Applying Record Value Theory in Combinatorial Optimization with Application to Environmental Statistics

by

Yu Wang

B.Sc., Simon Fraser University, 2016

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF
THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF SCIENCE

in

The Faculty of Graduate and Postdoctoral Studies

(Statistics)

THE UNIVERSITY OF BRITISH COLUMBIA

(Vancouver)

August 2018

© Yu Wang 2018
Committee Page

The following individuals certify that they have read, and recommend to the Faculty of Graduate and Postdoctoral Studies for acceptance of this thesis.

Examining Committee:
James V. Zidek, Professor, Department of Statistics
Supervisor
Nhu D. Le, Adjunct Professor, Department of Statistics
Co-supervisor
Abstract

We consider the problem of optimal subset selection from a set of correlated random variables. In particular, we consider the associated combinatorial optimization problem of maximizing the determinant of a symmetric positive semidefinite matrix that characterizes the chosen subset. This problem arises in many domains, such as experimental designs, regression modelling, and environmental statistics. In this thesis, we attempt to establish an efficient polynomial-time algorithm for approximating the optimal solution to the problem. Firstly, we employ determinantal point processes, a special class of spatial point processes, to develop an easy-to-implement sampling-based stochastic search algorithm for the task of finding approximations to the combinatorial optimization problem. Secondly, we establish theoretical tools for assessing the quality of those approximations using statistical results from record value theory, the study of record values and related statistics from a sequence of observations.
Lay Summary

The increasing recognition of the association between adverse human health conditions and many environmental substances as well as processes has led to the need to monitor them. An important problem that arises in environmental statistics is the design of the locations of the monitoring stations for those environmental processes of interest. One particular design criterion for monitoring networks that tries to reduce the uncertainty about predictions of unseen processes is called the maximum-entropy design. However, this design criterion involves a hard optimization problem that is computationally intractable for large data sets. The contributions of this thesis are twofold. Firstly, we examine a probabilistic model that can be implemented efficiently to approximate the underlying optimization problem. Specifically, we compare its computational performance with existing methods. Secondly, we attempt to establish statistically sound tools for assessing the quality of the approximations.
Preface

This thesis is an original work of the author, Yu Wang, under the supervision of Dr. James V. Zidek and Dr. Nhu D. Le.

Dr. Camila M. Casquilho-Resende provided her code, which was used at the early stages of development of chapter 3. Dr. Jon Lee and Dr. Maurice Queyranne provided valuable suggestions on parts of chapter 3 and recommended a software for computational experiments conducted in the chapter. A version of chapter 3 was submitted for peer review. The manuscript is named “Stochastic Approximation Algorithms in Combinatorial Optimization” by Wang, Y., Le, N. D., Zidek, J. V. The idea was jointly developed by myself, Dr. Nhu D. Le, and Dr. James V. Zidek. I conducted all the computational experiments and the majority of the writing. An electronic version can be found online at arxiv:1709.00151.
Table of Contents

Abstract .............................................................. iii
Lay Summary ............................................................ iv
Preface ................................................................. v
Table of Contents ...................................................... vi
List of Tables ............................................................ ix
List of Figures ........................................................... x
List of Algorithms ...................................................... xii
Acknowledgements ...................................................... xiii
Dedication ............................................................... xiv

1 Introduction ......................................................... 1
   1.1 Maximum-Entropy Design Problem ......................... 2
   1.2 Spatial Point Processes ..................................... 4
   1.3 Record Values Theory ...................................... 5

2 Overview of the Combinatorial Optimization Problem ........ 6
   2.1 Exact Methods for Finding the Optimum .................. 7
      2.1.1 Branch-and-bound .................................. 7
   2.2 Heuristics for Approximating the Optimum .............. 10
      2.2.1 Greedy algorithm .................................. 10
      2.2.2 Genetic algorithm ................................. 12
# Table of Contents

2.3 Discussion ................................................. 13

3 Determinantal Point Processes .............................. 14
  3.1 Definitions ............................................. 14
    3.1.1 General Poisson point processes ................. 15
    3.1.2 General determinantal point processes .......... 15
    3.1.3 $k$-DPPs ........................................... 17
  3.2 Sampling from $k$-DPPs ................................ 17
    3.2.1 Independent sampling ............................ 18
    3.2.2 MCMC sampling .................................. 21
  3.3 Sampling-based solution strategy ....................... 23
  3.4 Computational Performances ............................ 24
    3.4.1 Optimizing maximum-entropy designs for monitoring networks ......................... 24
    3.4.2 Synthetic data with a large number of points ...... 26
  3.5 Discussion ............................................. 28

4 Record Values Theory ....................................... 30
  4.1 Introduction .......................................... 30
    4.1.1 Definitions and notation ......................... 31
  4.2 Basic Distributional Results .......................... 33
    4.2.1 Record values from the classical model .......... 33
    4.2.2 Record times and related statistics ............. 35
    4.2.3 Markov chains .................................... 37
  4.3 $k$-DPP Approximations as Record Values .............. 38
    4.3.1 Jittering the log-determinants .................. 38
    4.3.2 Distribution of the jittered log-determinants .. 39
    4.3.3 Log-determinants records ....................... 43
  4.4 Discussion ............................................. 47

5 Concluding Remarks ........................................ 49
  5.1 Future Work ........................................... 50
    5.1.1 MCMC $k$-DPP sampling ........................... 50
# Table of Contents

5.1.2 Combination of $k$-DPP and other stochastic search methods .............................................. 50

Bibliography ................................................................................................................................. 51

Appendices

A R Code for Sampling from a $k$-DPP .................................................................................. 58

B Python Code for Genetic Algorithm ...................................................................................... 62
List of Tables

3.1 Results obtained from 100 realizations of \( k \)-DPP and GA for both the real and synthetic data. Sample size refers to the number of DPP simulations and the number of generations of GA for each realization; SD refers to standard deviation. 28

4.1 The first two columns provides a summary of the occurrence time of records and their values; the second and the third columns provides the conditional probabilities given the current record values and the conditional probabilities given the greedy approximation respectively; the last column gives the expected waiting times for the next record. 46
List of Figures

3.1 From left to right, realizations of a homogeneous Poisson point process, DPP, and DPP configuration with the highest probability on a 2-D fine grid. From Kulesza et al. (2012). 16

3.2 Geometric interpretations of the determinant of a kernel matrix. The leftmost two graphs from left represent the determinants of a matrix consisting of two column vectors, and the graph on the right represents the case with three column vectors. 17

3.3 Trace of 10,000 log-determinants generated by a 10-DPP. 23

3.4 Occurrence of maximum log-determinants of the restricted conditional hypercovariance matrix when increasing the number of simulated 10-DPP samples. The optimum solution is marked by the red horizontal dashed red line. The inset shows a zoomed-in view of the first 100 samples. 25

3.5 Occurrence of maximum log-subdeterminants of the synthetic kernel matrix when increasing the number of simulated 60-DPP samples. The greedy and the GA solutions are marked by the green and purple solid horizontal lines, respectively. The inset shows a zoomed-in view of the first 1,000 samples. 27

4.1 Top row: Histograms of $k$-DPP log-determinants with sample sizes 100,000 and 500,000 respectively; bottom row: Kernel density estimates of $k$-DPP log-determinants with sample sizes 100,000 and 500,000 respectively. 30
4.2 From top to bottom: histograms and estimated theoretical density functions (left) and quantile-to-quantile plot (right) displayed for the data above 90th percentile for GPD, Weibull, Normal distributions, respectively.
# List of Algorithms

1. Branch-and-bound algorithm ........................................... 10
2. Greedy algorithm ......................................................... 11
3. Genetic algorithm ........................................................ 13
4. Sampling from a DPP ....................................................... 19
5. Sampling from a $k$-DPP ................................................ 21
6. MCMC sampling from a $k$-DPP ........................................ 22
7. Sampling-based solution strategy using $k$-DPP ................. 24
Acknowledgements

I am deeply grateful to my supervisors Prof. Jim Zidek and Dr. Nhu Le for the opportunities and immense support they have provided me throughout the course of my graduate work at UBC. I am also indebted to my dear friends at UBC and SFU who believed in me and gave me continuous mental support to carry on this journey.
Dedication

This thesis is dedicated to my parents and YJX.
Chapter 1

Introduction

We consider the situation where we wish to make inferences regarding a set of random variables from observations made on a subset of these random variables. This is closely related to the concept of experimental design in classical statistics, in which we use a carefully selected experiment to be performed by the experimenter. The aim of this so-called design is to optimize a process or system by performing each experiment and to draw conclusions about the significant behaviour of the studied object from the results of the experiment. In environmental statistics, for example, the experiment yields observations of a certain environmental process (such as temperature, air pollution, rainfall, etc) taken from a set of monitoring stations. Maintaining all stations is costly and hence infeasible, and one may only need to select only a subset of them. Another example is given by variable selection in regression models, where the problem refers to the task of finding a small subset of the available independent variables that can effectively predict the dependent variable. In this chapter, we describe three fields in statistics that seem unrelated at first glance - a design strategy called the maximum-entropy design, a special class of spatial point processes called determinantal point processes, and the theory of record values. Because chapters 3 and 4 contain detailed developments of the latter two fields, we forego these details here. Later chapters then extends these methodologies, especially those concerning determinantal point processes and record values, to attempt to solve the maximum-entropy design problem.
1.1 Maximum-Entropy Design Problem

Well-defined optimality criteria are required to evaluate designs. Formally, consider a set $N$ of $n$ points, called the design space, and a design size $k$, such that $k \leq n$. Our goal is to select a subset $K$ of $k$ points, such that the observations made at these points are maximally informative. Information can be measured by entropy, for example, and our goal is then to choose a subset that minimizes the resulting entropy, i.e., maximizes the amount by which uncertainty will be reduced by the information provided by the experiment. This is referred to as the maximum-entropy design problem (or sometimes, the maximum-entropy sampling problem). The problem was first introduced by Caselton and Zidek (1984) who applied it to the design of environmental monitoring networks and later considered by Shewry and Wynn (1987); Guttorp et al. (1993); Wu and Zidek (1992).

We provide a brief introduction into the problem and its application in spatial statistics. First note that the uncertainty about a random vector $Y$ can be represented by the entropy of its distribution

$$H(Y) = \mathbb{E}_Y \left[ -\log \left( \frac{f(Y)}{h(Y)} \right) \right],$$

where $h(Y)$ denotes a reference density, which does not need to be integrable, allowing the entropy to be invariant under one-to-one transformations of the scale of $Y$ (Jaynes, 1963). In hierarchical models for environmental processes, $Y$ is usually defined conditionally on some hyperparameters, which we denote by $\Psi$. The total entropy can then be defined as

$$H(Y, \Psi) = \mathbb{E} \left[ -\log \left( \frac{f(Y, \Psi)}{h_Y(Y, \Psi)} \right) \right] = \mathbb{E} \left[ -\log \left( \frac{f(Y|\Psi)f(\Psi)}{h_Y(Y)h(\Psi)} \right) \right] = H(Y|\Psi) + H(\Psi).$$

We can define the total entropy in the context of the monitoring network design by first partitioning the data as $Y = (Y^{(u)}, Y^{(g)})$, where $Y^{(u)}$ denotes
the measurements at potential sites, currently ungauged, and \( Y^{(g)} \) relates to the existing sites, referred to as gauged locations. We then have

\[
H(Y^{(u)}, Y^{(g)}, \Psi) = H(Y^{(u)}|Y^{(g)}, \Psi) + H(Y^{(g)}) = H(Y^{(u)}, \Psi|Y^{(g)}) + H(Y^{(g)}).
\]

The design criterion is based on minimizing \( H(Y^{(u)}, \Psi|Y^{(g)}) \), which measures the uncertainty about \( Y^{(u)} \) after \( Y^{(g)} \) is observed. Since the total entropy \( H(Y^{(u)}, Y^{(g)}, \Psi) \) is fixed, an equivalent criterion is to maximize \( Y^{(g)} \). Moreover, the same criterion for maximizing \( Y^{(g)} \) would be similarly obtained had we decomposed \( H(Y^{(u)}, Y^{(g)}) \) instead of \( H(Y^{(u)}, Y^{(g)}, \Psi) \).

Rarely are networks designed from scratch, so the redesign of an existing network is often of interest. When the goal is to augment a network, the objective is to find a subset of \( u^+ \) sites among the \( u \) ungauged ones to add to the existing network. We denote the remaining sites that are not the selected as \( u^- \). The resulting network will then consists of \( (Y^{(u^+)}, Y^{(g)}) \).

We have

\[
H(Y^{(u^+)}, Y^{(u^-)}, Y^{(g)}) = H(Y^{(u^+)}, Y^{(u^-)}|Y^{(g)}) + H(Y^{(g)}) = H(Y^{(u^-)}|Y^{(u^+)}, Y^{(g)}) + H(Y^{(u^+)}, Y^{(g)}).
\]

Since the total entropy \( H(Y^{(u^+)}, Y^{(u^-)}, Y^{(g)}) \) is fixed, it will be optimal to augment the network with the \( u^+ \) sites so as to maximize \( H(Y^{(u^+)}, Y^{(g)}) \). For Gaussian random fields (i.e., where \( Y \) follows multivariate normal with covariance \( \Sigma \)), it can be shown (details can be found in Le and Zidek (2006)) that

\[
H(Y^{(u^+)}, Y^{(g)}) = H(Y^{(u^+)}|Y^{(g)}) + H(Y^{(g)}) \propto \frac{1}{2} \log \det(\Sigma_{u^+|g}) + \frac{1}{2} \log \det(\Sigma_{gg}),
\]

and hence it will be optimal to maximize \( \frac{1}{2} \log \det(\Sigma_{u^+|g}) \), where \( \Sigma_{u^+|g} \) is the conditional covariance matrix of random fields measured by added sites given those gauged sites. The entropy criterion for augmenting the network
1.2. Spatial Point Processes

is thus
\[ \arg\max_{u \in C} \frac{1}{2} \log \det(\Sigma_{u+|g}), \]
where \( C \) is the set of candidate sites.

Concretely, consider a monitoring network redesign problem where the objective is to choose from a set of \( n \) ungauged sites a subset of \( s \) sites to augment the existing network. With the maximum-entropy design strategy, one would seek an \( s \times s \) sub covariance matrix that has the largest log-determinant from an \( n \times n \) conditional covariance.

The criterion described above coincides with what is termed the \( D \)-optimal design. The problem now is to find a design from the set of all feasible designs that maximizes information. In classical regression models, the optimization criteria are generally related to the notion of the (Fisher) information matrix. In this context, the \( D \)-optimal design objective is to maximize the determinant of the corresponding information matrix.

1.2 Spatial Point Processes

Spatial point processes are useful as statistical models in the analysis of observed patterns of points, where the points represent the locations of some object of study (e.g., trees in a forest, bird nests, disease cases, or petty crimes). Point processes also play a special role in stochastic geometry, as the building blocks of more complicated random set models (such as the Boolean model), and as simple instructive examples of random sets. Since chapter 3 provides a thorough explanation of determinantal point processes as a special class of spatial point processes, and we forego this development here and instead simply provide some useful references. The work of Moller and Waagepetersen (2003) offers an up-to-date, unified collection of theoretical advances and applications in simulation-based inference for spatial point processes, while Lavancier et al. (2015) provides details about statistical inference procedures using determinantal point processes.
1.3 Record Values Theory

A record value or record statistic is a value that is larger or smaller than the preceding values obtained in a sequence of observations of random variables. The theory of records is connected very closely to the theory of order statistics. The term was first introduced by Chandler (1952). The study of record values was subsequently carried out by a relatively small but highly talented group of individuals and has garnered the attention of researchers from various fields. For example, Rényi (1962) has extensively studied on the limit theorems for record sequences, while Glick (1978) reviewed the work of Shorrock (1972a,b, 1974) who introduced the application of records in modelling the slowest cars in a traffic. Besides its use in real-world applications, knowledge of certain distributional functions of the record value sequence is adequate to determine the common distribution of the underlying observations. As is to be expected, the catalog parallels the corresponding list of characterizations based on order statistics. Although the literature is extensive, we provide details in chapter 4 only on the aspects of records that are most closely related to the subject of this thesis.
Chapter 2

Overview of the Combinatorial Optimization Problem

In this chapter, we formally define the underlying optimization problem that arises in the maximum-entropy sampling problem. Furthermore, we present an overview of existing methods for finding (both exactly and approximately) the solution. We start by providing the mathematical definitions and setting the notations for future discussions.

Let \( N = \{1, 2, 3, ..., n\} \) where \( n \) is a positive integer. We use \( C \) to denote a real symmetric positive semidefinite matrix indexed by elements in \( N \). Further, let \( S \) be an \( s \)-element subset of \( N \) with \( 1 \leq s \leq n \). Let \( C_{S,S} = C_S \) denote the principal submatrix of \( C \) having rows and columns indexed by elements in \( S \) (\( C_{S_1,S_2} \) with \( S_1 \neq S_2 \) denotes a submatrix of \( C \) having rows indexed by \( S_1 \) and columns indexed by \( S_2 \)). Our optimization problem is to determine

\[
\max_{S:|S|=s, S \subseteq N} \det(C_S),
\]

(2.1)

and the associated maximizer \( S \). For ease of future discussions, we let \( H_N(S) = \det(C_S) \) denote the determinant of the matrix \( C_S \). We also let

\[
v(C, F, E, s) = \max_{S:|S|=s, S \subseteq N} \det(C_S),
\]

(2.2)

where \( F \) is an \( f \)-element subset of \( N \) (\( 0 \leq f \leq n \)) and \( E \) is an \( e \)-element subset of \( N \) (\( 0 \leq e \leq n - f \)) satisfying \( F \cap E = \emptyset \). Any \( s \)-element subset \( S \) of \( N \) satisfying \( F \subseteq S \subseteq F \cup E = N \) is a feasible solution to the problem.
2.1. Exact Methods for Finding the Optimum

The set $F$ contains indices that are forced into the solution set or that are fixed in the solution set. The set $E$ contains indices that are eligible for consideration into a solution set. Usually, we start with $F = \emptyset$ and $E = \mathbb{N}$.

Note that this can be viewed as a subset selection problem, where the selection criterion is to maximize the determinant of the matrix indexed by the subset. This is a combinatorial optimization problem, whereby the problem consists of finding an optimal object from a finite set of objects. Usually, an exhaustive search is not feasible. In fact, as proved by Ko et al. (1995), this particular combinatorial optimization problem is NP-hard. To gain an idea of the reason why the problem is hard, imagine having a set $\mathbb{N}$ with 50 elements and the goal is to find a subset of 25 elements that maximize $H_N(S)$. The solution space contains $\binom{50}{25} \approx 1.26 \times 10^{14}$ subsets, which is computationally intractable in both time and storage.

2.1 Exact Methods for Finding the Optimum

Although its NP-hardness places the problem beyond those for which theoretically efficient algorithms exist for general instances, the theoretical result is based on worst-case analysis and are of an asymptotic nature. So hope rests in developing practical methods for solving the problem instances with moderate sizes. For example, efficient software implementation, such as that implemented in the EnviroStat v0.4-0 R package, can solve problem $\binom{50}{25}$ within hours. Ko et al. (1995) first introduced a branch-and-bound algorithm for this problem that guarantees optimality. In the remainder of this section, we provide the details of their algorithm.

2.1.1 Branch-and-bound

Branch-and-bound is a technique originally developed for solving integer linear programs (see Nemhauser and Wolsey (1989) for an example). This algorithm works by forming the solution space as a rooted tree with the full candidate set at the root, and exploring branches or regions of the tree,
which represent subsets of the full solution set. Before enumerating the candidate solutions of a branch, the branch is checked against the upper and lower estimated bounds of the optimal solution. It is then discarded if it cannot produce a better solution than the best one found so far by the algorithm. Accordingly, the algorithm depends on efficient estimation of the lower and upper bounds of regions or branches of the search space. The branch-and-bound algorithm can be thought of as an improved exhaustive search method, and if no good bounds are available, the algorithm degenerates to a simple exhaustive search.

Ko et al. (1995) used a spectral bounding method to compute the upper bound on the optimal value. Specifically, they established the spectral upper bound as

\[ v(C, \emptyset, E, s) \leq b(C, \emptyset, E, s) = \prod_{i=1}^{s} \lambda_i(C_E), \tag{2.3} \]

and if \( F \) is non-empty and \( C_F \) is invertible, then

\[ v(C, F, E, s) \leq b(C, F, E, s) = \det(C_F) \prod_{i=1}^{s-f} \lambda_i(C_{E|F}), \tag{2.4} \]

where

\[ C_{E|F} = C_E - C_{E,F}C_F^{-1}C_{F,E}. \tag{2.5} \]

Note that if \( C \) is a covariance for a set of random variables indexed by \( N \), then \( C_{E|F} \) is the covariance matrix of random variables indexed by \( E \), conditioned on the random variables indexed by \( F \).

The branch-and-bound algorithm in this case starts with an initial solution \( S^* \), which is typically obtained by using a heuristic method (see section 2.2 for a discussion). A branching strategy essentially tests different sets \( F \) and \( E \) with the aim to eventually discovering the optimal solution. The algorithm maintains a list of active subproblems, each of which is determined by its sets \( F \) and \( E \). Initially, the only subproblem is initialized to be the original problem (i.e., \( F = \emptyset, E = N \)). At each stage, an active subproblem is selected and an upper bound is calculated using the bounding methods described in Equation (2.3). If the upper bound is less than
2.1. Exact Methods for Finding the Optimum

det\((C_S)\), then the subproblem is discarded. Otherwise, an index \(j \in E\) is selected, and the subproblem is replaced by two new subproblems: one with \(j\) adjoined to \(F\) and deleted from \(E\), and the other with \(j\) deleted from \(E\). Then, for each of the these two potential subproblems, we check if there is one and only one \(S'\) having \(s\) elements with \(F \subset S' \subset F \cup E\). If there is, and this \(S'\) satisfies \(H_N(S') > H_N(S^*)\), then we replace \(S^*\) with \(S'\) and discard the subproblem. A more detailed pseudo-code of the implementation is provided in Algorithm 1. Note that one may attempt to increase the lower bound at any step by returning the heuristic based on information obtained from the current state of \(L\). This is necessary to speed up the algorithm whenever the size of \(L\) becomes intolerably large. The algorithm is finite and terminates with \(S^*\) optimal when \(LB = UB\). We can terminate early if we are willing to be satisfied when \(LB\) is sufficiently close to \(UB\).

The efficiency of the algorithm essentially depends on the sharpness of the theoretical upper bound. Ko et al. (1995) performed computational experiments using real data from environmental monitoring and solved a problem of size \((36, 26)\) within two hours. Meanwhile, Anstreicher et al. (1996, 1999) introduced a continuous relaxation of the problem and derived a continuous nonlinear programming upper bound on \(H_N(S)\). They reported success in solving the optimization problem with an \(n\) as large as 63. Furthermore, Lee (2000) discussed how the problem can be relaxed as a semidefinite program and thus the corresponding semidefinite programming upper bound can be derived. However, it does not seem that any computational results have been produced with that bound. Anstreicher et al. (1996, 1999) also demonstrated how any bound for the complementary problem of choosing a maximum entropy set of \(n - s\) points with respect to \(C^{-1}\) translates to a bound for the original problem. Hoffman et al. (2004) showed how the original spectral bound can be strengthened by combining the use of Fischer’s inequality (Fischer, 1908) and partitioning. In any case, the branch-and-bound algorithm has been found successful in solving real problems of a size as large as 75. Most recently, Anstreicher (2018) introduced a new bound “linx” based on a simple but previously unexploited determinant identity. The bound can be computed very efficiently and is superior to all previously
2.2. Heuristics for Approximating the Optimum

Algorithm 1 Branch-and-bound algorithm

Input: An initial solution $S^*$ and an $n \times n$ kernel matrix $C$.

Set $LB = \det(C_{S^*})$, $UB = b(C, \emptyset, N, s)$, $L = \{(C, \emptyset, N, s)\}$.

while $UB > LB$ do

Select and remove $(C, F', E', s) \in L$ and compute $b(C, F', E', s)$.

if $b(C, F', E', s) > LB$ then

Select $j \in E'$.

Set $L = L \cup \{(C, F', \emptyset, \{j\}, s)\}$ and compute $b(C, F', E' \setminus \{j\}, s)$.

if $|F'| + |E'| - 1 = s$ then

Set $S = F' \cup E'$.

if $\det(C_S) > LB$ then

Set $S^* = S$ and $LB = \det(C_S)$.

end if

end if

end if

Set $L = L \cup \{(C, F' \cup \{j\}, E' \setminus \{j\}, s)\}$ and compute $b(C, F' \cup \{j\}, E' \setminus \{j\}, s)$.

if $|F'| + |E'| - 1 = s$ then

Set $S = F' \cup E'$

if $\det(C_S) > LB$ then

Set $S^* = S$ and $LB = \det(C_S)$.

end if

end if

end if

Set $UB = \max_{L \in L} b(L)$.

end while

Output: Set $S^*$ optimal with $s$ elements.

known bounds for the maximum-entropy sampling problem on most benchmark test problems. With linx, the branch-and-bound algorithm can solve some instances of the problem of size $n = 124$.

2.2 Heuristics for Approximating the Optimum

2.2.1 Greedy algorithm

For large intractable problems, heuristics, though all lacking some degree of generality and theoretical guarantees on the proximity to the optimum,
2.2. Heuristics for Approximating the Optimum

can be used to find reasonably good solutions. One of the best known is the DETMAX algorithm (Mitchell, 1974), based on the idea of exchanges, which is widely used by statisticians for finding approximate $D$-optimal designs. Although the algorithm does not guarantee $D$-optimality, it has performed adequately in cases where the optimal design is known (Mitchell, 1974). Applied to the case of this work, the DETMAX begins from a set $S$ of the desired size, and while possible, choose $i \in N \setminus S$ and $j \in S$ so that $H_N(S \cup \{i\} \setminus \{j\}) > H_N(S)$, and replace $S$ with $S \cup \{i\} \setminus \{j\}$.

Due to a lack of readily available alternatives, Guttorp et al. (1993) use a greedy approach, which is summarized in Algorithm 2. Ko et al. (1995) experiment with a backwards version of Algorithm 2: start with $S = N$, then, for $j = 1, 2, \ldots, n - s$, choose $l \in S$ so as to maximize $H_N(S \setminus \{l\})$, and then remove $l$ from $S$.

**Algorithm 2** Greedy algorithm

**Input:** Size $k$ and an empty set $S = \emptyset$.

for $i = 1, \ldots, k$ do

Choose $s \in N \setminus S$ so as to maximize $H_N(S \cup \{s\})$.

Set $S = S \cup \{s\}$.

end for

**Output:** Set $S$ with $k$ elements.

The greedy-type algorithm has been popular in solving such optimization problems. Besides its simplicity, it attains good theoretical properties for special functions. Consider a function $f : 2^N \to \mathbb{R}$ defined by $f(S) = \log(\det C_S)$. Kelmans and Kimelfeld (1983) show that this is a submodular set function, that is

$$f(S) + f(T) \geq f(S \cup T) + f(S \cap T) \quad \text{for all } S, T \subseteq N.$$  

Nemhauser et al. (1978) proved that the greedy algorithm offers a good approximation to the optimal solution of the NP-hard optimization problem involving a submodular objective function. In fact, Nemhauser et al. (1978) proved that any algorithm that is allowed to only evaluate $f$ at a polynomial number of sets will be unable to obtain an approximation guarantee better
than $1 - 1/e$. Although this celebrated result does provide a guarantee on the proximity of a class of approximations to their optimum, the error bound is loose and hence it is of little practical interest.

2.2.2 Genetic algorithm

Genetic algorithms (GA) are a family of computational models inspired by evolution (Holland, 1992). In general, these algorithms encode candidate solutions to a specific problem into a sample chromosome-like data structure. The algorithms then seek to improve a population of potential solutions by applying recombination operators to these structures while preserving critical information. In particular, the recombination operators are inspired by using principles of genetic evolution such as natural selection, crossover, and mutation. Although the range of problems to which GAs have been applied is quite broad, GAs are often viewed as methods for function optimization problems. Moreover, they have been known to work well for optimizing hard, black-box functions with a potentially large number of local optima (Goldberg and Holland, 1988; Whitley, 1994).

Implementation of a GA begins with a population of (typically random) potential solutions to a problem (encoded on chromosomes). These structures are then evaluated and reproductive opportunities are allocated in such a way that those chromosomes which represent a better solution to the target problem are given more chances to reproduce than those chromosomes which are poorer solutions. The goodness of a solution is typically defined with respect to the current population.

Recently, Ruiz-Cárdenas et al. (2010) proposed a stochastic search procedure based on a GA for approximating the optimal design for environmental monitoring networks. The authors test the algorithm on a set of simulated datasets of different sizes, as well as on a real application involving the redesign of a large-scale environmental monitoring network. The GA considered here consists of the general steps described in Algorithm 3.
Algorithm 3 Genetic algorithm
1: Choose at random an initial population of size $N_0$, that is, a set of $N_0$ possible solutions $S_1, \ldots, S_{N_0}$.
2: Compute the fitness, that is, the value of the objective function $H_N(S_i), i = 1, \ldots, N_0$, for each of the solutions in the population.
3: Crossover: choose a proportion, $p_{\text{cross}}$, of solutions from the population. These solutions are selected according to a fitness-dependent selection scheme. Among these selected solutions, pairs of solutions are formed at random.
4: Mutation: choose a proportion, $p_{\text{mutprop}}$, of solutions from the population with equal probability. For each selected solution, each gauged site may be swapped, according to a mutation probability $p_{\text{mut}}$, with a randomly chosen ungauged neighbour site.
5: Compute the fitness of the solutions obtained by crossover and mutation. Include these solutions in the current population, creating an augmented population.
6: Selection: the population of solutions of the new generation will be selected from this augmented population. A proportion of solutions with best fitness, called the elite, enter directly into the new generation while the remaining members of the new generation are randomly chosen according to a certain fitness-dependent selection scheme.
7: Stop the algorithm if the stop criterion is met. Otherwise, return to step 3.

2.3 Discussion

In this chapter we have reviewed commonly used exact algorithms and heuristics for solving the combinatorial optimization problem of interest. Although the list of algorithms we discuss here is by no means exhaustive, it covers the mainstream of methods used in machine learning, statistics, and optimization communities for tackling similar problems and many other methods are developed based upon it. In fact, the optimization of this type of problems has always been a hot topic (see Krause and Golovin (2014); Nemhauser et al. (1978); Fisher et al. (1978) for a representative overview). This chapter mainly describes the theoretical development of important algorithms, and computational experiments involving the methods discussed here will be presented in the following chapter.
Chapter 3

Determinantal Point Processes

Determinantal point processes (DPPs) constitute a special class of repulsive spatial point processes and offer efficient and exact algorithms for sampling, marginalization, conditioning, and other inferential tasks. These processes were first studied by Macchi (1975), as fermion processes, to model the distribution of fermions at thermal equilibrium. Borodin and Olshanski (2000) and Hough et al. (2006) subsequently popularized the name “determinantal” and offered probabilistic descriptions of DPPs. More recently, DPPs have attracted attention in the machine learning and statistics communities. The seminal work of Kulesza et al. (2012) provides a thorough and comprehensive introduction to the applications of DPPs that are most relevant to machine learning problems. Since then, DPPs have been gaining attention in the machine learning community where their repulsive character has been used to enforce the idea of diversity in subset selection problems. These problems have been encountered in a variety of applications such as image searching, neuroscience and wireless network configuration. More recently, Lavancier et al. (2013) studies rigorously statistical models and methods for DPPs and how this class of point processes can be useful models for the description of spatial point pattern datasets where nearby points repel each other.

3.1 Definitions

We view a point process as a random locally finite subset X of a Borel set $S \subseteq \mathbb{R}^d$. Denoting by $X_B = X \cap B$, the restriction of X to a set $B \subseteq S$, and $N(B)$ the number of events (random elements) in $X_B$ ($N(B)$ also called
3.1. Definitions

the *counting variable*), the local finiteness of \( X \) means that \( N(B) < \infty \) almost surely whenever \( B \) is bounded - any bounded region contains only a finite number of points. We also assume that the point process is *simple*: \( N(\{x\}) \leq 1 \) for all \( x \in \mathbb{R}^d \). That is, with probability 1, no two points of the process are coincident.

3.1.1 General Poisson point processes

As an example of spatial point processes, recall that the Poisson process can be generalized to a general \( d \)-dimensional space. To define a uniform Poisson point process in \( \mathbb{R}^d \), or an inhomogeneous Poisson process in \( \mathbb{R}^d \), or a Poisson point process on some other space \( S \), the following general definition can be used.

Let \( S \) be a space, and \( \Lambda \) a measure on \( S \) (we require \( S \) to be a locally compact metric space, and \( \Lambda \) a measure which is finite on every compact set and which has no atoms). The Poisson process on \( S \) with intensity measure \( \Lambda \) is a point process on \( S \) such that

- For every compact set \( B \subset S \), the count \( N(B) \) has a Poisson distribution with mean \( \Lambda(B) \);
- If \( B_1, \ldots, B_n \) are disjoint compact sets, then \( N(B_1), \ldots, N(B_n) \) are independent.

For example, for the inhomogeneous Poisson process on \( \mathbb{R}^2 \) with intensity function \( \beta(u) \), \( u \in \mathbb{R}^2 \) is defined by taking \( S = \mathbb{R}^2 \) and \( \Lambda(B) = \int_B \beta(u)du \).

3.1.2 General determinantal point processes

The DPPs define a point process on \( S \subseteq \mathbb{R}^d \), that is, a random point configuration \( X = \{x_1, \ldots, x_n\} \) with \( x_n \in S \). Here we are only concerned with DPPs on a finite state space, \( S = \{s_1, \ldots, s_N\} \). Let \( C \) denote an \( N \times N \) positive semidefinite matrix constructed as \( C_{ij} = C(s_i, s_j) \) with a covariance function \( C(s_i, s_j) \). For a generic random point configuration \( X \), define

\[
P(X = A) = \frac{\det(C_A)}{\det(C + I)}
\]

(3.1)
3.1. Definitions

as a probability distribution on the $2^N$ possible point configurations $X$.

The above argument actually defines a subclass of DPPs known as $L$-ensembles. In practice, it is more convenient to characterize DPPs via these $L$-ensembles (Borodin and Rains, 2005; Kulesza et al., 2012), which directly define the probability of observing each subset of $S$. It can be demonstrated that the normalizing constant $\det(C + I) = \sum_{A \subseteq S} \det(C_A)$ and $I$ is an $n \times n$ identity matrix.

An important characteristic of DPPs is that they assign higher probability to sets of items that are diverse (unlike the Poisson point processes introduced earlier, which present complete spatial randomness). An illustration of the repulsive character of DPPs is shown in Figure 3.1. This is due to the fact that the intensity function of a DPP depends on the determinant of a kernel matrix that defines a global measure of similarity between pairs of items. To illustrate the diversity idea, recall that one can interpret the determinant as the volume of a parallelotope spanned by the column vectors of $C_A$ - equal or similar column vectors span less volume than highly diverse ones. Figure 3.2 provides a geometric view of this idea. Note that the column vectors can be thought of as features for items, and those with features that are more different have larger probabilities of being sampled from the ground set of items.

Figure 3.1: From left to right, realizations of a homogeneous Poisson point process, DPP, and DPP configuration with the highest probability on a 2-D fine grid. From Kulesza et al. (2012).
3.2. Sampling from $k$-DPPs

Efficient sampling algorithms for DPPs have been the focus for many branches in machine learning and applied mathematics. The first of these algorithms was developed in Hough et al. (2006), and relies on eigendecomposition of the kernel matrix. Due to the high computational expense of the eigendecomposition for large scale datasets, much research has been concentrated on developing faster sampling methods. Affandi et al. (2013) employ a
3.2. Sampling from $k$-DPPs

Nystrom approximation on the kernel matrix and develop an approximate sampling algorithm based on the approximated kernel matrix, which generally has a lower dimension. Meanwhile, Li et al. (2015) explore the idea of coresets for the ground set of items and propose an approximate sampling algorithm that uses the ground set with reduced size. More recently, Launay et al. (2018) propose an exact sampling methods for DPPs that does not rely on eigendecomposition but rather exploits the properties of the point processes and construct a sequential sampling procedure based on thinning. Another alternative, recently studied by Anari et al. (2016), takes a different approach and uses Markov chain Monte Carlo (MCMC) to generate approximate samples for $k$-DPPs.

In this section, we discuss sampling algorithms for DPPs, and more specifically, for $k$-DPPs. We particularly focus on two of them: the exact sampling algorithm developed in Hough et al. (2006) and extensively discussed in Kulesza et al. (2012); and the MCMC procedure discussed in Anari et al. (2016).

3.2.1 Independent sampling

Algorithm 4, after Hough et al. (2006), offers an efficient algorithm for generating a point configuration $Y$ from a DPP. The algorithm takes an eigendecomposition of the DPP kernel $C$ as the input. Kulesza et al. (2012) proves that Algorithm 4 samples random point patterns that follow a DPP distribution. Note that in the first loop of the algorithm, a subset of eigenvectors is selected at random with selection probability depending on the associated eigenvalue. In the second loop, a sample $Y$ is generated based on the selected eigenvectors.

Before providing further details, we introduce an important way of expressing a DPP as a mixture of elementary DPPs (Kulesza et al., 2012), also referred to as determinantal projection processes (Hough et al., 2006). Elementary DPPs, denoted as $P^V$, are a particular type of DPP whereby every eigenvalue of its marginal kernel is either zero or one. Its marginal
3.2. Sampling from k-DPPs

**Algorithm 4** Sampling from a DPP

**Input:** eigendecomposition \( \{v_n, \lambda_n\} \) of \( L \).

\[ J \leftarrow \emptyset. \]

\[ \text{for } n = 1, \ldots, N \text{ do} \]
\[ \quad J \leftarrow J \cup \{n\} \text{ with probability } \frac{\lambda_n}{\lambda_n + 1} \]
\[ \text{end for} \]

\[ V \leftarrow \{v_n\}_{n \in J} \]

\[ Y \leftarrow \emptyset \]

\[ \text{while } |V| > 0 \text{ do} \]
\[ \quad \text{Select } y_i \text{ from } Y \text{ with probability given by } \frac{1}{|V|} \sum_{v \in V} (v^T e_i)^2 \]
\[ \quad Y \leftarrow Y \cup \{y_i\} \]
\[ \quad V \leftarrow V \perp, \text{ an orthonormal basis for the subspace of } V \text{ orthogonal to } e_i \]
\[ \text{end while} \]

**Output:** \( Y \).

The kernel can thus be decomposed as

\[ K^V = \sum_{v \in V} vv^T, \quad (3.3) \]

where \( V \) is a set of orthonormal vectors. From this decomposition, note that elementary DPPs have their cardinality fixed as the cardinality of \( V \).

Furthermore, denoting \( V_J \) as \( \{v_n\}_{n \in J} \), the mixture is (Kulesza et al., 2012)

\[ P_L(Y) = \frac{1}{\det(L + I)} \sum_{J \subseteq \{1, \ldots, N\}} P^{V_J}(Y) \prod_{n \in J} \lambda_n. \quad (3.4) \]

From this notion of elementary DPPs, note that the normalization con-
3.2. Sampling from $k$-DPPs

stant of the $k$-DPP is given by (Kulesza et al., 2012):

$$
\sum_{|Y'|=k} \det(L_{Y'}) = \det(L + I) \sum_{|Y'|=k} \mathcal{P}_{L}(Y')
$$

(3.5)

$$
= \sum_{|Y'|=k} \sum_{J \subseteq \{1, \ldots, N\}} \mathcal{P}^{V_{J}}(Y') \prod_{n \in J} \lambda_n
$$

(3.6)

$$
= \sum_{J \subseteq \{1, \ldots, N\}} \prod_{n \in J} \lambda_n
$$

(3.7)

$$
\equiv \mathcal{E}^{N}_{k},
$$

(3.8)

which can be computed recursively, noting that

$$
\mathcal{E}^{N}_{k} = \mathcal{E}^{N-1}_{k} + \lambda_{N} \mathcal{E}^{N-1}_{k-1}
$$

(3.9)

where $\lambda_{1}, \ldots, \lambda_{N}$ are the eigenvalues of the $L$-ensemble. Kulesza et al. (2012) have illustrated that the first phase of Algorithm 4 chooses an elementary DPP according to its mixing coefficient, and the second phase samples are selected from the elementary DPP chosen in the previous phase.

Since $k$-DPPs are simply DPPs conditioned on size, a comparably fast algorithm which is similar to Algorithm 4 can be developed. Kulesza et al. (2012) shown that a $k$-DPP is also a mixture of elementary DPPs, but it only gives nonzero weight to those of the desired dimension. As previously mentioned, the second loop of Algorithm 4 can sample from any given elementary DPP; therefore, we can sample from a $k$-DPP if we can sample index sets $J$ according to appropriate mixture components. From Kulesza et al. (2012), $\Pr(J = J) = \frac{1}{e^{k}} \prod_{n \in J} \lambda_n$ when $|J| = k$, and zero otherwise. Therefore, the first loop of Algorithm 5 yields a realization of $J$ and the second loop is identical to that of the DPPs.

Kulesza et al. (2012) discussed thoroughly the computational complexity of sampling DPPs and $k$-DPPs. In summary, sampling an elementary DPP (the second loop of Algorithm 4) can be done in $O(Nk^3)$ time, where $k = |V|$ is the number of eigenvectors selected in phase one. For $k$-DPPs, the elementary symmetric polynomials can be precomputed and runs in $O(Nk)$
3.2. Sampling from $k$-DPPs

**Algorithm 5** Sampling from a $k$-DPP

**Input:** size $k$ and $\{v_n, \lambda_n\}$ eigenvectors and eigenvalues of $L$.

1. $J \leftarrow \emptyset$.
2. Compute elementary DPPs $E_1^n, \ldots, E_k^n$, for $n = 0, \ldots, N$.
3. for $n = N, \ldots, 1$ do
   4. Sample $u \sim U[0, 1]$.
   5. if $u < \frac{\lambda_n E^0_{n-1}}{E_k^n}$ then
      6. $J \leftarrow J \cup \{n\}$
      7. $k \leftarrow k - 1$
      8. if $k = 0$ then
         9. break
     end if
   end if
end for

10. $V \leftarrow \{v_n\}_{n \in J}$
11. $Y \leftarrow \emptyset$
12. while $|V| > 0$ do
13. Select $y_i$ from $Y$ with probability given by $\frac{1}{|V|} \sum_{v \in V} (v^\top e_i)^2$
14. $Y \leftarrow Y \cup \{y_i\}$
15. $V \leftarrow V_{\perp}$, an orthonormal basis for the subspace of $V$ orthogonal to $e_i$
end while

**Output:** $Y$.

The first loop of Algorithm 5 then iterates at most $N$ times and requires a constant number of operations. The remaining task is then to select $J$ eigenvectors with a fixed size and sample from the elementary DPPs. So sampling $k$-DPP is $O(Nk^3)$ overall in computational time, assuming that we have an eigendecomposition of the kernel in advance. Therefore, sampling a $k$-DPP is no more expensive than sampling a standard DPP.

### 3.2.2 MCMC sampling

Recent work of Anari et al. (2016) has shown that a natural Metropolis-Hastings sampler for MCMC defined on the support of a homogeneous strongly Rayleigh distribution $\mu$ mixes rapidly, and can be used to efficiently generate approximate samples from $\mu$. The authors also indicated
3.2. Sampling from $k$-DPPs

that DPPs are special cases of strongly Rayleigh measures, so that the same Markov chain efficiently generates random samples of a $k$-DPP. The MCMC procedure is summarized in Algorithm 6. The state space of the chain is the ground set of items $S = \{1, \ldots, N\}$.

**Algorithm 6** MCMC sampling from a $k$-DPP

**Input:** Size $k$ and ground set $S$.

Start with any $A \subseteq S$ with $|A| = k$

for Maximum number of iterations do

Let $T = S \setminus \{i\} \cup \{j\}$, where $i \in S$ and $j \notin S$

Sample $u \sim \text{Uniform}(0, 1)$

if $u \leq \min(1, \frac{\mu(T)}{\mu(S)})$ then

Move to a new state $T$

else

Stay in $A$

endif

end for

**Output:** A Markov chain of random samples of a $k$-DPP.

It is easy to see that the chain with the described transition is reversible, since the probability of moving forwards from the current state to the new state is the same as moving in the reverse direction. The chain is also irreducible, with $\mu$ being the stationary distribution due to the Metropolis-Hastings step. The convergence of the Markov chain is measured via its mixing time: the mixing time of the chain indicates the number of iterations $t$ that we must perform (starting from $S_0$) before we can consider $S_t$ as an approximately valid sample from $\mu$. Anari et al. (2016) proved that the chain described in Algorithm 6 mixes after polynomial steps. Here, instead of stating the theorem, we show in Figure 3.3 the trace plot of an MCMC chain of 10,000 log-determinants generated by a 10-DPP. The plot illustrates how quickly the MCMC chain mixes practically.
3.3 Sampling-based solution strategy

To handle the NP-hard optimization problem described in Equation (2.3), the $k$-DPP sampling approach involves generating such $k$-DPP subsets repeatedly and calculating the objective function $H_N(S)$, such that successively better approximations, as measured by $H_N(S)$, can be found. The approximate solution to the problem is then given by the best attained $H_N(S)$ up to a certain number of simulations and its associated indices of points, as described in Algorithm 7.

Note that eigendecomposition of the kernel matrix can be conducted as a pre-processing step and thus does not need to be performed before each sampling step. Therefore, assuming that we have an eigendecomposition of the kernel in advance, sampling one $k$-DPP runs in $O(Nk^3)$ time (Kulesza et al., 2012) and the computation of the determinant of a submatrix typically takes $O(k^3)$ time. Overall, Algorithm 7 runs in $O(Nk^3)$ time per iteration.

Figure 3.3: Trace of 10,000 log-determinants generated by a 10-DPP.
3.4 Computational Performances

In this section, we compare the performances of the greedy algorithm, the GA, and the k-DPP approaches discussed above in two examples; one tractable and another intractable.

3.4.1 Optimizing maximum-entropy designs for monitoring networks

For the first study we consider the data supplied by the U.S. Global Historical Climatology Network–Daily (GHCND), which is an integrated database of climate summaries from land surface stations across the globe. For illustrative purposes, we selected 97 temperature monitoring stations where the maximum daily temperature was recorded. A subset of 67 stations was selected among the 97 stations to constitute a hypothetical monitoring network. An additional 30 stations were selected and designated as potential sites for new monitors. Casquilho-Resende et al. (2018) has successfully approximated a maximum-entropy design using k-DPP for an instance of this problem. In this case study, the goal is to select a subset of 10 stations from among the additional 30 to augment the network based on the maximum-entropy design criterion.

Using the notation presented in Equation (2.1), $C$ here is the estimated covariance matrix of 30 candidate sites and $S$ is the subset of 10 sites that maximize $H_N(S)$. For tractable optimization problems, the maximal value of the objective function (or equivalently the optimal design) can be obtained...
3.4. Computational Performances

efficiently by the branch-and-bound algorithm detailed in Algorithm 1. In this study, the maximal value is 80.09011.

For the comparison, we first performed the greedy algorithm discussed in Algorithm 2, which yielded a solution of 80.07284. Using Algorithm 3 with the tuning parameters suggested in Ruiz-Cárdenas et al. (2010) \((N_0 = 100, p_{\text{cross}} = 0.75, p_{\text{mut}} = 0.05, \text{and a tournament selection scheme with four competitors})\), the GA yielded a solution of 80.09011 after 1000 generations. Similarly, the proposed 10-DPP achieved the optimal value after about 80,000 simulations. As illustrated in Figure 3.4, the maximal value of the log-determinant among the simulations increases as the number of simulations escalates.

![Figure 3.4: Occurrence of maximum log-determinants of the restricted conditional hypercovariance matrix when increasing the number of simulated 10-DPP samples. The optimum solution is marked by the red horizontal dashed red line. The inset shows a zoomed-in view of the first 100 samples.](image)

In terms of computation time, for this particular problem about 20 minutes of wall clock time was taken to simulate 100,000 subsets from the
3.4. Computational Performances

10-DPP using R programming language (R Core Team, 2018) on a laptop with a 2.5 GHz Intel Core i7 processor and a 16 GB 1,600 MHz DDR3 RAM. In the same computational environment, 5 minutes of wall clock time was needed to simulate 1,000 generations of GA.

3.4.2 Synthetic data with a large number of points

Exact methods quickly become impractical for large data sets, and one must then resort to heuristics. Greedy heuristics are known to be fast and efficient, but they can be quite inaccurate. As an illustrative example, let an \( n \times n \) real symmetric positive definite matrix be

\[
A = \begin{bmatrix}
  x_{11} & x_{12} & x_{13} & \ldots & x_{1n} \\
  x_{21} & x_{22} & x_{23} & \ldots & x_{2n} \\
  x_{31} & x_{32} & x_{33} & \ldots & x_{3n} \\
  \vdots & \vdots & \vdots & \ddots & \vdots \\
  x_{n1} & x_{n2} & x_{n3} & \ldots & x_{nn}
\end{bmatrix},
\]

where the diagonal elements are

\[
x_{ii} = \begin{cases} 
  d, & \text{if } i < n - k + 1, \\
  d + \delta, & \text{otherwise}.
\end{cases}
\]

And the off-diagonal elements are

\[
x_{ij} = \begin{cases} 
  a, & \text{if } i > j, i \neq n - k + 1, n - k + 2, \ldots, n - k + 10 \\
  b, & \text{if } i > j, i = n - k + 1, \\
  c, & \text{if } i > j, i = n - k + 2, \text{ or } n - k + 3, \text{ or } \ldots, n - k + 10 \\
  a, & \text{if } i < j, j \neq n - k + 1, n - k + 2, \ldots, n - k + 10 \\
  b, & \text{if } i < j, j = n - k + 1, \\
  c, & \text{if } i < j, j = n - k + 2, \text{ or } n - k + 3, \text{ or } \ldots, n - k + 10,
\end{cases}
\]

where \( n \) is the size of the matrix and \( k \) is the desired size of the subset one would like to select.
Suppose we seek a 60-by-60 submatrix with maximal determinant from a 100-by-100 matrix, with \( a = 0.2, \ b = 0.9, \ c = 0.65, \ d = 7, \) and \( \delta = 1. \) For this particular matrix, running a greedy algorithm results in the selection of subsets \( \{31, \ldots, 40, 51, \ldots, 100\} \) at termination and an associated log-subdeterminant of 122.8217. We also ran GA with the same tuning parameters as in the previous section and the best solution obtained was 123.6158 in 1,000 iterations. For comparison, we simulated 100,000 60-DPP samples and found a better solution of 123.7503, as shown in Figure 3.5.

![Figure 3.5](image)

Figure 3.5: Occurrence of maximum log-subdeterminants of the synthetic kernel matrix when increasing the number of simulated 60-DPP samples. The greedy and the GA solutions are marked by the green and purple solid horizontal lines, respectively. The inset shows a zoomed-in view of the first 1,000 samples.

The computational burden increases significantly when dealing with large matrices, but parallel simulations can be exploited to reduce the computational time. For this example, 1 hour of wall clock time was required to simulate 100,000 samples of 60-DPP on a Compute Canada cluster with 32 cores 2.1GHz Intel Broadwell CPUs and 128 GB RAM. In the same computational environment, running 1,000 generations of GA required 1
3.5. Discussion

In summary, the GA and the DPP approaches produced fairly comparable solutions that are more accurate than those produced by the greedy algorithm. The DPP methods seemed to require more computing resources. To observe the variations from run to run, we repeated the above two studies 100 times with the GA and DPP approaches. The results for DPP with 100,000 and 1,000,000 simulations and for the GA with 1,000 and 10,000 generations are shown in Table 3.1. Overall, the performances of the two approaches are fairly comparable.

<table>
<thead>
<tr>
<th>Sample Size</th>
<th>(k = 10)</th>
<th></th>
<th>(k = 60)</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Maximum</td>
<td>Mean</td>
<td>SD</td>
<td>Maximum</td>
</tr>
<tr>
<td>DPP-100,000</td>
<td>80.09</td>
<td>80.00</td>
<td>0.05</td>
<td>123.75</td>
</tr>
<tr>
<td>DPP-1,000,000</td>
<td>80.09</td>
<td>80.07</td>
<td>0.01</td>
<td>123.89</td>
</tr>
<tr>
<td>GA-1,000</td>
<td>80.09</td>
<td>80.07</td>
<td>0.06</td>
<td>123.62</td>
</tr>
<tr>
<td>GA-10,000</td>
<td>80.09</td>
<td>80.08</td>
<td>0.01</td>
<td>124.02</td>
</tr>
</tbody>
</table>

Table 3.1: Results obtained from 100 realizations of \(k\)-DPP and GA for both the real and synthetic data. Sample size refers to the number of DPP simulations and the number of generations of GA for each realization; SD refers to standard deviation.

3.5 Discussion

This chapter introduces a sampling-based approach for approximating the combinatorial optimization problem of subdeterminant maximization. By sampling from a \(k\)-DPP, which can be done in polynomial time, we approach the optimal solution by using the maximum simulated value as an approximation.

We demonstrated the potential application of the \(k\)-DPP based algorithm for finding optimal designs of spatial monitoring networks, and found it successful in obtaining the exact solution for small, tractable problems. When the size of the problem makes exact methods impractical, we showed (for a certain type of matrix) that our algorithm outperforms the greedy

hour of wall clock time.

In summary, the GA and the DPP approaches produced fairly comparable solutions that are more accurate than those produced by the greedy algorithm. The DPP methods seemed to require more computing resources. To observe the variations from run to run, we repeated the above two studies 100 times with the GA and DPP approaches. The results for DPP with 100,000 and 1,000,000 simulations and for the GA with 1,000 and 10,000 generations are shown in Table 3.1. Overall, the performances of the two approaches are fairly comparable.

<table>
<thead>
<tr>
<th>Sample Size</th>
<th>(k = 10)</th>
<th></th>
<th>(k = 60)</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Maximum</td>
<td>Mean</td>
<td>SD</td>
<td>Maximum</td>
</tr>
<tr>
<td>DPP-100,000</td>
<td>80.09</td>
<td>80.00</td>
<td>0.05</td>
<td>123.75</td>
</tr>
<tr>
<td>DPP-1,000,000</td>
<td>80.09</td>
<td>80.07</td>
<td>0.01</td>
<td>123.89</td>
</tr>
<tr>
<td>GA-1,000</td>
<td>80.09</td>
<td>80.07</td>
<td>0.06</td>
<td>123.62</td>
</tr>
<tr>
<td>GA-10,000</td>
<td>80.09</td>
<td>80.08</td>
<td>0.01</td>
<td>124.02</td>
</tr>
</tbody>
</table>

Table 3.1: Results obtained from 100 realizations of \(k\)-DPP and GA for both the real and synthetic data. Sample size refers to the number of DPP simulations and the number of generations of GA for each realization; SD refers to standard deviation.

3.5 Discussion

This chapter introduces a sampling-based approach for approximating the combinatorial optimization problem of subdeterminant maximization. By sampling from a \(k\)-DPP, which can be done in polynomial time, we approach the optimal solution by using the maximum simulated value as an approximation.

We demonstrated the potential application of the \(k\)-DPP based algorithm for finding optimal designs of spatial monitoring networks, and found it successful in obtaining the exact solution for small, tractable problems. When the size of the problem makes exact methods impractical, we showed (for a certain type of matrix) that our algorithm outperforms the greedy
3.5. Discussion

algorithm and is comparable to the GA for a relatively small cost in computational time. When solving a large problem where the exact methods do not work, the proposed DPP method is guaranteed to ultimately approach the optimum with a sufficient number of iterations while the greedy and GA potentially converge to a local optimum after a certain number of iterations. Moreover, although GA usually runs faster in terms of computational time than our DPP approximation and returns fairly accurate solutions, it requires careful calibrations of its tuning parameters. In fact, finding the right balance between crossover/selection, which pulls the population towards a local maximum, and mutation, which explores potentially better solution spaces, is a known issue for GA. Inappropriate choices of tuning parameters could adversely affect the convergence of the algorithm; see Goldberg and Holland (1988) and Whitley (1994) for more detailed discussions. The DPP approximation, on the other hand, can be run naively to obtain comparably accurate solutions. Another major advantage of the DPP is that the algorithm can be easily implemented with parallelization without material modifications, which enables potential usage of free-access supercomputers to further reduce computational time.

In future work, approximate sampling algorithms for $k$-DPP will be explored. Work has recently been published which attempts to reduce the dimension of the matrix, such as the one introduced in Li et al. (2015). Other studies mainly focus on the approximations of the kernel matrix using some lower dimensional structures or alternate representations of the matrix in lower dimensional forms, such as those in Affandi et al. (2013) and Kulesza et al. (2012). These methods could help to reduce the sampling complexity of the $k$-DPPs and eventually could further reduce the computational time required to obtain the approximate solutions. Currently, analytical theory is being developed to describe the number of iterations needed for successive improvements in the approximate DPP solutions, as well as to estimate the expected time duration, until an optimal solution is obtained.
Chapter 4

Record Values Theory

“Records, Record, Record!” “The oldest Olympic Record was beaten!” “New World Record result in 100-metre dash!”. Every day, we see such kind of headlines in newspapers, on the Internet, on the TV. Suppose we register the annual total inches of precipitation in Vancouver, what is the probability that next year will be a new maximum? Or, if we collect data on annual amounts of precipitation in Vancouver in the previous one hundred years, how many of them set a record - being higher in value than all previously registered values? It turns out that breakthroughs are less likely later than earlier in a sequence of observations.

In this chapter, we present statistical results concerning record values theory (also referred to as record statistics, or just simply records). The literature there is enormous. Here we only introduce results that are most relevant to the work reported in this thesis. In particular, we connect record values to the \( k \)-DPP approximations discussed in previous chapters and try to provide insights into the behaviour of those approximations.

4.1 Introduction

The study of record values dates back to the early nineteen-fifties. Chandler (1952) first introduced basic record models involving i.i.d. observations and documented many of the basic properties of the records. As one can imagine, record values are inextricably related to order statistics, and the study of record values in many ways parallels the study of order statistics. In fact, several early books on order statistics and related processes have chapters on record values. Many of them emphasize the parallel between record value theory and the theory of sample maxima. In particular, Chapter 6 of Galambos.
bos (1977) and Chapter 4 of Resnick (1987) provide extensive discussion of record values. An early survey paper by Glick (1978) also merits attention and introduces an interesting applications of record value theory in traffic - curiously, the simple model of random record values says something about how cars tend to bunch together behind a slow vehicle. Papers of similar spirit by Nevzorov (1988) and Nagaraja (1988) provide excellent resources for anyone interested in the field. The book by Ahsanullah (1995) provides a limited coverage of certain topics in the domain of record statistics. The coverage in Arnold et al. (1998) is much more extensive and inevitably more up to date. We follow the presentations in Arnold et al. (1998) to provide a brief overview of record values theory.

4.1.1 Definitions and notation

The standard record value process corresponding to an infinite sequence of i.i.d. observations is our focus. Let $X_1, X_2, \ldots$ be an infinite sequence of random variables having the same distribution as the random variable $X$. Denote the cumulative distribution function (cdf) of $X$ by $F$. Usually an assumption that $F$ is continuous is invoked to avoid the possibility of ties and allows a cleaner theoretical development.

An observation $X_j$ is called an upper record value (or simply a record) if its value exceeds that of all previous observations, that is, $X_j > X_i$ for all $i < j$. The times at which records appear are often of interest. For simplicity, let us assume that $X_j$ is observed at time $j$. Then the sequence of record time $\{T_n, n \geq 0\}$ is defined as follows:

$$T_n = \min \{j : X_j > X_{T_{n-1}}\}$$

(4.1)

for $n \geq 1$ and $T_0 = 1$ with probability 1. The sequence of record values $\{R_n\}$ is then defined by

$$R_n = X_{T_n}, \quad n = 0, 1, 2, \ldots.$$  

(4.2)

Since the first observation in any sequence will always be a record, here $R_0$
4.1. Introduction

is sometimes referred to as the reference value or the trivial record.

The above definition of record sequence implicitly assumes that the cdf $F$ will not produce any unbreakable record. This will not be the case if there exists some value $x_0$ such that $F(x_0) - F(x_0-) > 0$ and $F(x_0) = 1$, that is, if there is a largest possible real value that can be obtained with positive probability. Though such cases are sometime of interest and in fact, this is the case in our study, we will eliminate them from our discussion for now and assume that the cdf $F$ cannot yield unbreakable records so that \{$R_n, n \geq 0$\} will be a strictly increasing non-terminating sequence (we will explain in later sections why we still can apply such record values theory to our study). Note that this definition does not imply that \{$R_n$\} is unbounded. For example, if the cdf $F$ is uniform on the interval (0, 1), then \{$R_n$\} satisfies the definition and is bounded above by 1.

We may also define a record increment (jump) sequence \{$J_n, n \geq 0$\} by

$$J_n = R_n - R_{n-1}$$ (4.3)

for $n \geq 1$ and $J_0 = R_0$. An inter-record time sequence $\Delta_n$, is also of interest and can be defined as

$$\Delta_n = T_n - T_{n-1}, \quad n = 1, 2, \ldots.$$ (4.4)

Finally, the number of records may be tracked by a counting process \{$N_n, n \geq 1$\}, where

$$N_n = \text{number of records among } X_1, \ldots, X_n.$$ (4.5)

As we shall see, in the setup of the classical record model defined above, where $X_i’s$ are i.i.d. observations from a continuous distribution $F$, the record counting statistics $T_n$, $\Delta_n$, and $N_n$ will not be affected by $F$. The distributions of the record values $R_n$ are predictably affected by $F$. 

32
4.2 Basic Distributional Results

In this section, we discuss distributional results for record values and related statistics (i.e., $R_n$, $T_n$, $\Delta_n$, $N_n$) from the classical model. It turns out that, for the classical model, strong arguments can be made in favour of studying i.i.d. exponentially distributed $X'_i$s. In fact, we build relationship between a standard exponential distribution and a general continuous distribution to derive distributional results for record values from a general continuous cdf $F$.

4.2.1 Record values from the classical model

Let $\{X'_i, j \geq 1\}$ denote a sequence of i.i.d. exponentially distributed random variables with the rate parameter equal to 1. Due to the lack of memory property of the exponential distribution, the differences between successive records will be i.i.d. standard exponential as well, i.e., $\{J'_n\} = \{R^*_n - R^*_{n-1}\}$ are i.i.d. Exp(1) random variables. It follows that the $n$th record, $R^*_n$, corresponding to an i.i.d. Exp(1) sequence, have a Gamma distribution (Arnold et al., 1998), i.e.,

$$R^*_n \sim \text{Gamma}(n + 1, 1), \quad n = 0, 1, 2, \ldots.$$ \hfill (4.6)

We may use this result to obtain the distribution of the $n$th record corresponding to an i.i.d. sequence of random variables $\{X_i\}$ with common continuous cdf $F$. First note that if $X$ has a continuous cdf $F$, then

$$H(X) = -\log(1 - F(X))$$ \hfill (4.7)

has a standard exponential distribution. Consequently, $X \overset{d}{=} F^{-1}(1 - \exp(-X*))$ where $X^*$ is Exp(1). Since $X$ is a monotone function of $X^*$, the $n$th record of the $\{X_i\}$ sequence can be expressed as a function of the $n$th record of the $\{X'_i\}$ sequence. Specifically, we have

$$R_n \overset{d}{=} F^{-1}(1 - \exp(-R^*_n)), \quad n = 0, 1, 2, \ldots.$$ \hfill (4.8)
4.2. Basic Distributional Results

Since $n$ is an integer, it is easy to show by repeated integration by parts that the survival function for $R^*_n$ (Gamma($n + 1, 1$)) is

$$P(R^*_n > r^*) = e^{-r^*} \sum_{k=0}^{n} \frac{(r^*)^k}{k!}, \quad r^* > 0. \quad (4.9)$$

We may then use the relation established in Equation (4.8) to derive the survival function of the $n$th record corresponding to an i.i.d. sequence of random variables with cdf $F$:

$$P(R_n > r) = (1 - F(r)) \sum_{k=0}^{n} \frac{(- \log(1 - F(r)))^k}{k!}, \quad (4.10)$$

or equivalently, the incomplete Gamma function:

$$P(R_n \leq r) = \frac{1}{n!} \int_{0}^{- \log(1 - F(r))} v^n e^{-v} dv \quad (4.11)$$

If $F$ is absolutely continuous with probability density function (pdf) $f$, we may differentiate Equation (4.10) to obtain the pdf for $R_n$:

$$f_{R_n}(r) = f(r) \frac{(- \log(1 - F(r)))^n}{n!}. \quad (4.12)$$

The result presented in Equation (4.10) was first proved by Chandler (1952) using a different approach. The relationship between the distribution of record values and exponential distribution is not new. Ahsanullah (1978) proved that if a sequence of independent random variables follows common exponential distribution with rate parameter $\lambda$, then the resulting $n$th record is distributed as Gamma($n + 1, \lambda$) and the $n$th jump statistic ($R_n - R_{n-1}$) follows the same exponential distribution with rate parameter $\lambda$.

Applying transformation (4.8) coordinstewise we obtain the joint pdf of
the set of records \( R_0, R_1, \ldots, R_n \), i.e.,

\[
f_{R_0, R_1, \ldots, R_n}(r_0, r_1, \ldots, r_n) = \frac{\prod_{i=0}^{n} f(r_i)}{\prod_{i=0}^{n-1} (1 - F(r_i))} \tag{4.13}
\]

\[
= f(r_n) \prod_{i=0}^{n-1} h(r_i), \tag{4.14}
\]

where \( h(r) = \frac{dH(r)}{dr} = \frac{f(r)}{1-F(r)} \) represents the hazard function.

### 4.2.2 Record times and related statistics

As mentioned earlier, under the assumption that \( F \), the common cdf of \( X_i's \), is continuous, the distribution of record times and counts \( (T_n, N_n, \Delta_n) \) does not depend on \( F \). In order to discuss their distributional properties, we introduce a sequence of record indicator random variables as follows:

\[
I_n = I(X_n > \text{max}\{X_1, \ldots, X_{n-1}\}), \tag{4.15}
\]

for \( n > 1 \) and \( I_1 = 1 \) with probability 1. It is not difficult to verify that the \( I_n's \) are independent variables with

\[
P(I_n = 1) = \frac{1}{n}, \quad n \geq 1. \tag{4.16}
\]

In other words, \( I_n \) is a Bernoulli random variable with success probability \( p = \frac{1}{n} \) with \( n \geq 1 \). It is more convenient to give an intuitive interpretation to this result: after the first record (which is the first observation in a sequence with probability 1) the second observation has a probability of \( \frac{1}{2} \) of beating the first record; the third observation could either smaller than the first observation, between the first and the second observation, or larger than the second observation, which give it a probability of \( \frac{1}{3} \) for beating the previous record. In general, each random variable has the same chance of being the largest and hence, the \( n \)th observation has a probability of \( \frac{1}{n} \) for being a record. With this reasoning, the results given by (4.16) easily follows.
4.2. Basic Distributional Results

With the help of $I_n$, we have the following facts about the record counting process $\{N_n, n \geq 1\}$:

$$N_n = \sum_{i=1}^{n} I_i.$$  \hspace{1cm} (4.17)

Since $I_i's$ are independent Bernoulli random variables, we immediately see that

$$\mathbb{E}N_n = \sum_{i=1}^{n} \frac{1}{i} \approx \ln n$$  \hspace{1cm} (4.18)

and

$$\text{Var}N_n = \sum_{i=1}^{n} \frac{1}{i} (1 - \frac{1}{i}) \approx \ln n.$$  \hspace{1cm} (4.19)

As we just confirmed, records are clearly not common. In a sequence of 1000 observations we expect to see only about 7 records. Another fact that immediately follows is that the expected number of records goes to infinity as the number of samples go to infinity, which is due to the divergence of harmonic series. In fact, we also have that $N_n \to \infty$ as $n \to \infty$ (see Glick (1978) for a proof). The precise distribution of $N_n$ is complicated, but the expressions have been given by several authors (Rényi, 1962; David and Barton, 1962; Karlin, 1966):

$$P(N_n = k) = \frac{S_n^k}{n!} \approx \frac{\ln(n)^k}{nk!}$$  \hspace{1cm} (4.20)

for large sample size $n$, where $S_n^k$ is the Stirling number of the first kind.

We may use the information regarding the distribution of $N_n$ and $I_n$ to discuss the distribution of the kth non-trivial record time $T_k$. Note that the events $\{T_k\}$ and $\{N_n = K = 1, N_{n-1} = k\}$ are equivalent. Consequently,

$$P(T_k = n) = P(N_n = K = 1, N_{n-1} = k) = P(I_n, N_{n-1} = k).$$  \hspace{1cm} (4.21)

Since the events $\{I_n = 1\}$ and $\{N_{n-1} = k\}$ are independent, we have

$$P(T_k = n) = \frac{1}{n (n-1)!} \frac{S_{n-1}^k}{n!} = \frac{S_{n-1}^k}{n}.$$  \hspace{1cm} (4.22)
Note that $T_k$ grows rapidly as $k$ increases. In fact, Glick (1978) verified that

$$\mathbb{E}T_k = \infty, \quad \forall k \geq 1$$

(4.23)

and

$$\mathbb{E}\Delta_k = \mathbb{E}(T_k - T_{k-1}) = \infty, \quad \forall k \geq 1.$$  (4.24)

Neuts (1967) first developed an approach for computing the exact distribution of $\Delta_k$. By conditioning on the value of $R_{k-1}$ (assuming for convenience without loss of generality that the $X’s$ are standard exponential random variable), we have, for $j = 1, 2, \ldots, k = 1, 2, \ldots$

$$P(\Delta_k > j) = \int_0^\infty \frac{x^{k-2}}{\Gamma(k)} e^{-x}(1 - e^{-x})^j \, dx.$$  (4.25)

The integration is readily performed if we expand the term $(1 - e^{-x})^j$, which yields

$$P(\Delta_k > j) = \sum_{m=0}^j \binom{j}{m} (-1)^m \frac{1}{(1 + m)^k},$$

(4.26)

a result in Ahsanullah (1988). Using the above two equations, one may verify that

$$P(\Delta_k = j) = \sum_{m=0}^{j-1} \binom{j-1}{m} (-1)^m \frac{1}{(2 + m)^k}.$$  (4.27)

Note that summing the expression over $j$ provides an alternative proof that, as reported earlier, $\mathbb{E}\Delta_k = \infty$.

4.2.3 Markov chains

There are several Markov chains lurking in the background of any discussion of record values and related statistics. We list some of them that are most helpful for characterizing the behaviour of record values and record times.

First we observe that the record counting process $\{N_n, n \geq 1\}$ is a non-stationary Markov process with transition probabilities $P(N_n = j|N_{n-1} = i)$
4.3. \( k \)-DPP Approximations as Record Values

given by

\[
p_{ij} = \begin{cases} \frac{n-1}{n}, & j = i \\ \frac{1}{n}, & j = i + 1. \end{cases} \tag{4.28}
\]

Next note that \( \{T_n, n \geq 0\} \) forms a stationary Markov chain with \( P(T_0 = 1) = 1 \) and the transition probabilities \( P(T_n = j|T_{n-1} = i) \) given by

\[
p_{ij} = \frac{i}{j(j-1)}, \quad j > i. \tag{4.29}
\]

It is also obvious that \( \{R_n\} \) is a Markov chain with \( R_0 \sim F \) and the transitions governed by

\[
f_{R_n|R_{n-1}} = \frac{f(r_n)}{1 - F(r_{n-1})}, \quad r_n > r_{n-1}. \tag{4.30}
\]

Finally, an interesting observation due to Strawderman and Holmes (1970) is that \( \{(R_n, \Delta_{n+1}), n \geq 0\} \) is also a Markov chain. Given the sequence \( \{R_n\} \), the \( \Delta_{n+1} \)'s are conditionally independent geometric random variables with

\[
P(\Delta_{n+1} > k|\{R_n, n \geq 0\}) = P(\Delta_{n+1} > k|R_n = r_n) = (F(r_n))^k. \tag{4.31}
\]

4.3 \( k \)-DPP Approximations as Record Values

In this section, we use results from record value theory presented previously as tools to study the statistical behaviour of the log-determinants generated from the \( k \)-DPP approximations developed in Section 3.

4.3.1 Jittering the log-determinants

As most of the analytical results for records are developed under the assumption that the random sequence is independently generated from a common continuous distribution, we would like to transform the random sequence of log-determinants, which has finite support, to its continuous counterpart. Another justification, in favour of record value theory, is that we do not
4.3. \(k\)-DPP Approximations as Record Values

want to concern ourselves about ties in the random sequence. The reason is obvious: we would like to study how often strictly better approximations appear.

We refer the method we use for the transformation to Gaussian jittering. Let \(\{X_i, i \geq 1\}\) be a sequence of log-determinants generated using Algorithm 7. For each \(X_i, i = 1, \ldots, n\), where \(n\) is the sample size, we create

\[
Y_i|X_i \sim \text{Gaussian}(X_i, \sigma^2), \quad i = 1, \ldots, n,
\]

so that we have a sequence of independent \(\{Y_i\}\) generated from a continuous cdf. By choosing \(\sigma^2\) to be very small, the values of \(Y_i\)'s and the behaviour of the random sequence should be identical to that of \(X_i\)'s except in that

\[
P(Y_l = Y_m) = 0, \quad \forall l \neq m,
\]

and the corresponding random sequence of records will be strictly increasing and non-terminating.

We argue based on heuristic reasoning that we can consider the \(\{Y_i\}\) a sequence of log-determinants with the corresponding subsets generated from a \(k\)-DPP distribution. More importantly, we are able to instead study any statistical behaviour of \(\{Y_i\}\) and generalize them to that of \(\{X_i\}\) with ignorable errors.

4.3.2 Distribution of the jittered log-determinants

The next step before applying record value theory is to find an appropriate common cdf \(F\) for the jittered log-determinants. Since it would be unnecessarily complicated to study the precise analytical cdf of \(Y\) due to the Gaussian jittering transformations, we fit a distribution from some parametric family that best describes the data. Figure 4.1 shows histograms and corresponding density estimates of log-determinants generated from a \(k\)-DPP using the kernel described in Section 3.3.1. Note that the two plots with different sample sizes consistently suggest that \(Y\) is distributed like slightly skewed Gaussian. In fact, a member of the Gaussian distribution
family would fit the data well for most parts except for the tails. However, the reason that we do not consider such a distribution is that the “right” extremes are the ones that most concern us. For example, for the particular kernel we used to generate our plots, the approximations are starting to get very close to the truth after two standard deviations from the centre. For reasons such as these, we consider distributions from the extreme value theory literature to fit our data. In particular, we use a method generally referred to as “Peaks-Over-Threshold” (Leadbetter, 1990), which relies on extracting, from a continuous phenomenon, the peak values reached for any period during which values exceed a certain threshold.

Figure 4.1: Top row: Histograms of $k$-DPP log-determinants with sample sizes 100,000 and 500,000 respectively; bottom row: Kernel density estimates of $k$-DPP log-determinants with sample sizes 100,000 and 500,000 respectively.
Extreme value theory and peaks-over-threshold

We first provide a very brief introduction to extreme value theory and the two most common distribution families to model extreme values. Let \( X_1, \ldots, X_n \) be a sequence of i.i.d. random variables with a common cdf \( F \). Let \( M_n = \max\{X_1, \ldots, X_n\} \). Suppose there exists normalizing constants \( a_n > 0 \) and \( b_n \) such that

\[
P\left( \frac{M_n - b_n}{a_n} \leq y \right) = F^n(a_ny + b_n) \to G(y)
\]

as \( n \to \infty \) for all \( y \in \mathbb{R} \), where \( G \) is a non-degenerate distribution function. According to the Extremal Types Theorem (Fisher and Tippett, 1928), \( G \) must be either Fréchet, Gumbel or negative Weibull. Jenkinson (1955) noted that these three distributions can be merged into a single parametric family: the Generalized Extreme Value (GEV) distribution. The GEV has a distribution function defined by

\[
G(y) = \exp\left[ -\left(1 + \frac{y - \mu}{\sigma}\right)^{-1/\xi} \right],
\]

where \((\mu, \sigma, \xi)\) are the location, scale and shape parameters respectively with \( \sigma > 0 \). Note that \( z_+ = \max\{z, 0\} \). The Fréchet distribution is obtained when \( \xi > 0 \), the negative Weibull case is obtained when \( \xi < 0 \), and the Gumbel case is obtained when \( \xi \to 0 \).

From this result, Pickands III (1975) showed that the limiting distribution of normalized excesses over a threshold \( \mu \) as the threshold approaches the endpoint \( \mu_{\text{end}} \) of the variable of interest, is the Generalized Pareto Distribution (GPD). That is, if \( X \) is a random variable which satisfies (4.34), then

\[
P(X \leq y|X > \mu) \to H(y), \quad \mu \to \mu_{\text{end}}
\]

with

\[
H(y) = 1 - (1 + \xi \frac{y - \mu}{\sigma})^{-1/\xi},
\]

where again \((\mu, \sigma, \xi)\) are the location, scale and shape parameters respectively with \( \sigma > 0 \). Note that the Exponential distribution is obtained by
4.3. *k*-DPP Approximations as Record Values

continuity as $\xi \to 0$.

In practice, these two asymptotic results motivated modelling block maxima with a GEV, while peaks-over-threshold with a GPD.

**Fitting GPD to jittered log-determinants**

We now use the peaks-over-threshold scheme to fit a GPD to the tail of the jittered log-determinants data. Although many estimators have been proposed in the literature (see Coles et al. (2001), for example, for a detailed discussion), for simplicity, we follow the maximum likelihood estimation procedure for point estimation for the parameters $(\mu, \sigma, \xi)$.

Note that the location parameter $\mu$ for the GPD or equivalently the threshold is a particular parameter as most often it is not estimated as the other ones, and it is possible to estimate the other two parameters using MLE with varying threshold. The main goal of threshold selection is to select enough events to reduce the variance; but not too much as we could select events coming from the central part of the distribution (i.e., not extreme events) and induce bias. In practice, threshold selection is usually done with the aid of exploratory tools, however, decisions are not so clear-cut for real data example. In our study, since we have quite a lot of data we set $\mu$ to equal the 90th percentile of the log-determinants. For experiments, we have also tried the 70th and 80th percentiles for $\mu$ and found no material difference in the fitted distributions.

The results for 100,000 log-determinants generated from the 10-DPP with the same kernel as the one described in Section 3.3.1 are shown in Figure 4.2. For comparison, we also fit a Weibull and a Normal distribution to the entire dataset. Note that the fitted GPD shows advantage in terms of the goodness of fit in the tail part (i.e., data above the 90th percentile). Again the right tail of the data is what we are really interested in as the good approximations appear in this region. As a result, we can ignore the goodness of fit for parts of the data below our threshold.
4.3. *k*-DPP Approximations as Record Values

Figure 4.2: From top to bottom: histograms and estimated theoretical density functions (left) and quantile-to-quantile plot (right) displayed for the data above 90\(^{th}\) percentile for GPD, Weibull, Normal distributions, respectively.

4.3.3 Log-determinants records

The end goal of analyzing the candidate distributions for modelling jittered log-determinants is to study the sequence of records generated from them. Recall from Equations (4.12) and (4.10) that the distribution of records can
be derived from the distribution of the original sequence. Formally, let $R_d$ where $d \geq 0$ represents $d$th upper record from the jittered log-determinants sequence $\{Y_n, n \geq 1\}$. Then, we have

$$f_{R_d}(r) = f(r)\left(\frac{\log(1 - F(r))}{d!}\right)^d,$$

where

$$f(y) = f_Y(y) = \frac{1}{\sigma}(1 + \hat{\xi}y - \hat{\mu})^{-1/(1+\hat{\xi})}$$

and

$$F(y) = F_Y(y) = 1 - (1 + \hat{\xi}y - \hat{\mu})^{-1/\hat{\xi}}.$$

Note that the support of the last two equations is such that $y \geq \hat{\mu}$ for $\hat{\xi} \geq 0$ and $\hat{\mu} \leq y \leq \hat{\mu} - \hat{\sigma}/\hat{\xi}$ for $\hat{\xi} < 0$. Here $(\hat{\mu}, \hat{\sigma}, \hat{\xi})$ are MLE for $(\mu, \sigma, \xi)$.

Before delving further into the distributional results, many distribution-free results from record value theory can be applied to study the behaviour of our log-determinants approximations. Recall that the expected number of records in a sample of size $n$ can be approximated by $\log(n)$. In Figure 3.4 where we show the progression of 10-DPP approximations using 100,000 samples, the black dots represent the occurrence of records which count to 12. Note that this is really close to the expected value, which is $\log(100,000) \approx 11.51$. Observe in Figures 3.4 and 3.5 is that the occurrence of records is more frequent at the beginning when sample sizes are small than when the sample sizes are big. This can be seen from the “diminishing return” property of the logarithm function and it is also an immediate application of the Markov proper of the record counting process. Recall that $\{N_n, n \geq 1\}$ is a Markov process with

$$P(N_n = j|N_{n-1} = i) = \begin{cases} \frac{n-1}{n}, & j = i \\ \frac{1}{n}, & j = i + 1. \end{cases}$$

Therefore, the probabilities of record-breaking in the first few hundreds of samples are much higher than those in the later samples.
### 4.3. \( k \)-DPP Approximations as Record Values

**Conditional probabilities of record increments**

A key distributional result we use to model the behaviour of the \( k \)-DPP approximations is the conditional probability relating any two consecutive record values. From Equation 4.30, we have

\[
P(R_{d+1} > r | R_d = r_d) = \frac{1 - F(r)}{1 - F(r_d)}.
\]  

(4.42)

By letting \( r = (1 + \epsilon) r_d \) for some \( \epsilon > 0 \), we have

\[
\delta_d = P(R_{d+1} > (1 + \epsilon) r_d | R_d = r_d) = \frac{1 - F((1 + \epsilon) r_d)}{1 - F(r_d)}.
\]

(4.43)

For \( \epsilon < \epsilon_{\text{small}} \) and \( \delta_d < \delta_{\text{small}} \), we can say that the probability of a small increment in the next record value is very small. In the context of \( k \)-DPP approximations, this is equivalent to saying that the chance of getting a better approximation is small. Although \( P(R_{d+1} > r_d) = 1 \) by definition, the \( \epsilon \) term enables us to quantify how good the next approximation can be. One other use of Equation 4.43 is: if we have an approximation \( m \) from another algorithm, then by setting \( r_d = m \) the conditional probability tells us the chance that the next approximation from the \( k \)-DPP is better than \( m \). Intuitively, this probability is an increasing function of \( d \). In fact, it is possible to inversely compute \( d \) such that the probability of beating \( m \) is higher than some user defined threshold.

To illustrate how the conditional probabilities behave, we present numerical examples using the same sample of (jittered) log-determinants generated from the 10-DPP in Section 3.4.1. Table 4.1 shows the occurrence of records and the associated conditional probabilities of observing better approximations (with \( \epsilon = 0.0005 \)) given the current record values. Note that the sequence of probabilities is decreasing with the number of records, with the first few of them being equal to or close to one - record-breaking is more frequent at the beginning and the jumps are higher. The last record observed within this sample, though jittered, is actually the optimum value for the specific kernel. Note that the associated conditional probability is
identically 0. We also compute the conditional probabilities for the same sample but conditioning on the greedy approximation. The probabilities are now increasing with the progression of records - the chance of achieving better approximations (better than the greedy approximation) is higher. Note that the last record, which is better than the greedy approximation, has an associated conditional probability of 1.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>78.03975</td>
<td>1.0000</td>
<td>0.0000</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>78.04162</td>
<td>1.0000</td>
<td>0.0000</td>
<td>1</td>
</tr>
<tr>
<td>7</td>
<td>78.52475</td>
<td>0.8957</td>
<td>0.0000</td>
<td>1</td>
</tr>
<tr>
<td>16</td>
<td>78.85338</td>
<td>0.8691</td>
<td>0.0000</td>
<td>3</td>
</tr>
<tr>
<td>37</td>
<td>79.36103</td>
<td>0.7856</td>
<td>0.0000</td>
<td>34</td>
</tr>
<tr>
<td>234</td>
<td>79.39688</td>
<td>0.7756</td>
<td>0.0000</td>
<td>43</td>
</tr>
<tr>
<td>861</td>
<td>79.66752</td>
<td>0.6549</td>
<td>0.0001</td>
<td>357</td>
</tr>
<tr>
<td>1758</td>
<td>79.89616</td>
<td>0.3823</td>
<td>0.0031</td>
<td>9707</td>
</tr>
<tr>
<td>3573</td>
<td>79.93738</td>
<td>0.2873</td>
<td>0.0408</td>
<td>26276</td>
</tr>
<tr>
<td>44293</td>
<td>80.09011</td>
<td>0.0000</td>
<td>&gt; 1.0000</td>
<td>$1.3386 \times 10^{10}$</td>
</tr>
</tbody>
</table>

Table 4.1: The first two columns provides a summary of the occurrence time of records and their values; the second and the third columns provides the conditional probabilities given the current record values and the conditional probabilities given the greedy approximation respectively; the last column gives the expected waiting times for the next record.

**Conditional probabilities of inter-record times**

Recall from Equation 4.31 that given the current sequence of record values, the next inter-record times are conditionally independent geometric variables with $1 - p = F(r_d)$ where $p$ is the parameter for geometric distribution and $r_d$ is the $d$th record value. This distributional result gives us another way (besides the conditional probabilities of record increments) to design a stopping criteria for our $k$-DPP approximation algorithm. Specifically, we can stop sampling if the conditional probability of the waiting time for the next better approximation higher than a pre-specified value is high, or
the expected waiting time for the next better approximation is long. For illustration, we include in Table 4.1 the expected waiting times given the current sequence of record values. As anticipated, the last expected waiting time is huge, which indicates a stopping point for the algorithm. In fact, as discussed before, the last record value is already the theoretical maximum and hence should be the stopping point.

Essentially, a combination of the conditional probabilities and the expected waiting times provides us with an incisive tool to analyze our $k$-DPP approximations. In the numerical example above, these values tell us that we have reached an approximation such that the probability of obtaining a better solution is very low and the waiting time for this hypothetical better solution is expected to be very long. In other words, we should stop sampling.

4.4 Discussion

In this chapter we have explored mainly non-asymptotic properties of record values and the related statistics and applied them to model jittered log-determinants generated by $k$-DPP. There are many intriguing asymptotic results for record values (see Resnick (1973) for detailed discussions) that concern their behaviours when the number of records is large. Although these results are quite mathematically elegant and do provide insightful ideas, we prefer to rely on exact non-asymptotic theory and, it is fairly expensive to sample a large number of records. However, we have had interesting results based only on simple distributional facts (i.e. distribution and density functions). In particular, many of the observed behaviours seen in the $k$-DPP approximations, such as the quick escalation at the beginning and the diminishing return near the end, can be explained by record value theory. We also developed informative stopping rules for the search algorithm using a combination of distributional results for record values and inter-record times.

In order for the log-determinants sampled from the $k$-DPP to fit into the classical record model, we introduced the idea of Gaussian jittering.
4.4. Discussion

Although it worked quite well in our examples, in future work, we plan to rigorously quantify the approximation errors resulted from the jittering. In addition to the classical records model, there are more general record models, such as the $F^n$ record model (Yang, 1975) and the Pfeifer model (Pfeifer, 1982) where records are generated from a dependent sequence determined by the record history. Future research can focus on modelling the $k$-DPP approximations using non-classical record models.
Chapter 5

Concluding Remarks

This thesis was motivated by the combinatorial optimization problem that arises in the maximum-entropy approach for designing monitoring networks, which plays a critical role in the surveillance of environmental processes. We introduced a sampling-based stochastic search method using $k$-DPPs, and described how the methodology is able to handle datasets with different structures and sizes. We illustrated how the $k$-DPP based method is able to provide more accurate and more robust approximations than the deterministic greedy-type algorithm, and how the former is comparable to another stochastic search method - the GA. The proposed method is easily parallelizable so that it takes advantage of the extensive high-performance computing resources that are available and saves computation time.

More importantly, we explored the theory of record values in order to assess the quality of the stochastic approximations made by $k$-DPPs. In particular, using basic distributional results for record values and the peaks-over-threshold method from extreme value theory, we approximated the conditional distributions of the next-best $k$-DPP approximations given the current ones. With this result, we were also able to compare $k$-DPP approximations with those produced by other methods and gain insights into their quality. In addition, we developed a guideline for developing stopping criteria for the proposed method using expected waiting times between records.
5.1 Future Work

5.1.1 MCMC k-DPP sampling

In chapter 3, we developed the approximation method using independent sampling of k-DPPs. We relied on independent sampling mainly for the ease of applying the classical records models and the parallelizability of the sampling algorithm. However, the implementation quickly becomes expensive due to the \(O(Nk^3)\) computational complexity of Algorithm 7. It has been shown that the MCMC sampling algorithm described in the same chapter can be run in \(O(N^2k)\) time, as demonstrated by Li et al. (2017) using efficient calculation of the transition probabilities. This offers a huge reduction in computational time for large datasets. Future research then aims to compare the accuracies of the stochastic approximations based on independent k-DPP sampling and MCMC k-DPP sampling, respectively. Furthermore, we also aim to explore more general records models, again in order to assess the quality of these stochastic approximations.

5.1.2 Combination of k-DPP and other stochastic search methods

An observation from the numerical experiments presented in chapter 3 is that approximations from the proposed method often accelerates quickly at the beginning of the iterations. As confirmed by record value theory, our method experienced more jumps at the beginning, in comparison, the GA in comparison needed more iterations to “warm up” but required less computational time in total. We saw that the k-DPP approximation method achieved good solutions after a short time, while the GA started to outperform the k-DPP solutions after warming up. The study of the combination of the two methods and how well the combination compares to the original methods is a subject for future research.
Bibliography


Bibliography

R Core Team. *R: A Language and Environment for Statistical Computing*. 
R Foundation for Statistical Computing, Vienna, Austria, 2018. URL 
https://www.R-project.org/.


Appendix A

R Code for Sampling from a $k$-DPP

```
# $k$-DPP Sampler
# Note: Based on Alex Kulezsa’s sampling algorithms for DPPs
# MATLAB code available online at
# http://web.eecs.umich.edu/~kulesza/code/dpp.tgz
# Check Kulezsa’s thesis (2012) for description of the algorithms
# The code assumes eigenvectors LV and the associated
# eigenvalues lambda of the kernel in advance

# Compute Elementary Symmetric Polynomials #

Epoly = function(lambda, k){
  N = length(lambda)
  E = matrix(0, nrow = k+1, ncol = N+1)
  E[1,] = 1
  for (l in 2:(k+1)){
    for (n in 2:(N+1)){
      E[l,n] = E[l,n-1] + lambda[n-1]*E[l-1,n-1]
    }
  }
}
```
Appendix A. R Code for Sampling from a $k$-DPP

```r
return(E)
}

# # # # # # # # # # # # # # # # # # # # # # # # #
# Select k eigenvalues  #
# # # # # # # # # # # # # # # # # # # # # # # # #

ChooseEigenVals = function(lambda,k){
  # Obtain elementary symmetric polynomials
  E = Epoly(lambda,k)

  # Initialize to store k selected eigenvalues
  S = array(0, dim = k)

  i = length(lambda)
  remaining = k

  while (remaining > 0){

    # Compute marginal of i given that we have choosen
    # the remaining values from 1:i
    if (i == remaining){
      marg = 1
    } else{
      marg = lambda[i] * E[remaining,i] / E[remaining +1,i+1]
    }

    # Sample marginal
    if (runif(1) < marg){
      S[remaining] = i
      remaining = remaining - 1
    }
  i = i-1
```

59
Appendix A. R Code for Sampling from a k-DPP

```r
DPPSample = function(L, k, lambda, LV){
    # Initializing vector to save k-DPP sample
    y = array(0, dim = k)

    # Selecting eigenvalues for k-DPP
    v = ChooseEigenVals(lambda, k)

    # Corresponding columns in LV matrix
    V = LV[,v]

    for (i in (k:1)){
        # Compute probabilities for each item
        P = rowSums(t(t(V))^2)
        P = P / sum(P)

        # Choose a new item to include
        y[i] = which.max(runif(1) <= cumsum(P))

        if (k == 1) break;

    # Choose a vector to eliminate
```
Appendix A. R Code for Sampling from a k-DPP

```r
j = which.max(V[y[i],] != 0)
Vj = t(t(V[,j]))
V = t(t(V[-j]))

# update V
V = V - Vj %*% (V[y[i],] / Vj[y[i]])

# orthogonalize
if (ncol(V) != 0) {
  normV = max(svd(V)$d)
  if (normV != 1) {
    if (ncol(V) != 1) {
      for (a in which((1:(i-1))>1)) {
        for (b in which((1:(a-1))>1)) {
          V[,a] = t(t(V[,a])) -
          as.numeric(t(V[,a])%*%t(t(V[,b]))) * t(t(V[,b]))
        }
      }
      V[,a] = V[,a] / max(svd(V[,a])$d)
    }
  }
}
else { V[,1] = V[,1] / max(svd(V[,1])$d) }

y = sort(y)
return(y)
```
Appendix B

Python Code for Genetic Algorithm

The implementation is based on DEAP (Fortin et al., 2012), an evolutionary computation framework developed in Python.

# This script uses some functionalities of DEAP.

import random
import numpy as np

from deap import base
from deap import creator
from deap import tools

creator.create("FitnessMax", base.Fitness, weights=(1.0,))
creator.create("Individual", list, fitness=creator.FitnessMax)

toolbox = base.Toolbox()

# Attribute generator
# define 'attr_bool' to be an attribute ('gene')
# which corresponds to integers sampled uniformly
# from the range [0,1] (i.e. 0 or 1 with equal probability)
toolbox.register("attr_bool", random.randint, 0, 1)

# Structure initializers
# define 'individual' to be an individual
# consisting of 30 'attr_bool' elements ('genes')
toolbox.register("individual", tools.initRepeat,
                 creator.Individual, toolbox.attr_bool, 30)

# define the population to be a list of individuals
toolbox.register("population", tools.initRepeat,
                 list, toolbox.individual)

# the goal ('fitness') function to be maximized
def evalMaxDet(individual):

    Kernel = np.loadtxt("covEntropy30-86.txt")
    nonzeroInd = np.nonzero(individual)[0]
    subKernel = Kernel[nonzeroInd][:, nonzeroInd]

    if sum(individual) == 10:
        return (np.linalg.det(subKernel),)
    else:
        return (0,)

# Operator registration
# register the goal / fitness function
toolbox.register("evaluate", evalMaxDet)

# register the crossover operator
toolbox.register("mate", tools.cxTwoPoint)
# register a mutation operator with a probability to
# flip each attribute/gene of 0.05
toolbox.register("mutate", tools.mutFlipBit, indpb =0.05)

# operator for selecting individuals for breeding the
# next
generation: each individual of the current
generation
# is replaced by the 'fittest' (best) of three
# individuals
drawn randomly from the current generation.
toolbox.register("select", tools.selTournament, tournsize=3)

#-----

def main():
    # random.seed(750)
    # create an initial population of 300 individuals
    # (where
    # each individual is a list of integers)
    pop = toolbox.population(n=300)

    # CXPB is the probability with which two
    # individuals
    # are crossed
    #
    # MUTPB is the probability for mutating an
    # individual
    CXPB, MUTPB = 0.5, 0.2
Appendix B. Python Code for Genetic Algorithm

```python
print("Start of evolution")

# Evaluate the entire population
fitnesses = list(map(toolbox.evaluate, pop))
for ind, fit in zip(pop, fitnesses):
    ind.fitness.values = fit

print("Evaluated %i individuals" % len(pop))

# Extracting all the fitnesses of
fits = [ind.fitness.values[0] for ind in pop]

# Variable keeping track of the number of
generations

# Begin the evolution
while g < 1000:
    # A new generation
    g = g + 1
    print("Generation %i" % g)

    # Select the next generation individuals
    offspring = toolbox.select(pop, len(pop))
    # Clone the selected individuals
    offspring = list(map(toolbox.clone, offspring))

    # Apply crossover and mutation on the offspring
    for child1, child2 in zip(offspring[::2], offspring[1::2]):
```

65
Appendix B. Python Code for Genetic Algorithm

```
# cross two individuals with probability CXPB
if random.random() < CXPB:
    toolbox.mate(child1, child2)

# fitness values of the children
# must be recalculated later
del child1.fitness.values
del child2.fitness.values

for mutant in offspring:
    # mutate an individual with probability MUTPB
    if random.random() < MUTPB:
        toolbox.mutate(mutant)
        del mutant.fitness.values

# Evaluate the individuals with an invalid fitness
invalid_ind = [ind for ind in offspring if not ind.fitness.valid]
fitnesses = map(toolbox.evaluate, invalid_ind)
for ind, fit in zip(invalid_ind, fitnesses):
    ind.fitness.values = fit

print("Evaluated\%i\ individuals" % len(invalid_ind))

# The population is entirely replaced by the offspring
pop[:] = offspring
```
Appendix B. Python Code for Genetic Algorithm

```python
# Gather all the fitnesses in one list and print the stats
fits = [ind.fitness.values[0] for ind in pop]

length = len(pop)
mean = sum(fits) / length
sum2 = sum(x*x for x in fits)
std = abs(sum2 / length - mean**2)**0.5

print("Min\%s" % min(fits))
print("Max\%s" % max(fits))
print("Avg\%s" % mean)
print("Std\%s" % std)

print("---End_of_successful_evolution---")

best_ind = tools.selBest(pop, 1)[0]
print("Best individual is \%s, \%s" %
      (best_ind, best_ind.fitness.values))
# print(best_ind.fitness.values[0])

if __name__ == '__main__':
    main()
```