresponding variable nodes are incident to the factor node.

on this model will therefore determine the most likely variable assignment. The factor model variables are computed using the *max-sum* algorithm [126], which is simple and interpretable, however it also offers no optimality guarantee on the solution.

The **FLO** problem can be modelled as an affinity propagation problem using the following formulation, which extends the model for the uncapacitated facility location problem by Lazic *et al.* [131] to include outliers. A similar model was proposed by Ott *et al.* [158], however the model presented here requires fewer variables—$2n(n+1)$ compared to $2n(n+\ell)$—and uses different factor function definitions, which result in messages that are similar in form to the functions used in the LH algorithm.

The Affinity Propagation Formulation The factor model is a graph containing a *variable node* for each variable in the primal problem, and a *factor node* for each *factor function* on a set of the variables. Edges connect a factor node to variable nodes where the variables that are arguments of the associated factor function. The factors jointly describe a distribution over the possible assignments of the variables $\mathbf{x}$ and $\mathbf{o}$, as shown in Fig. 5.2. The MAP inference task is to find the most likely assignment over $\mathbf{x}$ and $\mathbf{o}$ that is a maximizer for the following energy function.

$$\mathbf{FLO}_{AP} \equiv \max_{\mathbf{x},\mathbf{o}} \sum_i \sum_j S_{ij}(x_{ij}) + \sum_j E_j(x_{:j}) + \sum_i I_i(x_{i:}, o_i) + A(\mathbf{o})$$

$$\text{subject to} \quad x_{ij}, o_i \in \{0,1\} \quad \forall i, j$$

where

$$S_{ij}(x_{ij}) = -\, d_{ij} x_{ij}$$

$$E_j(x_{:j}) = \begin{cases} 0 & \text{if } \sum_i x_{ij} = 0 \\ -f_j & \text{otherwise} \end{cases}$$

$$I_i(x_{i:}, o_i) = \begin{cases} 0 & \text{if } \sum_j x_{ij} + o_i = 1 \\ -\infty & \text{otherwise} \end{cases}$$

$$A(\mathbf{o}) = \begin{cases} 0 & \text{if } \sum_i o_i = \ell \\ -\infty & \text{otherwise} \end{cases}$$

Here, we use the notation $x_{i:}$ and $x_{:j}$ to represent the i-th row and j-th column vector of $\mathbf{x}$, respectively. The factor functions can be interpreted as imposing costs and enforcing the constraints defined in the primal model. Specifically, the factor functions can be interpreted as

- $S_{ij}(x_{ij})$ is the negative distance charge if point i is connected to exemplar j. If $x_{ij} = 1$ then this function ensures that the cost of connecting i to j is paid.
- $E_j(x_{:j})$ charges the facility cost f_j if any point is connected to exemplar j.
- $I_i(x_{i:}, o_i)$ enforces the constraint that point i can only be connected to exactly one exemplar, or be assigned as an outlier. Clearly, if the point is unassigned or multiply assigned, then the resulting value of the energy function will trivially be sub-optimal.

- $A(\mathbf{o})$ enforces the constraint that exactly ℓ points must be assigned as outliers.

The energy function can be maximized using the *max-sum* algorithm by Kschischang *et al.* [126] to obtain a MAP assignment of the variables. This is a message-passing algorithm and the particular messages are derived from the following general formulas for *factor-to-variable* and *variable-to-factor* messages [126].

$$\mu_{x\to f}(x') = \sum_{f_t\in\delta(x)\setminus f} \mu_{f_t\to x}(x')$$
$$\mu_{f\to x}(x') = \max_{x_i\in\delta(f)\setminus x} \Big[f(x_1,\ldots,x=x',\ldots,x_M) + \sum_{k:\, x_k\in\delta(f)\setminus x} \mu_{x_k\to f}(x_k)\Big]$$

The eight messages, shown if Fig. 5.3, were derived (see Appendix C for the full derivation of the messages).

$$\begin{aligned}
\rho_{ij} =& \eta_{ij} - d_{ij}.\\
\gamma_{ij} =& \alpha_{ij} - d_{ij}\\
\tau_i =& \chi_i\\
\sigma_i =& \omega_i\\
\alpha_{ij} =& \min\Big[0, -f_j + \sum_{k\neq i} \max\big[\rho_{kj}, 0\big]\Big]\\
\eta_{ij} =& \min\Big[\min_{k\neq j} -\gamma_{ik}, -\sigma_i\Big]\\
\chi_i =& -\max_j \gamma_{ij}\\
\omega_i =& -\begin{cases} \tau_{u_{\ell+1}} & \text{if } i\in\{u_1,\ldots,u_\ell\}\\ \tau_{u_\ell} & \text{otherwise}\end{cases}
\end{aligned}$$

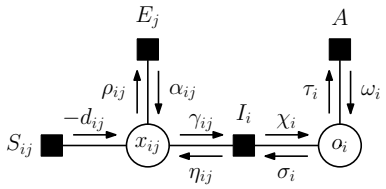


Figure 5.3: Sub-graph of the factor graph showing the messages that are passed along each edge. Each message conveys the sending node's *belief* about the value of the variable in the incident variable node.

Of the eight derived messages, four were eliminated by substitution, leaving the following factor-to-variable messages.

$$\alpha_{ij} = -\max\Big[f_j - \sum_{k \neq i} \max\big[\eta_{kj} - d_{kj}, 0\big], 0\Big] \tag{5.5}$$

$$\eta_{ij} = -\max\Big[\max_{k \neq j}\big[\alpha_{ik} - d_{ik}\big], \omega_i\Big] \tag{5.6}$$

$$\chi_i = -\max_j\Big[\alpha_{ij} - d_{ij}\Big] \tag{5.7}$$

$$\omega_i = -\begin{cases} \chi_{u_{\ell+1}} & \text{if } i \in \{u_1, \ldots, u_\ell\} \\ \chi_{u_\ell} & \text{otherwise} \end{cases} \tag{5.8}$$

For (5.8), let $u_1, \ldots, u_n \to \mathcal{C}$ be an ordering such that $\chi_i \geq \chi_k$ for all $u_i < u_k$.

The variables x_{ij} and o_i in the model all accept values of $\{0, 1\}$ and in general, each message contains a belief about the value of the corresponding variable. However, given that the variables are binary, it is sufficient to pass a single value that is the difference between the beliefs, and so for a message $\mu(x)$ on a variable $x \in \{0, 1\}$, let $\mu = \mu(1) - \mu(0)$. Thus, each message can be interpreted as $\mu > 0$ indicating a stronger belief that $x = 1$, and $\mu < 0$ that $x = 0$.

Interpretation of the Message Variables The messages in the affinity propagation model pass a value that quantifies the sending node's degree of belief about the local state of the model to a receiving node in its neighbourhood. We have eliminated all the variable-to-factor messages and thus only have factor-to-variable messages which can be interpreted as passing a degree of belief as to whether the variable node satisfies the

constraint encoded in the factor function of the sending node. The following messages are useful when interpreting the current beliefs about the MAP assignment of variables.

- α_{ij} is a measure of the current belief that point j should be marked as an exemplar, and that data point i be connected to it, and is a log-probability in the range $(-\infty, 0]$. Frey and Dueck [82] call this value the *availability* of exemplar j to accept connection from point i.

- ρ_{ij} $(= \eta_{ij} - d_{ij})$ is a measure of belief that, for each data point i, the exemplar that it connects to will be j. This value is a log-probability ratio and Frey and Dueck name this the *responsibility*, i.e. the belief that point i has that j should be its exemplar. For a given i, if $\rho_{ij} \ll 0$ for all j then i believes that it is an outlier.

- χ_i is the belief that data point i is one of the ℓ outliers, again measured as a log-probability ratio. In the spirit of Frey and Dueck's nomenclature, we observe that this measure is analogous to the availability, and name it the *separation* of point i.

The Affinity Propagation Algorithm The affinity propagation algorithm is also a simple iterative algorithm. The message beliefs are initialised to 0. The algorithm then sends all of the messages to update the beliefs in the nodes of the model, and the process is repeated until convergence—i.e. the beliefs of the nodes are consistent with their neighbours.

The algorithm is not guaranteed to converge, and oscillatory behaviour is possible in the message values. This can be mitigated by *dampening*, where an algorithm parameter $\theta \in [0, 1)$ is provided. Then at iteration t the message μ is computed as

$$\mu^t = \theta\mu' + (1 - \theta)\mu^{t-1}$$

where μ' is computed according to the message update rule for μ.

At this point, we can recover the assignment from the variable nodes by summing the values of the incoming labels to compute the belief and setting the variable to 1 when the belief is > 0. Specifically, let $x_{ij}^{(B)} = \alpha_{jj} + \eta_{jj} - d_{ij}$ be the belief that data point i is connected to exemplar j, and let $o_i^{(B)} = \chi_i + \omega_i$ be the belief that i is an outlier. Select the exemplars $\mathcal{E} \equiv \{j \mid x_{jj}^{(B)} > 0\}$ as the variables along the diagonal who believe themselves to be exemplars. The assignment of points as outliers is $\mathcal{O} \equiv \{i \mid o_i^{(B)} > 0\}$ and the assignment of points to exemplars is $\mathcal{A} \equiv \{(i,j) \mid i \in \mathcal{C} \setminus \mathcal{O}, j = \arg\max_{j'} x_{ij'}^{(B)}\}$.

The full algorithm is detailed in Algorithm 4.

5.2.3 Visualising the Auxiliary Variables

The Lagrangian heuristic and affinity propagation algorithms are similar in that they both have a set of *auxiliary variables* that contain the current "belief state" of the algorithm about the optimal assignment. These variables are the Lagrangian multipliers λ in the LH algorithm and the messages α,η,χ and ω in the AP algorithm. Moreover, both algorithms update their auxiliary variables at each iteration to refine the beliefs. The auxiliary variables can thus be captured during each iteration of the algorithm's execution to create a time-series that reflects how the belief state has evolved.

We claim that these time-series are useful data that can be readily visualised to provide insights about the execution of the algorithm, see Figure 5.1. We use this visualisation in the case study in Section 5.3 and the experiments in Section 5.4. Furthermore, additional auxiliary variables can be derived by combining the variables with the algorithm parameters, and we outline these next.

Each time-series is a $n \times t_{\text{max}}$ matrix, where t_{max} is the number of iterations that the algorithm ran for. We define the time series for the two algorithms here.

Lagrangian Heuristic The principal time-series captured for this algorithm are based on the Lagrangian multipliers λ_i, and the derived variables μ_j and δ_i, defined above. For each data point i at iteration $t \in \{1, \ldots, t_{\text{max}}\}$ the available time-series are as

Algorithm 4 The Affinity Propagation Algorithm

```
procedure AFFINITYPROPAGATION(d, f, ℓ, θ)
    α_ij, η_ij ← 0   ∀i, j                                  ▷ Initialise messages
    χ_i, ω_i ← 0   ∀i
    t ← 0

    while t < t_limit do
        Set α'_ij using (5.5)                               ▷ Update messages
        Set χ'_i using (5.7)
        Set ω'_i using (5.8)
        Set η'_ij using (5.6)

        α ← θα' + (1 − θ)α                                  ▷ Dampen messages
        η ← θη' + (1 − θ)η
        χ ← θχ' + (1 − θ)χ
        ω ← θω' + (1 − θ)ω
    end while

    x^B ← −d + α + η                                        ▷ Recover beliefs
    o^B ← χ + ω

    Get ordering u_1, ..., u_n of C according to o^B
    O ← {u_1, ..., u_ℓ}

    x, o ← 0                                                ▷ Initialise assignments
    for all i ∈ O do                                        ▷ Set outliers
        o_i ← 1
    end for
    for all j ∈ C do                                        ▷ Set exemplars
        if x^B_jj > 0 then
            x_jj ← 1
        end if
    end for
    for all i ∈ C \ O do                                    ▷ Connect i to most likely j
        j ← arg max_j' x^B_ij'
        x_ij ← 1
    end for
    Ẑ_p = Σ_j f_j x_jj + Σ_i Σ_j d_ij x_ij                  ▷ Primal objective value
    return x, o, Ẑ_p
end procedure
```

follows:

- $\lambda_{it}^{(S)}$ represents the amount data point i is willing to expend to connect to an exemplar in iteration t.
- $\delta_{it}^{(S)}$ is the surplus (deficit) budget that data point i is willing to expend to connect to the closest established exemplar in iteration t.
- $\mu_{jt}^{(S)}$ it the total budget deficit (surplus) that all data-points are willing to contribute to the establishment of an exemplar j.

Affinity Propagation Extracting time-series from the message variables is complicated by the fact that several of the relevant message types such as the availability α and the responsibility ρ are of dimensionality $n \times n$. However, it turns out that structure of the values in these matrices makes it possible to extract an n-dimensional vector from which a time series can be captured, that also gives some insight into the operation of the AP algorithm. Again, for each data point i at iteration t the captured time-series are:

- $\alpha_{jt}^{(S)}$ represents for point j the belief that j is an exemplar, and is thus captured in each iteration as the diagonal values of the matrix α, i.e. $\alpha_{jt}^{(S)} = \{\mathrm{Tr}\,\alpha\}_j$ at iteration t.
- $\rho_{it}^{(S)}$ is the belief that data point i is connected to *some* exemplar, and is computed in iteration t as $\rho_{it}^{(S)} = \max_j \rho_{ij}$.
- $\chi_{jt}^{(S)}$ is the belief at iteration t that i is an outlier.

5.3 Case Study: Improving the Lagrangian Heuristic Algorithm

In this case study, we use the auxiliary variable visualisation technique introduced in Subsection 5.2.3 to analyse the strategy for solving the relaxed problem by inspection

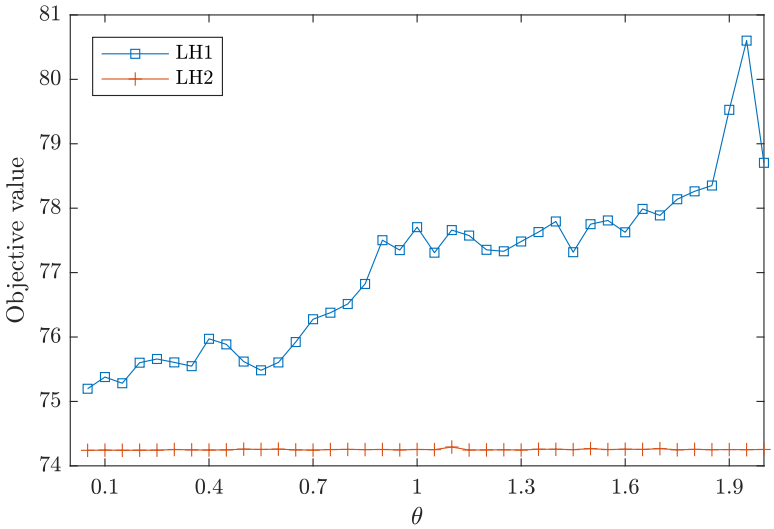
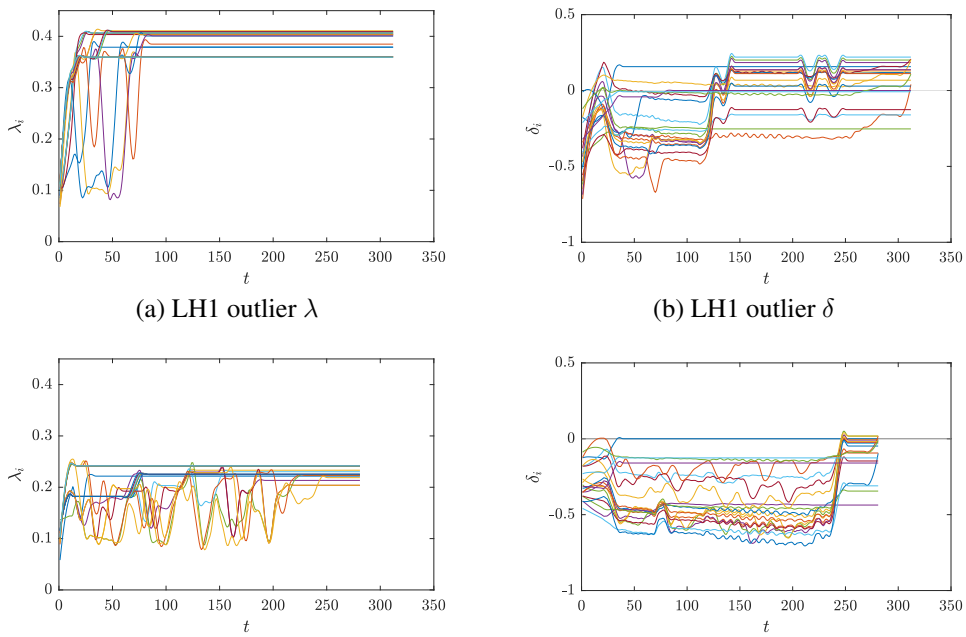


Figure 5.4: Objective function value obtained for LH1 and LH2 algorithms on identical inputs and configurations, for varying values of the algorithm parameter θ. The LH2 algorithm is numerically stable for variations in θ in comparison to the LH1 algorithm, and consistently obtains a lower objective function value. The configuration used is from the synthetic data experiments, see Table 5.1

from the LH algorithm. We identify a sub-optimal design choice made in the algorithm presented in Ott *et al.* [158] using the visualisation of the time-series of the auxiliary variables, then make a refinement to improve the solution, and verify that it yields improved solutions. To distinguish between the two Lagrangian heuristic algorithms, let LH1 algorithm be the algorithm presented in [158] and let LH2 be the modified algorithm presented in this paper.

Recall that the LH1 algorithm uses a simple strategy for selecting the data points to assign as outliers, namely choosing the ℓ points whose dual variable value λ_i is largest. Furthermore, any point that has been assigned as an outlier does not contribute to the establishment cost of an exemplar.

While this approach has the benefit of simplicity and often works well in practice, it is not difficult to see that there are situations where this approach may not choose the optimal assignment. First, the value of λ_i may be large as it is contributing to the establishment cost of an exemplar, and thus assigning the data point as an outlier would remove that contribution from the exemplar, potentially causing the cluster to



(a) LH1 outlier λ (b) LH1 outlier δ

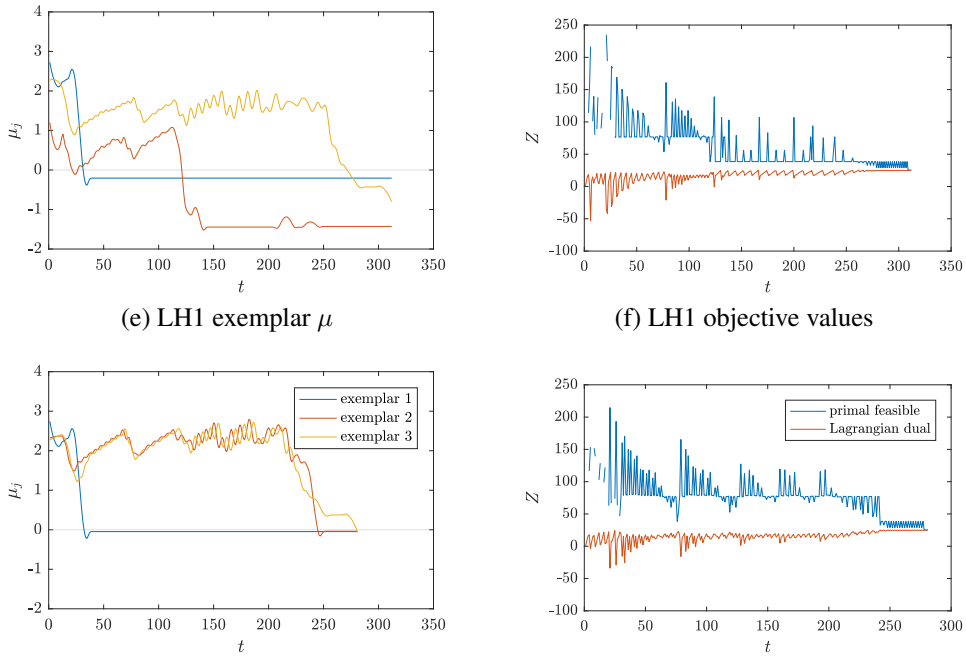
(c) LH2 outlier λ (d) LH2 outlier δ

Figure 5.5: Time-series visualisations of the auxiliary variables from the LH1 and LH2 algorithms executed on the same input. The time-series were smoothed using lowess smoothing with span 25. The $\lambda_{it}^{(S)}$ converge to larger values for the (a) LH1 algorithm than for the (c) LH2 algorithm. Furthermore, the time-series for LH2 satisfy the conditions in Proposition 5.1. (continues...)

have insufficient contribution to cover the establishment cost. Second, observe that once a data point is assigned as an outlier it cannot be assigned to an exemplar. This means that the point will not be *over-assigned*, and thus λ_i will not decrease in the subsequent iteration of the algorithm, via the mechanism of a negative subgradient.

We ran experiments to evaluate the performance of this algorithm for various values of the parameters. We observed that the obtained objective function value is sensitive to changes in the algorithm parameters, for example, the choice of value for θ—the initial step-length used for computing the subgradient step, see Figure 5.4. This implies that the algorithm is not numerically stable.

To investigate further, we investigated the operation of the algorithm on a synthetic input where θ was set to $\frac{1}{10}$ of the median distance in $\mathbf{d}$. The time-series visualisations are shown in Figure 5.5. It is clear from the plots that the LH1 algorithm selects



(e) LH1 exemplar μ (f) LH1 objective values

(g) LH2 exemplar μ (h) LH2 objective values

Figure 5.5: (contd.) Time-series visualisations of the auxiliary variables from the LH1 and LH2 algorithms executed on the same input.

the final outliers relatively early in its execution, and the value of λ of these outliers can be relatively high. The outliers are stable after $\approx$75 iterations, see Figure 5.5(a), however the exemplars—the distance from which the outliers are selected—are not finalised until the algorithm nears termination, after $\approx$300 iterations, see Figure 5.5(e). The algorithm is thus selecting the outliers before the clusters have been finalised, and points could thus be assigned as outliers even if an exemplar was established close by in a subsequent iteration. The execution of the algorithm in this case appears to be that the outliers are selected early, and then the regular facility location problem (without outliers) is solved for the remaining points. In contrast, the LH2 algorithm is constantly reevaluating the outliers whenever the exemplars change.

Furthermore, the final values of λ_i for the outliers are greater under the LH1 algorithm, compared to that the LH2 algorithm, see Figure 5.5(a) and (c). As we observed in Section 5.2.1, the values of λ_i for the outliers are an upper bound on the values for the non-outliers. This gap allows for a sub-optimal solution, which is the case, as the

LH1 algorithm obtains an objective function value of 25.16, and the LH2 algorithm obtains 24.11.

These visualisations facilitated the diagnosis of the issue with the LH1 algorithm. Moreover, when considering the conditions in Proposition 5.1 for an optimal assignment of values to λ, we can see that the LH2 algorithm finds an assignment that is closer to the optimal than the LH1 algorithm, see Figure 5.5(d) and (g). Furthermore, it is apparent that conditions in Proposition 5.1 appear to be satisfied in for the execution of the LH2 algorithm, as items 1 and 2 are evident in the time-series, and 3 holds by construction. And, as Figure 5.4 shows, the improved LH2 algorithm is numerically stable. This is in contrast to the LH1 algorithm, where the computed objective value is sensitive to the choice of algorithm parameter θ.

5.4 Experiments

The improved formulations of the LH2 and AP algorithms for the **FLO** problem were tested experimentally in order to evaluate the performance. We executed the algorithms on a synthetic dataset to determine the quality of the obtained solutions, and compared performance with the LH1 algorithm presented in Ott *et al.* [158], and the Gurobi mixed-integer linear program solver (the LR algorithm).

Furthermore, we present auxiliary variable visualisations for two real-world datasets based on spatial trajectories: the HURDAT2 hurricane dataset and a dataset of player trajectories from football matches. The purpose of these experiments is to evaluate the cluster visualisations as a technique for gaining insight into the structure of the underlying dataset.

5.4.1 Synthetic Data

We carried out experiments on a synthetic dataset to validate the proposed algorithm. The objective was to assess the runtime, obtained objective value and numeric stability of the algorithm for varying input sizes, number of outlier and number of instance-level

constraints.

The synthetic dataset was characterised by the following parameters. Let c be the number of *ground-truth* clusters, m be the number of samples per cluster, d be the dimensionality of the space, σ be the parameter for the standard deviation of the distribution the samples are drawn from, and let ℓ' be the ground-truth number of outliers. Let $n \equiv c \times m + \ell'$ be the number of generated samples. The output is a distance matrix $d \in \mathbb{R}^{n \times n}$.

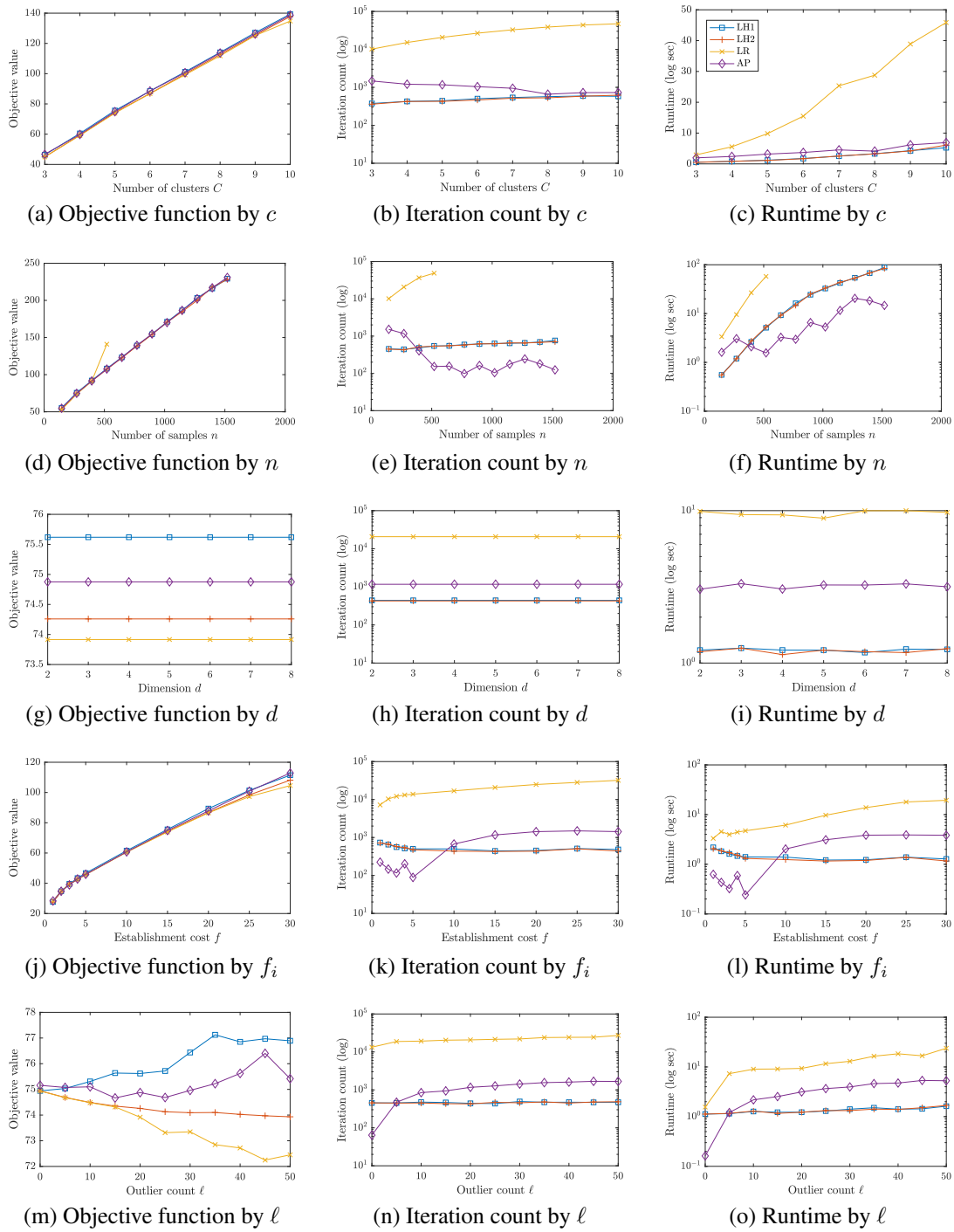
Define a surjective function $g : \{1, \ldots, n\} \to \{0, \ldots, c\}$, such that $g(j) = i$ indicates that the ground-truth cluster of data point p_j is $i \in \{1, \ldots, c\}$, or that p_j is outlier if $i = 0$. The function g is defined as

$$g(j) = \begin{cases} \lceil j/n \rceil & \text{if } j \leq cm \\ 0 & \text{if } j > cm \end{cases}. \tag{5.9}$$

The sample points $P = \{p_1, \ldots, p_n\}$ are generated from c ground-truth cluster centers. The centers $C^* = \{C_1, \ldots, C_c\}$ are sampled uniformly from the unit cube $[0, 1]^d$. For each cluster, define a variance matrix Σ_j by uniformly sampling each component from $[\sigma, 2\sigma]$. For each $i = 1, \ldots, cm$, point $p_i \in P$ is sampled from the Gaussian distribution $\mathcal{N}(C_{g(i)}, \Sigma_{g(i)}^{\frac{1}{2}})$, i.e. from the distribution of its associated cluster. The ℓ outlier points $\{p_{cm+1}, \ldots, p_n\}$ are sampled uniformly from $[0, 1]^d$.

The distance matrix $\mathbf{d}$ is computed from the set P of n samples, such that component d_{ij} of $\mathbf{d}$ denotes the distance between two samples i and j in the Euclidean distance metric. The matrix was then normalised by dividing all entries by the median of the non-diagonal entries, i.e. $\text{median}_{i,j\,:\,i \neq j} \left[d_{ij} \right]$.

The four algorithms were used to compute clusters for varying parameterisations, and statistics about the execution time and output clustering quality were captured. The parameters that configure the inputs and algorithms, and their default values, are summarised in Table 5.1.



(a) Objective function by c (b) Iteration count by c (c) Runtime by c

(d) Objective function by n (e) Iteration count by n (f) Runtime by n

(g) Objective function by d (h) Iteration count by d (i) Runtime by d

(j) Objective function by f_i (k) Iteration count by f_i (l) Runtime by f_i

(m) Objective function by ℓ (n) Iteration count by ℓ (o) Runtime by ℓ

Figure 5.6: Execution statistics for varying input parameters. All other parameters are set according to the defaults in Table 5.1.

Table 5.1: Configuration parameters used to generate the input datasets, and default values for algorithm parameters used in experiments.

Parameter	Description	Default Value
c	Number of ground truth cluster centers	3
m	Number of sample points per cluster center	50
d	Dimension of sample space	3
ℓ'	Number of outlier sample points	20
n	Total number of sample points	$m \times c + \ell'$
σ	Standard deviation of ground truth distribution	0.15
w	Number of trials for each configuration	30
$\mathbf{f}_i$	Facility establishment cost	15
ℓ	Number of outliers to select	20
θ_{LH}	Initial step-length used in Lagrangian heuristic algorithm	0.50
α	Step-size used in Lagrangian heuristic algorithm	0.99
θAP	Dampening parameter used in AP algorithm	0.80
T	Iteration limit for LH1, LH2 and AP algorithms	2000
V	Time limit for LR algorithm (sec)	600

5.4.1.1 Algorithm Execution

The performance of the algorithms were experimentally evaluated by running trials while varying the parameters to generate the input. Experiments were performed by varying the following parameters:

- Number of ground truth cluster centers c
- Number of samples n
- Sample point dimensionality d
- Establishment cost $\mathbf{f}_i$
- Number of outliers ℓ

The relative performance of the algorithms was compared using the following evaluation measures:

- Objective function value obtained
- Number of iterations

- Runtime in seconds

Each experimental configuration was used to generate 30 randomly sampled inputs, and the mean for each measure over the 30 trials for each configuration was computed. The experimental results are summarised in Figure 5.6. The results showed that all the algorithms obtained similar objective function values for each of the experiments, which the LH2 algorithm marginally outperforming the LH1 algorithm, as discussed in Section 5.3. All three heuristics had similar performance in terms of the execution time and number of iterations, and they significantly outperformed the LR algorithm. Figure 5.6 shows plots for the experiments when varying the parameters.

5.4.1.2 Cluster and Outlier Quality

In the next set of experiments, we evaluated the quality of the clustering and outlier detection. The output of an execution of any algorithm is a surjection for each sample to a predicted exemplar $\hat{g} : 1, \ldots, N \rightarrow 0, \ldots, N$, where an obtained value of 0 for $\hat{g}(i)$ denotes that i is an outlier. Given the ground truth surjection g defined in Equation (5.9), the quality of the obtained clusters and outliers are evaluated using the following metrics.

Homogeneity, Completeness and v-measure. These metrics, presented in Rosenberg and Hirschberg [169], are conditional-entropy based metrics for cluster quality. Given a contingency table $\mathbf{A} = \{a_{cj}\}$, where $a_{cj} = \sum_i \mathbb{I}[g(i) = c \wedge \hat{g}(i) = j]$, we define the conditional entropy and entropy as follows. Recall that C^* is the set of ground-truth clusters and $\mathcal{E}$ the set of predicted exemplars.

$$H(C|K) = -\sum_{c \in C^*} \sum_{j \in \mathcal{E}} \frac{a_{cj}}{N} \log_2 \frac{a_{cj}}{\sum_{c' \in C^*} a_{c'j}}$$

$$H(C) = -\sum_{c \in C^*} \frac{\sum_{j \in \mathcal{E}} a_{cj}}{N} \log_2 \frac{\sum_{j \in \mathcal{E}} a_{cj}}{N}$$

Homogeneity quantifies the extent to which the computed clusters are pure, and

completeness is the extent to which a ground truth class is recovered in a single cluster. They are defined as:

$$h = \begin{cases} 1 & \text{, if } H(C|K) = 0 \\ 1 - \frac{H(C|K)}{H(C)} & \text{, otherwise.} \end{cases}$$

$$c = \begin{cases} 1 & \text{, if } H(K|C) = 0 \\ 1 - \frac{H(K|C)}{H(K)} & \text{, otherwise} \end{cases}$$

The v-measure is the harmonic mean of h and c:

$$v = \frac{hc}{h + c}$$

Jaccard Measure The Jaccard coefficient is used to determine the extent to which outliers are recovered, and is defined as:

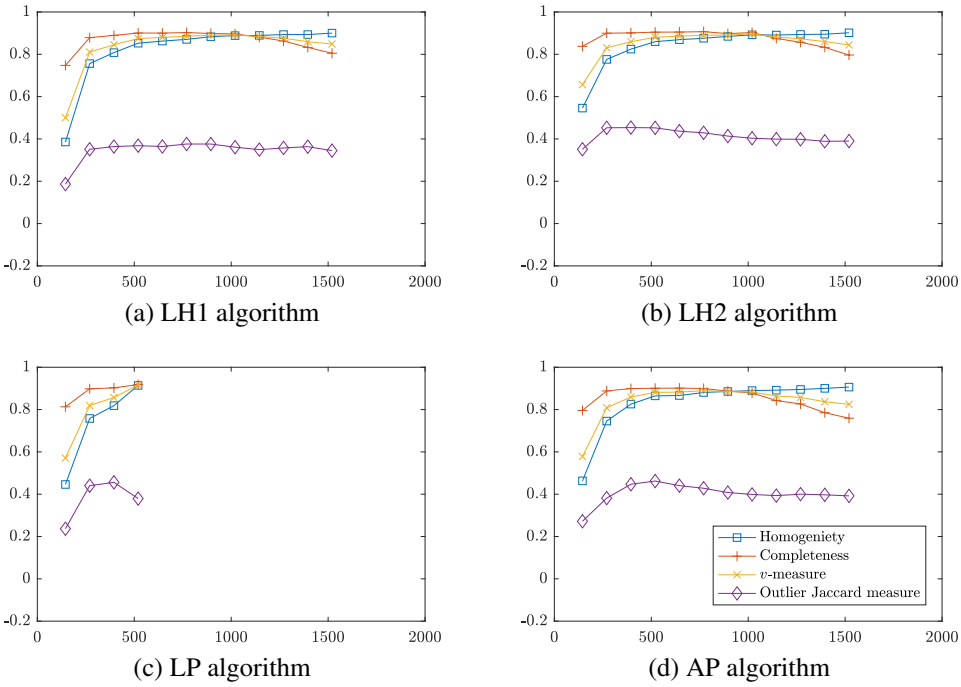
$$J(O, \hat{O}) = \frac{|O \cap \hat{O}|}{|O \cup \hat{O}|}$$

where $O = \{i \mid g(i) = 0\}$ is the set of ground-truth outliers, and $\hat{O} = \{i \mid \hat{g}(i) = 0\}$ is the set of predicted outliers.

All the measures h,c,v and J yield a result in the range $[0, 1]$, and 1 is optimal. The quality measures obtained by all algorithms showed that the performance in recovering the ground-truth assignments were very similar. The LH1 algorithm underperforms the other algorithms at recovering the ground-truth outliers, and again this is consistent with the analysis of the deficiency in the design of this algorithm, see Figure 5.7.

5.4.2 Dual Variable Conditions

In Section 5.2.1 we presented an interpretation of the dual variables λ in the LH algorithms that implied that an optimal solution of the **FLO** problem would satisfy the three properties in Proposition 5.1. During the experiments, the following statistics



(a) LH1 algorithm (b) LH2 algorithm

(c) LP algorithm (d) AP algorithm

Figure 5.7: Cluster quality results for algorithms on differing input sizes.

were captured about dual variables at the termination of the algorithm in each trial. The objective was to determine the extent to which the LH1 and LH2 algorithms met the conditions in Proposition 5.1.

Mean positive outliers computes the mean number of outliers that have a positive value of δ_i, i.e. determine the number of violations of condition 1:

$$\frac{1}{\ell}\sum_{i\in\mathcal{O}}\mathbb{I}[\delta_i > 0]$$

Mean close outliers finds the mean number of outliers that are closer to an exemplar than the connected point that is furthest from its exemplar, i.e. satisfy the necessary optimality condition in Observation 5.1:

$$\frac{1}{\ell}\sum_{i\in\mathcal{O}}\mathbb{I}[\min_{j\in\mathcal{E}} d_{ij} < \max_{i'\in\mathcal{C}\setminus\mathcal{O}}\min_{j\in\mathcal{E}} d_{i'j}]$$

Max. μ_j divergence is the maximum divergence of μ_j from 0 as a fraction of f_j, and

Table 5.2: Summary results for all experiments of the statistics described in Subsection 5.4.2. The results are the mean value over all 30 trials for each algorithm and experiment type, and for all statistics, a value of 0 is optimal.

Algorithm	Exp. Type	Mean Pos. Outlier δ_i	Mean Close Outliers	Max μ_j Div.
LH1	α	0.9867	0.6775	0.6106
LH1	θ	0.9795	0.7200	1.2658
LH1	ℓ	0.9869	0.6309	0.8013
LH1	c	0.9862	0.6725	0.6261
LH1	m	0.9745	0.5748	0.5339
LH1	f	0.9710	0.6011	1.0242
LH1	d	0.9883	0.5900	0.6036
LH2	α	0.5748	0	0.0127
LH2	θ	0.5830	0	0.0124
LH2	ℓ	0.5713	0	0.0128
LH2	c	0.4840	0	0.0135
LH2	m	0.3078	0	0.0166
LH2	f	0.3962	0	0.0237
LH2	d	0.5967	0	0.0121

thus tests condition 2:

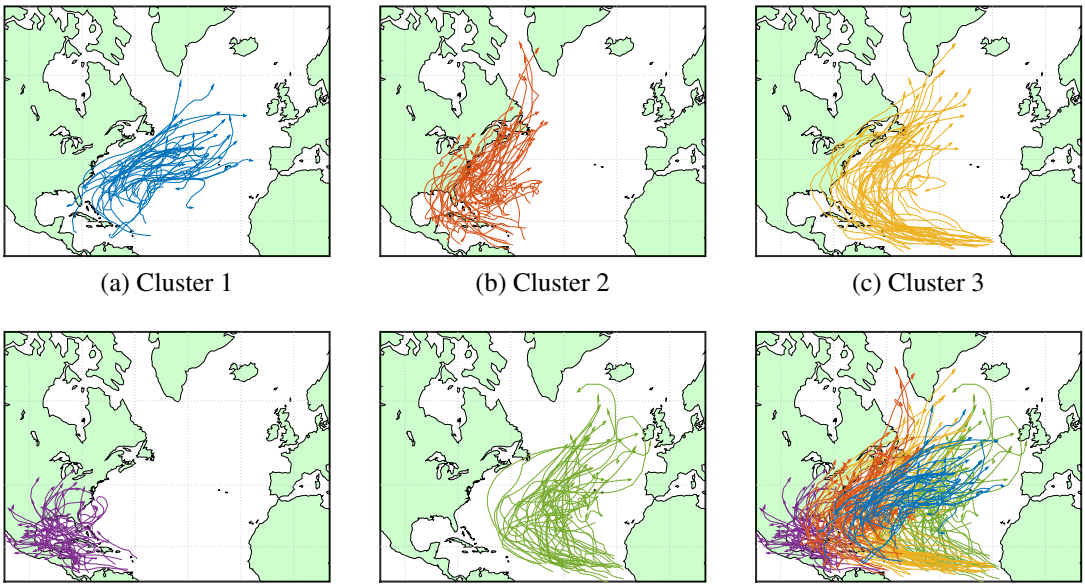
$$\max_{j \in \mathcal{E}} \frac{|\mu_j|}{f_j}$$

Here $\mathbb{I}[\cdot]$ is the indicator function on the condition in the brackets. The statistics were computed for each trial and then aggregated to compute the mean over all trials by algorithm and experiment type. The results are in Table 5.2.

Clearly, the LH2 algorithm outperforms the LH1 algorithm under these measures. In particular, the LH2 algorithm always finds a solution that satisfies the condition in Observation 5.1—and derived from condition 2—that μ_j is close to 0. However, in the case of the mean number of positive outliers, it appears that condition 1 is not met in many situations. This is likely a consequence of the fact that in both algorithms, the points that are assigned as outliers are not considered for connection to exemplars and therefore cannot be over-assigned, which would subsequently reduce the value of λ for the outlier.

5.4.3 Hurricane Trajectory Data

In this experiment, we used the auxiliary variable visualisation techniques for the LH2 algorithm on a clustering of hurricane trajectories, extracted from the HURDAT2 dataset [130]. The dataset contains trajectories of North Atlantic tropical storms from



(a) Cluster 1 (b) Cluster 2 (c) Cluster 3

(d) Cluster 4 (e) Cluster 5 (f) All Clusters

Figure 5.8: Computed clustering of hurricane trajectories with $\ell = 60$ outliers, with the exemplar for each cluster drawn with a thick line. The clusters are coherent in respect to their location and shape. The trajectories assigned as outliers are shown in Figure 5.10(b).

1851–2016. The purpose of these experiments was to explore the utility of the auxiliary variable visualisation on a real dataset, and some key observations are described in the following paragraphs.

The discrete Fréchet distance [78] was computed between each pair of trajectories using the dynamic programming algorithm by Eiter and Mannila [68] to obtain the distance matrix $\mathbf{d}$. The matrix $\mathbf{d}$ was then normalised by dividing all entries by the median of the non-diagonal entries, i.e. $\mathrm{median}_{i,j:\, i \neq j}\left[d_{ij}\right]$. The dataset was filtered to contain the 280 tropical storms since 1970 that were of hurricane-strength, and the LH2 algorithm was run with the following parameter values: $\mathbf{f} = 5$; $\ell = 60$; $\theta = 1/20$; and $\alpha = 0.99$. The algorithm converged in 656 iterations after approximately 2.5 s and the resulting clustering had 5 exemplars, see Figure 5.8.

Trajectory clustering can be visualised in the original space, as in Figure 5.8(f), however the trajectories often intersect with themselves and other trajectories, and thus can be occluded. Moreover, important features of the trajectories such as the length

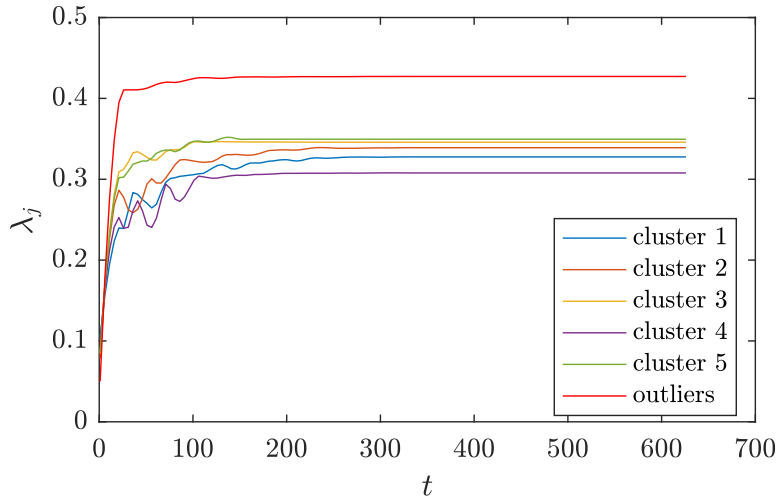
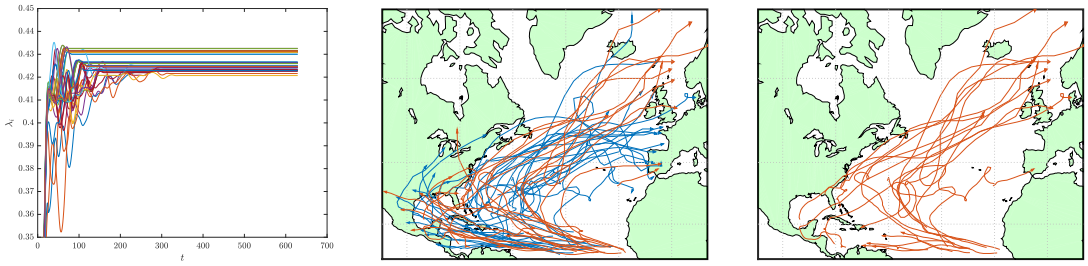


Figure 5.9: Time-series plot of the mean value of $\lambda_{it}^{(S)}$ from the hurricane trajectories for each computed cluster, and for the outliers. Two distinct features are observable: the value that each cluster mean converges to, and the number of iterations required to converge, and these provide an indication of the *density* and *coherence* of the clusters, respectively.

and direction are also difficult to discern. The time-series visualisation of the dual variables provides an alternative visualisation of the clustering, and an analysis on several observations of this is provided below.

Cluster formation In the first visualisation, we examine the time-series of the mean value of $\lambda_{it}^{(S)}$ for each cluster and for the outliers, shown in Figure 5.9. There are two features of this time-series that are readily apparent: the value that $\lambda_{it}^{(S)}$ converges to for each cluster, and the number of iterations that each value takes to converge.

The mean value that each cluster converges to is the mean of the total connection and contribution cost that each data point is willing to expend to connect to its exemplar. If the points in a cluster are generally relatively close to the exemplar, then the average connection cost will be relatively small; and if there are a relatively large number of points in a cluster, then the average contribution to the connection cost will be small. Thus, a small mean $\lambda_{it}^{(S)}$ for a cluster suggests that the cluster has many members and the members are relatively near the exemplar. The value is thus a measure of the *density* of the cluster. Comparing the mean converged values of $\lambda_{it}^{(S)}$ in Figure 5.9



(a) Outlier λ (b) Outliers (c) Outliers

Figure 5.10: Visualisation of HURDAT2 outliers from clustering where $\ell = 60$. The time-series plot in (a) shows a bifurcation in values, which suggests a natural parameter value for the number of outliers ℓ. The outliers are plotted in (b), with the upper cohort in the bifurcation coloured red, and (c) shows the outliers from a clustering where $\ell = 24$.

with the trajectories in Figure 5.8(f) there appears to be some correlation, with clusters 3 and 5 being the clusters covering the largest geographical area, and cluster 4 being a relatively tight cluster.

The second feature that can be determined from the time-series plot is number of iterations taken for the cluster to converge. Clusters that converge relatively quickly tend to be well separated from the other clusters, and also from the outliers, and thus this feature is a measure of the *coherence* of the cluster. Again, comparing the time-series plot and trajectory map in Figure 5.8, it is clusters 3 and 5 that converge earliest. Geographically, the trajectories in these clusters are coherent in their shape and length, describing a crescent-shaped path through the Atlantic Ocean. The remaining clusters comprise trajectories that tend to be shorter and more variable in shape.

"Natural" outlier parameter Next we consider the visualisation of the time-series for each outlier in Figure 5.10(a). The time series shows a distinct bifurcation between the obtained values of $\lambda_{it}^{(S)}$, suggesting that there may be some significant difference between the outliers. The outliers were plotted in Figure 5.10(b), with the cohort that obtained the larger values of $\lambda_{it}^{(S)}$ in red. This cohort contains 24 trajectories and appears to contain two distinct groups: trajectories that continue far into the North Atlantic, and those that head in a westerly direction towards and over Mexico.

The bifurcation in the values of $\lambda_{it}^{(S)}$ suggests that there may be "natural" outlier parameter of 24, so the algorithm was re-run using the same input and the same parameter values other than $\ell = 24$. The resulting outliers are plotted in Figure 5.10(c), and clearly captures many of the same trajectories, in particular the trajectories that extend into the North Atlantic towards Iceland.

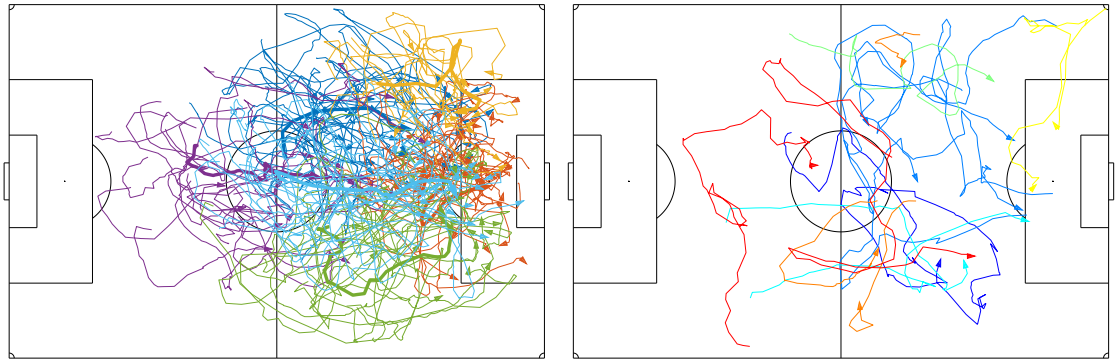
5.4.4 Football Trajectory Data

In the second experiment on a real-world dataset, the auxiliary variables from the affinity propagation algorithm were used to visualise the clusters. The input was a set of sub-trajectories extracted from the experimental dataset football dataset described in Section 1.2, and generated according to the following process. The raw trajectories were partitioned into sub-trajectories where a single team was in possession during an active phase of the match. A subset of these trajectories was selected according to the criteria that they were generated by a midfield or attacking player from the home team—Arsenal—and where the final event in the possession was a shot at goal by a player. This subset of 217 trajectories was used as the input, and the discrete Fréchet distance was computed for each pair of trajectories, to obtain the distance matrix $\mathbf{d}$, which was then normalised by dividing all entries by the median of the non-diagonal entries.

The following parameter values were used: $\mathbf{f} = 5$; $\ell = 20$; and $\theta = 0.8$. The algorithm converged in 60 iterations after approximately $0.3\,\mathrm{s}$ and the resulting clustering had 6 clusters, see Figure 5.11.

Visualising the trajectories in the original (planar) space provides some information about the spatial arrangement of the clusters and the outliers, however again it is difficult to determine properties such as the density and coherence of the clusters, see Figure 5.11(a).

Visualising the clusters and outliers using the affinity propagation message time-series provides an alternate view that provides information about the cluster formation



(a) Exemplars (b) Outliers

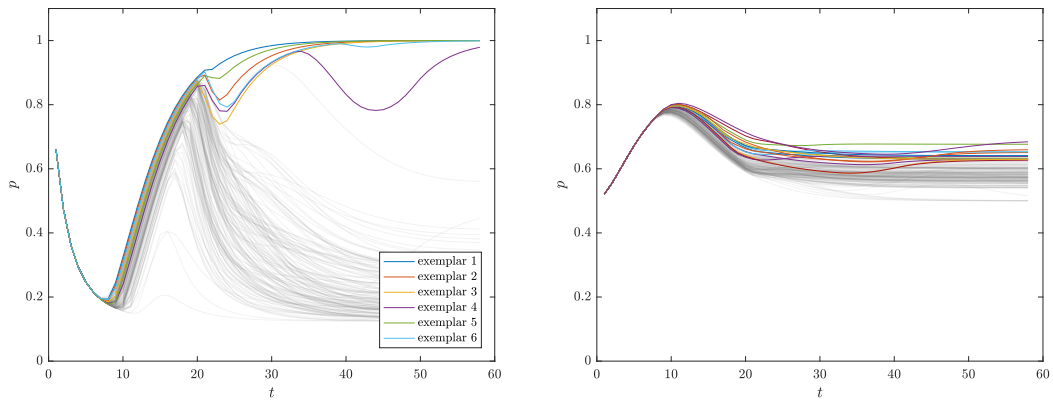
Figure 5.11: (a) Clusters and (b) outliers of football trajectories computed by the AP algorithm. The colours of the clusters, shown in (a), correspond to the legend in Figure 5.12(a), and the exemplars—drawn with thick lines—appear to conform to the positions of various roles. Of the 15 outliers shown in (b), all but two occur simultaneously with at least on other outlier, and simultaneous trajectories have the same colour.

and outlier detection. The message values are the log probability ratios of the assignment implied by indices of the message value [82]. The probabilities for each message were recovered from the ratios, and displayed in the time-series in Figure 5.12.

The time-series visualisations of the AP algorithm show that the formation of clusters and identification of outliers is a comparatively smoother process than that for the LH2 algorithm. However, the time-series are also interpretable, and provide useful information about the inputs. In the initial phase, the algorithm has insufficient information to determine of *any* of the data points are exemplars, and the probabilities uniformly decrease. The probability that all points are outliers similarly increases.

After ≈ 10 iterations, the exemplars begin to emerge and the probability of the outliers stabilises, during which time the marginal decisions on which points are outliers are made. Apparently, the impact is more pronounced for clusters 2, 3, 4 and 6, and this effect can also be seen in Figure 5.11, where the regions of these clusters coincide with the locations of the selected outliers.

The dip in the series for exemplar 4 at ≈ 45 iterations suggests that there were two candidates for exemplar of the cluster, and the choice of exemplar had to be resolved.



(a) Exemplars (b) Outliers

Figure 5.12: Time-series visualisations of (a) $\alpha_j^{(S)}$ for the exemplars, and (b) $\chi_i^{(S)}$ for the outliers. The y-values are the probability of that the sample point is an exemplar or outlier, respectively. The grey lines in (a) are the time-series of all the non-exemplars, and in (b) are the time-series of the non-outliers. The time series in (a) that converges to ≈ 0.56 is assigned to exemplar 4, which explains the dip in the exemplar's time-series.

This is indeed the case, where the time-series for the non-exemplar that converges to ≈ 0.56 is assigned to exemplar 4, and is clearly visible in Figure 5.12(a).

5.5 Conclusion

The joint task of clustering and outlier detection are a common requirement in many applications, and there are several efficient algorithms for this problem. However, in cases of high- or infinite-dimensional input, it is often challenging to visualise the obtained clustering to validate and interpret its utility. In this chapter we introduce the technique of monitoring the auxiliary variables computed by two common clustering heuristics–the Lagrangian heuristic and affinity propagation algorithms—and then visualising the resulting time series. This approach gives an implicitly two-dimensional view of the data, and contains useful information about the execution of the algorithm and also about the inherent clusters and outliers in the input data.

Using these visualisations, we identified a sub-optimal design choice in the Lagrangian heuristic algorithm for the joint clustering and outlier detection problem by

Ott *et al.* [158], and designed an improved version of this algorithm, which was experimentally shown to outperform the existing version. Furthermore, we used a novel interpretation of the Lagrangian dual variables to determine a set of conditions for an optimal solution, and found that the new algorithm satisfies these conditions. This suggest that the approach is an effective tool for analysis of algorithm design and execution.

We experimentally evaluated the approach on two real datasets to cluster trajectories using the LH2 and AP algorithms under the discrete Fréchet distance. The resulting visualisations yielded insights that were interpretable and could be validated in planar visualisations of the trajectory clusters. The difficulty in interpreting these "raw" planar real datasets is apparent, and we believe that the time-series visualisation presented in this paper is a useful complement and aid for understanding.

Chapter 6

Trajectory Clustering with Bounded Complexity Exemplars

In Chapter 5 we presented two heuristic algorithms for the joint task of clustering and outlier detection. The problem was modelled as the facility location problem, which is a form of *exemplar-based* clustering, where for each cluster, one of the inputs is selected as an exemplar that is in some sense a representative of the cluster. However, if the inputs are *noisy*, then the exemplars will be noisy too. For example, the exemplars from the clustering of football trajectory data described in Subsection 5.4.4 are not smooth, see Figure 6.1. The noise may result from measurement error, or from local deviations in direction of the traced object. When clustering trajectories under the Fréchet distance, the noise manifests as intervals where the trajectory is not smooth, and, in aggregate, the noise may obscure the high-level "shape" of the curve. It would thus be preferable to find an exemplar for each cluster that is representative of the cluster, but is also free from such noise, and this is the problem considered in this chapter.

We investigate the problem of clustering an input set of trajectories under the Fréchet distance, such that the input set is partitioned into k clusters, and for each cluster we compute a *smooth* exemplar. This is achieved by bounding the number of vertices in the exemplar by the value $\ell \ll n$, where n is the maximum number

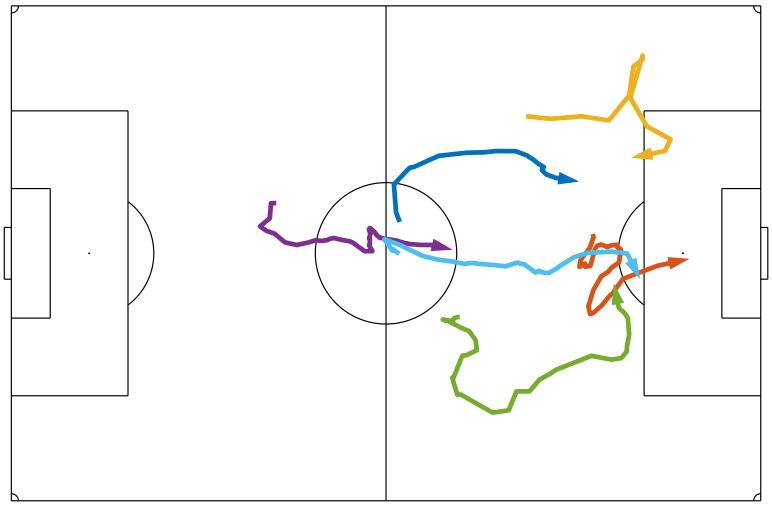


Figure 6.1: Cluster exemplars obtained in experiments described in Subsection 5.4.4. The exemplar trajectories are noisy, and this may obscure the important information about the clusters that they represent.

of vertices in any of the input trajectories. The objective of the clustering is to find the partition and set of exemplars, such that the maximum Fréchet distance between a trajectory in a cluster and its corresponding exemplar is minimised. This is a (k, ℓ)-clustering, and we present a family of approximation algorithms for which the cost of the optimal clustering is bounded to a multiplicative factor of the optimal cost, and where tighter approximation bounds result in slower runtime. We show that a bounded approximation for this problem can be achieved by first simplifying the input paths and then clustering the simplifications.

This chapter is structured as follows. Section 6.1 describes related work to this problem. In Section 6.2 we define the notation and terminology used in the sequel of the chapter, and formally describe the problems for which we present algorithms. An approximation algorithm for ℓ-simplification is given in Section 6.3 and a family of approximation algorithms for (k, ℓ)-clustering in Section 6.4. The chapter concludes in Section 6.5 with a discussion of open problems and other avenues of research.

6.1 Related Work

Effective algorithms for trajectory clustering have been researched in a number of different fields. Gaffney and Smyth used an algorithm based on a generative mixture model to cluster trajectories [85], [86]. Vlachos *et al.* [190] approached the problem by defining a non-metric similarity function based on the longest common subsequences appearing in the inputs, and suggest that this approach is robust to noise. Clustering has been used to detect *commuting patterns*—similar parts of a single trajectory—and Buchin *et al.* [33] gives several approximation algorithms for this problem. Driemel *et al.* [65] studies the (k, ℓ)-CENTER problem that we consider in this chapter, and present an $O(mn)$ time algorithm that computes a $(1 + \varepsilon)$-approximation on 1-dimensional (time-series) paths. It does not appear that this approach will extend to paths in 2 or more dimensions.

The approach presented here considers the sub-problems of trajectory simplification and clustering in general metric spaces. An early contribution to address the problem of trajectory simplification was the heuristic algorithm by Douglas and Peucker [62], and Imai and Iri presented several approximation algorithms that apply to trajectories with various properties [113], [114]. For simplification with respect to the Fréchet distance, Guibas *et al.* [98] presents an algorithm that for a given path P and error bound ε, computes a minimal-link simplification. Agarwal *et al.* [3] considered the problem of simplifying a curve where the vertices of the simplification are a subset of the vertices of the input, and presented a near-linear time approximation algorithms to compute an approximation for a given error ε whose complexity is at most that of an optimal curve of error $\varepsilon/2$. A practical $(1 + \varepsilon)$-approximation for computing the Fréchet distance for a family of realistic curves known as c-packed curves is given in Driemel *et al.* [64], where the distance was computed on simplified curves.

Clustering in general metric spaces is a classical problem in computer science, and there are many algorithms for the various formulation of the problem. Constant-factor polynomial-time approximation algorithms for clustering have been presented for the k-center [108], k-median and k-means [48], minimum diameter [91] and the

facility location [117] formulations. The best known approximation bound for the k-center clustering is 2, with the algorithm by Hochbaum and Shmoys [108] achieving this bound. The k-center clustering problem is known to be NP-hard to approximate within a factor of $(2 - \alpha)$ for any positive constant α [112].

6.2 Preliminaries

We wish to cluster trajectories under the Fréchet distance, see Section 1.2. We can view a trajectory as a polygonal path, where each location point is labelled with a time-stamp. For ease of exposition, we will use the following terminology. Denote $P = \langle p_1, \ldots, p_n \rangle$ as a path of length n such that each vertex p_i is a point in $\mathbb{R}^d$. Recall from Subsection 1.2.2 that a trajectory is denoted by $\tau = \langle (s_1, x_1, y_1), \ldots, (s_n, x_n, y_n) \rangle$. Clearly, there is an equivalent path P, where $p_i = (x_i, y_i) \in \mathbb{R}^2$ for all $p_i \in P$ and $(s_i, x_i, y_i) \in \tau$. The *complexity* of a polygonal path P is defined as the number of vertices in the path, and is denoted $|P|$.

Let $P[p, q]$ denote a minimum *sub-path* of P whose end-points are p and q. Furthermore, we use the following to denote an ordering of points along a path. Given a path P and two points $p, q \in P$, if $p \prec q$ then p is closer to the start of P than q. Formally, let p_1 be the first vertex of P, then $p \prec q \iff P[p_1, p] \subsetneq P[p_1, q]$. Similarly, denote $p \preceq q \iff P[p_1, p] \subseteq P[p_1, q]$, and $\succ$ and $\succeq$ are similarly defined.

We use (p, q) to denote an edge in a path P between vertices p and q. The path P is embedded in d-dimensional Euclidean space and thus $(p, q) \equiv \overline{pq}$. We use (p, q) to denote a line segment when doing so is unambiguous. We also define a *disk* as a d-dimensional hypersphere of a given radius r.

The algorithm described in Section 6.3 makes use of tangent lines to a disk that intersect at a point, see Figure 6.2. Given a disk D and a point $p \notin D$, there exist two tangent lines to D that intersect at p. Assume that p lies to the left of D, and perform a radial scan centered at p and starting in the direction of the positive y-axis. Let t_{cw} be the first tangent line encountered on the scan in the clockwise direction, and let t_{ccw} be

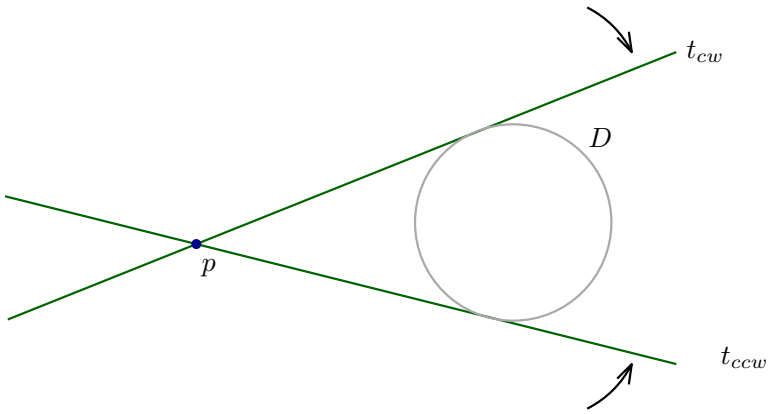


Figure 6.2: The tangent lines t_{cw} and t_{ccw} of disk D according to point p. The lines are named according to the tangent point that is first encountered in a radial sweep about p and whose origin does not intersect with D.

the tangent line encountered on the scan in the counter-clockwise direction.

6.2.1 Simplification

We require the following definition.

Definition 6.1 (ℓ-simplification)**.** For a polygonal path P, an *ℓ-simplification* of P is a polygonal path S of complexity ℓ, where there exists a homeomorphism (bicontinuous function) $\mu\colon P \to S$ that maps every point $p \in P$ to a point $\mu(p) \in S$, and the start of P is mapped to the start of S.

Let the *error* of the simplification be the Fréchet distance between P and S, hence $\mathrm{error}_\ell(P) = d_F(P, S)$. Given a polygonal path P, we wish to know if there exists an ℓ-simplification of minimal error:

PROBLEM ℓ-SIMPLIFICATION WITH MINIMAL ε

Instance: A polygonal path P, and an integer ℓ.

Question: Find the ℓ-simplification S of P, such that $\mathrm{error}_\ell(P)$ is minimised.

Let $\mathrm{error}^*_\ell(P)$ be the optimal error obtained from a solution to the ℓ-SIMPLIFICATION problem with respect to P.

We will use the following definition.

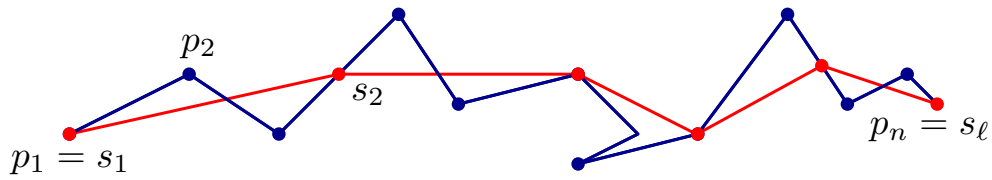


Figure 6.3: Example of a path-restricted ℓ-simplification $S = \langle s_1, s_2, \ldots, s_\ell \rangle$ of a polygonal path $P = \langle p_1, p_2, \ldots, p_n \rangle$. Observe that each vertex s_i of the simplification lies on P and that S is smoother than P.

Definition 6.2 (Path-restricted)**.** Given a polygonal path P, a simplification $S = \langle s_1, \ldots, s_\ell \rangle$ is *path-restricted* if $s_i \in P$ for all vertices $s_i \in S$.

In other words, all the vertices of a path-restricted simplification S of path P lie on P, see Figure 6.3.

6.2.2 Clustering

We use the following to define the clustering problem. Given a set of input paths $\mathcal{P} = \{P_1, \ldots, P_m\}$, and an integer k, let $\pi_k : \mathcal{P} \to \mathcal{C}$ define a *partition* of $\mathcal{P}$ to a set of *cluster centers* $\mathcal{C} = \{C_1, \ldots, C_k\}$, where π_k is a surjection, i.e. every $P \in \mathcal{P}$ maps to exactly one $C \in \mathcal{C}$, and for each C there is at least one P that maps to it.

We use two clustering problems that differ in the domain of the centers $\mathcal{C}$. In the first we wish to find a *k-center* clustering, where the center of each cluster is one of the input paths. For an input $\mathcal{P}$ and obtained clustering $\mathcal{C}$, let $\mathrm{cost}_k(\mathcal{P}) = \max_{P_i \in \mathcal{P}} d_F(P_i, \pi_k(P_i))$ be the *cost* of the clustering. Finding the set of centers $\mathcal{C}$ and surjection π_k that minimises the cost is the following problem:

PROBLEM k-CENTER

Instance: A set of polygonal paths $\mathcal{P} = \{P_1, \ldots, P_m\}$ each of complexity at most n and a positive integer k.

Question: Find the set of k exemplar paths $\mathcal{C} \subseteq \mathcal{P}$ and the surjection $\pi_k^* \colon \mathcal{P} \to \mathcal{C}$, such that the maximum Fréchet distance between a path $P \in \mathcal{P}$ and its cluster center $\pi_k^*(P)$ is minimized.

Let $\mathrm{cost}_k^*(\mathcal{P})$ be the optimal cost of the solution to the k-CENTER problem.

The main results in this chapter are for the (k, ℓ)-clustering task, formalised here.

Definition 6.3 ((k, ℓ)-clustering)**.** Given an input set $\mathcal{P}$, and positive integers k and ℓ, a (k, ℓ)-*clustering* is a partition of $\mathcal{P}$ defined by a set of k *cluster centers* $\mathcal{C}$ and a surjection $\pi_{(k,\ell)} \colon \mathcal{P} \to \mathcal{C}$, and where each $C_j \in \mathcal{C}$ is a polygonal path of complexity ℓ.

Finally, for an input $\mathcal{P}$ and obtained surjection $\pi_{(k,\ell)}$, define the *cost* of the clustering as the maximum Fréchet distance between a path $P_i \in \mathcal{P}$ and its assigned center $C_j \in \mathcal{C}$, i.e. $\mathrm{cost}_{(k,\ell)}(\mathcal{P}) = \max_{P_i \in \mathcal{P}} d_F(P_i, \pi_{(k,\ell)}(P_i))$.

The problem of finding the minimum-cost (k, ℓ)-clustering is defined as:

PROBLEM (k, ℓ)-CENTER

Instance: A set of polygonal paths $\mathcal{P} = \{P_1, \ldots, P_m\}$ each of length at most n, and positive integers k and ℓ.

Question: Find the set of k polygonal paths $\mathcal{C} = \{C_1, \ldots, C_k\}$, each with complexity ℓ and a surjection $\pi^*_{(k,\ell)} \colon \mathcal{P} \to \mathcal{C}$, such that the maximum $d_F(P_i, \pi^*_{(k,\ell)}(P))$ for each $P_i \in \mathcal{P}$ is minimised.

Similarly, let $\mathrm{cost}^*_{(k,\ell)}(\mathcal{P})$ be the cost of an optimal solution to the (k, ℓ)-CENTER problem.

6.3 Approximation Algorithm for the ℓ-SIMPLIFICATION Problem

We present an $O(n \log n)$ time greedy algorithm, that, given a polygonal path P and a parameter $\varepsilon > 0$, finds a path-restricted ℓ-simplification S of a path with maximum error 2ε—provided that there exists an ℓ-simplification of P with maximum error ε. The simplification S is obtained by computing an ordered stabbing of a set of ordered disks of radius r—whose centers lie on the vertices of P—with a polygonal path S of complexity ℓ. The computed simplification S is *path-restricted* to P, i.e. each vertex $s_i \in P$, and furthermore $s_1 = p_1$ and $s_\ell = p_n$. We show that the Fréchet distance

between P and S is no greater than r. Finally, we prove that if a path P admits an (optimal) ℓ-simplification of error at most ε, and ε is known, then the algorithm will find a simplification with complexity at most ℓ and error at most 2ε.

The algorithm is similar in construction to the algorithm by Guibas *et al.* [98] for the ordered stabbing of a set of ordered disks with a polygonal path, where the ordering is defined as:

Definition 6.4 (Guibas *et al.* Definition 4)**.** Line t hits points $p_1, p_2, \ldots, p_n$, with $p_j \in \ell \cap D_j$ in the correct order, for $i < j$, the point $p_j \prec p_j$.

Computing the Ordered Stabbing Path To compute the ordered stabber S, disks $D_1, D_2, \ldots, D_n$ of radius r are placed with centers at $p_1, p_2, \ldots, p_n$, and S is constructed such that it is path-restricted to P and stabs $D_1, D_2, \ldots, D_n$ in order according to Definition 6.4.

The algorithm constructs the path by finding a sequence of ℓ line segments (s_i, s_{i+1})—called *shortcuts*—where $s_i, s_{i+1} \in P$. The point s_1 is initialised to p_1. To find a shortcut starting at s_i, point s_{i+1} is initially placed at s_i and then moved along P as far as possible, such that $(s_i, s_{i+1}) \cap D_j \neq \emptyset$ for all j where $p_j \in P[s_i, s_{i+1}]$, see Figure 6.4. We say that all such disks (or, equivalently, their centers) are *charged* to shortcut (s_i, s_{i+1}). The process is repeated for s_{i+1} on the disks $D_{j+1}, \ldots, D_n$, until the end-point of the final shortcut of the ordered stabber reaches the final vertex p_n of P.

Computing each Shortcut In order to find each shortcut (s_i, s_{i+1}) in S, we find a stabbing line containing the point s_i that visits a set of the r-disks in order, starting with the first unvisited disk. This requires the maintenance of a data structure that stores the locus of points such that, for any point q in this locus, the line segment (s_i, q) contains points $d_1, \ldots, d_j$ such that $d_k \in D_k$ for $k = 1, \ldots, j$, and where $d_1, \ldots, d_j$ are visited in order, according to Definition 6.4. This locus is called the *line-stabbing wedge* by Guibas *et al.* and let Q_j denote the line-stabbing wedge after disk D_j has been stabbed.

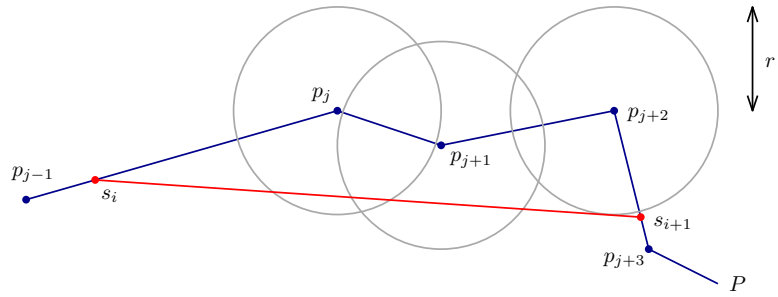


Figure 6.4: The shortcut (s_i, s_{i+1}) extends as far along P as possible such that $d_F(P[s_i, s_{i+1}], (s_i, s_{i+1})) < r$. The distance between p_{j+2} and the shortcut is r, and thus maximal.

The locus is bounded by two *limiting lines* t_{cw} and t_{ccw} that intersect at s_i and are not "to the left" of any the disks $D_1, \ldots, D_k$, see Figure 6.5. As we will see, the line-stabbing wedge will either be the whole space $\mathbb{R}^d$ or will be a region bounded by a chain containing two rays—each lying on a limiting line—and where the origins of the rays are connected by $O(k)$ arcs, each of which lie on the boundary of one of the disks $D_1, \ldots, D_j$.

This algorithm will thus only find stabbing lines containing the point s_i. In contrast, the algorithm by Guibas *et al.* has no such restriction. The restriction allows for a simpler algorithm for computing the stabbing line, as the limiting lines of the wedge always intersect at s_i. However, the definition of the line-stabbing wedge, the method by which it is incrementally constructed, and the complexity of the boundary of the wedge are broadly similar.

The line-stabbing wedge is constructed incrementally by processing each disk in order. Let $Q_0 \equiv \mathbb{R}^d$. To determine whether disk D_j can be stabbed, two conditions are tested. If the center p_j of D_j is outside the line-stabbing wedge, then we stop before adding disk D_j. Point s_{i+1} is then set as the intersection between the line segment (p_{j-1}, p_j) and the boundary of the line-stabbing wedge, thus completing the construction of shortcut (s_i, s_{i+1}).

We claim that such an intersection point always exists, and we use the following observation.

Observation 6.1. If vertex p_j of P is not contained within the line-stabbing wedge after

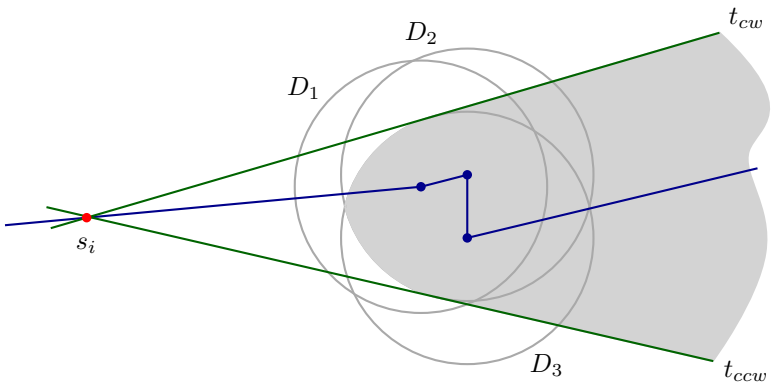


Figure 6.5: The *line-stabbing wedge* is the locus of points contained in lines that visit the disks in order between s_i and any point in q the wedge. That is they lie between the limit lines t_{cw} and t_{ccw} and are not to the left of any of the disks.

p_{j-1} was charged, i.e. $p_j \notin Q_{j-1}$, then edge (p_{j-1}, p_j) intersects with either t_{cw} or t_{ccw}.

This is true because D_{j-1} would only have been charged to the stabbing line if its center p_{j-1} was inside the line stabbing wedge Q_{j-2}. The limiting lines of Q_{j-2}—if they exist—could only have been updated to tangents of D_{j-1} and could not be placed so that p_{j-1} is outside of Q_{j-1}.

The second condition to check is that if $p_j \in Q_{j-1}$ and $j = n$, i.e. D_j is the last available disk to be stabbed, then point s_{i+1} is set to p_n, D_j is charged to shortcut (s_i, s_{i+1}), and the ordered stabbing is complete.

If the conditions are not met, then disk D_j can be added to the ordered stabbing by a line through s_i.

Stabbing a Disk For each disk D_j, there exists a region W_j that is the locus of points w such that for the line passing through s_i and w, there exists a point $d_j \in D_j$ for which $s_i \prec d_j \prec w$. The shape of the region W_j depends on whether $s_i \in D_j$, see Figure 6.6. If $s_i \in D_j$ then $W_j = \mathbb{R}^d$, as a line through s_i and any $w \neq s_i$ will also contain a point o such that $s_i \prec o \prec w$. If $s_j \notin D_j$ then W_j will be a region bounded by two rays and an arc on the circumference of D_j, see Figure 6.6(b). Each ray lies on one of the two tangent lines intersecting at s_i, originates at the point of tangency, and is directed away from s_i. The arc is the smaller section of the circumference of D_j subtended by

the two points of tangency, i.e. the arc that is closest to s_i.

The region W_j is used to update the line-stabbing wedge in each iteration. Recall Q_{j-1} be the line-stabbing wedge prior to processing disk D_j. If the disk D_j intersects the wedge Q_{j-1}, then the wedge $Q_j \subseteq Q_{j-1} \cap W_j$ can be computed, described below.

Observe that the boundary of Q_j, if it exists, always contains two rays and a chain of arcs. One of the rays will lie on a clockwise tangent for some disk, and the other on a counter-clockwise tangent for some disk. The chain of arcs will connect the origins of the two rays, see Figure 6.5.

The disks are added in order, and at each step the line-stabbing wedge is updated according to the following rules. To determine Q_j given W_j and Q_{j-1}, we need to consider the following cases, illustrated in Figure 6.6.

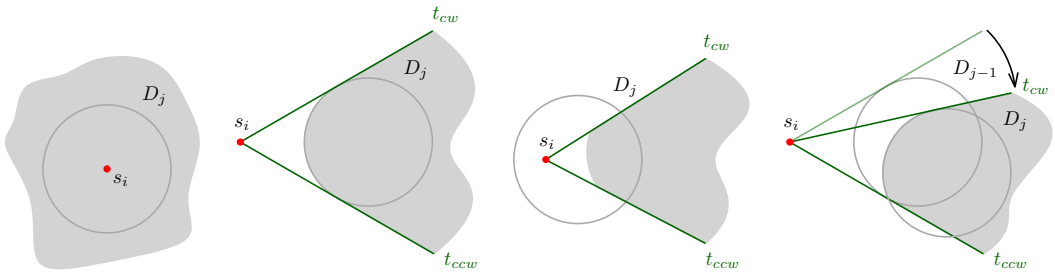
1. $Q_{j-1} = \mathbb{R}^d$ and $W_j = \mathbb{R}^d$: here $Q_j = \mathbb{R}^d$. This case will occur when all $D_1, \ldots, D_j$ contain the point s_i.

2. $Q_{j-1} = \mathbb{R}^d$ and $W_j \neq \mathbb{R}^d$: in this case $Q_j = W_j$.

3. $Q_{j-1} \neq \mathbb{R}^d$ and $W_j = \mathbb{R}^d$: s_i is in the interior of D_j, and there must exist an $k < j$ where $s_i \notin D_k$. We can set $Q_j = Q_{j-1}$. Observe that $D_j \cap Q_{j-1} \neq \emptyset$—if this were not the case then $p_j \notin Q_{j-1}$, which was handled by the first stopping condition.

4. $Q_{j-1} \neq \mathbb{R}^d$ and $W_1 \neq \mathbb{R}^d$: here s_i is not in the interior of D_j, and the limiting lines for W_j are the tangents that intersect at s_i.

The computation of Q_j is trivial for cases 1–3. In case 4 the limiting lines of the line stabbing wedge Q_j must be determined by comparing the clockwise limiting lines of D_{j-1} and W_j, and by comparing the two counter-clockwise limiting lines of the same. Here Q_j will be found by comparing the clockwise (resp. counter-clockwise) limiting lines of W_j and Q_{i-1} and choosing the one that obtains the smaller interior angle with the counter-clockwise (resp. clockwise) limiting line. In other words, Q_j is limited by the lines that result in a "narrower" wedge. The chain of arcs is also updated in case 4 if the bounding arc for W_j intersects the chain of arcs in Q_{j-1}.



(a) case 1 (b) case 2 (c) case 3 (d) case 4

Figure 6.6: Illustration of updates to line-stabbing wedge with respect to s_i when disk D_j is added.

Approximation Bound The greedy algorithm will compute a path S that is an ordered stabbing of a sequence of disks of radius r whose centers lie on the vertices of P, and where the vertices of S are path-restricted to P.

We show that, given an error $\varepsilon > 0$, such that the ℓ-SIMPLIFICATION problem admits an ℓ-simplification of error at most ε, then the greedy algorithm will find a simplification of error not greater than 2ε that has at most ℓ edges. Given an input path $P = \langle p_1, \ldots, p_n \rangle$ and an ℓ-simplification $O = \langle o_1, \ldots, o_\ell \rangle$ of error ε, the greedy algorithm will find a 2-approximate simplification $S = \langle s_1, \ldots, s'_\ell \rangle$ where $\ell' \leq \ell$.

Recall from Definition 6.1 that for the simplification O there exists a homeomorphism μ that assigns every point $o \in O$ to a point $p \in P$. The induction hypothesis is that the greedy simplification is not behind O after $k < \ell$ shortcuts, i.e. that $\mu(o_k) \prec s_k$ according to P.

The following lemma is used by the base case and inductive step.

Lemma 6.1. *Consider the points $s_i, s_{i+1} \in P$ such that (s_i, s_{i+1}) is a shortcut in the approximate simplification S of P. There exists a point $o \in O$ such that $\mu(o) = s_i$. Let $(o_{i'}, o_{i'+1})$ be the shortcut in the optimal simplification that contains o. Then $\mu(o_{i'+1}) \preceq s_{i+1}$.*

Proof. Let $p_j, \ldots, p_k$, $j \leq k$ be the vertices of P that are charged to (s_i, s_{i+1}), i.e. the corresponding 2ε-disks are stabbed by (s_i, s_{i+1}). There must exist a shortcut $(o_{i'}, o_{i'+1})$ in O that contains a point o where $\mu(o) = s_i$, and it is sufficient to show

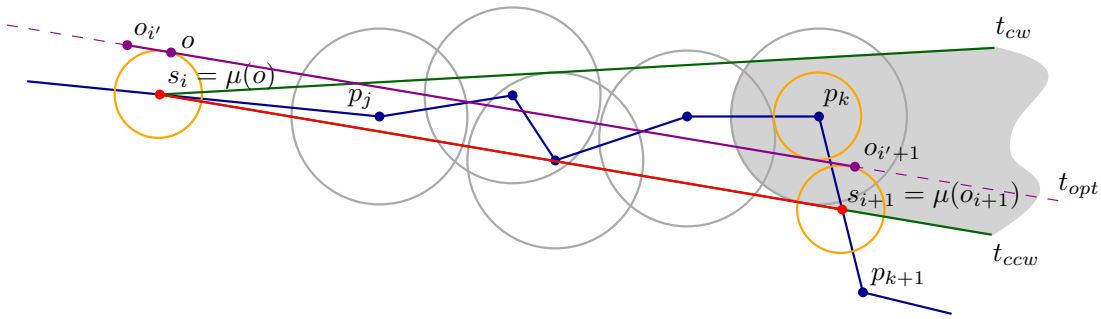


Figure 6.7: The greedy algorithm always stays ahead of the exact algorithm. Consider the shortcut (s_i, s_{i+1}) constructed using the greedy algorithm (in red). There must exist a shortcut (o_i, o_{i+1}) (magenta) on the exact simplification that contains a point o where $\mu^*(o) = s_i$. At best, the exact shortcut must lie on t_{opt} as it must be tangent to the ε-disks around s_i, p_k and s_{i+1}, and thus $\mu^*(o_{i+1}) \preceq s_{i+1}$.

that $\mu(o_{i'+1}) \preceq s_{i+1}$. By construction, point s_{i+1} lies on one of the limiting lines of the line-stabbing wedge, let us assume that this is t_{ccw}. The limiting line is a tangent to one of the 2ε-disks centered at $p \in \{p_j, \ldots, p_k\}$. The shortcut $(o_{i'}, o_{i'+1})$ must intersect the ε-disks centered at s_i and p, and hence, at best, must be parallel to t_{ccw}, and the distance between them will be ε. The endpoint of the optimal shortcut $o_{i'+1}$ may have $\mu(o_{i'+1}) = s_{i+1}$ at most, however it cannot be mapped to any point further along P, as (s_{i+1}, p_{k+1}) is below t_{ccw} and thus must be greater than ε from the optimal shortcut. □

See Figure 6.7 for an illustration of this construction.

Lemma 6.1 allows a simple proof by induction can be made on the end-point of each shortcut. The induction hypothesis is that for each $k \in 1, \ldots, \ell$, we have that $\mu(o_k) \preceq s_k$. For the base case $k = 1$, $s_i = p_i = \mu(o_i)$, by definition, and thus the ordering trivially holds. For the induction step, assume that $\mu(o_k) \preceq s_k$, and we must prove that $\mu(o_{k+1}) \preceq s_{k+1}$, which has been proven in Lemma 6.1.

Time Complexity During the execution of the algorithm on an input P with $|P| = n$, each point $p \in P$ will be evaluated, at most twice: it will be charged to a shortcut, and may be the endpoint of a shortcut that crosses one of the limiting lines, in which case it will be charged to the subsequent shortcut.

The algorithm must also maintain the boundary of the line-stabbing wedge for each shortcut. Guibas *et al.* [98, Lemma 9] states that it is possible to form all wedges from a sequence of n unit radius circles in $O(n \log n)$ total time.

Thus the total time required for the greedy algorithm is $O(n \log n)$ and we thus have the following:

Theorem 6.2. *Given an input polygonal chain P where $|P| = n$ and a constant $\varepsilon > 0$, if there exists a simplification of P with error at most ε and complexity ℓ, then the greedy algorithm will find an ℓ-simplification of P with error at most 2ε in $O(n \log n)$ time.*

6.4 Approximation Algorithms for the (k, ℓ)-CENTER Problem

Let $\mathcal{A}_\ell$ be an algorithm that finds an approximate solution to the ℓ-SIMPLIFICATION problem on a path P of complexity n, such that the error obtained is within a factor of K_1 of the optimal error ε, where $K_1 \geq 1$ is a constant. Similarly, let $\mathcal{A}_k$ be an algorithm that finds an approximate solution to the k-CENTER problem on a set of m paths, such that the cost of the obtained solution is within a factor of K_2 of the optimal cost, where $K_2 \geq 1$ is a constant.

Now, consider the following algorithm $\mathcal{A}_{(k,\ell)}$ for the (k, ℓ)-CENTER problem. We have as input $\mathcal{P} = \{P_1, \ldots, P_m\}$ of polygonal paths such that $|P_i| \leq n$ for all $P_i \in \mathcal{P}$, positive integers k and ℓ, and $\delta > 0$. We then run algorithm $\mathcal{A}_l$ on each $P \in \mathcal{P}$ to obtain the set $\mathcal{S} = \{S_1, \ldots, S_m\}$, such that $S_i \in \mathcal{S}$ is an ℓ-simplification of P_i. The simplified paths $\mathcal{S}$ are then clustered using algorithm $\mathcal{A}_k$ to obtain a k-clustering $\mathcal{C} = \{C_i, \ldots, C_k\}$ where $\mathcal{C} \subseteq \mathcal{S}$.

If there exists a (k, ℓ)-center clustering of cost δ, then applying $\mathcal{A}_\ell$ and $\mathcal{A}_k$ will output a (k, ℓ)-center clustering of cost at most $2(K_1 + 1)K_2\delta$.

Correctness The algorithm clearly obtains a (k, ℓ)-center clustering of $\mathcal{P}$, as the ℓ-simplifications $S \in \mathcal{S}$ obtained by algorithm $\mathcal{A}_\ell$ ensure that $|S_i| = \ell$ for all $S_i \in \mathcal{S}$, and algorithm $\mathcal{A}_k$ finds a k-clustering where the $\mathcal{C} \subseteq \mathcal{S}$. Thus, we need to show that $\mathcal{C}$ is, in fact, a $2(K_1 + 1)K_2$-approximation of the cost δ of the optimal solution to the (k, ℓ)-CENTER problem on $\mathcal{P}$.

Let $\mathcal{C}^* = \{C_1^*, \ldots, C_m^*\}$ be an optimal clustering of $\mathcal{P}$ obtained as a solution to the (k, ℓ)-CENTER problem, and let π^* be the corresponding surjection from $\mathcal{P}$ onto $\mathcal{C}^*$. We have for all $P_i \in \mathcal{P}$ that

$$d_F(P_i, \pi^*(P_i)) \leq \delta. \tag{6.1}$$

Now consider the set of ℓ-simplifications $\mathcal{S}$ of $\mathcal{P}$ obtained using algorithm $\mathcal{A}_\ell$, for each $P_i \in \mathcal{P}$ and its corresponding $S_i \in \mathcal{S}$ the following inequality holds:

$$d_F(S_i, \pi^*(P_i)) \leq d_F(S_i, P_i) + d_F(P_i, \pi^*(P_i)) \leq K_1\varepsilon + \delta. \tag{6.2}$$

The first inequality holds because of the triangle inequality property of the Fréchet distance, and the second inequality uses the approximation bound of $\mathcal{A}_\ell$ together with (6.1).

Next, for each center $C_j^* \in \mathcal{C}^*$ of the optimal solution, choose any $S_i \in \mathcal{S}$ where $i \in \{i \in 1, \ldots, n \mid \pi^*(P_i) = C_j^*\}$, and set $C_j = S_i$. Let $\mathcal{C} = \{C_1, \ldots, C_k\}$ be the set of simplified centers thus chosen. We then have, for each P_i and corresponding S_i, that

$$d_F(S_i, C_j) \leq d_F(S_i, C_j^*) + d_F(C_j^*, C_j) \leq 2(K_1\varepsilon + \delta). \tag{6.3}$$

Again, the first inequality holds because of the triangle-inequality property of the Fréchet distance, and since C_j was chosen as one of the ℓ-simplifications in the cluster of C_j^* then both the distances are bounded by (6.2).

The inequalities in (6.1)–(6.3) are upper-bounded by both δ and ε. Next, we see that ε has an upper bound of δ. Given that δ is the optimal cost of a solution to the (k, ℓ)-CENTER problem with respect to $\mathcal{P}$, and S_i^* is the solution to the ℓ-SIMPLIFICATION

problem for each $P_i \in \mathcal{P}$, then $d_F(C_j^*, P_i) \geq d_F(S_i^*, P_i)$ for all $P_i \in \mathcal{P}$. Thus, we have

$$\varepsilon = d_F(S_i^*, P_i) \leq d_F(C_j^*, P_i) = \delta. \tag{6.4}$$

The inequality holds as C_j^* is a feasible ℓ-simplification of P_i and thus must have a Fréchet distance to P_i that is at least as large as the distance to the optimal ℓ-simplification S_i^* to P_i.

With these observations, we are now able to prove an approximation bound for the algorithm. Substituting (6.4) into (6.3) we have

$$d_F(S_i, C_j) \leq 2(K_1\varepsilon + \delta) = 2(K_1 + 1)\delta. \tag{6.5}$$

Thus, there exists a k-center clustering with respect to $\mathcal{S}$ that is a $2(K_1+1)$-approximation of the optimal solution to the (k, ℓ)-CENTER problem with respect to $\mathcal{P}$.

From this and the fact that algorithm A_k obtains a K_2-approximation of the solution to the k-CENTER problem, we have the following result.

Lemma 6.3. *If there exists a solution to* (k, ℓ)-CENTER *problem with respect to* $\mathcal{P}$ *of cost* δ *then the algorithm* $\mathcal{A}_k$ *with respect to* $\mathcal{S}$ *obtains a* $2(K_1 + 1)K_2$*-approximation.*

Time Complexity Let $T_\ell(n)$ and $T_k(m)$ be the running time of algorithms $\mathcal{A}_\ell$ and $\mathcal{A}_k$ respectively. The approximation algorithm $\mathcal{A}_{(k,\ell)}$ for the (k, ℓ)-CENTER problem executes $\mathcal{A}_\ell$ on each of the m input paths, and then executes $\mathcal{A}_k$ on the resulting m shortcuts. Thus, we have:

Lemma 6.4. *The overall running time of algorithm* $\mathcal{A}_{(k,\ell)}$ *is* $mT_\ell(n) + T_k(m)$.

6.4.1 Results

In the analysis of algorithm $\mathcal{A}_{k,\ell}$, we assumed the existence of approximation algorithms $\mathcal{A}_\ell$ and $\mathcal{A}_k$ for the ℓ-SIMPLIFICATION and k-CENTER problems, respectively.

Table 6.1: Approximation factor and time complexity obtained for algorithms for the (k, ℓ)-CENTER problem for a given δ.

$\mathcal{A}_\ell$	$\mathcal{A}_k$	K_1	K_2	Approximation	Running Time
Guibas	Brute-force	1	1	4	$O(mn^2 \log^2 n + m(m\ell^2 \log \ell + k^m))$
Guibas	Hochbaum	1	2	8	$O(mn^2 \log^2 n + km\ell^2 \log \ell)$
Greedy	Brute-force	2	1	6	$O(mn \log n + m(m\ell^2 \log \ell + k^m))$
Greedy	Hochbaum	2	2	12	$O(mn \log n + km\ell^2 \log \ell)$

We can obtain the results for algorithm $\mathcal{A}_{k,\ell}$ by replacing $\mathcal{A}_\ell$ and $\mathcal{A}_k$ with known algorithms for these problems. By substituting the approximation factors and running times for the algorithms into Lemma 6.3 and Lemma 6.4, results for approximations of the (k, ℓ)-CENTER problem are obtained.

However, we are unaware of any exact or approximate algorithms for the ℓ-simplification problem. For a given ε, if an ℓ-simplification exists, the algorithm by Guibas *et al.* [98] gives an exact result for the ℓ-SIMPLIFICATION problem in $O(n^2 \log^2 n)$ running time, and the greedy algorithm in Section 6.3 gives a 2-approximation for the problem in $O(n \log n)$ time. To utilise these algorithms, consider the following:

Definition 6.5 ((k, ℓ, δ)-clustering)**.** Given a set of polygonal paths $\mathcal{P}$, positive integers k and ℓ, and $\delta > 0$, a (k, ℓ, δ)-*clustering* is a (k, ℓ)-clustering of cost at most δ.

Lemmas 6.3 and 6.4 imply the following lemma for (k, ℓ, δ)-clustering.

Lemma 6.5. *Given a set of polygonal paths $\mathcal{P}$, positive integers k and ℓ, and $\delta > 0$, if there exists a solution to the (k, ℓ)-CENTER problem with cost no greater than δ, then algorithm $\mathcal{A}_{(k,\ell)}$ outputs a $(k, \ell, 2(K_1+1)K_2\delta)$-clustering in $O(mT_\ell(n)+T_k(m))$ time.*

The k-CENTER problem can be solved exactly in $O(m(mT_d(n) + k^m))$ time using a brute force algorithm, and a 2-approximation can be obtained in $O(kmT_d(n))$ time using the algorithm by Hochbaum and Shmoys [108]. Here $T_d(n)$ is the cost of computation of the distance between inputs, and for the Fréchet distance this is $O(\ell^2 \log \ell)$ using the algorithm by Alt and Godau [7]. A small improvement may also be obtained using the algorithm by Buchin *et al.* [34] to compute the Fréchet distance in $O(\ell^2(\log \log \ell)^2)$ time in the word RAM model of computation.

Substituting these known algorithms for $\mathcal{A}_\ell$ and $\mathcal{A}_k$ as required, Lemma 6.5 gives the results in Table 6.1.

6.5 Conclusion

This chapter investigates the problem of finding an exemplar-based clustering of a set of polygonal paths—such as trajectories—that is of minimal cost and where the computed exemplars are smooth. The smoothness of the exemplars is enforced by restricting the exemplars to be paths of a bounded complexity in the number of edges. We formalise this as the (k, ℓ)-CENTER problem, and for a given error δ and show that it is possible to obtain a bounded approximation by first simplifying the input paths and then clustering the simplifications. A greedy algorithm for computing a 2-approximation of the ℓ-SIMPLIFICATION problem is presented, and using this, along with previously known algorithms for simplification and clustering, we are able to provide a family of bounded approximation results with varying running times.

The ℓ-simplification algorithms used in these results rely on the assumption that an upper bound on the simplification error ε is known. This is not a realistic assumption in practical situations, and efficiently determining this value is an open problem.

Chapter 7

Conclusion

The availability of large trajectory datasets on players in football matches and other invasion sports has motivated a consequent need for algorithms and tools to analyse and extract information from them. The main insight is that the tactics, strategies and objectives of the players are, to a certain extent, realisable from the trajectories. Teammates will coordinate their movements to achieve shared objectives, and at the same time the opposition players will move in an adversarial manner, and there is thus a rich latent structure within the trajectory data to be exploited.

There is also an apparent need for tools that can efficiently identify and present this underlying structure for tactical and strategic sports analysis. Professional sports is a significant industry, and tools to improve player and team performance—even marginally—can have a significant effect. The ability to identify patterns and structure in team play will be beneficial to team management for tasks such as player assessment and recruitment, opposition analysis and planning and strategic planning. Tasks that have heretofore been labour-intensive manual tasks could be partially or fully automated, such as video match analysis. Similarly, secondary industries such as broadcasting and fan engagement could improve their offering by providing analyses and visualisations that are automatically generated. The algorithms and techniques described here also have application in other domains, such as behavioural ecology, urban and city planning.

The object tracking systems that generate trajectory data have only recently proliferated, and the research into algorithms and tools to exploit the data is at an early stage. There are many open problems for which there is no consensus on the best approaches to solving them.

It is in this context that we present the contributions described in this thesis. Each contribution describes an algorithmic approach to extracting information from the same underlying input of player trajectory and event log data.

In Chapter 3 we presented a framework that used machine learning algorithms to address the classification problem of labelling each pass made during a football match with a rating of the quality of the pass. The insight that motivated this approach was that by computing geometric data structures it would be possible to incorporate higher-level structure about the state of play when the pass was made, and from this derive predictor variables that would be important to the learned classifier. The framework we constructed was able to achieve $90\,\%$ accuracy on the classification task. Moreover, we found that the predictor variables constructed from geometric data structures such as the *dominant region* were of high importance to the classifiers when making the rating. The subjective nature of the classification task meant that there was no "right" answer, and we found that the learned classifier would obtain a similar level of agreement with an expert observer as the level of agreement between two experts.

The research in Chapter 4 investigated a method for summarising a set of *state sequences*—of which trajectories are a specific example—in a compact structure that could be readily visualised. The goal was to segment the state sequences according to a set of criteria describing properties of the states, and then to produce a *group segmentation* using a *flow diagram* of a minimal number of nodes, such that for each state sequence, a valid segmentation was represented in the flow diagram as an s–t path. This problem was found to be NP-hard and also hard to approximate, and hence two heuristic algorithms were designed that were found to work well in practice—inducing flow diagrams that were close to optimal in the number of nodes they contained. We evaluated the utility of this approach on two use-cases using football data and found

that the resulting flow diagrams were interpretable and that several well-known insights into football strategy were apparent.

Chapter 5 presented algorithms for the joint task of clustering and outlier detection. The clustering problem was modelled as the facility location problem, and two heuristic algorithms were designed, and were based on distinct integer program formulations of the problem, one using Lagrangian relaxation and the other using affinity propagation. We observed that both algorithms were iterative and maintained their state in a set of *auxiliary variables*, and that by monitoring these variables at each iteration of the algorithm execution, a time-series of the state was captured. These time-series yield an implicitly two-dimensional representation of the clustered points and of the outliers—regardless of the dimensionality of the input. The visualisation offers insights into the structure of the input and also the execution of the algorithm, and we were able to use this technique to identify an improved design for the Lagrangian heuristic algorithm. This improved algorithm was found to outperform other techniques, and furthermore the induced time-series visualisations on real trajectory datasets revealed interesting insights into the structure of the input data.

The trajectory clustering problem was also considered in Chapter 6 but from a very different perspective. We observed that exemplar-based clustering of trajectories—such as those computed by the algorithms in Chapter 5—obtain exemplars for each cluster that are one of the input trajectories, and thus may be noisy, which may make them harder to visualise and interpret. To overcome this, we investigated the (k, ℓ)-CENTER problem that finds clusters and associated exemplars that have bounded complexity, and thus will be robust to noise. The chapter contained a formal analysis of the problem and presented an algorithmic framework for the problem. Using previously known algorithms for trajectory simplification and clustering, we obtained four approximation algorithms with bounded costs relative to the optimum cost.

The contributions in this thesis fit into a taxonomy of problems and tasks for spatio-temporal analysis of invasion sports, presented in Chapter 2. This taxonomy, was based on the systematic survey of research carried out on player trajectory data from

a number of sporting codes. In this survey, we identified a number of open problems, and we expand on these here to speculate on the directions that we believe that research in this area will take.

We anticipate that in future, the input trajectory data will contain a much richer set of location points. Currently the location points typically contain a two- or three-dimensional point location, but it is likely that in future additional details will be available, such as the direction that a player is facing. Furthermore, multiple parts of the player could be tracked and thus the player's pose at each location point could be ascertained. Consequently, the input trajectory data will contain more information, and hence present more challenges as to how to process and exploit this additional information.

Currently, the object tracking systems require intensive post-processing to output the trajectories, and thus many of the current analytic approaches are offline processes. There is an apparent demand for real-time analysis during matches, and thus analytic tools that operate in a streaming or online model will be required. Such algorithms will need to execute in near-linear time, and while there has been some research on fundamental problems such as curve simplification in this setting [1], [64], this remains an open field.

Finally, the techniques for extracting information from spatio-temporal data are currently fragmented and there is a need for common data structures and a common language through which queries can be made against a trajectory database. Tasks such as trajectory querying, clustering and segmentation are currently addressed using algorithms that rely on specific data structures. These algorithms exploit many of the same properties, and a unified data structure that supported algorithms that were competitive with those on bespoke structures would be an important contribution. Similarly, a unified language for trajectory query and analysis would facilitate the adoption of the techniques and algorithms discussed in Chapter 2. Such a language would need to support queries on the spatial and structural properties of the trajectories and should be

extensible to domain-specific requests. The language would also support more sophisticated queries and would help to unify the heretofore disparate techniques currently available for trajectory analysis.

In conclusion, there are many open problems—both fundamental and applied—in the area of trajectory analysis and mining, and advances in this area could have a significant impact. These problems have interested researchers from many fields, and we can expect innovative and diverse contributions to the state of the art in the near future.

Appendix A

Input Match Data Details

A.1 Event Types

The following list event types are used in the event logs.

Table A.1: List of events and their types used in event logs.

Event Name	Event Type	Event Name	Event Type
Ball Out Of Play	Technical event	Goalkeeper Save Catch	Player event
Block	Player event	Goalkeeper Throw	Player event
Clearance	Player event	Handball	Player event
Corner Cross	Player event	Header	Player event
Corner Pass	Player event	Header Shot	Player event
Cross	Player event	Indirect Free Kick Pass	Player event
Direct Free Kick Cross	Player event	Kick Off	Technical event
Direct Free Kick Pass	Player event	Offside	Technical event
Direct Free Kick Shot	Player event	Own Goal	Player event
Dribble	Player event	Pass	Player event
Drop Ball	Player event	Penalty Shot	Player event
End Of Half	Technical event	Post	Player event
Foul	Technical event	Shot	Player event
Goal	Player event	Start Of Half	Technical event
Goal Kick	Player event	Stoppage	Technical event
Goalkeeper Catch	Player event	Substitution	Technical event
Goalkeeper Kick	Player event	Tackle	Player event
Goalkeeper Pick Up	Player event	Throw In	Player event
Goalkeeper Punch	Player event	Touch	Player event
Goalkeeper Save	Player event	Yellow Card	Technical event

Appendix B

Pass Classifier Feature Descriptions

The features functions used to compute the predictor variables are briefly described in the Table B.1.

Table B.1: The feature functions implemented for the pass classification model.

Feature Name	Description	Variable Class	Variable Type	Domain
Half	The half of the match that the pass occurred in.	N/A	Discrete	$\{1, 2\}$
Time-stamp	The time-stamp of the pass in the half.	N/A	Discrete	$\mathbb{N}$
Player Sequence ID	A unique identifier of the player possession sequence.	Sequential	Discrete	$\mathbb{N}$
Player Sequence Index	The ordinal position within the player possession sequence when the pass event occurred.	Sequential	Discrete	$\mathbb{N}$
Possession Sequence ID	The unique identifier of the team possession sequence.	Sequential	Discrete	$\mathbb{N}$
Possession Sequence Index	The ordinal position within the team possession sequence when the pass event occurred.	Sequential	Discrete	$\mathbb{N}$
Possession Start Flag	Indicator that the event is the first event in the team possession sequence.	Sequential	Binary	$\{0, 1\}$
Possession End Flag	Indicator that the event is the final event in the team possession sequence.	Sequential	Binary	$\{0, 1\}$

Table B.1: (contd.) The feature functions.

Feature Name	Description	Variable Class	Variable Type	Domain
Possession First Action	Event type of the first action in the possession sequence.	Sequential	Categoric	
Play Sequence ID	The unique identifier of the play sequence.	Sequential	Discrete	$\mathbb{N}$
Play Sequence Index	The ordinal position within the play sequence when the pass event occurred.	Sequential	Discrete	$\mathbb{N}$
Play Start Flag	Indicator that the event is the first event in the play sequence.	Sequential	Binary	$\{0, 1\}$
Play End Flag	Indicator that the event is the final event in the play sequence.	Sequential	Binary	$\{0, 1\}$
Play First Action	Event type of the first action in the play sequence.	Sequential	Categoric	
Passing Player	The name of the player who makes the pass.	N/A	Categoric	
Receiving Player	The name of the player who receives the pass.	N/A	Categoric	
Passing Player X-coordinate	The x-coordinate of the passing player.	Geometric	Discrete	$\mathbb{Z}$
Passing Player Y-coordinate	The y-coordinate of the passing player.	Geometric	Discrete	$\mathbb{Z}$
Receiving Player X-coordinate	The x-coordinate of the receiving player.	Geometric	Discrete	$\mathbb{Z}$
Receiving Player Y-coordinate	The y-coordinate of the receiving player.	Geometric	Discrete	$\mathbb{Z}$
Dominant Region - Passer	The total area of the passing player's dominant region in mm^2.	Strategic	Continuous	$\mathbb{R}$
Dominant Region - Receiver	The total area of the receiving player's dominant region in mm^2.	Strategic	Continuous	$\mathbb{R}$
Dominant Region Net Change - Passer	The difference in the area in the passing player's dominant region in mm^2 between when the pass was made and when it was received.	Strategic	Continuous	$\mathbb{R}$
Dominant Region Net Change - Reciever	The difference in the area in the receiving player's dominant region in mm^2 between when the pass was made and when it was received.	Strategic	Continuous	$\mathbb{R}$

Table B.1: (contd.) The feature functions.

Feature Name	Description	Variable Class	Variable Type	Domain
Dominant Region - Team	The total area of the receiving player's team's dominant region in mm^2.	Strategic	Continuous	$\mathbb{R}$
Dominant Region Net Change - Team	The difference in the area in the receiving player's team's dominant region in mm^2 between when the pass was made and when it was received.	Strategic	Continuous	$\mathbb{R}$
Passer Pressure	The pressure exerted on the passing player using the pressure measure defined by [183].	Physiological	Continuous	$\mathbb{R}$
Receiver Pressure	The pressure exerted on the receiving player using the pressure measure defined by [183].	Physiological	Continuous	$\mathbb{R}$
Passer-Receiver Pressure Net Change	The net change in the pressure exerted on the passing player and the receiving player using the pressure measure defined by [183].	Physiological	Continuous	$\mathbb{R}$
Distance To Goal	The distance in mm from the passing player's position to the centre of the goal that the player is attacking.	Geometric	Continuous	$\mathbb{R}$
Opposition To Goal	The number of opposition players in the funnel between the passing player and the goal the player is attacking.	Geometric	Discrete	$\mathbb{N}$
Opposition To Goal Net Change	The difference in the number of opposition players in the funnel between the passing player and the goal the player is attacking from when the pass is made to when it is received.	Geometric	Discrete	$\mathbb{N}$
Opposition To Goal-line	The number of opposition players between the passing player and the goal-line he/she is attacking.	Geometric	Discrete	$\mathbb{N}$

Table B.1: (contd.) The feature functions.

Feature Name	Description	Variable Class	Variable Type	Domain
Opposition To Goal-line Net Change	The difference in the number of opposition players in the area between the passing player and the goal-line, and the receiving player and the goal-line.	Geometric	Discrete	$\mathbb{N}$
Teammates To Goal-line	The number of teammates between the player making the pass and the goal-line the player is defending.	Geometric	Discrete	$\mathbb{N}$
Teammates To Goal-line Net Change	The difference in the number of teammates between the passing player and the goal-line, and the receiving player and the goal-line.	Geometric	Discrete	$\mathbb{N}$
Opposition To Receiver	The number of opposition players in the reachable area between the passing player and the receiving player.	Geometric	Discrete	$\mathbb{N}$
Controlled By Receiver	Indicator to determine whether the receiving player controlled the pass, either by making two or more touches or by passing the ball again.	Physiological	Binary	$\{0, 1\}$
Supporting Players	The number of teammates of the passing player who are open to receive the pass.	Physiological	Discrete	$\mathbb{N}$
Pass Distance	The distance in mm of the pass.	Geometric	Continuous	$\mathbb{R}$
Pass Speed	The speed of the pass in m/sec.	Geometric	Continuous	$\mathbb{R}$
Pass Angle	The angle of the trajectory of the pass.	Geometric	Continuous	$[-\pi, \pi]$
Facing Angle	The angle that the passing player is facing when the pass is made.	Geometric	Continuous	$[-\pi, \pi]$
Receiver Angle	The difference between the angle that the receiving player is facing and the angle to the point where the pass is received.	Geometric	Continuous	$[-\pi, \pi]$

Table B.1: (contd.) The feature functions.

Feature Name	Description	Variable Class	Variable Type	Domain
Reachable Angle	The angle of the "cone" that the passer must pass the ball through in order that the receiver can reach the ball before any other player, given the speed of the pass.	Geometric	Continuous	$[-\pi, \pi]$
Distance From Nearest Touchline	The distance in mm to the touchline nearest to the passing player.	Geometric	Continuous	$\mathbb{R}$
Distance From Defensive Goal-line	The distance in mm to the goal-line that the passing player is defending.	Geometric	Continuous	$\mathbb{R}$
Distance From Attacking Goal-line	The distance in mm to the goal-line that the passing player is attacking.	Geometric	Continuous	$\mathbb{R}$
Receiver Distance To Ball	The distance in mm from the location of the receiving player when the pass is made to the location of the point where the ball is received.	Geometric	Continuous	$\mathbb{R}$
Passer Touch Count	The total number of touch events made by the passing player prior to the pass.	Sequential	Discrete	$\mathbb{N}$
Passer Distance	The total distance in mm travelled by the passing player prior to the pass.	Sequential	Continuous	$\mathbb{R}$
Passer Possession Time	The total time in ms that the passing player is in possession prior to the pass being made.	Sequential	Discrete	$\mathbb{N}$
Receiver Touch Count	The total number of touch events made by the receiving player after receiving the pass.	Sequential	Discrete	$\mathbb{N}$
Receiver Distance	The total distance in mm travelled by the receiving player after receiving the pass.	Sequential	Continuous	$\mathbb{R}$
Receiver Possession Time	The total time in ms that the receiving player is in possession after receiving the pass.	Sequential	Discrete	$\mathbb{N}$

Appendix C

Affinity Propagation Message Formulation

The messages described in Subsection 5.2.2 were derived as follows. The two general message forms depend on whether the message is from a factor node with function f to a variable node with variable x, or vice versa.

$$\mu_{x\to f}(x') = \sum_{f_t \in \delta(x)\setminus f} \mu_{f_t \to x}(x') \tag{C.1}$$

$$\mu_{f\to x}(x') = \max_{x_i \in \delta(f)\setminus x} \Big[f(x_1, \ldots, x = x', \ldots, x_M) + \sum_{k\colon x_k \in \delta(f)\setminus x} \mu_{x_k \to f}(x_k) \Big] \tag{C.2}$$

Here $\delta(v)$ is the neighbourhood of v, i.e. all nodes that are incident to v. Informally, a message from a node x to f is the sum of all messages to x from all its incident nodes, except f. A message from a node f to x is the maximum value over all variable assignments of the factor function plus the values of all messages into f from nodes that are incident to f.

The general form (C.1) is used to derive messages ρ_{ij}, γ_{ij}, σ_i and τ_i, and (C.2) is used to derive α_{ij}, η_{ij}, χ_{ij} and ω_{ij}.

C.1 Variable-to-factor Messages

We first derive the messages that originate at variables. Using (C.1), we have

$$\begin{aligned}\rho_{ij}(1) =& \eta_{ij}(1) - d_{ij}(1)\\ \rho_{ij}(0) =& \eta_{ij}(0) - d_{ij}(0)\end{aligned}$$

And thus

$$\begin{aligned}\rho_{ij} =& \rho_{ij}(1) - \rho_{ij}(0)\\ =& \eta_{ij}(1) - \eta_{ij}(0) - d_{ij}(1) + d_{ij}(0)\\ =& \eta_{ij} - d_{ij}.\end{aligned}$$

Similarly

$$\begin{aligned}\gamma_{ij} =& \alpha_{ij} - d_{ij}\\ \tau_i =& \chi_i\\ \sigma_i =& \omega_i\end{aligned}$$

C.2 Factor-to-variable Messages

The factor-to-variable messages are derived using (C.2). For these messages, the derived functions can be different for each of the values of x, and thus the derivation is more complicated.

Message α_{ij}: The update rule for $\alpha_{ij}(1)$ follows. When $x_{ij} = 1$, then $E_j = -f_j$ and all other variables $x_{kj} : k \neq i$ can be either 1 or 0.

$$\begin{aligned}
\alpha_{ij}(1) = &\max_{x_{kj}\,:\,k\neq i} \Big[E_j(x_{ij}, \ldots, x_{ij} = 1, \ldots, x_{nj}) + \sum_{t\neq i} \rho_{tj}(x_{tj})\Big] \\
=&-f_j + \max_{x_{kj}\,:\,k\neq i} \Big[\sum_{t\neq i} \rho_{tj}(x_{tj})\Big] \\
=&-f_j + \sum_{k\neq i} \Big[\max_{x_{kj}} \rho_{kj}(x_{kj})\Big]
\end{aligned}$$

For $x_{ij} = 0$, we need to find the maximizer of two values obtained first when all $x_{kj} = 0, k \neq j$ and then when x_{kj} can be either 0 or 1.

$$\begin{aligned}
\alpha_{ij}(0) = &\max_{x_{kj}\,:\,k\neq i} \Big[E_j(x_{ij}, \ldots, x_{ij} = 0, \ldots, x_{nj}) + \sum_{t\neq i} \rho_{tj}(x_{tj})\Big] \\
=&\max \Big[0 + \sum_{k\neq i} \rho_{kj}(0), -f_j + \sum_{k\neq i} \big[\max_{x_{kj}} \rho_{kj}(x_{kj})\big]\Big]
\end{aligned}$$

Computing the difference is thus

$$\begin{aligned}
\alpha_{ij} =&\rho_{ij}(1) - \rho_{ij}(0) \\
=&-f_j + \sum_{k\neq i} \Big[\max_{x_{kj}} \rho_{kj}(x_{kj})\Big] - \max \Big[\sum_{k\neq i} \rho_{kj}(0), -f_j + \sum_{k\neq i} \big[\max_{x_{kj}} \rho_{kj}(x_{kj})\big]\Big] \\
=&\min \Big[-f_j + \sum_{k\neq i} \Big[\max_{x_{kj}} \big[\rho_{kj}(x_{kj}) - \rho_{kj}(0)\big], 0\Big] \\
=&\min \Big[0, -f_j + \sum_{k\neq i} \max \big[\rho_{kj}, 0\big]\Big]
\end{aligned}$$

Messages η_{ij} and σ_i: The update rules for η_{ij} and σ_i are derived by a similar sequence of operations. First we consider $\eta_{ij} = 1$, observing that unless exactly one of $x_{:j} \cup \{o_i\}$ has value 1, then the factor function will be trivially sub-optimal.

$$\begin{aligned}\eta_{ij}(1) &= \max_{\{x_{ik}:\, k\neq j\}\cup\{o_i\}} \Big[I_i(x_{i1},\ldots,x_{ij}=1,\ldots,x_{in},o_i) + \sum_{k\neq j}\gamma_{ik}(x_{ik}) + \sigma_i(o_i)\Big] \\ &= \sum_{k\neq j}\gamma_{ik}(0) + \sigma_i(0)\end{aligned}$$

For $\eta_{ij}(0)$, the maximal assignment will have exactly one of the remaining variables with value 1.

$$\begin{aligned}\eta_{ij}(0) &= \max_{\{x_{ik}:\, k\neq j\}\cup\{o_i\}} \Big[I_i(x_{i1},\ldots,x_{ij}=0,\ldots,x_{in},o_i) + \sum_{t\neq j}\gamma_{it}(x_{it}) + \sigma_i(o_i)\Big] \\ &= \max\Big[\max_{k\neq j}\big[\gamma_{ik}(1) + \sum_{t\notin\{j,k\}}\gamma_{it}(0) + \sigma_i(0)\big], \sum_{t\neq j}\gamma_{it}(0) + \sigma_i(1)\Big]\end{aligned}$$

η_{ij} is the difference between the two.

$$\begin{aligned}\eta_{ij} =& \eta_{ij}(1) - \eta_{ij}(0) \\ =& \sum_{k\neq j}\gamma_{ik}(0) + \sigma_i(0) - \\ & \max\Big[\max_{k\neq j}\big[\gamma_{ik}(1) + \sum_{t\notin\{j,k\}}\gamma_{it}(0) + \sigma_i(0)\big], \sum_{t\neq j}\gamma_{it}(0) + \sigma_i(1)\Big] \\ =& \min\Big[\min_{k\neq j}\big[\gamma_{ik}(0) - \gamma_{ik}(1)\big], \sigma_i(0) - \sigma_i(1)\Big] \\ =& \min\Big[\min_{k\neq j} -\gamma_{ik}, -\sigma_i\Big]\end{aligned}$$

In a similar manner, χ_i can be derived.

$$\chi_i = \chi_i(1) - \chi_i(0) \qquad = -\max_j \gamma_{ij}$$

Message ω_i: The update rule for ω_i requires that two cases be handled in order to derive the message, however the resulting message is relatively simple.

$$\omega_i(1) = \max_{o_k\,:\,k\neq i} \Big[A(o_1,\ldots,o_i=1,\ldots,o_n) + \sum_{t\neq i}\tau_t(o_t)\Big]$$

and

$$\omega_i(0) = \max_{o_k\,:\,k\neq i} \Big[A(o_1,\ldots,o_i=0,\ldots,o_n) + \sum_{t\neq i}\tau_t(o_t)\Big]$$

The factor function $A(\mathbf{o})$ will be trivially suboptimal unless exactly ℓ of the variables in $\mathbf{o}$ are set to 1, and thus we need to find such a set $C^* \in \mathcal{C}$ where $|C^*| = \ell$ that maximises ω. Recall that $\mathcal{C} = \{1,\ldots,n\}$ is the set of all indices and let $\mathcal{T}_\ell = \{C \subset \mathcal{C} \mid |C| = \ell\}$ be the set of all subsets of $\mathcal{C}$ of cardinality ℓ. Consider the function

$$h^*(\ell) = \max_{C\in\mathcal{T}_\ell} \Big[\sum_{t\in C}\tau_t(1) + \sum_{t\in\mathcal{C}\setminus C}\tau_t(0)\Big],$$

and let $C^*_\ell \in \mathcal{T}_\ell$ be a set that maximises $h^*(\ell)$. Furthermore let $u_1,\ldots,u_n \to \mathcal{C}$ be an ordering such that $\tau_{u_i} \geq \tau_{u_j}$for all $u_i < u_j$. In other words $u_1,\ldots,u_n$ order the values of τ_{u_i} in decreasing order. We need the following lemma.

Lemma C.1. *The value of $h^*(\ell)$ is the sum of the ℓ largest values of τ_i, i.e. $h^*(\ell) = \sum_{k=1}^{\ell}\tau_{u_k}$ and thus $C^*_\ell = \{u_1, u_2, \ldots, u_\ell\}$.*

Proof. To prove a contradiction, assume that C^*_ℓ is a maximizer for $h^*(\ell)$ and contains an index u_k, such that $k > \ell$. There must exist a $u_m \notin C^*_\ell$ such that $m <= \ell$.

Consider the objective function value if u_k is replaced with u_m. We have

$$\begin{aligned} h^*(\ell) - \tau_{u_k}(1) + \tau_{u_k}(0) + \tau_{u_m}(1) - \tau_{u_m}(0) &< h^*(\ell) \\ h^*(\ell) - \tau_{u_k} + \tau_{u_m} &< h^*(\ell) \\ \tau_{u_m} &< \tau_{u_k} \end{aligned}$$

However $m \leq \ell < k$, thus contradicting the definition of the ordering on u_i. $\square$

To compute $\omega_i(1)$ and $\omega_i(0)$, consider an ordering $u_1, \ldots, u_n$. Then i is either a member of $u_1, \ldots, u_\ell$—i.e. the indices yielding the ℓ largest values of τ_i—or not, and each case must be handled separately.

1. $i \in \{u_1, \ldots, u_\ell\}$ We can restate the message formula for $\omega_i(1)$ and $\omega_i(0)$ in terms of C^*_ℓ as $\omega_{i_\leq}$:

$$\omega_{i_\leq}(1) = \sum_{k \in C^*_\ell \setminus \{\ell\}} \tau_k(1) + \sum_{k \in \mathcal{C} \setminus C^*_\ell} \tau_k(0)$$
$$\omega_{i_\leq}(0) = \sum_{k \in C^*_{\ell+1} \setminus \{\ell\}} \tau_k(1) + \sum_{k \in \mathcal{C} \setminus C^*_{\ell+1}} \tau_k(0)$$

And thus we can compute

$$\begin{aligned}
\omega_{i_\leq} =& \omega_{i_\leq}(1) - \omega_{i_\leq}(0) \\
=& \sum_{k \in C^*_\ell \setminus \{\ell\}} \tau_k(1) - \sum_{k \in C^*_{\ell+1} \setminus \{\ell\}} \tau_k(1) + \sum_{k \in \mathcal{C} \setminus C^*_\ell} \tau_k(0) - \sum_{k \in \mathcal{C} \setminus C^*_{\ell+1}} \tau_k(0) \\
=& - \tau_{u_{\ell+1}}(1) + \tau_{u_{\ell+1}}(0) \\
=& - \tau_{u_{\ell+1}}
\end{aligned}$$

2. $i \notin \{u_1, \ldots, u_\ell\}$ In a similar manner, we have $\omega_{i_>}$:

$$\omega_{i_>}(1) = \sum_{k \in C^*_{\ell-1}} \tau_k(1) + \sum_{k \in \mathcal{C} \setminus (C^*_{\ell-1} \cup \{i\})} \tau_k(0)$$
$$\omega_{i_>}(0) = \sum_{k \in C^*_\ell} \tau_k(1) + \sum_{k \in \mathcal{C} \setminus (C^*_\ell \cup \{i\})} \tau_k(0)$$

And thus we can compute

$$\begin{aligned}\omega_{i_>} =& \omega_{i_>}(1) - \omega_{i_>}(0) \\ =& -\tau_{u_\ell}(1) + \tau_{u_\ell}(0) \\ =& -\tau_{u_\ell}\end{aligned}$$

Finally we can combine the two cases:

$$\omega_i = -\begin{cases} \tau_{u_{\ell+1}} & \text{if } i \in \{u_1, \ldots, u_\ell\} \\ \tau_{u_\ell} & \text{otherwise} \end{cases} \tag{C.3}$$

Bibliography

[1] M. A. Abam, M. de Berg, P. Hachenberger, and A. Zarei, "Streaming algorithms for line simplification," *Discrete & Computational Geometry*, vol. 43, no. 3, pp. 497–515, 2010. DOI: 10.1007/s00454-008-9132-4.

[2] N. Adams, "Issue information—special issue: Sports analytics," *Statistical Analysis and Data Mining: The ASA Data Science Journal*, vol. 9, no. 5, pp. i–iv, Sep. 2016. DOI: 10.1002/sam.11292.

[3] P. K. Agarwal, S. Har-Peled, N. H. Mustafa, and Y. Wang, "Near-linear time approximation algorithms for curve simplification," *Algorithmica*, vol. 42, no. 3-4, pp. 203–219, May 2005. DOI: 10.1007/s00453-005-1165-y.

[4] C. C. Aggarwal, *Outlier Analysis*. Springer International Publishing AG, Jan. 6, 2017. DOI: 10.1007/978-3-319-47578-3.

[5] S. P. A. Alewijnse, K. Buchin, M. Buchin, A. Kölzsch, H. Kruckenberg, and M. A. Westenberg, "A framework for trajectory segmentation by stable criteria," in *Proceedings of the 22nd. ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems (SIGSPATIAL '14)*, ACM, Nov. 2014, pp. 351–360. DOI: 10.1145/2666310.2666415.

[6] H. Alt, A. Efrat, G. Rote, and C. Wenk, "Matching planar maps," *Journal of Algorithms*, vol. 49, no. 2, pp. 262–283, Nov. 2003. DOI: 10.1016/s0196-6774(03)00085-3.

[7] H. Alt and M. Godau, "Computing the fréchet distance between two polygonal curves," *International Journal of Computational Geometry and Applications*, vol. 5, pp. 75–91, 1995. DOI: 10.1142/S0218195995000064.

[8] B. Aronov, A. Driemel, M. J. van Kreveld, M. Löffler, and F. Staals, "Segmentation of trajectories on nonmonotone criteria," *ACM Transactions on Algorithms*, vol. 12, no. 2, pp. 1897–1911, 2016. DOI: 10.1145/2660772.

[9] S. Baccianella, A. Esuli, and F. Sebastiani, "Evaluation measures for ordinal regression," in *Proceedings of the 9th. International Conference on Intelligent Systems Design and Applications*, IEEE, 2009, pp. 283–287. DOI: 10.1109/ISDA.2009.230.

[10] P. Balkundi and D. A. Harrison, "Ties, leaders, and time in teams: Strong inference about network structures effects on team viability and performance," *Academy of Management Journal*, vol. 49, no. 1, pp. 49–68, Feb. 2006. DOI: 10.5465/AMJ.2006.20785500.

[11] A. Bavelas, "Communication patterns in task-oriented groups," *The Journal of the Acoustical Society of America*, vol. 22, no. 6, pp. 725–730, 1950.

[12] J. Beasley, "Lagrangean heuristics for location problems," *European Journal of Operational Research*, vol. 65, no. 3, pp. 383–399, 1993. DOI: http://dx.doi.org/10.1016/0377-2217(93)90118-7.

[13] M. Beetz, N. von Hoyningen-Huene, B. Kirchlechner, S. Gedikli, F. Siles, M. Durus, and M. Lames, "ASPOGAMO: Automated sports games analysis models," *International Journal of Computer Science in Sport*, vol. 8, no. 1, pp. 1–21, 2009.

[14] M. de Berg, O. Cheong, M. van Kreveld, and M. Overmars, *Computational Geometry*. Heidelberg, Germany: Springer Berlin Heidelberg, Mar. 7, 2008.

[15] D. Bertsimas and R. Weismantel, *Optimization over Integers*. Athena Scientific, 2005.

[16] A. Bialkowski, P. Lucey, G. P. K. Carr, Y. Yue, and I. Matthews, "Win at home and draw away: Automatic formation analysis highlighting the differences in home and away team behaviors," in *Proceedings of the 8th. Annual MIT Sloan Sports Analytics Conference*, MIT, Feb. 2014, pp. 1–7.

[17] A. Bialkowski, P. Lucey, G. P. K. Carr, Y. Yue, S. Sridharan, and I. Matthews, "Identifying team style in soccer using formations learned from spatiotemporal tracking data," in *Proceedings of the IEEE International Conference on Data Mining Workshops (ICDM Workshops)*, IEEE, Dec. 2014, pp. 9–14. DOI: 10.1109/ICDMW.2014.167.

[18] A. Bialkowski, P. Lucey, P. Carr, Y. Yue, S. Sridharan, and I. Matthews, "Large-scale analysis of soccer matches using spatiotemporal tracking data," in *Proceedings of the IEEE International Conference on Data Mining (ICDM '14)*, IEEE, Dec. 2014, pp. 725–730. DOI: 10.1109/ICDM.2014.133.

[19] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent dirichlet allocation," *The Journal of Machine Learning Research*, vol. 3, pp. 993–1022, Mar. 2003.

[20] F. Bonchi, C. Castillo, D. Donato, and A. Gionis, "Taxonomy-driven lumping for sequence mining," *Data Mining and Knowledge Discovery*, vol. 19, no. 2, pp. 227–244, Jul. 2009. DOI: 10.1007/s10618-009-0141-6.

[21] S. P. Borgatti, "Centrality and network flow," *Social Networks*, vol. 27, no. 1, pp. 55–71, Jan. 2005. DOI: 10.1016/j.socnet.2004.11.008.

[22] A. Borrie, G. K. Jonsson, and M. S. Magnusson, "Temporal pattern analysis and its applicability in sport: An explanation and exemplar data.," *Journal of Sports Sciences*, vol. 20, no. 10, pp. 845–52, 2002. DOI: 10.1080/026404102320675675.

[23] J. Bourbousson, G. Poizat, J. Saury, and C. Seve, "Team coordination in basketball: Description of the cognitive connections among teammates," *Journal of Applied Sport Psychology*, vol. 22, no. 2, pp. 150–166, Apr. 2010. DOI: 10.1080/10413201003664657.

[24] J. Bourbousson, C. Sève, and T. McGarry, "Space-time coordination dynamics in basketball: Part 1. intra- and inter-couplings among player dyads," *Journal of Sports Sciences*, vol. 28, no. 3, pp. 339–347, Feb. 2010. DOI: 10.1080/02640410903503632.

[25] P. Bradley, P. O'Donoghue, B. Wooster, and P. Tordoff, "The reliability of ProZone MatchViewer: A video-based technical performance analysis system," *International Journal of Performance Analysis in Sport*, vol. 7, no. 3, pp. 117–129, 2007.

[26] D. Braess, A. Nagurney, and T. Wakolbinger, "On a paradox of traffic planning," *Transportation Science*, vol. 39, no. 4, pp. 446–450, Nov. 2005. DOI: 10.1287/trsc.1050.0127.

[27] U. Brefeld and A. Zimmermann, "Guest editorial: Special issue on sports analytics," *Data Mining and Knowledge Discovery*, Jul. 2017. DOI: 10.1007/s10618-017-0530-1.

[28] S. Brin and L. Page, "The anatomy of a large-scale hypertextual web search engine," *Computer Networks and ISDN Systems*, vol. 30, no. 1-7, pp. 107–117, Apr. 1998. DOI: 10.1016/S0169-7552(98)00110-X.

[29] K. Buchin, M. Buchin, J. Gudmundsson, M. Löffler, and J. Luo, "Detecting commuting patterns by clustering subtrajectories," *International Journal on Computational Geometry and Applications*, vol. 21, no. 3, pp. 253–282, 2011. DOI: 10.1142/S0218195911003652.

[30] K. Buchin, M. Buchin, M. J. van Kreveld, M. Löffler, R. I. Silveira, C. Wenk, and L. Wiratma, "Median trajectories," *Algorithmica*, vol. 66, no. 3, pp. 595–614, 2013.

[31] K. Buchin, M. Buchin, J. Gudmundsson, M. Horton, and S. Sijben, "Compact flow diagrams for state sequences," in *Proceedings of the 15th. International Symposium on Experimental Algorithms (SEA '16)*, ser. Lecture Notes in Computer Science, vol. 9685, Springer, Jun. 2016, pp. 89–104. DOI: 10.1007/978-3-319-38851-9_7.

[32] ——, "Compact flow diagrams for state sequences," *Journal of Experimental Algorithmics*, vol. 22, pp. 1–23, Dec. 2017. DOI: 10.1145/3150525.

[33] K. Buchin, M. Buchin, J. Gudmundsson, M. Löffler, and J. Luo, "Detecting commuting patterns by clustering subtrajectories," *International Journal of Computational Geometry & Applications*, vol. 21, no. 03, pp. 253–282, Jun. 2011. DOI: 10.1142/s0218195911003652.

[34] K. Buchin, M. Buchin, W. Meulemans, and W. Mulzer, "Four soviets walk the dog: Improved bounds for computing the fréchet distance," *Discrete & Computational Geometry*, vol. 58, no. 1, pp. 180–216, Feb. 2017. DOI: 10.1007/s00454-017-9878-7.

[35] M. Buchin, A. Driemel, M. van Kreveld, and V. Sacristan, "Segmenting trajectories: A framework and algorithms using spatiotemporal criteria," *Journal of Spatial Information Science*, vol. 3, pp. 33–63, 2011. DOI: 10.5311/JOSIS.2011.3.66.

[36] M. Buchin, H. Kruckenberg, and A. Kölzsch, "Segmenting trajectories based on movement states," in *Proceedings of the Advances in Spatial Data Handling: Geospatial Dynamics, Geosimulation and Exploratory Visualization (SDH)*, Springer-Verlag, 2013, pp. 15–25.

[37] O. F. Camerino, J. Chaverri, M. T. Anguera, and G. K. Jonsson, "Dynamics of the game in soccer: Detection of t-patterns," en, *European Journal of Sport Science*, vol. 12, no. 3, pp. 216–224, May 2012. DOI: 10.1080/17461391.2011.566362.

[38] H. Cao, O. Wolfson, and G. Trajcevski, "Spatio-temporal data reduction with deterministic error bounds," *The VLDB Journal*, vol. 15, no. 3, pp. 211–228, 2006. DOI: 10.1007/s00778-005-0163-7.

[39] L. Cao and J. Krumm, "From GPS traces to a routable road map," in *Proceedings of the 17th. ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems (GIS '09)*, ACM Press, 2009. DOI: 10.1145/1653771.1653776.

[40] Catapult Sports Ltd. (2015). Catapult USA - Wearable Technology for Elite Sports, [Online]. Available: http://www.catapultsports.com/ (visited on 11/17/2015).

[41] D. Cervone, A. D'Amour, L. Bornn, and K. Goldsberry, "POINTWISE: Predicting points and valuing decisions in real time with NBA optical tracking data," in *Proceedings of the 9th. Annual MIT Sloan Sports Analytics Conference*, MIT, Feb. 2014, pp. 1–9.

[42] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM computing surveys (CSUR)*, vol. 41, no. 3, p. 15, 2009.

[43] Y.-H. Chang, R. Maheswaran, J. Su, S. Kwok, T. Levy, A. Wexler, and K. Squire, "Quantifying shot quality in the NBA," in *Proceedings of the 8th. Annual MIT Sloan Sports Analytics Conference*, Feb. 2014.

[44] M. Charikar, S. Khuller, D. M. Mount, and G. Narasimhan, "Algorithms for facility location problems with outliers," in *Proceedings of the 12th. Annual Symposium on Discrete Algorithms*, S. R. Kosaraju, Ed., ACM/SIAM, Jan. 2001, pp. 642–651.

[45] S. Chawla, J. Estephan, J. Gudmundsson, and M. Horton, "Classification of passes in football matches using spatiotemporal data," *ACM Transactions on Spatial Algorithms and Systems*, vol. 3, no. 2, pp. 1–30, Aug. 2017. DOI: 10.1145/3105576.

[46] S. Chawla and A. Gionis, "k-means--: A unified approach to clustering and outlier detection," in *Proceedings of the 13th. SIAM International Conference on Data Mining*, Society for Industrial & Applied Mathematics (SIAM), May 2013, pp. 189–197. DOI: 10.1137/1.9781611972832.21.

[47] J. Chen, H. M. Le, P. Carr, Y. Yue, and J. J. Little, "Learning online smooth predictors for realtime camera planning using recurrent decision trees," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, Jun. 2016. DOI: 10.1109/cvpr.2016.507.

[48] K. Chen, "On coresets for k-median and k-means clustering in metric and euclidean spaces and their applications," *SIAM Journal on Computing*, vol. 39, no. 3, pp. 923–947, Jan. 2009. DOI: 10.1137/070699007.

[49] ChyronHego Corporation. (2015). Tracab Player Tracking System, [Online]. Available: http://chyronhego.com/sports-data/player-tracking (visited on 08/28/2017).

[50] P. Cintia, F. Giannotti, L. Pappalardo, D. Pedreschi, and M. Malvaldi, "The harsh rule of the goals: Data-driven performance indicators for football teams," in *Proceedings of the IEEE International Conference on Data Science and Advanced Analytics (DSAA)*, IEEE, Oct. 2015, pp. 1–10. DOI: 10.1109/DSAA.2015.7344823.

[51] F. M. Clemente, M. S. Couceiro, F. M. L. Martins, and R. S. Mendes, "Using network metrics to investigate football team players' connections: A pilot study," *Motriz: Revista de Educação Física*, vol. 20, no. 3, pp. 262–271, Sep. 2014. DOI: 10.1590/S1980-65742014000300004.

[52] ——, "Using network metrics in soccer: A macro-analysis," *Journal of Human Kinetics*, vol. 45, no. March, pp. 123–134, 2015. DOI: 10.1515/hukin-2015-0013.

[53] F. M. Clemente, F. Manuel, L. Martins, M. S. Couceiro, R. S. Mendes, and A. J. Figueiredo, "A network approach to characterize the teammates interactions on football: A single match analysis," *Cuadernos de Psicología del Deporte*, vol. 14, no. 3, pp. 141–148, 2014.

[54] F. M. Clemente, F. M. L. Martins, M. S. Couceiro, R. S. Mendes, and A. J. Figueiredo, "Inspecting teammates coverage during attacking plays in a football game: A case study," *International Journal of Performance Analysis in Sport*, vol. 14, no. 2, pp. 384–400, 2014.

[55] F. M. Clemente, F. M. L. Martins, D. Kalamaras, J. Oliveira, P. Oliveira, and R. S. Mendes, "The social network analysis of switzerland football team on FIFA World Cup 2014," *Acta Kinesiologica*, vol. 9, pp. 25–30, 2015.

[56] F. M. Clemente, F. Silva, F. M. L. Martins, D. Kalamaras, and R. S. Mendes, "Performance analysis tool for network analysis on team sports: A case study of FIFA Soccer World Cup 2014," en, *Proceedings of the Institution of Mechanical Engineers, Part P: Journal of Sports Engineering and Technology*, vol. 229, no. 3, pp. 1–13, Jul. 2015. DOI: 10.1177/1754337115597335.

[57] J. Cohen, "Weighted kappa: Nominal scale agreement provision for scaled disagreement or partial credit," *Psychological Bulletin*, vol. 70, no. 4, pp. 213–220, 1968.

[58] H. Collignon, N. Sultan, and C. Santander. (Jul. 2011). The sports market, [Online]. Available: https://www.atkearney.com/documents/10192/6f46b880-f8d1-4909-9960-cc605bb1ff34 (visited on 08/03/2017).

[59] G. Cornuejols, R. Sridharan, and J. Thizy, "A comparison of heuristics and relaxations for the capacitated plant location problem," *European Journal of Operational Research*, vol. 50, no. 3, pp. 280–297, Feb. 1991. DOI: 10.1016/0377-2217(91)90261-s.

[60] C. Cotta, A. M. Mora, J. J. Merelo, and C. Merelo-Molina, "A network analysis of the 2010 FIFA world cup champion team play," *Journal of Systems Science and Complexity*, vol. 26, pp. 21–42, 2013. DOI: 10.1007/s11424-013-2291-2.

[61] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the EM algorithm," *Journal of the Royal Statistical Society. Series B (Methodological)*, vol. 39, no. 1, pp. 1–38, 1977.

[62] D. Douglas and T. Peucker, "Algorithms for the reduction of the number of points required to represent a digitized line or its caricature," *Cartographica: The International Journal for Geographic Information and Geovisualization*, vol. 10, no. 2, pp. 112–122, Dec. 1973. DOI: 10.3138/fm57-6770-u75u-7727.

[63] Z. Drezner and H. W. Hamacher, Eds., *Facility Location: Applications and Theory*. Springer, 2002.

[64] A. Driemel, S. Har-Peled, and C. Wenk, "Approximating the fréchet distance for realistic curves in near linear time," *Discrete & Computational Geometry*, vol. 48, no. 1, pp. 94–127, 2012. DOI: 10.1007/s00454-012-9402-z.

[65] A. Driemel, A. Krivošija, and C. Sohler, "Clustering time series under the fréchet distance," in *Proceedings of the27th. Annual ACM-SIAM Symposium on Discrete Algorithms (SODA '16)*, Society for Industrial and Applied Mathematics, Jan. 2016, pp. 766–785. DOI: 10.1137/1.9781611974331.ch55.

[66] J. Duch, J. S. Waitzman, and L. A. N. Amaral, "Quantifying the performance of individual players in a team activity," *PLoS ONE*, vol. 5, no. 6, E. Scalas, Ed., e10937, Jun. 2010. DOI: 10.1371/journal.pone.0010937.

[67] N. Eagle, Y.-A. de Montjoye, and L. M. Bettencourt, "Community computing: Comparisons between rural and urban societies using mobile phone data," in *Proceedings of the International Conference on Computational Science and Engineering*, IEEE, 2009. DOI: 10.1109/cse.2009.91.

[68] T. Eiter and H. Mannila, "Computing discrete fréchet distance," Unpublished work, 1994.

[69] J. H. Fewell, D. Armbruster, J. Ingraham, A. Petersen, and J. S. Waters, "Basketball teams as strategic networks," *PLoS ONE*, vol. 7, no. 11, e47445, 2012.

[70] FIFA. (2017). Castrol Index, [Online]. Available: http://www.fifa.com/worldcup/statistics/castrol-index/index.html (visited on 08/04/2017).

[71] M. L. Fisher, "The lagrangian relaxation method for solving integer programming problems," *Management Science*, vol. 27, no. 1, pp. 1–18, Jan. 1981. DOI: 10.1287/mnsc.27.1.1.

[72] S. Fonseca, J. Milho, B. Travassos, and D. Araújo, "Spatial dynamics of team sports exposed by voronoi diagrams," *Human Movement Science*, vol. 31, no. 6, pp. 1652–1659, 2012. DOI: 10.1016/j.humov.2012.04.006.

[73] S. Fortune, "A sweepline algorithm for voronoi diagrams," *Algorithmica*, vol. 2, pp. 153–174, 1987. DOI: 10.1007/BF01840357.

[74] A. Franks, A. Miller, L. Bornn, and K. Goldsberry, "Characterizing the spatial structure of defensive skill in professional basketball," *The Annals of Applied Statistics*, vol. 9, no. 1, pp. 94–121, Mar. 2015. DOI: 10.1214/14-AOAS799.

[75] ——, "Counterpoints: Advanced defensive metrics for NBA basketball," in *Proceedings of the 9th. Annual MIT Sloan Sports Analytics Conference*, MIT, 2015, pp. 1–8.

[76] I. M. Franks and G. Miller, "Eyewitness testimony in sport," *Journal of Sport Behavior*, vol. 9, no. 1, pp. 38–46, 1986.

[77] C. B. Fraser and R. W. Irving, "Approximation algorithms for the shortest common supersequence," *Nordic Journal of Computing*, vol. 2, no. 3, pp. 303–325, 1995.

[78] M. M. Fréchet, "Sur quelques points du calcul fonctionnel," *Rendiconti del Circolo Matematico di Palermo*, vol. 22, no. 1, pp. 1–72, 1906. DOI: 10.1007/bf03018603.

[79] L. C. Freeman, "A set of measures of centrality based on betweenness," *Sociometry*, vol. 40, no. 1, pp. 35–41, Mar. 1977. DOI: 10.2307/3033543.

[80] ——, "Centrality in social networks conceptual clarification," *Social Networks*, vol. 1, no. 3, pp. 215–239, Jan. 1978. DOI: 10.1016/0378-8733(78)90021-7.

[81] W. Frencken, K. Lemmink, N. Delleman, and C. Visscher, "Oscillations of centroid position and surface area of soccer teams in small-sided games," *European Journal of Sport Science*, vol. 11, no. 4, pp. 215–223, Jul. 2011. DOI: 10.1080/17461391.2010.499967.

[82] B. J. Frey and D. Dueck, "Clustering by passing messages between data points," *Science*, vol. 315, no. 5814, pp. 972–976, Feb. 2007. DOI: 10.1126/science.1136800. eprint: http://science.sciencemag.org/content/315/5814/972.full.pdf.

[83] M. J. Fry and J. W. Ohlmann, "Introduction to the special issue on analytics in sports, part i: General sports applications," *Interfaces*, vol. 42, no. 2, pp. 105–108, Apr. 2012. DOI: 10.1287/inte.1120.0633.

[84] A. Fujimura and K. Sugihara, "Geometric analysis and quantitative evaluation of sport teamwork," *Systems and Computers in Japan*, vol. 36, no. 6, pp. 49–58, 2005. DOI: 10.1002/scj.20254.

[85] S. J. Gaffney, A. W. Robertson, P. Smyth, S. J. Camargo, and M. Ghil, "Probabilistic clustering of extratropical cyclones using regression mixture models," *Climate Dynamics*, vol. 29, no. 4, pp. 423–440, Mar. 2007. DOI: 10.1007/s00382-007-0235-z.

[86] S. J. Gaffney and P. Smyth, "Trajectory clustering with mixtures of regression models," in *Proceedings of the 5th. ACM SIGKDD international conference on Knowledge discovery and data mining (KDD '99)*, ACM Press, 1999. DOI: 10.1145/312129.312198.

[87] G. Gan and M. K.-P. Ng, "*k*-means clustering with outlier removal," *Pattern Recognition Letters*, vol. 90, pp. 8–14, 2017.

[88] I. E. Givoni and B. J. Frey, "A binary variable model for affinity propagation," *Neural Computation*, vol. 21, no. 6, pp. 1589–1600, Jun. 2009. DOI: 10.1162/neco.2009.05-08-785.

[89] K. Goldsberry, "Courtvision: New visual and spatial analytics for the NBA," in *Proceedings of the 6th. Annual MIT Sloan Sports Analytics Conference*, MIT, Mar. 2012, pp. 1–7.

[90] K. Goldsberry and E. Weiss, "The dwight effect: A new ensemble of interior defense analytics for the NBA," in *Proc. 7th. Annual MIT Sloan Sports Analytics Conference*, MIT, Feb. 2013, pp. 1–11.

[91] T. F. Gonzalez, "Clustering to minimize the maximum intercluster distance," *Theoretical Computer Science*, vol. 38, pp. 293–306, 1985. DOI: 10.1016/0304-3975(85)90224-5.

[92] P. Gould and A. Gatrell, "A structural analysis of a game: The liverpool v manchester united cup final of 1977," *Social Networks*, vol. 2, no. 3, pp. 253–273, Jan. 1979. DOI: 10.1016/0378-8733(79)90017-0.

[93] T. U. Grund, "Network structure and team performance: The case of English Premier League soccer teams," *Social Networks*, vol. 34, no. 4, pp. 682–690, 2012. DOI: 10.1016/j.socnet.2012.08.004.

[94] J. Gudmundsson and M. Horton, "Spatio-temporal analysis of team sports," *ACM Computing Surveys*, vol. 50, no. 2, 22:1–22:34, 2017. DOI: 10.1145/3054132.

[95] J. Gudmundsson and T. Wolle, "Football analysis using spatio-temporal tools," in *Proceedings of the 20th. International Conference on Advances in Geographic Information Systems (SIGSPATIAL '12)*, ACM, Nov. 2012, pp. 566–569. DOI: 10.1145/2424321.2424417.

[96] ——, "Football analysis using spatio-temporal tools," *Computers, Environment and Urban Systems*, vol. 47, pp. 16–27, 2014. DOI: 10.1016/j.compenvurbsys.2013.09.004.

[97] S. Guha and S. Khuller, "Greedy strikes back: Improved facility location algorithms," *Journal of Algorithms*, vol. 31, no. 1, pp. 228–248, Apr. 1999. DOI: 10.1006/jagm.1998.0993.

[98] L. J. Guibas, J. Hershberger, J. S. B. Mitchell, and J. Snoeyink, "Approximating polygons and subdivisions with minimum link paths," *International Journal of Computational Geometry & Applications*, vol. 3, no. 04, pp. 383–415, Dec. 1993. DOI: 10.1142/S0218195993000257.

[99] S. Gupta, R. Kumar, K. Lu, B. Moseley, and S. Vassilvitskii, "Local search methods for k-means with outliers," *Proceedings of the VLDB Endowment*, vol. 10, no. 7, pp. 757–768, 2017.

[100] I. Guyon and A. Elisseeff, "An introduction to variable and feature selection," *Journal of Machine Learning Research*, vol. 3, pp. 1157–1182, 2003.

[101] H. Haken, J. A. S. Kelso, and H. Bunz, "A theoretical model of phase transitions in human hand movements," *Biological Cybernetics*, vol. 51, no. 5, pp. 347–356, Feb. 1985. DOI: 10.1007/BF00336922.

[102] C.-S. Han, S.-X. Jia, L. Zhang, and C.-C. Shu, "Sub-trajectory clustering algorithm based on speed restriction," *Computer Engineering*, vol. 37, no. 7, 2011.

[103] F. Hausdorff, *Mengenlehre*. Walter de Gruyter, 1927.

[104] M. Held and R. M. Karp, "The traveling-salesman problem and minimum spanning trees," *Operations Research*, vol. 18, no. 6, pp. 1138–1162, Dec. 1970. DOI: 10.1287/opre.18.6.1138.

[105] ——, "The traveling-salesman problem and minimum spanning trees: Part II," *Mathematical Programming*, vol. 1, no. 1, pp. 6–25, Dec. 1971. DOI: 10.1007/bf01584070.

[106] A. Hervieu and P. Bouthemy, "Understanding sports video using players trajectories," in *Intelligent Video Event Analysis and Understanding*, ser. Studies in Computational Intelligence, J. Zhang, L. Shao, L. Zhang, and G. A. Jones, Eds., vol. 332, Springer Berlin Heidelberg, 2011, pp. 125–153. DOI: 10.1007/978-3-642-17554-1_7.

[107] A. Hervieu, P. Bouthemy, and J.-P. L. Cadre, "Trajectory-based handball video understanding," in *Proceeding of the ACM International Conference on Image and Video Retrieval (CIVR '09)*, ACM Press, Jul. 2009, p. 1.

[108] D. S. Hochbaum and D. B. Shmoys, "A best possible heuristic for the k-center problem," *Mathematics of Operations Research*, vol. 10, no. 2, pp. 180–184, May 1985. DOI: 10.1287/moor.10.2.180.

[109] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997. DOI: 10.1162/neco.1997.9.8.1735.

[110] M. Horton, "Automated classification of passes in football matches," Masters by Coursework Thesis, The University of Sydney, 2013.

[111] M. Horton, J. Gudmundsson, S. Chawla, and J. Estephan, "Automated classification of passing in football," in *Proceedings of the 19th. Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining (PAKDD '15), Part II*, ser. Lecture Notes in Computer Science, vol. 9078, Springer, May 2015, pp. 319–330. DOI: 10.1007/978-3-319-18032-8_25.

[112] W.-L. Hsu and G. L. Nemhauser, "Easy and hard bottleneck location problems," *Discrete Applied Mathematics*, vol. 1, no. 3, pp. 209–215, Nov. 1979. DOI: 10.1016/0166-218x(79)90044-1.

[113] H. Imai and M. Iri, "Computational-geometric methods for polygonal approximations of a curve," *Computer Vision, Graphics, and Image Processing*, vol. 36, no. 1, pp. 31–41, Oct. 1986. DOI: 10.1016/s0734-189x(86)80027-5.

[114] ——, "Polygonal approximations of a curve," *Computational Morphology*, pp. 71–86, 1988.

[115] Impire AG. (Aug. 2017). Impire AG, [Online]. Available: http://www.bundesliga-datenbank.de/en/products/ (visited on 08/28/2017).

[116] S. S. Intille and A. F. Bobick, "A framework for recognizing multi-agent action from visual evidence," in *Proceedings of the 16th. National Conference on Artificial Intelligence and Eleventh Conference on Innovative Applications of Artificial Intelligence*, AAAI Press / The MIT Press, Jul. 1999, pp. 518–525.

[117] K. Jain and V. V. Vazirani, "Approximation algorithms for metric facility location and k-median problems using the primal-dual schema and lagrangian relaxation," *Journal of the ACM*, vol. 48, no. 2, pp. 274–296, Mar. 2001. DOI: 10.1145/375827.375845.

[118] H. Janetzko, D. Sacha, M. Stein, T. Schreck, D. A. Keim, and O. Deussen, "Feature-driven visual analytics of soccer data," in *Proceedings of the IEEE Conference on Visual Analytics Science and Technology (VAST '14)*, IEEE, Oct. 2014, pp. 13–22. DOI: 10.1109/VAST.2014.7042477.

[119] R. Jenatton, J. Audibert, and F. R. Bach, "Structured variable selection with sparsity-inducing norms," *Journal of Machine Learning Research*, vol. 12, pp. 2777–2824, 2011.

[120] C. Kang, J. Hwang, and K. Li, "Trajectory analysis for soccer players," in *Workshops Proceedings of the 6th. IEEE International Conference on Data Mining (ICDM '06)*, IEEE Computer Society, Dec. 2006, pp. 377–381. DOI: 10.1109/ICDMW.2006.160.

[121] H.-C. Kim, O. Kwon, and K.-J. Li, "Spatial and spatiotemporal analysis of soccer," in *Proceedings of the 19th. ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, ACM, Nov. 2011, pp. 385–388. DOI: 10.1145/2093973.2094029.

[122] K. Kim, M. Grundmann, A. Shamir, I. Matthews, J. Hodgins, and I. Essa, "Motion fields to predict play evolution in dynamic sport scenes," in *Pro. IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR '10)*, IEEE, Jun. 2010, pp. 840–847. DOI: 10.1109/CVPR.2010.5540128.

[123] H. Kitano, M. Asada, Y. Kuniyoshi, I. Noda, and E. Osawa, "RoboCup," in *Proceedings of the 1st. International Conference on Autonomous Agents (AGENTS'97)*, ACM Press, Feb. 1997, pp. 340–347. DOI: 10.1145/267658.267738.

[124] D. Kollar and N. Friedman, *Probabilistic Graphical Models: Principles and Techniques*. The MIT Press, 2009.

[125] J. B. Kruskal, "On the shortest spanning subtree of a graph and the traveling salesman problem," *Proceedings of the American Mathematical Society*, vol. 7, no. 1, pp. 48–50, 1956.

[126] F. R. Kschischang, B. J. Frey, and H. Loeliger, "Factor graphs and the sum-product algorithm," *IEEE Transactions on Information Theory*, vol. 47, no. 2, pp. 498–519, Feb. 2001. DOI: 10.1109/18.910572.

[127] J. Kubatko, D. Oliver, K. Pelton, and D. T. Rosenbaum, "A starting point for analyzing basketball statistics," *Journal of Quantitative Analysis in Sports*, vol. 3, no. 3, pp. 1–22, Jan. 2007. DOI: 10.2202/1559-0410.1070.

[128] H. W. Kuhn, "The Hungarian method for the assignment problem," *Naval Research Logistics (NRL)*, vol. 52, no. 1, pp. 7–21, 2005.

[129] D. Lamoreaux, "Baseball in the late nineteenth century: The source of its appeal," *The Journal of Popular Culture*, vol. 11, no. 3, pp. 597–613, 1977. DOI: 10.1111/j.0022-3840.1977.00597.x.

[130] C. Landsea, J. Franklin, and J. Beven. (Apr. 2014). The revised atlantic hurricane database (HURDAT2), [Online]. Available: http://www.aoml.noaa.gov/hrd/hurdat/newhurdat-format.pdf (visited on 08/28/2017).

[131] N. Lazic, B. J. Frey, and P. Aarabi, "Solving the uncapacitated facility location problem using message passing algorithms," in *Proceedings of the 13th. International Conference on Artificial Intelligence and Statistics (AISTATS)*, ser. JMLR Proceedings, vol. 9, JMLR.org, May 2010, pp. 429–436.

[132] H. M. Le, P. Carr, Y. Yue, and P. Lucey, "Data-driven ghosting using deep imitation learning," in *Proceedings of the 11th. Annual MIT Sloan Sports Analytics Conference*, MIT, Feb. 2015, pp. 1–8.

[133] H. M. Le, Y. Yue, P. Carr, and P. Lucey, "Coordinated multi-agent imitation learning," in *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, D. Precup and Y. W. Teh, Eds., ser. Proceedings of Machine Learning Research, vol. 70, PMLR, 2017, pp. 1995–2003.

[134] M. Lewis, *Moneyball: The Art of Winning an Unfair Game*. WW Norton and Company, 2004.

[135] R. Li and R. Chellappa, "Group motion segmentation using a spatio-temporal driving force model," in *Proc. IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR '10)*, IEEE, Jun. 2010, pp. 2038–2045. DOI: 10.1109/CVPR.2010.5539880.

[136] R. Li, R. Chellappa, and S. K. Zhou, "Learning multi-modal densities on discriminative temporal interaction manifold for group activity recognition," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR '09)*, IEEE, Jun. 2009, pp. 2450–2457. DOI: 10.1109/CVPR.2009.5206676.

[137] Y. S. Lincoln and E. G. Guba, *Naturalistic Inquiry*. SAGE Publications, Inc, Apr. 11, 1985, vol. 75, 416 pp.

[138] W. Liu and S. Chawla, "A quadratic mean based supervised learning model for managing data skewness," in *Proceedings of the 11th. SIAM International Conference on Data Mining (SDM '11)*, SIAM / Omnipress, Apr. 2011, pp. 188–198. DOI: 10.1137/1.9781611972818.17.

[139] P. Lucey, A. Bialkowski, G. P. K. Carr, E. Foote, and I. Matthews, "Characterizing multi-agent team behavior from partial team tracings: Evidence from the English Premier League," in *Proceedings of the 26th. AAAI Conference on Artificial Intelligence*, AAAI Press, Jul. 2012.

[140] P. Lucey, A. Bialkowski, G. P. K. Carr, S. Morgan, I. Matthews, and Y. Sheikh, "Representing and discovering adversarial team behaviors using player roles," in *Proceedings of the 26th. IEEE Conference on Computer Vision and Pattern Recognition (CVPR '13)*, IEEE, Jun. 2013, pp. 2706–2713. DOI: 10.1109/CVPR.2013.349.

[141] P. Lucey, A. Bialkowski, G. P. K. Carr, Y. Yue, and I. Matthews, "How to get an open shot?: Analyzing team movement in basketball using tracking data," in *Proceedings of the 8th. Annual MIT Sloan Sports Analytics Conference Conference*, MIT, Feb. 2014, pp. 1–10.

[142] P. Lucey, A. Bialkowski, M. Monfort, P. Carr, and I. Matthews, "'"quality vs quantity": Improved shot prediction in soccer using strategic features from spatiotemporal data," in *Proceedings of the 8th. Annual MIT Sloan Sports Analytics Conference*, MIT, Feb. 2014, pp. 1–9.

[143] P. Lucey, D. Oliver, G. P. K. Carr, J. Roth, and I. Matthews, "Assessing team strategy using spatiotemporal data," in *Proceedings of the 19th. ACM International Conference on Knowledge Discovery and Data Mining, (KDD '13)*, ACM, Aug. 2013, pp. 1366–1374. DOI: 10.1145/2487575.2488191.

[144] P. Lucey, Y. Yue, J. Wiens, and S. Morgan. (2016). SIGKDD 2016 Workshop on Large Scale Sports Analytics, [Online]. Available: http://www.large-scale-sports-analytics.org/ (visited on 08/26/2016).

[145] C. Lund and M. Yannakakis, "On the hardness of approximating minimization problems," *Journal of the ACM*, vol. 41, no. 5, pp. 960–981, 1994. DOI: 10.1145/185675.306789.

[146] R. Maheswaran, Y.-H. Chang, A. Henehan, and S. Danesis, "Deconstructing the rebound with optical tracking data," in *Proceedings of the 6th. Annual MIT Sloan Sports Analytics Conference*, Feb. 2012.

[147] R. Maheswaran, Y.-H. Chang, J. Su, S. Kwok, T. Levy, A. Wexler, and N. Hollingsworth, "The three dimensions of rebounding," in *Proceedings of the 8th. Annual MIT Sloan Sports Analytics Conference*, Feb. 2014.

[148] P. McCullagh, "Regression models for ordinal data," *Journal of the Royal Statistical Society. Series B (Methodological)*, vol. 42, no. 2, pp. 109–142, 1980.

[149] ——, "Generalized linear models," *European Journal of Operational Research*, vol. 16, no. 3, pp. 285–292, Jun. 1984. DOI: http://dx.doi.org/10.1016/0377-2217(84)90282-0.

[150] A. Miller, L. Bornn, R. P. Adams, and K. Goldsberry, "Factorized point process intensities: A spatial analysis of professional basketball," in *Proceedings of the 31st. International Conference on Machine Learning, (ICML '14)*, JMLR.org, Jun. 2014, pp. 235–243.

[151] R. von Mises and H. Pollaczek-Geiringer, "Praktische verfahren der gleichungsauflösung," *Journal of Applied Mathematics and Mechanics / Zeitschrift für Angewandte Mathematik und Mechanik*, vol. 9, no. 2, pp. 152–164, 1929. DOI: 10.1002/zamm.19290090206.

[152] MIT Sloan EMS Club. (2017). MIT Sloan Sports Analytics Conference, [Online]. Available: https://www.sloansportsconference.com/ (visited on 08/03/2017).

[153] R. Nakanishi, J. Maeno, K. Murakami, and T. Naruse, "An approximate computation of the dominant region diagram for the real-time analysis of group behaviors," in *Proceedings of the 13th. Annual RoboCup International Symposium*, Springer, Jun. 2009, pp. 228–239. DOI: 10.1007/978-3-642-11876-0_20.

[154] T. Narizuka, K. Yamamoto, and Y. Yamazaki, "Statistical properties of position-dependent ball-passing networks in football games," *Physica A: Statistical Mechanics and its Applications*, vol. 412, pp. 157–168, Oct. 2014. DOI: 10.1016/j.physa.2014.06.037.

[155] M. E. J. Newman, *Networks: An Introduction*. Oxford University Press, Mar. 2010, p. 784.

[156] M. Ojala and G. C. Garriga, "Permutation tests for studying classifier performance," *Journal of Machine Learning Research*, vol. 11, pp. 1833–1863, 2010.

[157] Opta Sports Ltd. (2015). Opta Live Performance Data, [Online]. Available: http://www.optasports.com/about/what-we-do/live-performance-data.aspx (visited on 08/01/2017).

[158] L. Ott, L. X. Pang, F. T. Ramos, and S. Chawla, "On integrated clustering and outlier detection," in *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems*, Dec. 2014, pp. 1359–1367.

[159] P. Passos, K. Davids, D. Araújo, N. Paz, J. Minguéns, and J. Mendes, "Networks as a novel tool for studying team ball sports as complex social systems," *Journal of Science and Medicine in Sport*, vol. 14, no. 2, pp. 170–176, Mar. 2011. DOI: 10.1016/j.jsams.2010.10.459.

[160] J. L. Peña and H. Touchette, "A network theory analysis of football strategies," in *Proc. Euromech Physics of Sports Conference*, LÉditions de l'École Polytechnique, Apr. 2012, pp. 517–528.

[161] C. Perin, R. Vuillemot, and J.-D. Fekete, "SoccerStories: A kick-off for visual soccer analysis," *IEEE Transactions on Visualization and Computer Graphics*, vol. 19, no. 12, pp. 2506–2515, 2013. DOI: 10.1109/TVCG.2013.192.

[162] M. Perše, M. Kristan, S. Kovačič, G. Vučkovič, and J. Perš, "A trajectory-based analysis of coordinated team activity in a basketball game," *Computer Vision and Image Understanding*, vol. 113, no. 5, pp. 612–621, May 2009. DOI: 10.1016/j.cviu.2008.03.001.

[163] K. Pietrzak, "On the parameterized complexity of the fixed alphabet shortest common supersequence and longest common subsequence problems," *Journal of Computer and System Sciences*, vol. 67, no. 4, pp. 757–771, 2003. DOI: 10.1016/S0022-0000(03)00078-3.

[164] H. Pileggi, C. D. Stolper, J. M. Boyle, and J. T. Stasko, "SnapShot: Visualization to propel ice hockey analytics," *IEEE Transactions on Visualization and Computer Graphics*, vol. 18, no. 12, pp. 2819–2828, Dec. 2012. DOI: 10.1109/TVCG.2012.263.

[165] P. Power, H. Ruiz, X. Wei, and P. Lucey, "Not all passes are created equal: Objectively measuring the risk and reward of passes in soccer from tracking data," in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD '17*, ACM Press, Aug. 2017. DOI: 10.1145/3097983.3098051.

[166] K. Räihä and E. Ukkonen, "The shortest common supersequence problem over binary alphabet is np-complete," *Theoretical Computer Science*, vol. 16, pp. 187–198, 1981. DOI: 10.1016/0304-3975(81)90075-X.

[167] C. Reep and B. Benjamin, "Skill and chance in association football," *Journal of the Royal Statistical Society. Series A (General)*, vol. 131, no. 4, pp. 581–585, 1968. DOI: 10.2307/2343726.

[168] B. J. Reich, J. S. Hodges, B. P. Carlin, and A. M. Reich, "A spatial analysis of basketball shot chart data," *The American Statistician*, vol. 60, no. 1, pp. 3–12, Feb. 2006. DOI: 10.1198/000313006X90305.

[169] A. Rosenberg and J. Hirschberg, "V-measure: A conditional entropy-based external cluster evaluation measure," in *Proceedings of the Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, ACL, Jun. 2007, pp. 410–420.

[170] W. Rossmann, *Lie Groups: An Introduction Through Linear Groups*. Oxford University Press, 2002, p. 265.

[171] H. Ruiz, P. Power, X. Wei, and P. Lucey, "'the leicester city fairytale?': Utilizing new soccer analytics tools to compare performance in the 15/16 & 16/17 epl seasons," in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD '17*, ACM Press, Aug. 2017. DOI: 10.1145/3097983.3098121.

[172] C. Seiffert, T. M. Khoshgoftaar, J. V. Hulse, and A. Napolitano, "RUSBoost: A hybrid approach to alleviating class imbalance," *IEEE Transactions on Systems, Man, and Cybernetics, Part A*, vol. 40, no. 1, pp. 185–197, 2010. DOI: 10.1109/TSMCA.2009.2029559.

[173] C. E. Shannon, "A mathematical theory of communication," *Mobile Computing and Communications Review*, vol. 5, no. 1, pp. 3–55, 2001. DOI: 10.1145/584091.584093.

[174] N. Z. Shor, *Minimization Methods for Non-Differentiable Functions and Applications (Springer Series in Computational Mathematics)*. Springer-Verlag New York, Inc., 1985.

[175] A. Shortridge, K. Goldsberry, and M. Adams, "Creating space to shoot: Quantifying spatial relative field goal efficiency in basketball," *Journal of Quantitative Analysis in Sports*, vol. 10, no. 3, pp. 303–313, Jan. 2014. DOI: 10.1515/jqas-2013-0094.

[176] B. Skinner, "The price of anarchy in basketball," *Journal of Quantitative Analysis in Sports*, vol. 6, no. 1, pp. 1–16, Jan. 2010. DOI: 10.2202/1559-0410.1217.

[177] Sports Reference LLC. (2017). Sports reference, [Online]. Available: https://www.sports-reference.com/ (visited on 08/04/2017).

[178] SportVision Inc. (2015). NHL, sportvision test program to track players, puck, [Online]. Available: http://www.sportvision.com/news/nhl-sportvision-test-program-track-players-puck (visited on 08/28/2017).

[179] STATS LLC. (2017). Basketball player tracking for pro teams, [Online]. Available: https://www.stats.com/sportvu-basketball/ (visited on 08/01/2017).

[180] ——, (Jul. 2017). STATS SportVU® football player tracking, [Online]. Available: https://www.stats.com/sportvu-football/ (visited on 07/01/2017).

[181] M. Stein, J. Häußler, D. Jäckle, H. Janetzko, T. Schreck, and D. Keim, "Visual soccer analytics: Understanding the characteristics of collective team movement based on feature-driven analysis and abstraction," *ISPRS International Journal of Geo-Information*, vol. 4, no. 4, pp. 2159–2184, Oct. 2015. DOI: 10.3390/ijgi4042159.

[182] T. Taki and J. Hasegawa, "Dominant region: A basic feature for group motion analysis and its application to teamwork evaluation in soccer games," in *Proc. 6th. IS&T/SPIE Conference on Videometrics*, International Society for Optics and Photonics, Dec. 1998, pp. 48–57. DOI: 10.1117/12.333797.

[183] ——, "Visualization of dominant region in team games and its application to teamwork analysis," in *Proceedings of the IEEE International Conference on Computer Graphics*, IEEE, Jun. 2000, pp. 227–235. DOI: 10.1109/CGI.2000.852338.

[184] T. Taki, J. Hasegawa, and T. Fukumura, "Development of motion analysis system for quantitative evaluation of teamwork in soccer games," in *Proceedings of the International Conference on Image Processing*, IEEE, Sep. 1996, pp. 815–818. DOI: 10.1109/ICIP.1996.560865.

[185] H. C. Tijms, *Stochastic Modeling and Analysis: A Computational Approach (Probability & Mathematical Statistics)*. John Wiley & Sons, Inc., 1986.

[186] M. R. Tora, J. Chen, and J. J. Little, "Classification of puck possession events in ice hockey," in *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, IEEE, Jul. 2017. DOI: 10.1109/cvprw.2017.24.

[187] F. Ueda, H. Masaaki, and H. Hiroyuki, "The causal relationship between dominant region and offense-defense performance—focusing on the time of ball acquisition," *Football Science*, vol. 11, pp. 1–17, 2014.

[188] J. Van Haaren, V. Dzyuba, S. Hannosset, and J. Davis, "Automatically discovering offensive patterns in soccer match data," in *Proceedings of the 14th. International Symposium of Advances in Intelligent Data Analysis (IDA'15)*, ser. Lecture Notes in Computer Science, vol. 9385, Springer, Oct. 2015, pp. 286–297. DOI: 10.1007/978-3-319-24465-5_25.

[189] V. N. Vapnik, *The Nature of Statistical Learning Theory*. Springer-Verlag, 1995, p. 188.

[190] M. Vlachos, D. Gunopulos, and G. Kollios, "Discovering similar multidimensional trajectories," in *Proceedings of the 18th. International Conference on Data Engineering (ICDE '02)*, R. Agrawal and K. R. Dittrich, Eds., IEEE Computer Society, Feb. 2002, pp. 673–684. DOI: 10.1109/ICDE.2002.994784.

[191] Q. Wang, H. Zhu, W. Hu, Z. Shen, and Y. Yao, "Discerning tactical patterns for professional soccer teams," in *Proceedings of the 21st. ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'15)*, ACM, Aug. 2015, pp. 2197–2206. DOI: 10.1145/2783258.2788577.

[192] S. Wasserman and K. Faust, *Social Network Analysis: Methods and Applications. Structural Analysis in the Social Sciences*. Cambridge University Press, 1997.

[193] X. Wei, P. Lucey, S. Vidas, S. Morgan, and S. Sridharan, "Forecasting events using an augmented hidden conditional random field," in *Proceedings of the 12th. Asian Conference on Computer Vision*, ser. Lecture Notes in Computer Science, vol. 9006, Springer International Publishing, Nov. 2015, pp. 569–582. DOI: 10.1007/978-3-319-16817-3_37.

[194] X. Wei, L. Sha, P. Lucey, S. Morgan, and S. Sridharan, "Large-scale analysis of formations in soccer," in *Proceedings of the International Conference on Digital Image Computing: Techniques and Applications (DICTA)*, IEEE, Nov. 2013, pp. 1–8. DOI: 10.1109/DICTA.2013.6691503.

[195] P. Weiner, "Linear pattern matching algorithms," in *Proceedings of the 14th. Annual Symposium on Switching and Automata Theory*, IEEE Computer Society, Oct. 1973, pp. 1–11. DOI: 10.1109/SWAT.1973.13.

[196] J. Wiens, G. Balakrishnan, J. Brooks, and J. Guttag, "To crash or not to crash: A quantitative look at the relationship between offensive rebounding and transition defense in the NBA," in *Proceedings of the 6th. Annual MIT Sloan Sports Analytics Conference*, MIT, Mar. 2013, pp. 1–7.

[197] J. Wilson, *Inverting the Pyramid: The History of Football Tactics*. Hachette UK, 2010.

[198] Y. Yue, P. Lucey, G. P. K. Carr, A. Bialkowski, and I. Matthews, "Learning fine-grained spatial models for dynamic sports play prediction," in *Proceedings of the IEEE International Conference on Data Mining (ICDM'14)*, IEEE, Dec. 2014, pp. 670–679. DOI: 10.1109/ICDM.2014.106.

[199] S. Zheng, Y. Yue, and J. Hobbs, "Generating long-term trajectories using deep hierarchical networks," in *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, D. D. Lee, M. Sugiyama, U. von Luxburg, I. Guyon, and R. Garnett, Eds., 2016, pp. 1543–1551.

[200] Y. Zheng and X. Zhou, *Computing with Spatial Trajectories*. Springer Publishing Company, Incorporated, 2011, p. 333.