# Inaugural-Dissertation

zur Erlangung der Doktorwürde

der

Naturwissenschaftlich-Mathematischen Gesamtfakultät

der

Ruprecht-Karls-Universität Heidelberg

vorgelegt von

Diplom-Mathematiker Peter Markowsky

aus Solingen

Tag der mündlichen Prüfung: . . . . . .

Thema:

# Model-based Stochastical Segmentation of Higher-dimensional Data

Gutachter: Prof. Dr. Christoph Schnörr
Ruprecht-Karls-Universität Heidelberg

**Zusammenfassung**

Der Ausgangspunkt dieser Arbeit ist das Problem der Segmentierung extrem verrauschter Bilder geometrischer Objekte. Zu diesem Zweck untersucht sie die Kombination eines randomisierten Ansatzes zur Lösung des kombinatorischen Mengenüberdeckungsproblems mit einem statistischen Modell der Objekt-Interaktion.

Die Auffassung als Mengenüberdeckungsproblem bietet die Möglichkeit einer stabilen Segmentierung selbst in Fällen, in denen viele traditionelle Methoden der Bilderkennung aufgrund von Rauschen und unvollständiger Bildinformation versagen. Das statistische Modell liefert zusätzliche Information, die nicht direkt durch die Bilddaten gegeben ist, und führt zu einer realistischeren Widergabe physischer Objekteigenschaften in der Segmentierung.

Diese Arbeit gliedert sich in drei Teile. Teil eins behandelt Themen der randomisierten kombinatorischen Optimierung. Dies beinhaltet die Verbesserung von Konvergenzgrenzen eines bestehenden Ansatzes und dessen Parallelisierung. Ferner werden Verbindungen zwischen verwandten Ansätzen zur Lösung verschiedener kombinatorischer Probleme aufgezeigt, wie zum Beispiel des Mengenüberdeckungproblems und der generellen linearen Optimierung.

Im zweiten Teil der Arbeit wird ein punktprozessbasiertes, auf die spätere Anwendung zugeschnittenes Modell von paarweiser Objekt-Interaktion erstellt. Diskutiert werden theoretische und praktische Schwierigkeiten bei dessen Simulation, Schätzung und Verbindung mit einem kombinatorischen Ansatz.

In Teil drei werden die bisher diskutierten Methoden empirisch ausgewertet. Dies umfasst den Vergleich auf simulierten Datensätzen sowie die Anwendung auf echte Daten in Form von mikroskopischen Zellaufnahmen und dreidimensionalen $\mu$CT-Scans faserverstärkter Kunststoffe.

**Abstract**

This thesis is motivated by the problem of segmenting extremely noisy images of geometric objects. To this end, it combines randomized combinatorial set cover optimization with a statistical model of object interaction. The set cover approach provides stability and applicability in cases in which many traditional methods of segmentation fail due to noise and imperfect data. The statistical model provides additional information that is not directly supplied by the image, and leads to a more realistic depiction of physical object properties in the resulting segmentation.

This dissertation is divided into three parts: The first covers topics of randomized combinatorial optimization. This includes improving bounds of convergence and establishing a method of parallelization for an existing approach, as well as linking solutions to different combinatorial problems, such as geometric set cover and a general linear program.

Part two is concerned with constructing a point process model of object interaction that fits later applications, and exploring some theoretical and practical pitfalls in its simulation, estimation, and coupling with a combinatorial approach.

Part three compares previously discussed methods empirically, and demonstrates the performance of the established combination of randomized optimization and statistical model on microscopic cell images and 3D $\mu$CT scans of fiber reinforced materials.

**Acknowledgements**

First, I would like to thank Prof. Christoph Schnörr for introducing me to the topic of optimization-based image segmentation, as well as the image and pattern analysis group Heidelberg, and providing the framework for the development of this thesis.

My colleague Tabea Zuber proved to be a constantly reliable and insightful partner in the discussion of our work. In addition, many other members of the IPA group offered interesting talks and useful insights, in particular on the topic of code optimization.

On the subject of the segmentation of cell images, P.D. Karl Rohr supported me with many encouraging and illuminating discussions and tips, in particular for publishing on this topic. I would also like to thank Svenja Reith for her enthusiastic and diligent attitude in our joint work.

In addition, my sincere thanks goes to Prof. Claudia Redenbach of the TU Kaiserslautern for acting as a second advisor for my thesis.

Lastly, I would like to thank the faculty for mathematics and computational science Heidelberg for providing the general framework that made it possible to complete this thesis.

# Contents

# Chapter 1

# Introduction

## 1.1 Overview and Motivation

Important properties of polymer materials reinforced with glass or carbon fibers, such as durability and flexibility, are determined by microscopic properties of the fibers themselves, such as length and relative orientation. These properties can be difficult to control during production, as during manifacturing processes such as molding fibers may bend or break. Therefore quality control based on microtomographical ($\mu$CT) scans of material samples such as the one seen in figure 1.1, and consequently the segmentation of these scans, has in recent years become an increasingly interesting, yet largely unsolved problem.

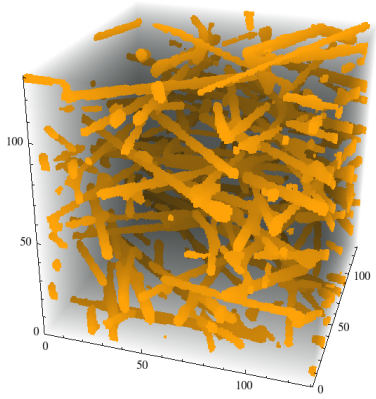


Figure 1.1: $\mu$CT data of Fiber Material

This dissertation originated as part of the AniS project of the German ministry of Education and Research. The project was conceived as a cooperative effort between several universities and companies; its aim was to further the state of the art in the modeling and segmentation of porous foam rubber materials, as well as fiber-reinforced plastics. The focus of the Heidelberg group was on micro CT scans of these fiber materials. Specifically, the goal was a sensible local pre-processing of the given CT data, as well the conception a probabilistic energy model that could be minimized to match physical fiber interaction in a possible segmentation. This was later extended to include the full segmentation and labeling of smaller artificial and real data sets.

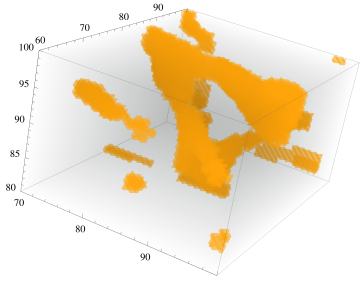


Figure 1.2: Blurred local borders in fiber data

In terms of image segmentation, fiber data presents several challenges: $\mu$CT data is usually noisy and dense – neighboring fibers often lack clear borders, even after local pre-processing – see figure 1.2. This limits the use of many traditional methods that are based on local morphology, such as thresholding, filtering, watershed transformations or identifying the connected components of foreground data.

At the same time, the sheer increase in data size makes the naive transfer of many 2D methods such as the Hough transform or the minimization of a global energy or linear program (LP) unfeasible.

On the other hand, the advantage in modeling and segmenting fiber data such as that shown in figure 1.1 is that the rough structure of a single fiber is already known: It will in good approximation be a long, thin cylinder, albeit with an unknown center, orientation and length. This naturally steers the problem towards the use of parameter-based segmentation methods.

The *Randomized Set Cover* (RSC) approach that served as the initial basis for this dissertation tries to make use of this combined prior knowledge of image data: Interpreting the problem of finding an optimal segmentation for a set of data points as the global problem of minimally covering these points with a collection of predefined geometric objects – namely the cylindrical fiber shapes – supplies high robustness towards local noise and blurred edges. Knowledge of the shape of objects allows us to parameterize the space of possible solutions to increase computational efficiency. At the same time, RSC combats the problem of large data by approximating the global problem in an iterative, randomized way, where each step of the approximation uses smaller, randomized collections of possible solutions.

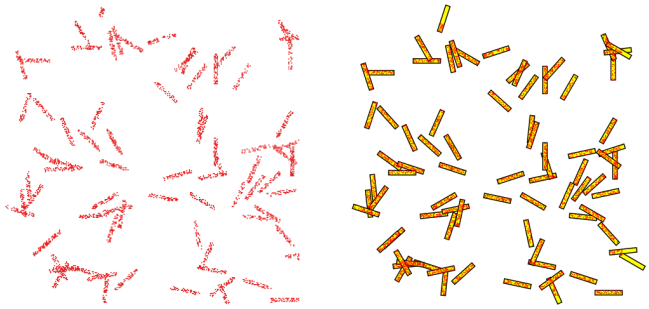


Figure 1.3: Segmentation as a minimal set cover of all data points.

The RSC algorithm operates by iteratively sampling a fixed number of pre-defined sets – which will be cylinders in the case of fiber segmentation – and updating the probability distribution from which these sets are sampled. To guarantee convergence to a full – if not necessarily minimal – cover of all object voxels, the probability of covering voxels that were previously covered with low probability is increased in each step, until all voxels are covered with a sufficiently high probability to sample a full cover.

Mathematically, we find points covered with low probability using $\varepsilon$-nets: Given a (discrete) collection of sets $R_i$ with a probability distribution $p_i$, we can define these as any collection of sets that covers all points that are covered with probability $\geq \varepsilon$ when sampling a single set $R$. See the beginning of chapter 2 for a more detailed examination of their mathematical background.

Although we can guarantee convergence for the RSC approach using $\varepsilon$-nets, a segmentation produced purely as a cover of data voxels will not necessarily reflect the location of the physical objects. Even simple properties, such as physical objects not overlapping, will not necessarily be reflected when approximating a minimal cover of all data points – see figure 1.4. Thus, a natural second step towards the segmentation and labeling of physical objects is to not only consider the given image data and the the geometrical shape of each *single* given object, but also any a priori knowledge of the *interaction* between two or more of these objects.

To achieve this, we modify the RSC algorithm so that the potential covers produced in every step are not a simple random sample, but the minimizer of an energy that reflects object behavior.

We derive this energy from the probability density of a statistical point process model. Specifically, we try to strike a balance between a relatively accurate model of object interaction and the practical applicability of said model by using a *Pairwise Interaction Point Process*, which we will later refer to somewhat casually as the Gibbs model.

Without any prior knowledge, this model can be as simple as prohibiting object overlap. However when given a prior segmentation, we can estimate a more refined pairwise interaction model in a relatively straightforward way by minimizing the quadratic difference of an estimated statistic of our model to that of the previous segmentation. This knowledge can then be used to better segment comparable data. If we do not start with the ground truth of a data set, we can first segment using the simplified model of non-overlap, then produce a second segmentation using the object interaction estimated from the first.

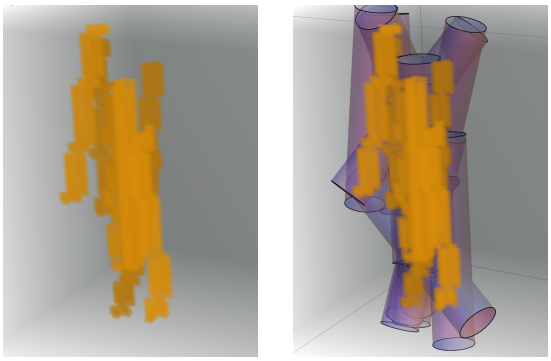


Figure 1.4: A naive approximate cover (right) will not reflect physical properties of data (left), such as the absence of overlap for solid objects.

## 1.2 Related Work

### 1.2.1 Randomized Set Cover Approach

#### Background

One of the first uses of iterative sampling to find a "critical set" was introduced by Clarkson [Cla88] in 1988 amongst several papers exploring the application of randomization in optimization algorithms. This took the form of an algorithm geared towards solving an LP-type problem by finding its *basis*, a minimal set of conditions that fully defined the problem. Initially, Clarkson uniformly sampled random sets of conditions until an approximation of the basis is found. This algorithm was refined by Clarkson in 1995 [Cla95]: conditions are no longer sampled uniformly, but a weight function is assigned to possible candidates, and updated in each step depending on their performance. Convergence is guaranteed by only increasing the probability of "light" elements, i.e. those that are assigned a relatively low weight in the current updating step.

This concept was later generalized by by Brönnimann and Goodrich [BG95] to construct the version of the RSC algorithm most cited today: It operates on a range space $(X, \mathcal{R})$, i.e. a set $X$ and a collection $\mathcal{R}$ of subsets of $X$; we will refer to the subsets that constitute the space as "ranges". Given such a range space, their algorithm aims to find a *hitting set*, i.e. a set of elements $x_i$ that "hits", or has a non-empty intersection with, every range $R \in \mathcal{R}$. It should be noted that the problem of finding a hitting set and the problem of finding a set cover are interchangeable, since the set cover of a range space is equivalent to a hitting set of its dual range space, and vice versa – see section 2.2.

Brönnimann and Goodrich aim to find this hitting set in a way similar to Clarkson's by iteratively sampling and updating the weight of "light" subsets. Finding light subsets to update is achieved by

randomly sampling $\varepsilon$-nets, approximations to a hitting set that do not hit all ranges, but those with a high relative weight. If a set of points is an $\varepsilon$-net for a weighted collection of ranges, then by definition ranges that are *not* hit by this $\varepsilon$-net have a relative weight of lower than $\varepsilon$, and are considered "light". Their proof of convergence to an approximate solution is based on the knowledge that in each step an $\varepsilon$-net can be constructed via random sampling with high probability.

Some more recent approaches modify the scheme of iterative sampling and reweighting by first calculating a weight system in which *every* range has a relative weight of $\varepsilon$ or more, and then constructing an $\varepsilon$-net *once* for these weights. By definition, the $\varepsilon$-net is then a hitting set for the range space. This approach is used e.g. by Even et al. [ERS05] – who find the $\varepsilon$-net via solving a linear program – and Agarwal et al. [AP14] – who use iterative reweighting – to achieve a higher degree of parallelization compared to Brönnimann/Goodrich [BG95]. This approach can however lead to a significant decrease in the quality of the approximation compared to an iterated construction in each step. For a discussion of the problem of large constants in applying reweighting schemes see [BMR15] . This paper was later developed by Bus in [BGMR16] to an algorithm with a similar idea as our approach of parallelization, though without a formal background or proof of convergence, and restricted to disks. We later compare the work of Agarwal et al. [AP14] to our own parallelization approach that calculates $\varepsilon$-nets significantly less often, but more than once – see sections 3.1 and following.

### $\varepsilon$-net Construction

Convergence of all schemes related to the one used by Clarkson and Brönnimann/Goodrich rely on the construction of $\varepsilon$-nets to guarantee their convergence; the specific method used for this construction, however, can vary greatly: Bounds on Clarkson's original method of random sampling were established using the Vapnik-Chernovenkis (VC)-dimension, a measure of combinatorical complexity for the range space $(X, \mathcal{R})$ – see section 2.2. The bounds on the cardinality required so that a randomly sampled set of elements is likely an $\varepsilon$-net were later tightened to their current state in [KPW92].

Elbassioni [Elb16] extended the VC-based approach to infinite range spaces. Some more recent approaches [Var10, Cha16, MDG16, KMP17] use a more complicated construction than random sampling under the assumption that the *shallow cell complexity*, which defines the complexity of the range space using the incidence matrix[1]of its elements and subsets, is known. It should be noted, however, that for large scale problems even explicitly formulating and storing the incidence matrix of the range space may not be computationally feasible. Similarly, methods that substitute solving the linear program with a greedy approach [CM96, You95] can quickly lead to extremely long running times; sampling-based-methods use global information only in a computationally inexpensive way, i.e. by sampling from a distribution over all feasible ranges. Even for the moderately large problems of our later applications this computational advantage far outweighs the worse bounds on net size. Lastly, constructions with improved bounds compared to the general case have been established in many specific geometric settings – see [MV17] or [PR08] for an overview.

Regardless of the specific method used for constructing an $\varepsilon$-net, the value of $\varepsilon$ used in its construction directly determines its cardinality, and thus the quality of the resulting solution to the minimal hitting set problem. One aim of the later chapter 2 is to further tighten the limits for values of $\varepsilon$ under

[1]The incidence matrix is a binary matrix that indicates for each point $p_i$ and set $r_j$ if the point is covered by the set (1) or not (0)

which convergence is still guaranteed by formulating a modified version of the original proof of convergence for the weight-update scheme, as seen e.g. in [BG95] in propositions 2.3.1 and A.1.1. One important property of our parallelized weight update scheme proposed in section 3.1 is that it does not compromise this new bound on $\varepsilon$, and consequently the quality of the produced segmentation. This is in constrast to e.g. the parallelized approach by Agarwal et al. [AP14], which trades the speedup via parallelization for a significant decrease in the quality of the result.

### Related Problems and Methods

One of the most prominent and intuitive methods to solve a minimal set cover problem is the greedy algorithm that iteratively selects the set covering most points that have not been covered in previous steps. While it can be guaranteed to have a cover cardinality of $\mathcal{O}(\ln(n)) * OPT$, where $OPT$ is the optimal cover cardinality – see e.g. in [Vaz01, chapter 2] – its speed tends to scale poorly for larger problems .

Similarly, LP-based approaches generally trade good results for poor speed or even complete inapplicability on larger problems. As mentioned in the previous section, even calculating and storing the incidence matrix of a larger problem can be prohibitively expensive. They can however be useful as a second step of postprocessing for larger problems. One of the most prominent approaches to solve a set cover problem directly, i.e. without the use of $\varepsilon$-nets, is the method of randomized rounding: We first solve the relaxed version of classical set cover LP, then interpret the solution $(x_i) \in [0,1]^n$ as the probabilities of picking a range $R_i$ for the solution. It is shown e.g. in [Vaz01, chapter 14.2] that if we sample ranges independently $t = O(\log n)$ times from the discrete distribution produced by $(x_i)$, a cover is produced with high probability.

One natural extension of the set cover problem is the problem of covering all, or some points multiple times. Chekuri, Clarkson and Har-Peled [CCHP09] solve this by defining a new set system based on creating multiple copies of each point; the multi-cover problem can then be reduced to solving the original set cover problem for the newly-defined range space. This is achieved by Chekuri et al. by bounding the VC-dimension of the new range space and then utilizing the same VC-based framework as Clarkson/Brönnimann/Goodrich. Bien et al. [BT11] tackle the same problem using two approaches similar to the randomized rounding and greedy methods for the classical set cover problem.

Instead of using a linear program to solve a set cover problem, we can also use set cover-related methods to solve a (general) linear program. As mentioned in the beginning of the section 1.2.1, the RSC reweighting scheme by Brönnimann and Goodrich [BG95] is based on the method used by Clarkson [Cla95] to find the basis of a linear program. We will later show in section 4.2 that – up to the change of some constants – Clarkson's algorithm can indeed be explicitly reduced to a special case of applying the B/G reweighting scheme [BG95] to solving a minimal set cover problem of real half-spaces, which represent the linear constraints of the LP.

The randomized B/G scheme for set covering shares some characteristics with the framework of Online learning, as seen e.g. in [CBL06]. Specifically, both repeatedly evaluate multiple non-deterministic elements – "experts" in the case of online learning, randomly sampled sets in the case of B/G – and adjustment their strategy depending on the current state of these elements sets in each step. However, interpreting the sets randomly sampled by the B/G scheme as a form of "expert advice",

we cannot meaningfully define cover quality for a single sampled set, but only for the collection of all sets: Even if a single set covers many points, the use of two such sets may still result in a bad solution to the overall minimal cover problem if they are almost identical. This distinguishes the B/G scheme from the online learning approach, which relies on a collection of experts that can both be judged individually via a loss function, and combined in a relatively straightforward way to define the resulting strategy (i.e. one that still permits useful bounding, such as a weighted average).

The most direct link of online learning and randomized set cover is the multiplicative weights method [AHK12], a framework that relies almost exclusively on the use of weights and sampling, and generalizes both approaches. It also subsumes many other well-known algorithms, such as the Ada boost algorithm [FS95], the Plotkin-Shmoys-Tardos algorithm [PST95] and the Winnow algorithm [Lit88]. This very generalized approach can however come at the cost of less precise bounds on convergence in the special cases. Indeed, applying the framework directly to the randomized set cover approach results in estimates that will be trivial in almost all cases, as we later see in section 4.5.

The RSC approach also shares some similarities with the generalized Hough transform [Bal87], which extended the method originally introduced to segment incomplete line segments via parameter voting – see [IK88] – to all parameterizable objects. Both methods sequentially use single image points to "vote", i.e. update a discrete weight function, on the same space of object parameters. These similarities are heightened further when comparing RSC to the later established randomized versions of the hough transform [KHXO95, KKA00, WR02], in particular the Probabilistic Hough Transform [KEB91] that votes on a random subset of image points; despite this, Hough-based approaches remain somewhat limited in both theory and practice when compared to RSC. We will discuss their relation in more detail in section 4.6 and give a numerical performance comparison in section 7.2.4.

### 1.2.2 Point Process Model

Possibly because of the potentially high computational requirements, literary examples of Bayesian segmentation approaches are still relatively sparse; Dong [DA07] and Khan [KGS15] et al. minimize energy (or, equivalently, maximize likelihood) over a Marked Point Processes (MPP) model for cell segmentation. Two approaches by Soubies and Poulain [SWD15, PPSD15] iteratively minimize a binary model (i.e. no object overlap) over a randomized sample. Regarding fiber segmentation, Jain and Dubuisson [JD92] used a conditional probability model for gray value thresholding based on Besag [Bes86]. The main advantage of a minimal set covering approach is its stability in the case of noisy data. The specific advantage of the iterative RSC approach by Brönnimann/Goodrich [BG95] is its ability to handle large data sizes by restricting itself to a randomized subset in each step. We can retain both of these advantages by combining a stepwise energy minimization over a random sample with the stepwise RSC probability update of Brönnimann/Goodrich.

However while a purely binary model, i.e. no object overlap, is not unfitting in the segmentation of physical objects, we may wish to include prior knowledge more extensively.

For both the generation of test data sets and the statistical estimation of a more refined model, we need to be able to sample from a given Gibbs density. We present two methods of simulation: a Markov Chain Monte Carlo algorithm that is straightforward and fast to implement, but only approx-

imates a given process, and a Dominated Coupling from the Past (DCFTP) algorithm that is able to sample exactly from a given Gibbs density. The drawback of the latter is that it can be extremely slow to converge, making it generally only feasible for single simulations, not statistical averaging over large numbers of samples.

For our MCMC simulations, and most importantly MCMC-based estimation, we use the somewhat straightforward algorithm of [HP99] which is itself based on [Rip77]. For our purposes, Markov chains with a fixed number of points are sufficient. MCMC methods can however be applied in different ways and a more general context – see e.g. [Møl99] for an overview of MCMC methods in point process simulation.

The "Dominated Coupling from the Past" method can be used to draw perfect (non-approximative) samples from a Gibbs density. While it is theoretically possible to replace the MCMC approximations used for Monte Carlo root-finding method (see 5.6.2) with averages over perfect samples for parameter approximation, the DCFTP method is far more computationally expensive in practice, too much so to make the calculation of several thousand samples feasible in a reasonable time-frame. It can however be useful for simulations of lower number, and to estimate the "burn-in" period of MCMC-based approximations, i.e. the number of starting steps a markov chain needs until realizations are close to its stationary target distribution. The DCFTP method used in later sections is based directly on [BM03, KM00], up to some considerations for our somewhat different process model.

The ability to sample directly from the density of a point process is generally not a given, though it is not unique to multiscale processes; as an example, in [Ken98] and [BVL95], Kendall and Baddeley describe methods of perfect simulation and parameter estimation for area-interaction processes that are comparable to the methods described in the later sections 5.5.2 and 5.6.2.

To estimate model parameters from given data, we minimize the quadratic difference of a sufficient statistic of the data to its expected value in the model, using the Levenberg-Marquardt method [Mar63]. Levenberg-Marquardt can be seen as an interpolation between the minimization methods of Gauss-Newton [Bjo96]and gradient descent [Avr03]. In the case of point processes, the necessary derivations can generally not be calculated explicitly, and are instead approximated by averaging over MCMC chains; the resulting method is described as the Monte Carlo Marquardt algorithm and applied to Gibbs processes in [HP99]. We later present a slightly different version that incorporates two-dimensional object interaction.

For an overview of other methods of parameter estimation for pairwise interaction processes see e.g. [DFG+94].

## 1.3 Contributions

(i) We generalize the *Randomized Set Cover* approach and tighten existing bounds for convergence for this generalized approach.

(ii) We introduce a new method of parallelization for this approach, and compare it to an established method of parallelization in both theory and practice. We also briefly discuss a hierarchical approach that achieves a speed up via coarsening of the parameter space.

(iii) We show that an established method of solving large linear programs can be understood as a special case of the randomized set cover approach (up to constants), and bound the dual VC-dimension, a measure of combinatorial complexity, for real half-spaces to this end. We also show that Randomized Set Cover can be understood as a special case of the *Multiplicative Weights* algorithm.

(iv) We modify existing point process models based on pairwise interaction between objects to be based on two measures of object interaction instead of one, and discuss both theoretical and practical pitfalls in applying existing methods of estimation and simulation for these point process models.

(v) We establish two algorithms that combine an energy minimization on an estimated model of object interaction with the Randomized Set Cover to approximate an image segmentation that reflects physical object interaction probabilistically; we again discuss convergence on both a theoretical and practical level.

(vi) We evaluate previously discussed approaches in practice; this includes a run-time and quality comparison on synthetic data, as well as an evaluation on real 2D cell and 3D fiber data.

## 1.4 Organization

This dissertation is divided into three parts:

Part I is concerned with the problem of solving the combinatorial set cover problem optimally and efficiently for large data sets. We give an introduction to the generalized Randomized Set Cover approach and tighten existing bounds for its convergence in chapter 2. We introduce methods of speeding it up up via parallel and hierarchical versions in chapter 3. Anything pertaining to related methods, including the link to linear programs and Multiplicative Weights, is discussed in chapter 4.

In part II the combinatorial approach to segmentation is extended to include a model of physical interaction of objects. We show how a Gibbs point process model can be adapted for cases in which object interaction is multi-dimensional, and discuss methods of simulation and estimation in chapter 5. We discuss the combination of Gibbs model and randomized set cover in theory and practice in chapter 6.

Part III is comprised of everything related to the application of the results of the previous two parts. This includes a numerical evaluation of the examined methods on synthetic simulations in chapter 7, a discussion of local pre- and post-processing methods for real data in chapter 8, as well as as exemplifying segmentations of real data sets in chapter 9. As examples of real data sets we use 2D cellular data provided by, and evalucated in cooperation with, the Biomedical Computer Vision Group (BMCV) Heidelberg in section 9.1 and 3D $\mu$CT fiber data provided by the BASF company in section 9.2.

# Part I

# Randomized Set Cover (RSC)

# Chapter 2

# Background and Convergence

## 2.1 Introduction

High amounts of noise in image data can make any threshold or region-based approach to image segmentation partially or completely unfeasible. To be able to segment images in which voxel data is difficult, but prior knowledge of object shapes is available, we can see image segmentation as a minimal set cover problem: Given a collection of points $X$ and a discrete collection of sets $R \in \mathcal{R}$, we want to find the smallest collection of sets $R_1, \ldots, R_n$ that cover all points, i.e. have a non-empty intersection with every element of $X$.

A minimal set cover problem itself is generally NP-hard [Kar72] and can thus in general only be solved approximately. To this end, many different approaches exist– two of the most notable ones are greedy set covering (see section 4.1) and solving a linear program (see section 4.2). However, both of these approaches tend to scale badly for large problems – a greedy algorithm needs to search through all available sets, or "ranges", several times, while for an LP-based approach even calculating the matrix representation of the cover problem [YNM+15] can be computationally expensive. To solve large set cover problems, we instead base our approach on an algorithm that utilizes small random subsets of the global problem:

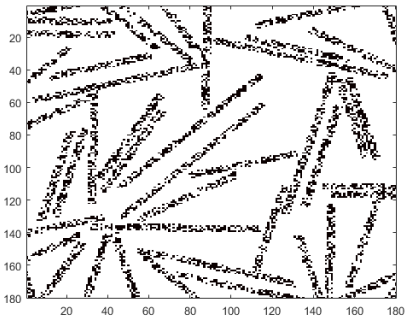
The *Randomized Set Covering* (RSC) algorithm is an iterative reweighting scheme designed to solve a large combinatorial problem in an approximate and randomized way. For its most prominent version, formulated by Brönnimann and Goodrich [BG95], this is a minimal set covering problem. On an intuitive level, the RSC algorithm can be seen as iteratively matching parameterized templates (the ranges) to a set of points: in each step, we sample a collection of ranges according to a probability distribution induced by weights $\omega_i$, then check if they cover all image points. As long as there is a point not covered by the currently sampled ranges, we increase the probability covering it in the next step by doubling the weight of ranges that cover this point. We repeat the process of sampling and reweighting until all image points are covered with high probability – see figure 2.1 – and we eventually sample a full cover. The key to guaranteeing convergence to an approximate solution is that the weight increase does not happen unconditionally, but only for points that are a priori covered with a probability below a set threshold of $\varepsilon$ – we will refer to this as the "$\varepsilon$-condition". See algorithm 1 for an overview of the B/G set covering approach.

**Algorithm 1:** Randomized Set Covering, Brönnimann and Goodrich (B/G)

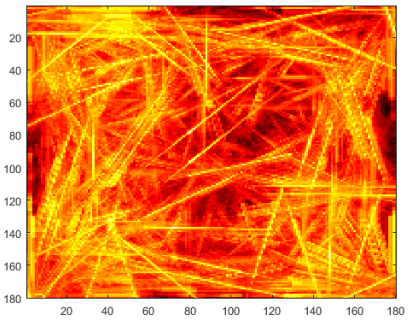
**Input:** $X, \mathcal{R}, \varepsilon, n$.
**Output:** set cover $R_1, \ldots, R_n$ of $X$.
**begin**
  set $\omega(R) = \frac{1}{|\mathcal{R}|},\ \forall R \in \mathcal{R}$ `// normalized starting weights/probabilties`
  sample sets $R_1, \ldots, R_n \sim \mu = \frac{1}{\omega(\mathcal{R})}\omega$ and `// random approximate cover`
  set $\overline{X}_c = X \setminus \{x \colon x \in R_i \text{ for some } R_i\}$ `// points not covered`
  **while** $\overline{X}_c \neq \emptyset$ **do**
    pick any point $x \in \overline{X}_c$
    **if** $\omega(\mathcal{R}_x) < \varepsilon * \omega(\mathcal{R})$ `// ε-condition`
    **then**
      $\omega(R) \longleftarrow 2\omega(R),\ \forall R \colon \{x\} \cap R \neq \emptyset$ `// update ranges covering` $x$
    sample sets $R_1, \ldots, R_n \sim \mu = \frac{1}{\omega(\mathcal{R})}\omega$ and `// repeat sampling`
    set $\overline{X}_c = X \setminus \{x \colon x \in R_i \text{ for some } R_i\}$
  return $R_1, \ldots, R_n$

Mathematically, we find points covered with low probability using $\varepsilon$-nets: a collection of sets that covers every point that is a priori covered with probability $\geq \varepsilon$, thus guaranteeing that uncovered points satisfy the $\varepsilon$-condition. For a given probability distribution and set system, $\varepsilon$-nets can be constructed in several different ways; similarly, the definitions of a set cover and hitting set are closely related, and can be applied in different contexts, such as the solution of a linear program. Consequently, many different variants and special cases of the RSC algorithm exist. In this part of the thesis, we formulate a proof of convergence for a generalized version of the RSC algorithm in section 2.3 that tightens existing bounds after introducing the necessary formal definitions in section 2.2.

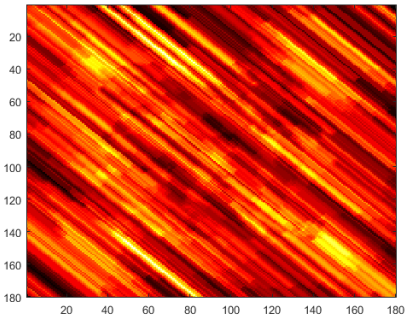
To facilitate fast run times for large problems in later applications, we introduce a method to parallelize the algorithm without loss of quality in section 3.1 and compare it to the current state of the art. We discuss how it can be sped up by coarsening the parameter space in section 3.2. In chapter 4 we discuss the relation of the RSC algorithm to several other iterative randomized methods; most importantly, in section 4.4 we show how an algorithm introduced for the solving of linear programs can be viewed as a special case of the RSC algorithm up to parameter values, leading to tightened bounds for convergence and the applicability of the previously introduced method of parallelization. Some sections in this part of the thesis, in particular those related to the proof of convergence and parallelization, bear strong similarities to an as of yet unpublished paper by Markowsky et al.[MZ17].



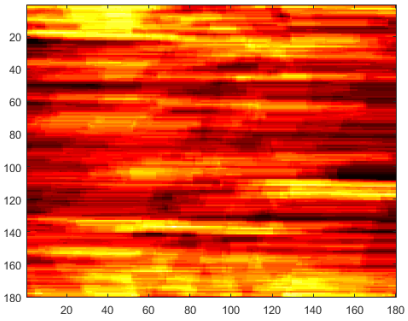

(a) Simulated data



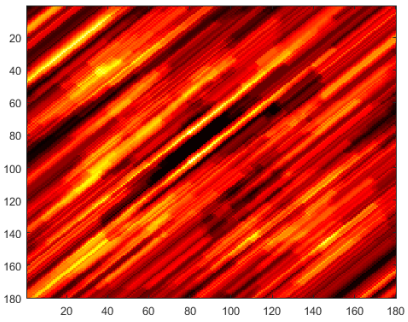

(b) Probabilities of object centers, summed over all orientations



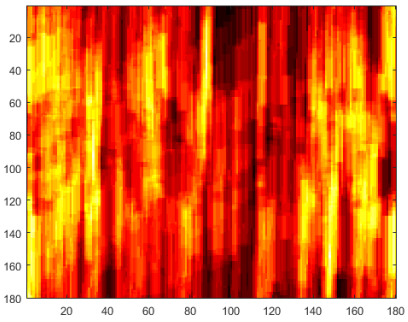

(c) Orientation 1




(d) Orientation 2




(e) Orientation 3




(f) Orientation 4

Figure 2.1: Simulated 2D example, probabilities of object centers assigned by the RSC algorithm at convergence (logarithmically scaled).

## 2.2 Basic Definitions and Preliminaries

This section introduces basic definitions and results used in the subsequent sections. Basic references include [HP11, Mat02] to which we refer for more background and details. Throughout this chapter, we only consider *finite* sets.

**Definition 2.2.1** (Range Space, Hitting Set, Set Cover)**.** Let $X$ be an arbitrary finite set, and let $\mathcal{R}$ be a family of subsets of $X$. Then the pair $(X, \mathcal{R})$ is called a *range space* with *points* $x \in X$ and *ranges* $R \in \mathcal{R}$.

A set of elements $N \subseteq X$ is called a *hitting set* for $(X, \mathcal{R})$ if any range contains at least one point of $N$, i.e. $\forall R \in \mathcal{R}\, \exists\, x \in N \colon x \in R$.

A set of ranges $\mathcal{C} \subseteq \mathcal{R}$ is called a *set cover* for $(X, \mathcal{R})$ if any point of $X$ is contained in at least one range in $\mathcal{C}$, i.e. $\forall x \in X\, \exists\, R \in \mathcal{C} \colon x \in R$.

**Definition 2.2.2** (Dual Range Space)**.** Given a range space $\mathsf{S} = (X, \mathcal{R})$ and a point $x \in X$, we define the *dual range* of $x$ as the set of all ranges containing $x$,

$$\mathcal{R}_x := \{R \in \mathcal{R} \colon x \in R\}\,. \tag{2.2.1}$$

The *dual range space* corresponding to $\mathsf{S}$ is

$$\mathsf{S}^* := (\mathcal{R}, \mathcal{R}_X) \quad \text{with} \quad \mathcal{R}_X := \{\mathcal{R}_x \colon x \in X\}. \tag{2.2.2}$$

We can see immediately that for any point $x$ and range $R$, $x$ is contained in $R$ *iff* $R$ is contained in the dual range of $x$:

$$x \in R \iff R \in \mathcal{R}_X. \tag{2.2.3}$$

As a direct consequence, the a set cover of $\mathsf{S} = (X, \mathcal{R})$ is a hitting set of its dual range space $\mathsf{S}^*$, and the dual range space of the dual range space $\mathsf{S}^*$ is equivalent to $\mathsf{S}$ itself.

**Definition 2.2.3** ($\varepsilon$-net)**.** Let $\mathsf{S} = (X, \mathcal{R})$ be a range space and $A \subseteq X$.

(i) a set $N \subseteq X$ is called an *$\varepsilon$-net* for $\mathsf{S}$ if, for any range $R \in \mathcal{R}$ with

$$|R \cap A| \geq \varepsilon |A|, \tag{2.2.4}$$

$R$ contains at least one point of $N$, i.e. $R \cap N \neq \emptyset$.

(ii) For a given probability measure $\mu$ on $X$, condition (2.2.4) reads

$$\mu(R \cap A) \geq \varepsilon \mu(A). \tag{2.2.5}$$

Conditions (2.2.4) and (2.2.5) are equivalent if $\mu$ is the uniform measure. Because any common factor does not change the relation (2.2.5) any measure can be used as well. We denote such weights by

$$\omega(R \cap A) \geq \varepsilon \omega(A). \tag{2.2.6}$$

(iii) If the subset $A \subseteq X$ is not specified in connection with an $\varepsilon$-net $N$, then it is tacitly assumed that $A = X$. In this case condition (2.2.5) reads

$$\mu(R) \geq \varepsilon, \tag{2.2.7}$$

i.e. $N$ must intersect any range $R$ with a probability measure not less than $\varepsilon$.

**Definition 2.2.4** (Shattering,VC-Dimension)**.** Let $\mathsf{S} = (X, \mathcal{R})$ be a range space. For any $Y \subset X$ the family of subsets

$$\mathcal{R}_{|Y} := \{R \cap Y \colon R \in \mathcal{R}\} \tag{2.2.8}$$

is called the *projection* of $R$ onto $Y$. If $\mathcal{R}_{|Y}$ contains all subsets of $Y$, i.e. $\mathcal{R}_{|Y} = 2^Y$, then $Y$ is *shattered* by $\mathcal{R}$. The *Vapnik-Chervonenkis dimension* $\dim_{VC}(\mathsf{S})$ is the largest cardinality of subsets $Y$ that can be shattered by $\mathcal{R}$. If subsets of arbitrary size can be shattered, then $\dim_{\mathrm{VC}}(\mathsf{S}) := \infty$.

**Theorem 2.2.1** (Size Upper Bound for $\varepsilon$-nets [KPW92, Thm. 3.1, Thm. 3.2])**.** *Let $\mathsf{S} = (X, \mathcal{R})$ be a range space with $\dim_{VC}(\mathsf{S}) = d$ and let $\mu$ be a probability measure defined on $X$ such that all ranges are $\mu$-measurable. If $\varepsilon > 0$ is chosen sufficiently small depending on $d$ such that $\exists N \in \mathbb{N}$ with*

$$p := 2\sum_{i=0}^{d} \binom{N}{i} \left(1 - \frac{n}{N}\right)^{(N-n)\varepsilon - 1} < 1, \tag{2.2.9}$$

*then any i.i.d. random sample $X^{(n)} := \{x^1, \ldots, x^n\} \subset X$ with $x^i \sim \mu$ of size*

$$n = \frac{d}{\varepsilon}\big[\log(1/\varepsilon) + 2\log\log(1/\varepsilon) + 3\big] \tag{2.2.10}$$

*is an $\varepsilon$-net for the measure $\mu$ with probability $\geq 1 - p$.*

## 2.3 Generalized Iterative Reweighting with Tight Bounds

Since a set cover can be seen as a hitting set of the dual of a range space, and vice versa, the RSC algorithm can be formulated equivalently as a minimal hitting set algorithm (see 2.2). This formulation makes some proofs a little more intuitive, and we will be using it for the more theoretical sections 2.3 and 3. Although different versions of $\varepsilon$-net based reweighting schemes can be constructed, they share the same key elements that are necessary to guarantee convergence. We first formulate a reweighting scheme that generalizes both the algorithms formulated by Brönnimann and Goodrich (B/G) [BG95] and a sequential version of the simplified scheme proposed by Agarwal and Pan (A/P) [AP14]:

**Algorithm 2:** Sequential weight update scheme

**Input:** $(X, \mathcal{R}), \varepsilon, \omega$.
**begin**
  **while** *Convergence is not achieved* **do**
    pick a range $R \in \mathcal{R}$
    **if** $\omega(R) < \varepsilon\omega(X)$ **then**
      $\omega(x) \longleftarrow 2\omega(x),\ \forall x \in \mathcal{R}$
  return $\omega$

Here, "pick a range" has a slightly different meaning depending on the reweighting scheme used: For approaches based on the B/G scheme, picking a range means constructing an $\varepsilon$-net in each step and picking a range that is not hit by this $\varepsilon$-net. Agarwal et al.[AP14] originally check several ranges $R$ consecutively, and update all that satisfy $\omega(R) < \varepsilon\omega(X)$. In this sequential scheme, we first only allow the update of one range per step. Similarly, "convergence is not achieved" has different meanings depending on the different versions of the reweighting scheme: For versions based on the B/G scheme the condition for termination is that the $\varepsilon$-net constructed in a step is a hitting set for the original problem. The original approach by Agarwal et al.[AP14] stops when less than $\lfloor \frac{1}{\varepsilon} \rfloor$ $\varepsilon$-light ranges are found after checking every range $R \in \mathcal{R}$. Our sequential version simply stops when no more $\varepsilon$-light ranges can be found.

In both cases, the key for guaranteeing convergence to a minimal hitting set is limiting the weight-doubling to sets $R$ that satisfy

$$\omega(R) < \varepsilon\omega(X). \tag{2.3.1}$$

Although the idea of the proof of convergence closely follows the general multiplicative weight update scheme [AHK12], and has been formulated many times (see e.g.[BG95],[HP11]), we include a modified version to establish tight bounds for $\varepsilon$:

**Proposition 2.3.1.** *If a hitting set of size $n^*$ exists, and*

$$\varepsilon < \varepsilon_{\max}(n^*) := 2^{\frac{1}{n^*}} - 1, \tag{2.3.2}$$

*then a range satisfying condition 2.3.1 can only be found for a finite number of steps. This bound on $\varepsilon$ cannot be improved in the general case.*

*Proof.* Assume $\varepsilon = 2^{\frac{1}{an^*}} - 1$ for some $a > 1$.

If an $\varepsilon$-light range, i.e. one satisfying condition 2.3.1, can be found in every step, and its weight is doubled, then the total weight of $\mathcal{R}$ after $i$ steps is bounded by

$$\omega^i(X) \leq (1+\varepsilon)\omega^{i-1}(X) \leq (1+\varepsilon)^i\omega^0(X) = 2^{\frac{i}{an^*}}\omega^0(X), \tag{2.3.3}$$

On the other hand, suppose $H$ is a hitting set of size $n^*$. Let $t_i(j)$ denote the number of times the weight of the $j$th element of $H$ was doubled after $i$ iterations. By the definition of a hitting set, at least one point is in each of the ranges which are doubled in weight, i.e. $\sum_{j=1}^{n^*} t_i(j) \geq i$.

Defining $\omega^i$ as the weight function after $i$ steps, and $\omega^0_{\min} = \min_{h\in H} \omega^0(h)$, we can lower bound the weight of $H$:

$$\omega^i(H) \geq \sum_{j=1}^{n^*} 2^{t_i(j)}\omega^0_{\min} \geq n^* 2^{\sum_{j=1}^{n^*} t_i(j)/n^*}\omega^0_{\min} \geq n^*\omega^0_{\min}2^{i/n^*}. \tag{2.3.4}$$

The second inequality is a direct consequence of the geometric mean lower-bounding the arithmetic mean. Comparing equations 2.3.3 and 2.3.4, this means that since $a > 1$, the weight of a subset of $X$ grows faster in $i$ than its total weight by a factor of $\mathcal{O}(2^i)$, leading to a contradiction for a finite value of $i$, and thus bounding the number of steps. See appendix section A.1 for the detailed bounds of the number of steps under the assumption that condition 2.3.1 is satisfied with a positive probability $q$ or

higher in all steps.

To prove general tightness, let $R_1, \ldots, R_{n^*}$ be a set of ranges that each cover exactly one element of an optimal hitting set $H$, and choose uniform starting weights $\omega(R) = \omega^0_{\min} = c$. Consecutively choosing these ranges for weight-doubling, not doubling the weight of one twice before the weight of all others has been doubled, will make the bound in equation 2.3.4 tight for step numbers that are an integral mutiple of $|H| = n^*$: After $tn^*$ steps, with $t \in \mathbb{N}$, every point $h$ in $H$ will have been selected for weight-doubling exactly $t$ times, and have the weight $2^t\omega^0(h) = 2^{i/n^*}\omega^0_{\min}$. This scenario generally occurs with a positive probability. Unless we can reliably bound the relative weight added in each step lower than $\varepsilon$, or can exclude the aforementioned scenario of iterative weight-doubling a priori, the bound of $\varepsilon_{\max}$ as defined in equation 2.3.2 is thus the best we can establish for the general case. □

It should be noted that in our experience the bounds on the cardinality of randomly sampled $\varepsilon$-nets (see theorem 2.2.1), as originally used by Brönnimann and Goodrich [BG95], will be far from tight in practice. However convergence is mathematically guaranteed by fulfilling the condition 2.3.1; For the B/G scheme, sampling a full $\varepsilon$-net is thus sufficient, but not necessary to guarantee convergence: Since the condition 2.3.1 is explicitly checked in every step, the theoretical bounds on sample sizes can be ignored without risking guaranteed convergence as long as the condition is still empirically satisfied in a sufficient percentage of steps.

If we can establish a bound on the probability of condition 2.3.1 being satisfied, e.g. using theorem 2.2.1, we can also bound the expected number of overall steps of the algorithm. We formulate this in theorem A.1.1 of the appendix.

# Chapter 3

# Parallel and Hierarchical RSC

## 3.1 Independent Weight Updates for a Parallelization with Tight Bounds

For any reweighting scheme in the form of algorithm 2, especially for B/G-type reweighting schemes which construct an $\varepsilon$-net in each step, an obvious way to save calculating time is to parallelize the weight updates, i.e. update the weight of multiple ranges in each step.

However, when looking at the proof of convergence formulated in proposition 2.3.1, specifically the context of equation 2.3.3, we can see that simply allowing the (successive) weight-doubling for all ranges that are $\varepsilon$-light in the current iteration will change the upper estimate for the added weight in one step from

$$\omega^{i+1}(X) = (1+\varepsilon)\,\omega^{i}(X)$$

to

$$\omega^{i+1}(X) = (1+2^{n_i}\varepsilon)\;\omega^{i}(X),$$

assuming $n_i$ is the number of ranges selected for weight-doubling: potentially the ranges successively doubled in weight can be almost identical; this sharply worsens the bound for the total weight of the range space, and thus makes a much smaller value of $\varepsilon$, and thus a much bigger sample size, a requirement.

Our approach towards circumventing this problem is to allow the successive doubling of the weights for any collection of ranges *that do not intersect in* $X$, i.e.

$$\{R_j \cap R_k\} = \emptyset \;\forall R_j, R_k \tag{3.1.1}$$

If all ranges $R_i$ are $\varepsilon$-light, they will satisfy the condition $\sum_i \omega(R_i) < \varepsilon M \omega(X)$. On the other hand, we can tightly bound total weight growth by using condition 3.1.1, and prove convergence of the parallelized algorithm for the same value of $\varepsilon$:

**Proposition 3.1.1.** *If a hitting set of size $n^*$ exists, and*

$$\varepsilon < \varepsilon_{\max}(n^*) = 2^{\frac{1}{n^*}} - 1, \tag{3.1.2}$$

*then for the parallelized algorithm 3 a range satisfying condition 2.3.1 can only be found for a finite number reweighting steps. The number of reweighted ranges is limited by the same bound as the number of steps of the non-parallelized algorithm 2.*

**Algorithm 3:** Independently Parallelized Reweighting

**Input:** $(X, \mathcal{R}), \varepsilon, \omega$.
**begin**
  **while** *Convergence is not achieved* **do**
    pick a set of ranges $\mathcal{R}_i, i = 1, \ldots, M \in \mathcal{R}$ such that $\{\mathcal{R}_j \cap \mathcal{R}_k\} \cap X = \emptyset \; \forall \, \mathcal{R}_j, \mathcal{R}_k$
    **if** $\sum_i \omega(R_i) < \varepsilon M \omega(X)$ **then**
      $\forall i \colon \omega(x) \longleftarrow 2\omega(x), \; \forall x \in R_i$
  return $\omega$

*Proof.* Let $i$ denote the total number of ranges $R$ selected for weight-doubling in all steps up to a certain point in the algorithm. Looking back at the proof of proposition 2.3.1, the original upper estimate

$$\omega^i(X) \leq \omega^0(X) \, (1 + \varepsilon)^i$$

will then still hold true: Let $l$ be the numbers of steps taken, $n_l$ is the number of ranges doubled in weight in step $l$. Condition 3.1.1 ensures that if all ranges $R_j$ are $\varepsilon$-light, no point $x \in \cup_j R_j$ has its weight doubled more than once in each step, and the total weight is not increased by a factor of more than $(1 + n_l \varepsilon)$. Thus after $l$ steps, and $i = \sum_l n_l$ updates in total,

$$\omega^i(X) \leq \omega^0(X) \prod_l (1 + n_l \varepsilon) \leq \omega^0(X) \, (1 + \varepsilon)^{\sum_l n_l} = \omega^0(X) \, (1 + \varepsilon)^i \, . \tag{3.1.3}$$

Similarly looking back at equation 2.3.4, after $i$ weight-doubling steps, the total weight of a minimal hitting set $H$ will still at least be

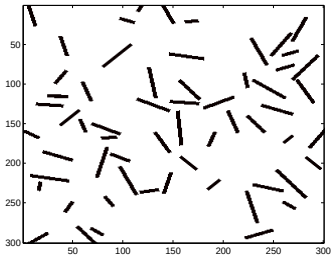
$$\sum_{j=1}^{n^*} 2^{t_i(j)} \omega^0_{\min}$$

since the weight of at least one element is still doubled for every weight-doubling step.

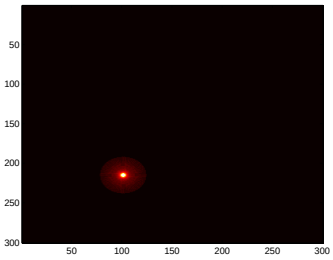
Since both upper and lower bound remain identical, with the exception of $i$ not being the number of steps, but the number of weight updates, convergence will still be guaranteed for the same value of $\varepsilon$, with the same limit[1] on $i$. □

Updating several weights instead of one per step (i.e. per constructed $\varepsilon$-net) can obviously provide a significant speed-up. See figure 3.1 for a graphical example and subsection 3.1.2 for a theoretical bound on running time.

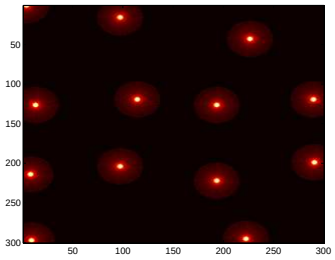
[1] See appendix section A.1 for the detailed bounds on the number of steps of the sequential reweighting scheme 2 under the assumption that condition 2.3.1 is satisfied with a positive probability $q$ or higher in all weight updates.



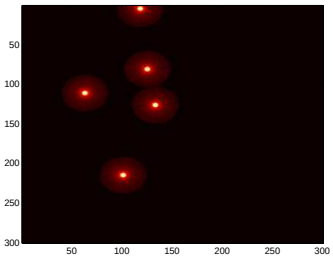

Original image



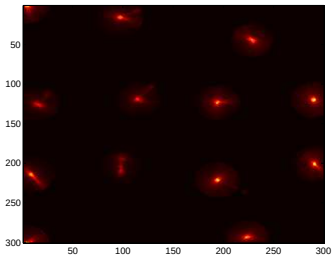

Singular updates, step 1



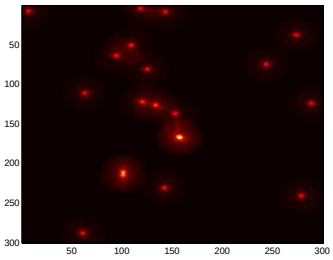

Multiple updates, step 1



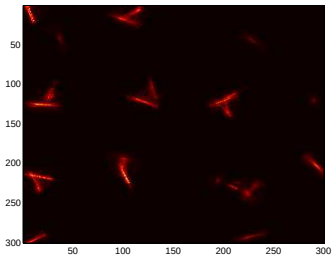

Singular updates, step 5




Multiple updates, step 5




Singular updates, step 20




Multiple upd., step 20

Figure 3.1: Probabilites of choosing a rectangle for the set cover dependent on its middle point. After 20 steps, the rectangular structures begin visibly emerging in the case of multiple weight updates per step, but not for singular ones.

### 3.1.1 Comparing independently parallelized RSC to Agarwal/Pan

The reweighting scheme described by Agarwal et al. [AP14] as a simpler, but more efficient version of the B/G RSC scheme (algorithm 1) follows a similar pattern, with two major differences:

- it does not use $\varepsilon$-nets to find $\varepsilon$-light ranges in each step. One $\varepsilon$-net is constructed when the reweighting terminates, i.e. a net finder is called once instead of multiple times.
- Several ranges are updated in one step, and multiple times, i.e. a parallelization without the constraint of non-intersection.

The obvious advantage of the first alteration by A/P is the decrease in running time: an $\varepsilon$-net needs to be constructed only once for the algorithm, instead of once for each step. The mathematical downside to this can be summarized as: A range that is $\varepsilon$-heavy *before* doubling the weight of one or several other ranges generally is not afterwards. Specifically, after doubling the weight of $n_i$ ranges that are all $\varepsilon$-light, the total weight of the space $X$ is bounded by

$$\omega^{i+1}(X) \leq (1+\varepsilon)^{n_i}\omega^i(X), \tag{3.1.4}$$

and a range that was $\varepsilon$-heavy before reweighting is only guaranteed to have a relative weight of $\varepsilon_2 = \frac{\varepsilon}{(1+\varepsilon)^{n_i}}$ or more afterwards. A/P combats this problem by limiting the number of reweightings to $n_i \leq \lfloor \frac{1}{\varepsilon} \rfloor$, establishing a bound of $\frac{\varepsilon}{\mathrm{e}}$ on $\varepsilon_2$. [2]

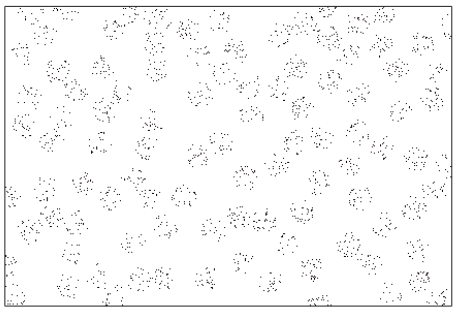
The proof of convergence formulated in proposition 2.3.1 remains true whether $\varepsilon$-nets are used in each step or not, meaning that reweighting can be performed with any $\varepsilon < \varepsilon_{\max}(n^*) = 2^{\frac{1}{n^*}} - 1$ for both schemes.

The contribution of choosing parallel weight updates independently, i.e. of non-intersecting ranges, is limiting the additional weight to a linear instead of exponential factor, namely $(1 + n_i\varepsilon)$. This in turn allows for a proof of convergence without changing the required value of $\varepsilon$, since the weight of an optimal subset is equally lower-bounded linearly. When parallelizing the B/G scheme, this means that the same value of $\varepsilon < \varepsilon_{\max}$ can still be used when constructing $\varepsilon$-nets as possible solutions. Following the A/P scheme, the bounds for $\varepsilon_2$ are still worsened, but linearly – when doubling the weight of no more than $\lfloor \frac{1}{\varepsilon} \rfloor$ ranges, as in the original paper, by a factor of $\frac{1}{2}$ instead of $\frac{1}{\mathrm{e}}$ for dependent updates.

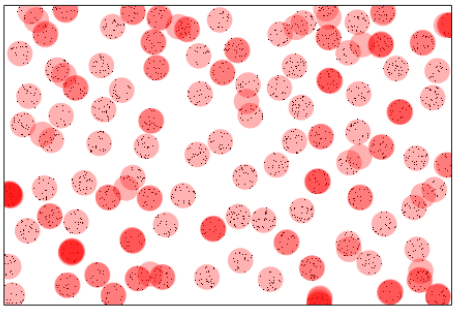
In total, both the decision whether to use $\varepsilon$-nets in every step or once after reweighting, and whether to parallelize independently or not, are a trade of running time versus the cardinality of the resulting approximation. While the latter options will almost universally lead to faster times per step, they both decrease the required value of $\varepsilon$, and thus increase the required cardinality of the $\varepsilon$-net, by a significant factor. In applications a good compromise may be found by first reweighting using the generally faster A/P scheme, and using the resulting weight function as input for an independently parallelized B/G scheme for optimal quality.

[2] In the original paper [AP14], the maximal number of reweights is $2k$ for a value of $\varepsilon = \frac{1}{2k}$. For a general $\varepsilon$, the value of $\lfloor \frac{1}{\varepsilon} \rfloor$ preserves the same bounds on $\varepsilon_2$.

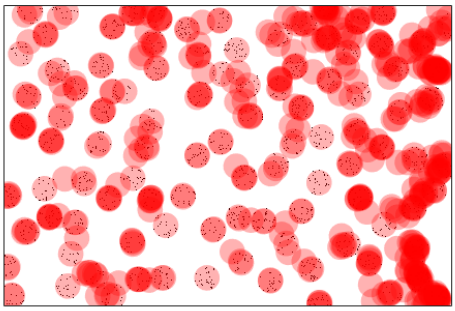
**Example**



Set of points $X$, minimal set cover cardinality $n^* = 100$



Independently parallelized B/G scheme $\varepsilon_1 = 2^{\frac{1}{n^*+1}} - 1$, cover cardinality 175



A/P scheme (dependent parallelization) $\varepsilon_2 = \frac{\varepsilon_1}{\mathrm{e}}$, cover cardinality 501

Figure 3.2: Graphical example of the results of our proposed parallelization of B/G vs. the A/P scheme, applied to a set covering problem. Darker areas indicate a cover by multiple circles. The lower value of $\varepsilon_2 \ll \varepsilon_1$ required for convergence of the A/P scheme leads to siginificantly larger cover cardinalities in theory and practice.

See figure 3.2 for a graphical example of the results of our proposed parallelization applied to the B/G scheme vs. the parallelized scheme proposed by Agarwal/Pan, applied to a set cover problem with circles. Here, $X \subset \mathbb{R}^2$ is the set of points depicted, $\mathcal{R}$ is the intersection of $X$ with all circles of a fixed radius, centered on the pixel grid. Since we wish to solve a set cover problem, we apply the hitting set algorithm 3 to the dual range space $\mathsf{S}^*$ of $\mathsf{S} = (X, \mathcal{R})$. Non-overlapping dual ranges are found via the distance of their center points.

$\varepsilon$-nets were constructed via random sampling for both algorithms. The sample sizes $n_i$ were chosen empirically: In the case of the B/G scheme that samples $\varepsilon$-nets repeatedly, a value $n_1 = 330$ was chosen that allows finding $\varepsilon$-light ranges and reweighting with high probability, empirically in 4394 out of 4546 steps. For the A/P scheme, an $\varepsilon_2$-net needs to be sampled only once, allowing for

more repetitions, and a relatively lower cardinality for the constructed $\varepsilon_2$-net. The Sampling size $n_2$ was chosen as a small value that empirically allows finding an $\varepsilon_2$-net via random sampling consistently in less than $10000$ attempts. In the depicted examples, a set of $n_2 = 660$ ranges had to be sampled $58$ times before finding the first $\varepsilon_2$-net. Double sampling leads cover sizes smaller than $n_i$ in the approximative solutions produced by both algorithms.

In this example, the sample sizes $n_i$ are somewhat arbitrarily chosen input parameters of the algorithms, and the specific number of attempts required to sample an $\varepsilon$-net randomly can fluctuate highly. However, the weight function that is produced upon termination of the reweighting algorithms tends to be much more consistent, and is a direct result of the algorithms' structure and the theoretical bounds on $\varepsilon_i$. See figure 3.3 for a depiction of the weights from which the set covers in figure 3.2 were sampled. Since we solve a set cover problem as a hitting set problem on the dual range space, weights are assigned to the cyclic ranges that make up its elements. They are normalized to a maximum of 1, and depicted as a function of their centers. As we would expect from the higher value of $\varepsilon_1$ compared to $\varepsilon_2$, the independently parallelized version of B/G shows a significantly sharper convergence to optimal ranges than the dependently parallelized A/P scheme.

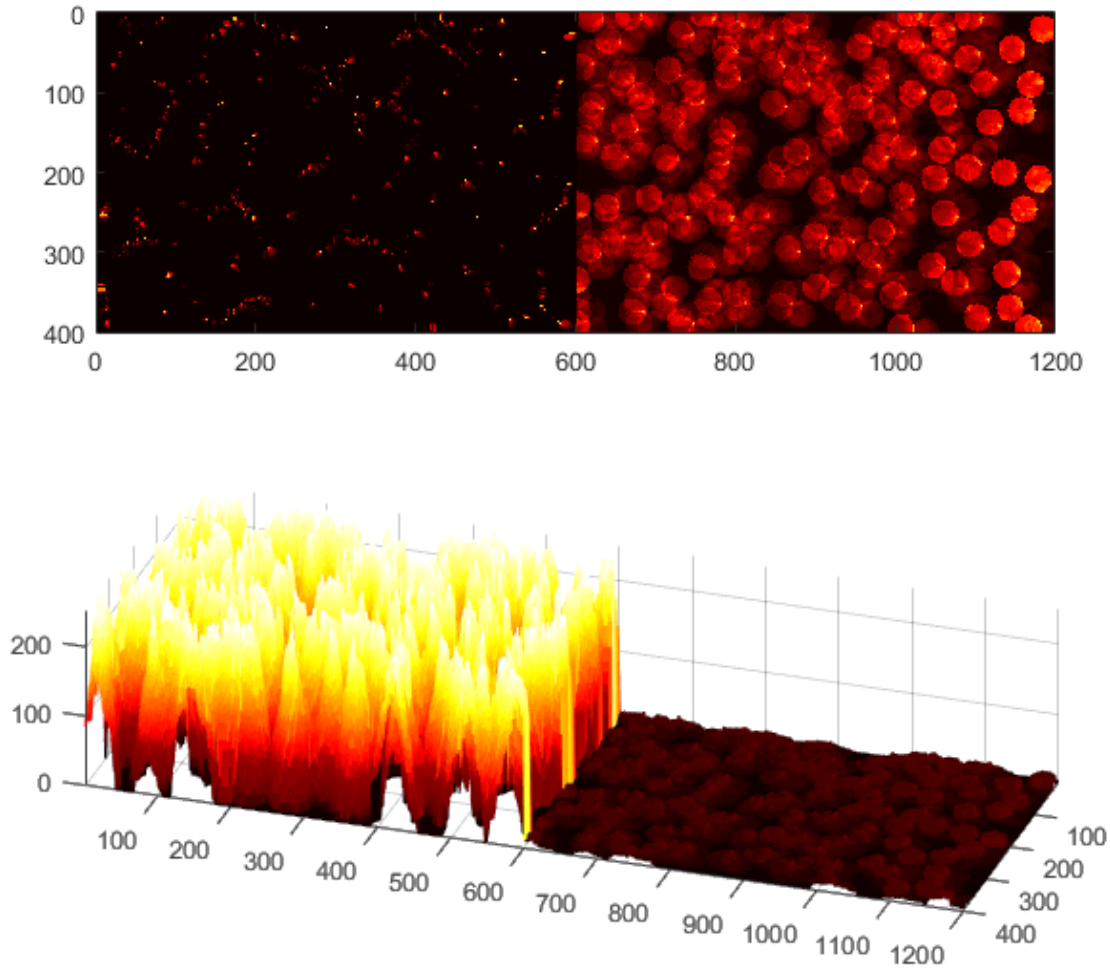



Figure 3.3: Weights produced by the independently parallelized B/G (left) and A/P (right) algorithms upon termination. Scaled logarithmically (top)/with an $8^{th}$ root (bottom). Convergence of weights to optimal values is far sharper for the independently parallelized B/G scheme.

### 3.1.2 Quality and Running Time Analysis

The quality, i.e. cardinality, of an $\varepsilon$-net based solution to a minimal hitting set problem depends on the value of $\varepsilon$ used in the construction of the net: In general it is $\mathcal{O}(\frac{1}{\varepsilon}\log(\frac{1}{\varepsilon}))$, except in some special geometric cases that allows $\varepsilon$-nets of size $\mathcal{O}(\frac{1}{\varepsilon})$, see [MV17]. Any improvement to the value of $\varepsilon$ will thus directly improve the quality of a hitting set by an equal or larger factor.

The bound of $\varepsilon_{\max}$ established in propositions 2.3.1 and 3.1.1 asymptotically approaches $\frac{\ln(2)}{n^*}$ – see section A.2 of the appendix. It applies to both any sequential reweighting, and an independently parallelized B/G scheme. To our knowledge, the currently best bound of $\varepsilon$ for a parallelization was established by Agarwal et al. [AP14] as $\frac{1}{2n^*e}$. [3] The new bound $\varepsilon_{\max}$ thus constitutes an improvement of roughly a factor of $2\mathrm{e}\ln 2 \approx 3.77$ or better in the quality of the solution of a parallelized approach. Using proposition 2.3.1 and reflections from the previous section 3.1.1, we can also establish a new bound of $\varepsilon_2 = \frac{\varepsilon_{\max}(n^*)}{\mathrm{e}}$ for the original A/P scheme, improving its original bounds by a factor of roughly $2\ln 2 \approx 1.39$.

For an independently parallelized approach, the increase in running time compared to the original A/P scheme is caused by the need to find a preferably large number of non-overlapping ranges. While in practice different approaches may be useful depending on the problem structure (see the end of the subsection), we first give a general approach in order to bound running times: Given a set of $\varepsilon$-light ranges found by either the B/G or A/P scheme, we construct a graph in which vertices represent the ranges of this subset, while edges represent mutual overlap. Solving the problem of finding a maximal independent subset of this graph – that is, a subset of vertices that are not connected by any edge – is then equivalent to finding maximal set of $\varepsilon$-light ranges that do not overlap. If $L$ denotes the number of vertices of the graph, several established algorithms solve the maximal independent set problem, such as Luby's [Lub86] and Blelloch's [BFS12] algorithm, with running times of $\mathcal{O}(\log(L))$ and $\mathcal{O}(\log^2(L))$, respectively. At the same time, if $\frac{Ld}{2}$ is the number of edges of the graph, Turan's theorem [NS08] establishes a lower bound of $\frac{L}{2d}$ for the cardinality of this set, i.e. the number of non-intersecting ranges the algorithms will find.

Parallelization can thus sharply decrease running times, especially for the B/G scheme. In the vocabulary of the original paper [BG95], assume we know a "finder" that constructs an $\varepsilon$-net in time $f(\varepsilon)$, and a "verifier" that supplies $\varepsilon$-light ranges that are not hit by the net in time $g_1(\varepsilon)$. By theorem A.1.1, the maximal number of weight-doubling steps $N$ is bound, with a finite expected value $\mathbb{E}[N]$. Let $\varepsilon_1 < \varepsilon_{\max}$. Then the expected running time for finder and verifier of the non-parallelized B/G scheme is

$$[f(\varepsilon_1, \cdot) + g_1(\varepsilon_1, \cdot)]\,\mathbb{E}[N]. \tag{3.1.5}$$

For our proposed independently parallelized version, assume that on average we can use the verifier to find $L$ ranges, of which no more than $\frac{Ld}{2}$ pairs overlap. We can find at least $\frac{L}{2d}$ non-overlapping ranges in time $\mathcal{O}(\log(L))$ using Luby's algorithm. The verifier thus needs to be executed $2d$ more times, but we can simultaneously perform at least $\frac{L}{2d}$ update steps after using the net-finder once, in

[3] Specifically, [AP14] defines $\varepsilon = \frac{1}{2ke}$, where $k/2 < n^* \leq k$. This value is largest (optimal) for $k = n^*$.

total establishing a running time of

$$2d \left[ \frac{f(\varepsilon_1, \cdot) + \mathcal{O}(\log(L))}{L} + g_1(\varepsilon_1, \cdot) \right] \mathbb{E}[N] \tag{3.1.6}$$

for an independently parallelized B/G scheme. Note that $f(\varepsilon, \cdot)$ varies depending on the exact method of $\varepsilon$-net construction used, but will in general be much larger than the cost $g_1(\varepsilon, \cdot)$ of afterwards finding an $\varepsilon$-light range.

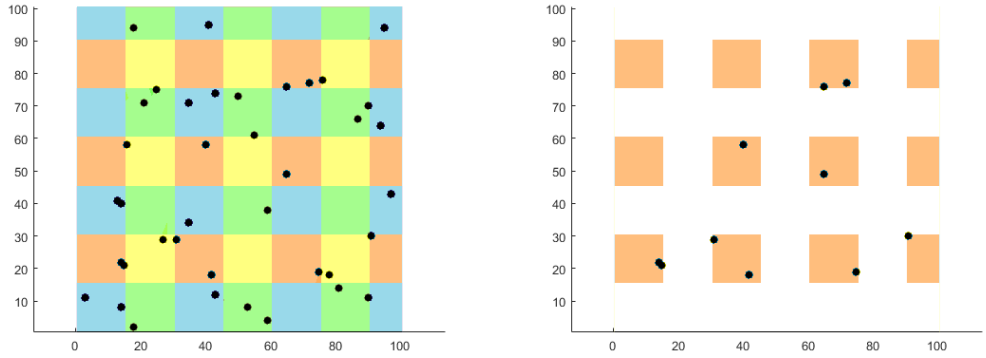
For the (parallel) A/P scheme, the number of reweightings is bound by the same value, but a $\varepsilon$-net is only constructed once, for $\varepsilon = \varepsilon_2$. In addition, the verifier checks whether a range is $\varepsilon$-light sequentially instead of using an $\varepsilon$-net, leading to different running times. We will denote this verifier by $g_2(\varepsilon)$. Given a range space $(X, \mathcal{R})$, Agarwal/Pan establish its running time per range as $\mathcal{O}(\log(|X|))$, and multiply this by a factor of $\mathcal{O}(|X| + \mathcal{R})$ to establish the running time of checking *all* ranges in $\mathcal{R}$. If we assume similarly that the number of ranges that need to be checked on average before finding the first $\varepsilon$-light range is $\mathcal{O}(|X| + |\mathcal{R}|)$, this establishes a total running time of

$$f(\varepsilon_2, \cdot) + g_2(\varepsilon_1, \cdot)\, \mathbb{E}[N] = f(\varepsilon_2, \cdot) + [\mathcal{O}((|X| + |\mathcal{R}|) \log |X|)]\, \mathbb{E}[N] \tag{3.1.7}$$

for finder and verifier. Note that while the value $\varepsilon_2$ used by the finder is different from the one used by B/G, the one used by the verifier $g_2$ during reweighting is not. We omit discussion of the time taken for steps that are the same for all reweighting schemes described

In a lot of applications we know that $X \subset \mathbb{R}^d$, ranges are characterized by a central point, i.e. for circle ranges or a dual range $R^x$, and the maximal distance of any point in the range from this central point is bounded. In such situations we can efficiently find a large number of non-intersecting $\varepsilon$-light (dual) ranges for independent parallelization ranges with minimal cost in run time:

Given a set of ranges, we can lay a grid of a length that is larger or equal to the maximum (generalized) diameter of a range over their central points – see figure 3.4. Now, choose one type of grid-sections (e.g. the light blue ones) in such a way that there is at least one point in these grid-sections and pick one central point in each of these (light blue) grid-sections. Since this ensures that the distance of all central points selected is greater than the diameter of the ranges, we know that no two ranges can intersect. While this method does not necessarily find a *maximal* independent subset of ranges, its running times will generally be significantly faster than any method which calculates the maximal independent subset precisely.




Points partitioned

Points in each grid separated by minimal distance

Figure 3.4: Separating ranges using a grid

## 3.2 Hierarchical Weight Updates

Though randomized approaches are generally the most apt to handle large data, the RSC algorithm still needs to assign and update probabilities for all possible cover candidates $R_i$. If higher accuracy is demanded in the result, the space of possible ranges can be extensively large.

To improve tractability of larger problems, one intuitive approach is to first solve the cover problem for a "rougher", smaller range space $\mathsf{S}_- = (X, \mathcal{R}_-)$. For the purpose of theory, we will assume that the ranges $R_{i_-} \in \mathcal{R}_-$ are a union of a number of the original ranges $R_i \in \mathcal{R}$.

The minimum cardinality $n$ of the input of the set cover algorithm 1 that guarantees convergences is

$$n = \frac{d}{\varepsilon}\big[\log(1/\varepsilon) + 2\log\log(1/\varepsilon) + 3\big] \tag{3.2.1}$$

$$\varepsilon < 2^{\frac{1}{n^*}} - 1 \tag{3.2.2}$$

where $d$ is the VC-dimension of the range space and $n^*$ is the smallest possible size of a cover, as stated in equations 2.2.10 and 2.3.2.

If a cover of size $n^*$ exists for the original range space, we will obviously also be able to find a cover of the same size for a modified range space that consists of unions of the original ranges; indeed, this will be true for any space of supersets, not just unions. The VC-dimension $d_-$ of the modified range $\mathsf{S}_-$ space on the other hand is not easily bounded based on the VC-dimension $d$ of the original range space $\mathsf{S}$, and can in fact be largely depended on the specific way original ranges are united and the shape of this union. In applications, the easiest way to circumvent this problem is to choose unions of ranges in such a way that simple geometric forms with a known VC-dimension are produced; in particular, we can first disregard orientation by uniting all cylinders with a particular point of origin to a ball (figure 3.6), or unite all cylinders of a certain orientation with points of origin in a

cylinder of similar orientation to get one larger cylinder (figure 3.5). In terms of computational feasibility, uniting several candidates allows us to reduce parameter space size via a rougher discretization.

If we do find an approximate cover using these larger candidates, we can run a second RSC algorithm in which we assign non-zero starting probabilities only to parameters which were combined into the candidates of the initial cover. In total, this replaces one RSC run on a large parameter space with two consecutive solutions on smaller subsets. Comparable approaches exist for the Hough transform which equally reweighs a discrete parameter space – [YTL92]. In practice, the hierarchical approach will usually lead to faster running times, but a drop in overall quality, as a good solution to the overall problem is not guaranteed to exist as a subset of the rougher solution. See section 7.2.3 for some numerical examples.

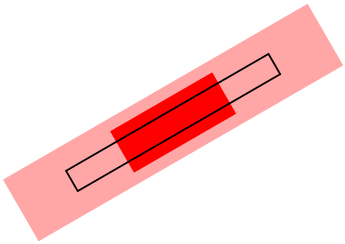
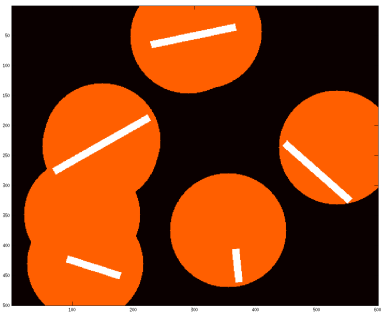
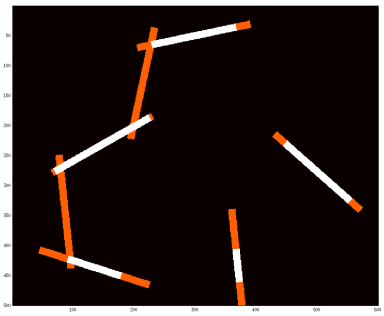


Figure 3.5: Union over all candidates in the form of a rectangle (black) with points of origin in a rectangle (dark red) is a larger rectangle (light red).



Initial cover of white rectangles disregarding orientation (circles)



Cover using rectangles with points of origin in the initial cover.

Figure 3.6: RSC in two phases

# Chapter 4

# Related Methods

We use this section to compare and contrast the RSC method of Brönnimann/Goodrich [BG95] algorithm to similar approaches, both for solving a minimal set cover problem and in a more general context. In particular, we show that a reweighting algorithm introduced by Clarkson [Cla95] to find the minimal base of a linear program can be shown to be a special case of the B/G RSC algorithm in subsection 4.4.

## 4.1 Greedy Set Covering

Perhaps the most intuitive method of determining a small cover of a given finite set of points $X$ from a collection of ranges $\mathcal{R}^{(n)}$ is to iteratively choose the ranges that cover the most points that previous ranges did not cover, or, when using a custom cost function, that minimizes the ratio of its cost to the number of these points:

**Algorithm 4:** Greedy Set Covering Algorithm

**Input:** $X = \{x_1, \ldots, x_n\}, \mathcal{R}^{(n)} = \{R_1, \ldots, R_m\}, C.$
**Output:** set cover $\mathcal{R}^{(c)}$ of $X$.
**begin**
  set $\mathcal{R}^{(c)} = \emptyset$ `// listing ranges already used for the cover`
  and $U = \emptyset$, `// union over ranges already used for the cover`
  **while** $X \setminus U \neq \emptyset$ **do**
    $i^* = \operatorname{argmin} \frac{C(i)}{|R_i \cap (P \setminus U)|}$ `// best ratio of cost to new points covered`
    $U = U \cup R_{i^*}$
    $\mathcal{R}^{(c)} = \mathcal{R}^{(c)} \cup \{R_{i^*}\}$
  return $\mathcal{R}^{(c)}$

Using this method, a cover cost of of $H_m\, n^*$ can be guaranteed, where

$$H_m := \sum_{i=1}^{m} \frac{1}{i}, \tag{4.1.1}$$

$m$ is the number of sets we choose from, and $n^*$ is the optimal cover cost – see e.g. in [Vaz01, chapter 2]. While a greedy approach supplies results of relatively high quality (or low cardinality), especially when compared to the theoretical limits for the results of the RSC algorithm as in theorem 2.2.1, it is as the name implies often not computationally efficient. We have found it to empirically be orders of magnitude slower than the RSC approach for even moderate problem sizes. Since sampling sizes needed for convergence of the RSC algorithm are however often much larger than those needed for a full cover, the greedy set covering algorithm lends itself as a relatively straight-forward method of post-processing, producing a smaller, fully covering subset from the output of an RSC algorithm or similar.

## 4.2 Solving RSC via Linear Programming

### 4.2.1 $\varepsilon$-nets for LP Set Cover

At the beginning of chapter **6.3** of "Geometric Approximation Algorithms" [HP11], a different approach to solving the Geometric Set Cover problem is introduced, here denoted with the slight modification of adding a cost function $C$:

If we assume we wish to cover the points $P = \{p_1, \ldots, p_n\}$ using ranges $\{R_1, \ldots, R_m\}$, we can formulate the problem by defining variables $y_i$ via

- $y_i = 1$ if the range $R_i$ is part of the cover and
- $y_i = 0$ otherwise, $i = 1, \ldots, m$

which lets us formulate the covering problem as

$$\min \quad \sum_{i=1}^{m} y_i \tag{4.2.1}$$

$$\text{subject to} \quad \sum_{i:p_j \in R_i} y_i \geq 1 \qquad \forall p_j \in P \tag{4.2.2}$$

$$y_i \in \{0, 1\} \qquad \forall i = 1, \ldots, m. \tag{4.2.3}$$

If no other information is available, this function will default to 1.

To solve this problem, we can use the commonly employed method of instead solving a linear programming relaxation of the problem:

$$\text{min} \quad f := \sum_{i=1}^{m} x_i \tag{4.2.4}$$

$$\text{subject to} \quad \sum_{i:p_j \in R_i} x_i \geq 1 \qquad \forall p_j \in P \tag{4.2.5}$$

$$x_i \in [0,1] \qquad \forall i = 1, \ldots, m \tag{4.2.6}$$

An equivalent formulation of the same problem is

$$\begin{aligned} \text{min} \quad & f \\ \text{s.t.} \quad & \sum_{i=1}^{m} x_i/f = 1 \\ \text{s.t.} \quad & \sum_{i:p_j \in R_i} x_i/f \geq 1/f \qquad \forall p_j \in P \\ & x_i \geq 0 \qquad \forall i = 1, \ldots, m \end{aligned}$$

or, if we define $\varepsilon = 1/f$ and $z_i = x_i/f$:

$$\text{max} \quad \varepsilon \tag{4.2.7}$$

$$\text{s.t.} \quad \sum_{i=1}^{m} z_i = 1 \tag{4.2.8}$$

$$\text{s.t.} \quad \sum_{i:p_j \in R_i} z_i \geq \varepsilon \qquad \forall p_j \in P \tag{4.2.9}$$

$$z_i \geq 0 \qquad \forall i = 1, \ldots, m. \tag{4.2.10}$$

The restriction $\sum_{i=1}^{m} z_i = 1$ lets us interpret the values of $z_i$ as probabilites assigned to the ranges $R_i$. Thus, if we are able to solve this problem, we can construct an $\varepsilon$-net of size $n = \frac{d}{\varepsilon}[\log(1/\varepsilon) + 2\log\log(1/\varepsilon) + 3]$ with respect to these probabilites using theorem 2.2.1. Via condition 4.2.9 the $\varepsilon$-net property necessitates that one range of our sample will be in each dual range $\mathcal{R}_{p_j} = \{R_i \colon p_j \in R_i\}$ for $j = 1, \ldots, n$, meaning that the set of points will be covered.

On the other hand, if we apply the original set covering Randomized Set Covering algorithm 1 with parameters $\varepsilon_0 < 2^{1/n^*} - 1$ and $n_0$ as given by theorem 2.2.1 , we have indirectly also found a (non-optimal) solution for value $\varepsilon = \frac{1}{n_0}$ for the LP 4.2.9 ff. if we choose $z_i = \frac{1}{n_0}$ for our found set of ranges. Since $n_0 \geq \frac{d}{\varepsilon_0}[\log(1/\varepsilon_0) + 2\log\log(1/\varepsilon_0) + 3]$, this solution will likely be far from optimal, but computationally inexpensive, particularly if we can use parallelization. It can be applied as long as we can bound the VC-dimension of the range space defined by the $R_i$.

If the original problem is formulated with a cost, i.e. has objective min $\sum_{i=1}^{m} C(i)\, y_i$, we can still use the first method of traditional LP solving followed by $\varepsilon$-net construction by defining $z_i = C(i)\, x_i/f$. However, RSC-based approaches are generally unable to take cost functions into account.

### 4.2.2 Randomized Rounding

If we want to use a different, arbitrary method to solve the relaxed set cover LP 4.2.4, we can produce a set cover from a non-integer solution via randomized rounding: The condition $x_i \in [0,1]$ lets us interpret $x_i$ as probabilities, specifically the probability of using a range $R_i$ in the cover. Following this intuition, it is shown in [Vaz01, chapter 14.2] that if we independently draw $t = O(\log n)$ times, and pick a range $R_i$ for our cover if is has been picked with probability $x_i$ in any of these $t$ draws, effectively picking $R_i$ with a probability of

$$1 - (1 - x_i)^t, \tag{4.2.11}$$

a cover of expected size $t f_{opt}$ is achieved with high probability, where $f_{opt}$ the optimal solution of problem 4.2.4.

## 4.3 Variants of the Set Cover Problem

### 4.3.1 Set Multi-Cover

One natural extension of the set cover problem is the problem of covering all, or some points multiple times. In LP formulation:

$$\text{min} \quad f := \sum_{i=1}^{m} x_i \tag{4.3.1}$$

$$\text{subject to} \quad \sum_{i:p_j \in R_i} x_i \geq d(p_j) \qquad \forall p_j \in P \tag{4.3.2}$$

$$x_i \in [0,1] \qquad \forall i = 1, \ldots, m, \tag{4.3.3}$$

where $d(p_j)$ indicates the number of times the point $p_j$ should be covered.

Chekuri, Clarkson and Har-Peled[CCHP09] solve this by defining a new set system based on creating multiple copies of each point; the multi-cover problem can then be reduced to solving the original set cover problem for the newly-defined range space. This is achieved by Chekuri et al. by bounding the VC-dimension of the new range space and then utilizing the same VC-based framework as Clarkson/Brönnimann/Goodrich.

### 4.3.2 Multi-class Set Cover

A similar, but slightly more complicated extension of the set cover problem is the Multi-class set cover problem, in which a class $l$ to both ranges and points $p_i \in P$:

$$\underset{\alpha_j^l, \xi_i, \eta_i}{\text{minimize}} \sum_i \xi_i + \sum_i \eta_i + \lambda \sum_{j,l} a_j^l \quad \text{s.t.} \tag{4.3.4}$$

$$\xi_i \geq 1 - \sum_{j\colon p_i \in R_j} a_j^{y_i} \quad \forall p_i \tag{4.3.5}$$

$$\eta_i \geq \sum_{j\colon p_i \in R_j, y_i \neq l} a_j^l \quad \forall p_i \tag{4.3.6}$$

$$a_j^l \in \{0, 1\} \tag{4.3.7}$$

Here, $a_j^l = 1$ indicates that the range $R_j$ is assigned to covering elements of class $l$; $y_i$ indicates which class $x_i$ is assigned to.

Intuitively, $\xi_i$ penalizes cases where a point $x_i$ is not covered, whereas $\eta_i$ grows larger when points of different classes are included in the same range.

Bien et al. [BT11] apply two approaches similar to the randomized rounding and greedy methods of sections 4.2.2 and 4.1. For randomized rounding, one can use constraint 4.3.7 to reduce the LP problem to solving

$$\underset{\alpha_j^l, \xi_i \in \mathcal{P}_l}{\text{minimize}} \sum_i \xi_i + \sum_{j,l} C_l(j) a_j^l \text{ s.t.} \tag{4.3.8}$$

$$\xi_i \geq 1 - \sum_{j\colon x_i \in R_j} a_j^{y_i} \quad \forall x_i \tag{4.3.9}$$

$$a_j^l \in \{0, 1\}, \tag{4.3.10}$$

where

$$\mathcal{P}_l = \{p_i \in P\colon y_i = l\} \tag{4.3.11}$$

$$C_l(j) := \lambda + |R_j \cap (P \setminus \mathcal{P}_l)|\,, \tag{4.3.12}$$

for each class $l$.

As before, $\xi_i$ penalizes cases where a point $x_i$ is not fully covered, whereas $C_l(j)$ is a fixed cost (to minimize total number of ranges used) plus a penalty for cases of a range $R_j$ covering points not in its class $l$.

This problem 4.3.8 can be relaxed and solved for each class $l$ seperately; the resulting $a_j^l \in [0, 1]$ can be seen as probabilites, and a cover can be estimated by independently drawing from $\mathrm{Ber}(a_j^l)$ distributed random variables $O(\log \mathcal{P}_l)$ times, and using a range $R_j$ to cover class $l$ if any of these draws came up positive.

Alternatively, Bien et al. [BT11] propose a modified greedy algorithm works very similarly to algorithm 4; however, instead of repeatedly minimizing over $\frac{C(i)}{|R_i \cap (P \setminus U)|}$, we introduce a new objective function

$$\Delta\text{Obj}(j,l) = \Delta\xi(j,l) - \Delta\eta(j,l) - \lambda \quad \text{with} \tag{4.3.13}$$

$$\Delta\xi(j,l) := |\mathcal{P}_l \cap (R_j \setminus \bigcup_{j' \in \mathcal{R}_l^{(c)}} R_{j'})| \tag{4.3.14}$$

$$\Delta\eta(j,l) := |R_j \cap (P \setminus \mathcal{P}_l)|\,, \tag{4.3.15}$$

where $\mathcal{R}_l^{(c)}$ is the set of all ranges $R_j$ assigned to cover class $l$ up to this step.

In words, $\Delta\text{Obj}(j,l)$ is the improvement of the cover gained by (newly) assigning range $R_j$ to cover (points of) class $l$

With this definition we can solve multi-class set cover using a greedy approach:

**Algorithm 5:** Modified Greedy Algorithm: Set Covering

**Input:** $P = \{p_1, \ldots, p_n\}, \mathcal{R} = \{R_1, \ldots, R_m\}$,
**Output:** set cover $\mathcal{R}^{(c)}$ of $X$.
**begin**
  set $\mathcal{R}_l^{(c)} = \emptyset \ \forall l$
  **while** $\Delta Obj(j,l) > 0$ **do**
    $(j^*, l^*) = \text{argmax}\, \Delta\text{Obj}(j,l)$
    $\mathcal{R}_{l^*}^{(c)} = \mathcal{R}_{l^*}^{(n)} \cup \{R(j^*)\}$
  return $\mathcal{R}_l^{(c)} \ \ l = 1, \ldots, L$

## 4.4 Solving a Linear Program via RSC

In this subsection, we show how the results of the previous sections can be applied to solve a general linear program. This is achieved by presenting an algorithm introduced by Clarkson [Cla95] as a special case of the B/G scheme, applied to minimally cover a subset of $\mathbb{R}^d$, consisting of possible solutions, with half-spaces representing the linear constraints. We also determine the dual VC-dimension (see section 2.2) of the corresponding range space to give reliable bounds on sampling sizes for $\varepsilon$-nets.

In the notation of [MR95], the weighted version [Cla95] of Clarkson's original randomized algo-

rithm for LP solving [Cla88] reads the following way:

**Algorithm 6:** IterSampLP

**Input:** A set of constraints $H$, where $|H| = n$, a vector $c$
**Output:** A maximizer of $cx, x \in \mathbb{R}^d$, under these constraints.
**begin**
  set $w_h = 1 \; \forall h \in H$
  **if** $n < 9n^{*2}$ **then**
    return **Simplex**(H)
  **else**
    $V \leftarrow H$
    **while** $|V| > 0$ **do**
      Choose $R \subset H$ with $|R| = 9d^2$ according to $w_h$
      $x \leftarrow \mathbf{Simplex}(R)$
      $V \leftarrow \{h \in H : x \text{ violates } h\}$
      **if** $\sum_{h \in V} w_h \leq \frac{2}{9n^* - 1} \sum_{h \in H} w_h$ **then**
        $\forall h \in V$ set $w_h \leftarrow 2w_h$
    return $x$

Here, $\mathbf{Simplex}(R)$ is the optimum under the conditions $R$ using the simplex method. The idea of IterSampLP is to find a basis of $H$, i.e. a minimal set of conditions that define all feasible solutions, through reweighting. Replacing the expression "$x \in \mathbb{R}^d$ violates conditon $h : (Ax)_i \leq b_i$" with the equivalent expression "$x \in S_h$", where

$$S_h = \{(Ax)_i > b_i\}, \tag{4.4.1}$$

the problem is equivalent to minimally covering all non-feasible solutions using the half-spaces $S_h$ – see figure 4.1.

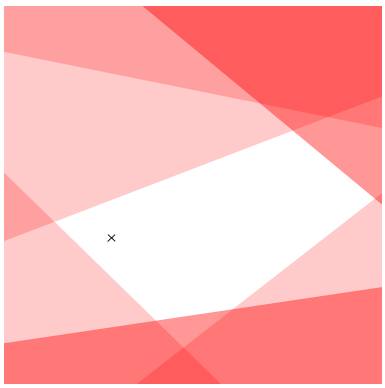


Figure 4.1: Finding the basis of an LP is equivalent to minimally covering the compliment of the space of admissable solutions with the half-spaces of solutions that violate a condition $h$

Similarly, defining

- $X = \{\mathbf{Simplex}(R) : R \subseteq H\}$ where, if the conditions in $R$ form an unbounded linear problem, $\mathbf{Simplex}(R)$ is an arbitrary point of high absolute value that satisfies all conditions in $R$
- $\mathcal{R} = S_H := \{X \cap S_h : h \in H\}$

then $\{s \in S_H : x \in s\}$, the set of all ranges that contain the point $x$, is the dual range of $x$ in the (finite) range space $(X, \mathcal{R})$.

Indeed, the algorithm is very similar to the B/G scheme applied to a set cover problem, and using random sampling as a net-finder [BG95, HP11] with the following exceptions:

- Some parameter values, e.g. the size of the random sample, are different
- The point $x = \mathbf{Simplex}(R)$ is a specific, not randomly chosen point not covered by $S_R$
- The condition for termination of the algorithm is not $\nexists x \in \overline{\bigcup_{h \in R} S_h}$, but rather $|V| > 0$

However, the latter two differences are not relevant for the proof of convergence as given in proposition 2.3.1.

Let $n^*$ be the cardinality of $\mathcal{B}(H)$, the basis of $H$. Set

$$\varepsilon < 2^{\frac{1}{n^*}} - 1 \tag{4.4.2}$$

and $n = n(\varepsilon)$ a sample size that guarantees a random sample is an $\varepsilon$-net with high probability for spaces of limited VC-dimension – see e.g. theorem 2.2.1.

With these two parameters, we can formulate a version of the algorithm with tighter bounds, where the proof of termination is still given by proposition 2.3.1, and the parallelization method of section 3.1 can be applied:

**Algorithm 7:** IterSampLP, RSC Version

**Input:** A set of conditions $H$, where $|S_H| = n$
**Output:** The optimum of a function $cx, x \in \mathbb{R}^d$, under these constraints.
**begin**
  set $w_h = 1 \, \forall R_h \in H$
  **while** $|V| > 0$ **do**
    Choose $R \subset H$ with $|R| = n(\varepsilon)$ according to $w_h$
    $x \leftarrow \mathbf{Simplex}(R)$
    $V \leftarrow \{h \in H : x \text{ violates } S_h\}$
    **if** $\sum_{h \in V} w_h < \varepsilon \sum_{h \in H} w_h)$
    **then**
      $\forall h \in V$ set $w_h \leftarrow 2w_h$
  return $x$

The only problem remaining is determining the dual VC dimension of the range space $(X, S_H)$ to bound the needed size of a random sample $n = n(\varepsilon)$.

**Lemma 4.4.1** (Dual VC-dimension of halfspaces)**.** *Let $X$ be a subset of $\mathbb{R}^d$.*

*Let $\mathcal{R} = \{\{x \in X : a_j x > b_j\} : j = 1, \ldots, N\}$ for some $a_j \in \mathbb{R}^d, b_j \in \mathbb{R}$. Then the dual VC-dimension of $(X, \mathcal{R})$ is bounded by $d$.*

*Proof.* The dual VC dimension of a space is the maximal number of ranges that can be shattered using dual ranges (see definition 2.2.4). A set of ranges $\{\{x \in X : a_i x > b_i\}_{i=1}^n\}$ can be shattered by intersecting with dual ranges *iff* there exists at least one point in every $X \cap H_1 \cap \ldots \cap H_m$ where

$$H_i \in \{\{a_j x > b_j\}, \{a_j x > b_j\}_c = \{a_j x \leq b_j\}\} \tag{4.4.3}$$

for some $j \in 1, \ldots, n, i_k \neq i_l \implies j_k \neq j_l$, i.e. in every non-trivial intersection of ranges from this set and their complements in $X$.

This can only be true if the vectors $a_1, \ldots, a_n$ are independent:

Assume without loss of generality that $a_n = \sum_{j=1}^{n-1} a_j \, u_j$. Then

$$a_j x > b_j \qquad \forall j \neq n : u_j \geq 0 \tag{4.4.4}$$
$$a_j x \leq b_j \qquad \forall j \neq n : u_j < 0 \tag{4.4.5}$$

implies that $u_j a_j x \geq u_j b_j \; \forall j \neq n$, and thus $a_n x \geq b_n^* := \sum_{j=1}^{n-1} u_j \, b_j$.

Similarly,

$$a_j x \leq b_j, \qquad \forall j \neq n : u_j \geq 0 \tag{4.4.6}$$
$$a_j x > b_j, \qquad \forall j \neq n : u_j < 0 \tag{4.4.7}$$

implies that $a_n x \leq b_n^*$.

Consequently, if $b_n^* > b_n$, then

$$a_j x > b_j, \qquad \forall j \neq n : u_j \geq 0 \tag{4.4.8}$$
$$a_j x \leq b_j, \qquad \forall j \neq n : u_j < 0 \tag{4.4.9}$$
$$a_n x \leq b_n \tag{4.4.10}$$

has no solution, since the first two lines imply $a_n x \geq b_n^* > b_n$.

Similarly, if $b_n^* \leq b_n$, then

$$a_j x \leq b_j, \qquad \forall j \neq n : u_j \geq 0 \tag{4.4.11}$$
$$a_j x > b_j, \qquad \forall j \neq n : u_j < 0 \tag{4.4.12}$$
$$a_n x > b_n \tag{4.4.13}$$

has no solution.

The maximal number of linearly independent vectors in $\mathbb{R}^d$ is obviously $d$.

□

In total, this establishes much better bounds than used in the original algorithm, while simultaneously allowing us to apply the methods of parallelization introduced in the previous section for RSC.

## 4.5 RSC as a Special Case of Multiplicative Weights

In [AHK12] Arora et al. give an excellent overview of multiple well-known algorithms, such as the Ada boost algorithm [FS95], the Plotkin-Shmoys-Tardos algorithm [PST95] and the Winnow algorithm [Lit88], and subsumes them as special cases or slight modifications of the *Multiplicative Weights algorithm* 8:

**Algorithm 8:** Multiplicative Weights algorithm

**Input:** $\eta \leq \frac{1}{2}$,
maximal step number $T \in \mathbb{N}$,
a set of decisions $i, i = 1, \dots n$, with a dynamic cost $m_i^t \ \forall i, t \leq T$
**Output:** Probability distribution $(p_i^T) := \frac{w_i^T}{\Phi^T}$ over the decisions, where $\Phi^T := \sum w_i^T$.
**begin**
  set $w_i^1 = 1 \ \forall i, t = 1$
  **while** $t < T$ **do**
    Choose decision $i$ according to the distribution $(p_i^t) := (\frac{w_i^t}{\sum w_i^t})$
    Observe the cost of the decision $m_i^T$
    Set $w_i^{t+1} := w_i^t(1 - \eta m_i^t) \ \forall i$
    $t = t + 1$
  return $(p_i^T)$

This algorithm supplies the following guarantee of convergence:

**Theorem 4.5.1** ([AHK12]). *Assume that all costs $m_i^t \in [-1, 1]$ and $\eta \leq \frac{1}{2}$. Then after $T$ rounds the cost of any decision $i$ is limited by the following equation:*

$$\sum_{t,j} m_j^t p_j^t \leq \sum_t m_i^t + \eta \sum_t \left|m_i^t\right| + \frac{\log n}{\eta} \ \forall i \tag{4.5.1}$$

If we formulate the algorithm with gains instead of losses, i.e. with costs $u_i^t = -m_i^t$, we obtain the result

**Theorem 4.5.2** ([AHK12]). *Assume that all gains $u_i^t \in [-1, 1]$ and $\eta \leq \frac{1}{2}$, and the algorithm is run with gains, i.e. positive update. Then after $T$ rounds:*

$$\sum_{t,j} u_j^t p_j^t \geq \sum_t u_i^t - \eta \sum_t \left|u_i^t\right| - \frac{\log n}{\eta} \ \forall i \tag{4.5.2}$$

The set covering algorithm 1, specifically the steps in which a weight update is actually performed, can be understood as a special case of the *Multiplicative Weights algorithm* with gains instead of losses: every "decision" $i$ is a range $R_i \in \mathcal{R}$, and we update with $\eta = \frac{1}{3}$ and $u_i^t = 1 \ \forall R \in \mathcal{R}_{x^t}$ where $x^t$ is a point not covered in step $t$, and $u_i^t = -1$ otherwise to double the weight of ranges in $\mathcal{R}_{x^t}$ relative to all others.

Assuming that for all points $x$ the size of their dual range $|\mathcal{R}_x| \geq D \in \mathbb{N}$, at least $D$ points will be updated with gain $u_i = 1$ in each step, and theorem 4.5.2 supplies us with the bound

$$\sum_{t,j} u_j^t p_j^t \geq \sum_t \left( 2 \mathbb{1}_{\{R_i \in \mathcal{R}_{x^t}\}} - 1 \right) - \eta \sum_t 1 - 3 \log(|\mathcal{R}|) \ \ \forall i \tag{4.5.3}$$

$$|\mathcal{R}| \sum_{t,j} u_j^t p_j^t \geq T \left[ 2D - |\mathcal{R}| - \frac{1}{3} |\mathcal{R}| \right] - 3 |\mathcal{R}| \log |\mathcal{R}|, \tag{4.5.4}$$

where the second inequality is obtained by summing over all ranges $R_i$.
Since we assume that in each step $\sum_{R_i \in \mathcal{R}_x} p_j^t \leq \varepsilon$, we know that

$$\sum_j u_j^t p_j^t = \sum_{R_i \in \mathcal{R}_x} p_j^t + \sum_{R_i \notin \mathcal{R}_x} -p_j^t \leq \varepsilon - (1 - \varepsilon) = 2\varepsilon - 1 \ \ \forall t \tag{4.5.5}$$

and thus

$$T |\mathcal{R}| (2\varepsilon - 1) \geq T[2D - \frac{4}{3} |\mathcal{R}|] - 3 |\mathcal{R}| \log |\mathcal{R}| \tag{4.5.6}$$

$$T \left[ 2D - 2\varepsilon |\mathcal{R}| - \frac{1}{3} |\mathcal{R}| \right] \leq 3 |\mathcal{R}| \log |\mathcal{R}| \tag{4.5.7}$$

However, since generally $\frac{1}{3} |\mathcal{R}| + 2\varepsilon |\mathcal{R}|$ will be much bigger than $2D$, the prefactor of $T$ will be negative, and the bound is not very useful.

Alternatively, we can sum equation 4.5.4 not over all ranges, but over a subset $\{R_{i^*}\}$ of size $n^*$ that we assume covers all points. Until all points are covered, at least one of these ranges will be updated in each step, i.e.

$$\sum_{i^*} \sum_{t=1}^{T} u_i^t = \sum_{t=1}^{T} \sum_{i^*} u_i^t \geq \sum_{t=1}^{T} [+1 + (n^* - 1)(-1)] = T [2 - n^*], \tag{4.5.8}$$

resulting in the inequalities

$$n^* \sum_{t,j} u_j^t p_j^t \geq T [2 - n^*] - T \frac{1}{3} n^* - 3n^* \log |\mathcal{R}| \tag{4.5.9}$$

$$T n^* (2\varepsilon - 1) \geq T [2 - \frac{4}{3} n^*] - 3n^* \log |\mathcal{R}| \tag{4.5.10}$$

$$3n^* \log |\mathcal{R}| \geq T [2 - (\frac{1}{3} + 2\varepsilon) n^*] \tag{4.5.11}$$

However, since $2 - (\frac{1}{3} + 2\varepsilon) n^*$ will generally not be positive, this will again not result in a useful upper bound for the number of steps $T$.

The given proof of the central theorem 4.5.1 follows the same basic idea as the proof of proposition 2.3.1 and theorem A.1.1, i.e. giving an upper bound to the factor by which the total weight grows in

each step, and thus the total weight growth, and comparing this to the growth of an interesting subset of ranges.

However, in the proof of theorem 4.5.1, several estimates are made that will not always be tight; on the other hand, the main bounding estimate for Theorem A.1.1, given in equation 2.3.3, will be close tight in a lot of cases: Every time the weight of a range is doubled the probability that it will not be selected for the cover and have its weight doubled in the next step decreases, increasing the probability of a periodic update of each optimal range, for which the estimate is tight.

Thus, while theorem 4.5.1 is formulated for and applicable in a much more general context, it is not surprising that the results do not improve the bounds stated in Theorem A.1.1.

## 4.6 Comparison to Hough transformation

The main advantage of the RSC algorithm is that, while it is designed for the detection of parameterizable objects, no additional assumptions are made. In particular, not relying on connectivity assumptions and providing the opportunity to incorporate greyscale values into parameter voting makes it robust in the detection of noisy objects; this makes it comparable to the relatively slow but stable method of Hough transform [IK88]. Though originally conceived as a parameter voting method for lines, it can be generalized to arbitrary parameterized shapes in a straightforward way – see e.g. [Bal87]. For the RSC algorithm, the dual ranges (see definition 2.2.2) that define which parameters an image point votes for (i.e. reweighs) are exactly the admissable parameters used in generalized Hough voting. On the other hand, the choice of image points that contribute to voting for the RSC algorithm is randomized and dependent on the current weights, while it is simply sequential for the original Hough transform.

Several randomized versions of the Hough transform have been established ([KHXO95, KKA00, WR02]).Since the RSC algorithm fundamentally is parameter voting on a randomized subset, the Probabilistic Hough Transform ([KEB91]) is the most directly comparable to RSC. Although both procedures are essentially parameter voting on a randomized set of data points, and many heuristic refinements of the basic Hough algorithm have been developed in the past, the theoretical foundation of the Randomized Set Covering Algorithm offers several advantages:

- The fact that the desired cardinality of the cover is an input parameter offers an intuitive way of regulating quality versus run time.
- The theoretical limits on $\varepsilon$-net size, and thus achievable cover quality, are chosen for the case of a uniform distribution, and are usually not exact in practice. However, they do supply a rough idea of which quality of cover might be achieved in a certain geometric situation.
- Although theoretical limits for cover cardinality are not exact, condition (2.3.3), and thus ensuring that condition (2.3.1) is satisfied in each update step, is close to a necessary condition for ensuring that optimal ranges receive the most votes. Being able to ensure this via an easily-checked condition is a big advantage compared to uncontrolled voting.

In practice, the probabilistic Hough transform performs far worse than the RSC scheme even for simple problems; see the later section 7.2.4 for numerical results.

# Part II

# Set Covering with a Gibbs Prior

# Chapter 5

# Gibbs Point Process Model

## 5.1 Introduction and Overview

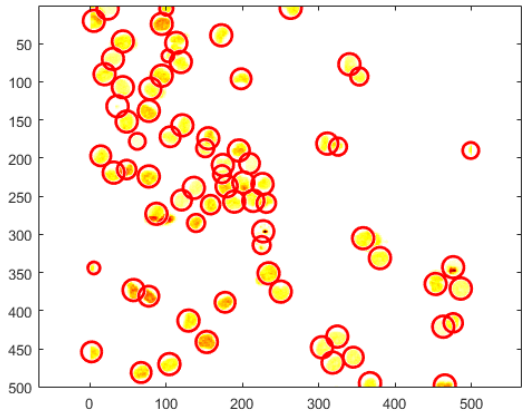
While the methods introduced in the previous section vary in their approach, they are all united in one aspect: the segmentation produced relies either exclusively, or at least overwhelmingly, on image data. Incorporation of prior knowledge of object behavior is either absent, or extremely simplified, like the punishment of large overlap for the LP-based methods. In cases where image data is difficult or unreliable, but a priori knowledge can be obtained, e.g. because we need to segment several samples of similar data, a natural refinement of a purely data-based method is to apply a Bayesian approach, i.e. combine image data and a prior model to find the most likely segmentation. The goal of this section is to introduce a model of object behavior that is simple enough to allow both repeated energy minimization and the estimation of model parameters for 3D data with an acceptable amount of computational expense, while being refined enough to approximate the pairwise behaviour of real physical objects – see figure 5.1. We then show exactly how we can combine the iterative RSC approach with this model in section 6.1, and demonstrate performance on self-generated, as well as real cell and fiber data sets in sections 7.3, 9.1, and 9.2, respectively.

## 5.2 Basic Definitions: Point Processes

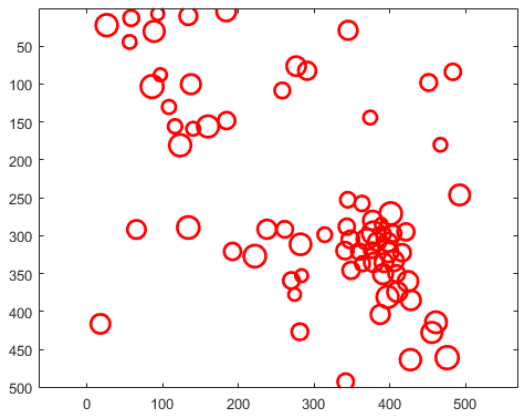
To establish a point process model of fiber, or more general object interaction, we first need a few basic definitions for point processes. See e.g. [MW04] for further elaboration on the existence and uniqueness of the following definitions.

**Definition 1** (Point Process)**.** A spatial point process is a random countable subset of a space $S$. We restrict ourselves to real point processes, i.e. $S \subseteq \mathbb{R}^d$, whose realizations $x$ are *locally finite*, i.e. $|x \cap B| < \infty$ for any bounded set $B$.

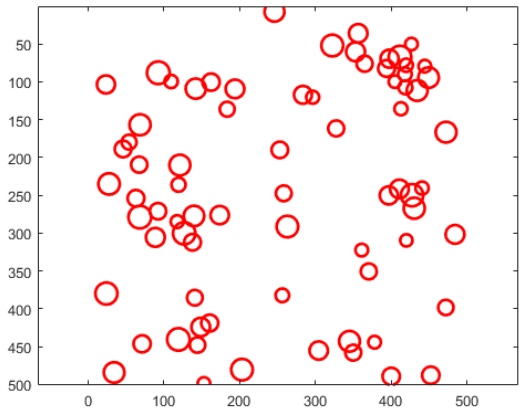
**Definition 2** (Intensity Function, Intensity Measure)**.** Given $S \subseteq \mathbb{R}^d$, an *intensity function* is a function $\rho \colon S \to [0, \infty)$ that is *locally integrable*, i.e. $\int_B \rho(\xi) d\xi < \infty$ for any bounded $B \subseteq S$.



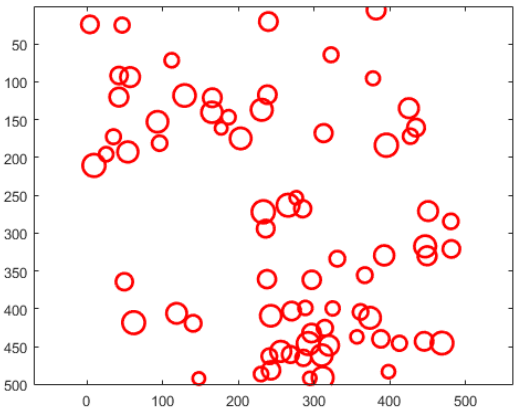

(a) Segmentation




(b) Simulation 1




(c) Simulation 2




(d) Simulation 3

Figure 5.1: Segmentation of cell data; 3 samples of Gibbs point processes with the (estimatedly) same clustering behaviour.

The *intensity measure* $\mu$ corresponding to $\rho$ is given by

$$\mu(B) = \int_B \rho(\xi) d\xi \;\; \forall B \subseteq S. \tag{5.2.1}$$

**Definition 3** (Poisson Point Process)**.** A point process $X$ on $S$ is called a *Poisson Point Process* with intensity function $\rho$ / intensity measure $\mu$ if the following properties are satisfied:

- for any $B \subseteq S$ with $\mu(B) < \infty$, $|X \cap B| \sim \text{po}(\mu(B))$, the Poisson distribution with mean $\mu(B)$.
- for any $n \in \mathbb{N}$ and $B \subseteq S$ with $0 < \mu(B) < \infty$, conditional on $|X \cap B| = n$, $|X \cap B| \sim \text{binom}(B, n, f)$ with $f(\xi) := \rho(\xi)/\mu(B)$.

We then write $X \sim \text{Poisson}(S, \rho)$.

**Definition 4** (Point Process Density)**.** Given $S \subseteq \mathbb{R}^d$, let $N_{lf}$ be the space of *locally finite point configurations*:

$$N_{lf} := \{x \subseteq S \colon \; |x \cap B| < \infty \text{ for all bounded } B \subseteq S\}\,. \tag{5.2.2}$$

If $X_1$ and $X_2$ are two point processes on $S$, then $X_1$ is *absolutely continuous* with respect to $X_2$ *iff*

$$P(X_2 \in F) = 0 \;\Rightarrow\; P(X_2 \in F) = 0 \;\; \forall\, F \in N_{lf}. \tag{5.2.3}$$

Equivalently, by the Radon-Nikodym theorem there exists a function $f \colon N_{lf} \to [0, \infty)$ so that

$$P(X_1 \in F) = \mathbb{E}\left[\mathbb{1}_{\{X_2 \in F\}} f(X_2)\right] \;\; \forall\, F \in N_{lf}. \tag{5.2.4}$$

We call $f$ a *density* for $X_1$ with respect to $X_2$.

In the following, unless specified otherwise, densities of a point process on $S$ are defined relative to the Poisson Point Process on $S$ with intensity measure $\rho \equiv 1$.

## 5.3 Pairwise Interaction/Gibbs Point Processes

A multiscale process is the natural extension of a binary/hard-core process, i.e. one that simply prohibits object distances below a minimal radius. Instead of assigning binary probabilities based on one minimal distance, we assign non-binary probabilities based on several object interaction radii – see figure 5.2.

A multiscale point process is a special case of a pairwise interaction point process, i.e. a point process that assigns probabilities based on pairs of points. It is defined by a density of the form

$$f(x) \sim \beta^{n(x)} \prod_{i=1}^{k} \theta_i^{y_i(x)}, \tag{5.3.1}$$

where $n(x)$ is the cardinality of the point vector $x$, $\beta > 0$ and

$$y_i(x) := \sum_{k<l} \mathbb{1}_{[r_i, r_{i+1})}(d(x_k, x_l)) \tag{5.3.2}$$

counts the number pairwise distances that fall into the discretized intervals $[r_i, r_{i+1})$. Although formally, Gibbs point processes are a generalization of multiscale processes [Der17], we will somewhat casually refer to this process as the Gibbs model.

Integrability is ensured [MW04, chapter 6.2] if either

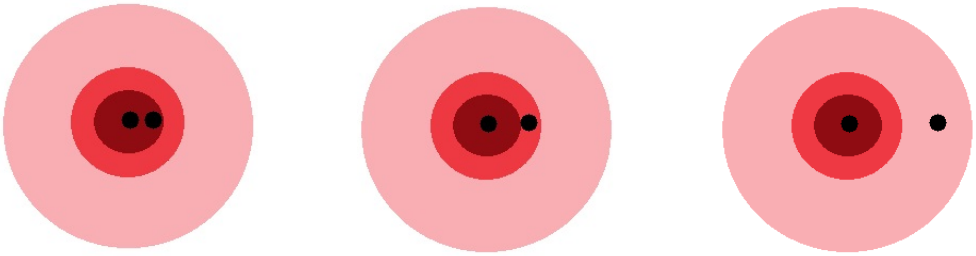
(i) $0 < \theta_1 \leq 1$ and $0 \leq \theta_2 \leq 1, \ldots, 0 \leq \theta_k \leq 1$, or

(ii) $\theta_1 = 0$ and $\theta_2 \geq 0, \ldots, \theta_k \geq 0$.

In the case of the modelling of physical objects, we can reasonably define a minimal distance, i.e. satisfy $\theta_1 = 0$ in our model. We thus choose the otherwise less restrictive condition (ii) and write our density in the equivalent form

$$f(x) \sim \beta^{n(x)} \exp(U_\zeta(x)) \tag{5.3.3}$$

$$U_\zeta(x) := \sum_{i=1}^{k} \zeta_i \, y_i(x) \tag{5.3.4}$$



(a) $\zeta_1$ (b) $\zeta_2$ (c) $\zeta_3$

Figure 5.2: Multiscale processes assign probabilities are based on the discretized pairwise distance of points. Depending on which domain the relative position of two points falls in, we assign a probability using different parameters $\zeta_i$ to the pair. A hard-core process corresponds to $\zeta_1 = -\infty$.

## 5.4 Gibbs Model for Fiber Data

Multiscale processes are usually defined based on the discretization of one pairwise object distance. We refine this definition to model fiber data: In a pattern that is generated from a physical flow through a constricted space, we expect objects of close distance to have a similar orientation as well. Intuitively, the probability of a configuration should rely not only on the distance of two fibers, but also their relative orientation.

We thus rewrite our model to define interaction based on two distances $d_1$ and $d_2$; For our purposes, $d_1(x_k, x_l)$ will be the distance of the two line segments that are the middle-axis of the two cylinders

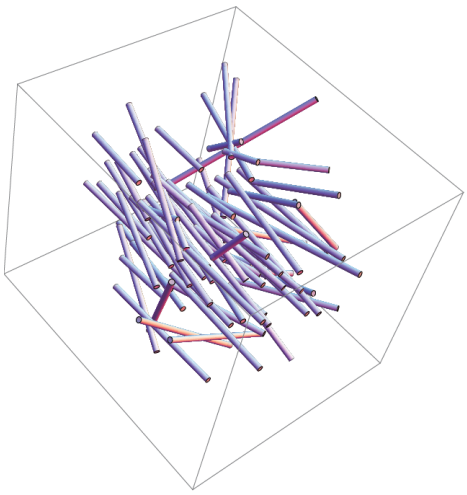


Figure 5.3: Flow: Similar orientation for close fibers (simulation)

$x_k$ and $x_l$; meanwhile $d_2$ will be the distance of normed orientation vectors modulo reflection. For details on $d_1$ see the subsection 5.4.1. For our two-dimensional model, we redefine our *Interaction Function* $\Phi$ as a two-dimensional step function of the two distances $d_1$ and $d_2$ with steps of height $\zeta_{ij}$ and their limits on the points $(r_i, s_j)$:

$$f_{\Phi,\beta}(x) := \frac{1}{c_{\beta,\zeta}} \beta^{n(x)} \exp\{U_\zeta(x)\} \tag{5.4.1}$$

$$U_\zeta(x) := \sum_{k<l} \Phi(d_1(x_k, x_l), d_2(x_k, x_l)) \tag{5.4.2}$$

$$\Phi(d_1, d_2) := \sum_{i,j} \zeta_{ij} \mathbb{1}_{[r_i, r_{i+1})}(d_1) \mathbb{1}_{[s_j, s_{j+1})}(d_2). \tag{5.4.3}$$

Although multiscale processes are usually defined based on the single Euclidean distance, the properties that are critical for the methods used in later sections remain for the modified model: well-definedness and applicability of the DCFTP method (5.5.2) are guaranteed for locally stable processes – see [MW04, chapter 6.2] and [KM00], respectively. Local stability is ensured for a standard multiscale process with a minimal distance because $f(x)$ will be 0 if $n(x) > n_0$ for some $n_0$ – see [MW04, chapter 6.2]. The same will be true, however, if we define a minimal distance for a distance $d_1$ that is smaller or equal to a Euclidean distance – like the distance between line segments, which is always smaller or equal to the distance of their middle points. Applicability of the Monte Carlo root-finding method in subsection 5.6.2 is based on the density being in the form of an exponential family, which it still is. We only need to exercise caution when using a smoothing prior for the method. See the subsection 5.6.2 for details.

### 5.4.1 Distance between two Cylinders

Let $V \subset \mathbb{R}^3$ be a finite volume. Let $S^3_+$ denote elements of the 3-sphere for which the first coordinate is positive.

Let $(\vec{x}, \vec{o}, L)$ be the parametrization of a cylinder by its central point $\vec{x} \in V$, its orientation $\vec{o} \in S^3_+$ and length $L \in \mathbb{R}_+$. Note that the orientation of a cylinder is only defined up to reflection across its central point, which is why we restrict to unit vectors with a positive first component.

We can calculate the difference of orientations of two cylindrical candidates in a straightforward way as the angle between the vectors defining their orientation:

$$M_2\left((\vec{x}, \vec{o}, L), (\vec{x'}, \vec{o'}, L')\right) = 2 \arcsin\left(0.5 * \|\vec{o} - \vec{o'}\|\right). \tag{5.4.4}$$

Since we will exclusively be looking at cases in which the length of a cylinder will be much larger than its radius, a good approximation for cylinder distance will be the distance between the two line segments that form their central axes, denoted here by

$$M_1\left((\vec{x}, \vec{o}, L), (\vec{x'}, \vec{o'}, L')\right). \tag{5.4.5}$$

### 5.4.2 Distance between two Line Segments

See e.g.[SE02]. To calculate the distance between two line segments, we will first calculate the distance $\text{dist}(s, \vec{p})$ between a line segment $s$ and a point $\vec{p}$, which we will need several times below. So let

$$s := \vec{x} + l\,\vec{o} \quad l \in [0, L] \tag{5.4.6}$$

be the central line segment of the cylinder defined by $(\vec{x}, \vec{o}, L)$. Given a point $\vec{p}$ we can solve

$$\vec{x} + l_0\,\vec{o} + r_0\vec{u} = \vec{p} \tag{5.4.7}$$
$$\langle \vec{u}, \vec{o} \rangle = 0 \tag{5.4.8}$$
$$\|\vec{u}\| = 1 \tag{5.4.9}$$

to find the point $\vec{x} + l_0\,\vec{o}$ closest to $\vec{p}$ on the full line that extends the line segment $s$. If $l_0 \in [0, L]$, this point is on the line segment, and

$$\text{dist}(s, \vec{p}) = \|\vec{x} + l_0\,\vec{o} - \vec{p}\| := r_0. \tag{5.4.10}$$

If $l_0 \in (-\infty, 0)$, the closest point in $s$ to $p$ is the starting point $\vec{x}$ and

$$\text{dist}(s, \vec{p}) = \|\vec{x} - \vec{p}\|. \tag{5.4.11}$$

If $l_0 \in (L, \infty)$, the closest point in $s$ to $p$ is the end point $\vec{x} + L\vec{o}$ and

$$\text{dist}(s, \vec{p}) = \|\vec{x} + L\vec{o} - \vec{p}\|. \tag{5.4.12}$$

Now to calculate the distance between two line segments $s_1$ and $s_2$, define

$$s_1 := \vec{x} + l\,\vec{o} \qquad l \in [0, L] \tag{5.4.13}$$

$$s_2 := \vec{x'} + m\,\vec{o'} \quad m \in [0, L']. \tag{5.4.14}$$

We again assume without loss of generality that the directional vectors $\vec{o}, \vec{o'}$ and later $\vec{u}$ are unit vectors, and that $\langle \vec{o}, \vec{o'} \rangle \geq 0$.

If the line segments are parallel, i.e. $\vec{o'} = \vec{o}$, solve

$$\vec{x} + l_0\,\vec{o} + r_0\vec{u} = \vec{x'} \tag{5.4.15}$$

$$\langle \vec{u}, \vec{o} \rangle = 0 \tag{5.4.16}$$

$$\|\vec{u}\| = 1 \tag{5.4.17}$$

to find a point in the line extension of line segment $s_1$ closest to the starting point $\vec{x'}$ of line segment 2.

If $l_0 \in [0, L]$, these are also a pair of closest line segment points, and the distance is $r_0$.

Otherwise, if $l_0 \in (L, \infty)$, we know that the whole line segment $s_2$ lies "above" line segment $s_1$, meaning

$$\langle \vec{p}, \vec{u} \rangle \leq \langle \vec{q}, \vec{u} \rangle \quad \forall p \in s_1, q \in s_2, \tag{5.4.18}$$

and the distance between the line segments is the distance of the end point of $s_1$ and the starting point of $s_2$, namely $\left\| \vec{x} + L\vec{o} - \vec{x'} \right\|$.

If $l_0 \in (-\infty, 0)$, we know that the starting point $\vec{x'}$ of $s_2$ lies "below" line segment $s_1$, implying that either all of $s_1$ lies "above" $s_2$, or the closest point to $\vec{x}$ in the line extension of $s_2$ lies in the line segment $s_2$. In both cases, the distance of the line segments is given by $\text{dist}(s_2, \vec{x})$.

If they are *not* parallel, the first problem is finding the two closest points of the line extensions of the line segments.

To accomplish this, solve

$$\langle \vec{u}, \vec{o} \rangle = 0 \tag{5.4.19}$$

$$\langle \vec{u}, \vec{o'} \rangle = 0 \tag{5.4.20}$$

$$\|u\| = 1 \tag{5.4.21}$$

for a $\vec{u} \in S^3$ that is orthogonal to both line segments, then solve

$$\vec{x} + l_0\vec{o} + r_0\,\vec{u} = \vec{x'} + m_0\,\vec{o'} \tag{5.4.22}$$

to get the parameters $l_0$ and $m_0$ of closest line points and the distance $r_0$ of the two lines.

If both $l_0 \in [0, L]$ and $m_0 \in [0, L']$, then $r_0$ is also the distance of line segments.

If exactly one parameter is not within limits, e.g. $l_0 \in [0, L]$ but $m_0 < 0$, the distance of line segments will be the distance of the line point closest to the calculated parameter, in this case $\vec{q} = \vec{x'} + m_1\,\vec{o'}$ with $m_1 = 0$, and the other line.

If both parameters are out of limits, e.g. $l_0 > L$ and $m_0 < 0$, again choose the parameters closest to the optimal ones, here $l_1 = L$ and $m_1 = 0$, and calculate distance to the other line segment for both corresponding points. The distance of the two line segments will then be the minimum of these two distances.

## 5.5 Simulation

### 5.5.1 Markov Chain Monte Carlo

Markov Chain Monte Carlo methods of simulation can be implemented in a straightforward and efficient way. Their main drawback is that they do not sample from a density directly, but rather approach it starting from a (usually) uniform distribution. The number of steps until the target distribution is approached sufficiently, usually referred to as "burn-in", can generally only be determined empirically in practice.

**Algorithm 9:** Point Process MCMC

**Input:** process space $S$, number of points $n$, number of steps $T$
**Output:** realization $x_1, \ldots, x_n$.
**begin**
  Set t=0
  Sample $x = \{x_1, \ldots, x_n\}$ where each $x_j$ is i.i.d. uniformly in $S$
  **while** $t < T$ **do**
    $t \leftarrow t + 1$
    Sample $i \in \{1, \ldots, n\}$ uniformly
    Sample $a$ uniformly $\sim \mathcal{U}[0, 1]$
    Sample a point $z_i$ from a uniform distribution in $S$ and define $z = (x \setminus \{x_i\}) \cup \{z_i\}$
    **if** $a < \min\left(1, \frac{f_{\Phi,\beta}(z)}{f_{\Phi,\beta}(x)}\right)$ **then**
      $x_i \leftarrow z_i$

### 5.5.2 Perfect Sampling: Dominated Coupling from the Past Algorithm (DCFTP)

In contrast to an MCMC approximation, DCFTP guarantees an exact sample from a distribution, but is significantly more computationally expensive. The DCFTP method described in this section is based directly on [BM03, KM00], but slightly modified to use two point distances instead of one.

We first generate a dominating Markov chain $D_i,\ i = 0, -1, -2, \ldots$ in the following way:
Let $\nu_\beta$ denote the hom. Poisson point process on $S$ with rate $\beta > 0$. Then $D_0 \sim \nu_\beta$. For every subsequent step $i$:

- with probability $\beta/(\beta+n(D_i))$ make a "birth": $D_{i-1} = D_i \cup \{\eta_i\}$ where $\eta_i$ is chosen uniformly in $S$
- else make a "death": Drawn $\xi_i$ uniformly from $D_i$ and set $D_{i-1} = D_i \setminus \{\xi_i\}$.

For every step $j$ of the dominating chain $D_i$ we generate an upper process $U_j$ and lower process $L_j^i$:
Initially set $U_j^j = D_j$ and $L_j^j = \emptyset$. For $i = j+1, \dots, 0$:

- Death, i.e. $D_i = D_{i-1} \setminus \{\eta_i\}$
  $\Rightarrow U_i^j = U_{i-1}^j \setminus \{\eta_i\}$ and $L_i^j = L_{i-1}^j \setminus \{\eta_i\}$
- Birth, i.e. $D_i = D_{i-1} \cup \{\xi_i\}$

$$\Rightarrow \quad U_i^j = \begin{cases} U_{i-1}^j \cup \{\xi_i\} & \text{if } M_i \leq \prod_{\eta \in L_{i-1}^j} \Phi(d_1(\xi_i, \eta), d_2(\xi_i, \eta)) \\ U_{i-1}^j & \text{otherwise} \end{cases} \tag{5.5.1}$$

$$\text{and } L_i^j = \begin{cases} L_{i-1}^j \cup \{\xi_i\} & \text{if } M_i \leq \prod_{\eta \in U_{i-1}^j} \Phi(d_1(\xi_i, \eta), d_2(\xi_i, \eta)) \\ L_{i-1}^j & \text{otherwise} \end{cases} \tag{5.5.2}$$

where $M_i \overset{i.i.d}{\sim} \text{Uniform}[0,1]$

Set

$$T_{\min} := \begin{cases} \inf\{-i \colon D_i \cap D_0 \neq \emptyset, D_{i-1} \cap D_0 = \emptyset\} & \text{if } D_0 \neq \emptyset \\ 0 & \text{otherwise} \end{cases} \tag{5.5.3}$$

$$j_k := 2^{-k} T_{\min} \tag{5.5.4}$$

$$T := \inf\{-j_k \colon U_0^{j_k} = L_0^{j_k}\} \tag{5.5.5}$$

Then

$$U_0^{-T} \sim f_{\beta,\Phi} \tag{5.5.6}$$

In total:

- Generate $D_0 \sim \nu_\beta$ and keep backwards generating $D_i$ until you reach the time $T_{\min}$ where no original point of $D_0$ is left.
- Then repeat for $k = 0, 1, 2, \dots$ until $U_0^{j_k} = L_0^{j_k}$:
  (i) Generate backwards $D_{j_{k-1}-1}, \dots, D_{j_k}$ and generate the associated $M_i$ for deaths
  (ii) Generate forwards $(U_{j_k}^{j_k}, L_{j_k}^{j_k}), \dots, (U_0^{j_k}, L_0^{j_k})$
- Return $U^{-T} \sim f_{\beta,\Phi}$

## 5.6 Estimation of Model Parameters

### 5.6.1 Use of Summary Statistics

Looking back at equation 5.3.4 ff., to determine a Gibbs model from data we need to estimate the interaction intervals $[r_i, r_{i+1})$ as well as the interaction strength parameters $\zeta_i$. [BM03] presented an approach to estimate both simultaneously – however, in practice the estimation of only the values of $\zeta_i$ tends to be slow to converge and computationally taxing, particularly for 3D data. We thus forego the estimation of interaction intervals, instead use summary statistics to visually determine distinct changes in attractive or repulsive behaviour depending on interaction radius, set interaction intervals manually. Summary statistics are a common tool to detect anomalous spatial behavior in point processes [RRSS17] – see e.g. [MW04, LB96, GDGF10] for an introduction.

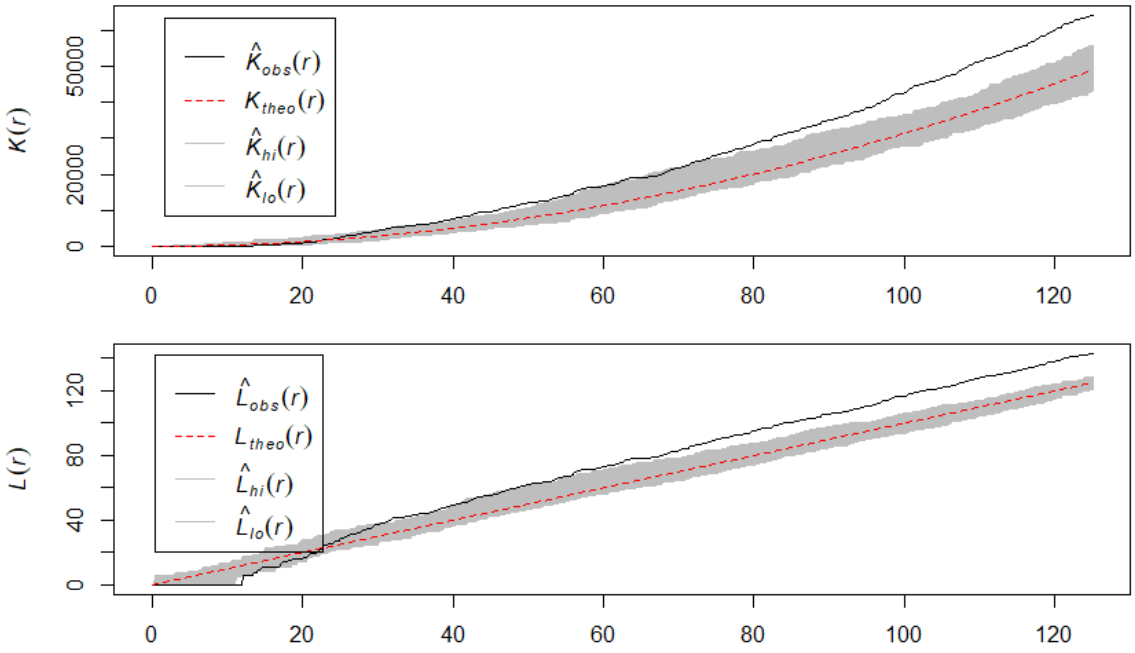



Figure 5.4: K and L summary statistics of an approximate segmentation of 74 cells; gray envelopes produced by 20 simulations

In particular we use Ripley's $K$-function (see [MW04, LB96]), estimated as

$$\hat{K}(r) = \frac{|W|}{n(n-1)} \sum_{x_i, x_j} \mathbb{1}(d(x_i, x_j) \leq r) e_{ij} \tag{5.6.1}$$

where $|W|$ is the area (or volume) of the observation window $e_{ij}$ is an edge correction factor, for which we use the standard Ripley's isotropic edge correction, see [Rip91]. The theoretical value of the $K$ function is $K(r) = \pi r^2$ ( $4/3\pi r^3$ in 3D) for a (homogeneous) Poisson process [MW04, Definition 4.4 ff.]. Although the information given by the statistic is usually seen as only a partial examination of the

underlying process, there are exceptions – for example [DGS87] derive a nonparametric estimator of an interaction function directly from the K-statistic.

The equally commonly used $L$ summary statistic, derived as

$$L(r) := \frac{\sqrt{K(r)}}{\pi} \qquad (2D) \tag{5.6.2}$$

$$L(r) := \frac{\sqrt[3]{K(r)}}{\frac{4}{3}\pi} \qquad (3D) \tag{5.6.3}$$

has a theoretical value of $L(r) = r$ in this case. To compare the behavior of a point process sample to that of a completely random Poisson process, it is common practice to plot the estimated $K$ or $L$ function of a sample of data points against both the theoretical values of the functions, as well as an envelope confined by their extreme values on a fixed number of Poisson point process samples [MW04, chapter 4.3.4.]. See figure 5.4 for a graphical example, produced from apporimately segmented cell data. The implementation used is the R *spatstat* package, see [BRT15].

### 5.6.2 Monte Carlo Root-finding

Assume that both for a given sample and in later uses for segmentation, the cardinality $n(x)$ of a realizations is fixed; We can thus for the purposes of segmentation and interaction parameter estimation simplify equation 5.4.1 by setting $n(x) \equiv n^*$. As a consequence, we can further set $\beta = 1$ without loss of generality, integrating it into a single scaling factor $c_{\beta,\zeta} = c_\zeta$.

Understanding $\zeta_{ij}$ as a vector with two indices in a slight abuse of notation, we can then express equation 5.4.1 as an exponential family in canonical form[NG09]:

$$U_\zeta(x) = \sum_{i,j} \zeta_{i,j} y_{ij}(x) = \langle \zeta, y(x) \rangle \tag{5.6.4}$$

$$y_{ij}(x) := \sum_{k<l} \mathbb{1}_{[r_i, r_{i+1})}(d_1(x_k, x_l))\ \mathbb{1}_{[s_i, s_{i+1})}(d_2(x_k, x_l)) \tag{5.6.5}$$

$$l(\zeta, x) = \log(f_\Phi(x)) = U_\zeta(x) - \log c_\zeta = \langle \zeta, y(x) \rangle - \log c_\zeta \tag{5.6.6}$$

$$\tag{5.6.7}$$

Here $l(\zeta, x)$ is the log-likelihood, $\zeta_{ij}$ are the natural parameters; $y(x)$ is the natural sufficient statistic, and $F(\zeta) = \log c_\zeta$ is the log-normalizer. By the properties of exponential families [NG09], we know that

$$\mathbb{E}_\zeta[y] = \nabla F(\zeta) \tag{5.6.8}$$

$$\mathrm{var}_\zeta[y] = \nabla^2 F(\zeta). \tag{5.6.9}$$

Consequently, for a fixed value of $x$ we can express the derivatives of the log-likelihood as

$$g(\zeta) := \nabla l(\zeta) = y(x) - \mathbb{E}_\zeta[y] \tag{5.6.10}$$

$$H(\zeta) := \nabla^T g(\zeta) = -\mathrm{var}_\zeta[y]. \tag{5.6.11}$$

Using a regular Gauss-Newton algorithm to maximize the maximum likelihood requires an update of the form

$$\zeta^{k+1} = \zeta^k - H(\zeta^k)^{-1} g(\zeta^k) \tag{5.6.12}$$

for a given parameter value $\zeta^k$.

We can replace Hessian and gradient if we know the expected values and variance of the sufficient statistic $y$ via 5.6.10 and 5.6.11. However, since we will usually be unable to calculate these values explicitly, we replace them with empirical approximations generated via a Markov Chain Monte Carlo simulation (see 9 or e.g. [MW04, chapter 7]):

$$\hat{g}(\zeta^k) = y(x) - \overline{y} \tag{5.6.13}$$

$$\hat{H}(\zeta^k) = -S_y. \tag{5.6.14}$$

With a way to approximate gradient and Hessian of the target function, we can in principle use any gradient-based optmization method. We follow the example of [HP99] and use a Levenberg-Marquardt algorithm [Mar63] in combination with these MCMC approximations, resulting in updates of the form

$$\zeta^{k+1} = \zeta^k - \hat{H}_\lambda(\zeta^k)^{-1} \hat{g}(\zeta^k), \tag{5.6.15}$$

where $H_\lambda$ is the result of multiplying the diagonal elements of the approximate Hessian $\hat{H}(\zeta^k)$ with a stabilizing factor of $1 + \lambda$; $\lambda$ starts at a fixed value $\lambda_0$ and gets multiplied (divided) by a factor of $\mu$ after each step that resulted in a lower (higher) value of the merit function

$$\chi^2 = \sum_i \hat{g}_i(\zeta^k)^2. \tag{5.6.16}$$

The idea behind, and advantage of, the use of this stabilizing factor in Levenberg-Marquardt is the dynamic interpolation between the faster Gauss-Newton and the more stable gradient descent, see also [Avr03] for details.

### 5.6.3 Smoothing Prior

One potential drawback of the Monte Carlo root-finding method is that depending on the discretization of distances, the number of points interacting in a specific way may be small, leading to small, highly varying entries of the sufficient statistic $y_{ij}(x)$. [HP99] combat this problem with a Bayesian approach: We can introduce a smoothing prior $\pi(x)$ to give a bias towards similar values for close intervals, leading to a modified log-likelihood $l(\zeta, x) + \log \pi(x)$with a gradient and Hessian of

$$g(\zeta) := \nabla \log \pi(x) + y(x) - \mathbb{E}_\zeta[y] \tag{5.6.17}$$

$$H(\zeta) := \nabla^T \nabla \log \pi(x) - \mathrm{var}_\zeta[y]. \tag{5.6.18}$$

In [HP99],the smoothing prior for the one-dimensional case was chosen in the following way:

$$Y_i \overset{i.i.d.}{\sim} \mathcal{N}(0, \sigma_i) \tag{5.6.19}$$

$$\zeta_i = \sum_{k=i}^{p} Y_i \tag{5.6.20}$$

In a similar manner, we define

$$X_i \overset{i.i.d.}{\sim} \mathcal{N}(0, \sigma_i^1) \tag{5.6.21}$$

$$Y_{i,j} \overset{i.i.d.}{\sim} \mathcal{N}(0, \sigma_{i,j}^2) \tag{5.6.22}$$

$$\zeta_{ij} = \sum_{k=i}^{p} X_i + \sum_{l=j}^{q} Y_{i,j} \tag{5.6.23}$$

In some cases, e.g. if we anticipate the interaction function to be minimal for small values and close to one for large, it may make sense to choose means that are not zero, but gradually increasing or decreasing.

One may wonder why the $X_i$ are chosen independently of $j$. However, if we assume that this were not the case, consistency dictates that:

$$\zeta_{(i-1),(j-1)} = \zeta_{i,(j-1)} + X_{(i-1),(j-1)} = \zeta_{i,j} + X_{(i-1),(j-1)} + Y_{i,(j-1)} \tag{5.6.24}$$

$$\zeta_{(i-1),(j-1)} = \zeta_{(i-1),j} + Y_{(i-1),(j-1)} = \zeta_{i,j} + X_{(i-1),j} + Y_{(i-1),(j-1)} \tag{5.6.25}$$

$$X_{(i-1),(j-1)} + Y_{i,(j-1)} = X_{(i-1),j} + Y_{(i-1),(j-1)} \tag{5.6.26}$$

$$X_{(i-1),(j-1)} = X_{(i-1),j} + Y_{(i-1),(j-1)} - Y_{i,(j-1)}. \tag{5.6.27}$$

Inductively, all $X_{i,j}$ are a function of $X_{i,p} =: X_i$ and $\{Y_{i,j} \colon j = 1, \ldots, q\}$, meaning only one family of variables may depend on both $i$ and $j$.

In practice, the optimal use and strength of a smoothing prior obviously depends on the question to which degree the assumption of a smooth transition between values of $\zeta_{ij}$ actually applies. In general, the higher the number of data points, the finer the discretization that can be meaningfully estimated (i.e. for which each component of sufficient statistic $l(\zeta, x)$ is adequately large), and the more likely the assumption of smoothness is justified. Sensible values of $\sigma_{ij}$ will however still obviously depend on the specific metric(s) measuring interaction and the specific discretization, and have to be determined empirically from case to case.

### 5.6.4 Restricting Parameters and Starting Values

In practice, one significant problem, or at least pitfall, in applying Monte Carlo root-finding, or indeed any gradient-based method with a merit function based on the sufficient statistic $y$, are the linear dependencies of its values that are present by its definition:

**Lemma 5.6.1.** *If $X_1, \ldots, X_n$ are linearly dependent, the determinant of their covariance matrix is zero.*

*Proof.* Assume without loss of generality that

$$X_n = \sum_{i=1}^{n-1} a_i X_i \tag{5.6.28}$$

Then

$$\operatorname{cov}(X_n, X_j) = \operatorname{cov}\left(\sum_{i=1}^{n-1} a_i X_i, X_j\right) = \sum_{i=1}^{n-1} \operatorname{cov}(X_i, X_j), \tag{5.6.29}$$

implying that the last row (column) of the joint covariance matrix is linearly dependent on the other rows (columns), and its determinant is zero. □

The entries of the sufficient statistic $y$ are oviously linearly dependent via $\sum y_i = n(x)(n(x) - 1)/2$, the number of pairs of points in $x$, being constant. This implies together with lemma 5.6.1 that without restricting the optimization parameters, $\operatorname{var}_\zeta[y]$ is generally not invertible. The same is true if the covariance matrix includes any $y_i$ that is constant – like values corresponding to distance intervals below the minimal distance. Any gradient based method should thus be restricted to a subset of $\{\zeta_i\}$, using the corresponding submatrix of $S_y$ and subvector of $\overline{y}$ for parameter updates.

To determine starting value of $\zeta_0$, [HP99] follow the somewhat intuitive idea of using a normalization of the sufficient statistic of the data $y(x)$,

$$\zeta_0 = \log\left(\frac{y(x)}{(K_i)}\right) \tag{5.6.30}$$

$$K_i = \frac{1}{\nu(E)} \frac{n^2}{2} \pi (r_i^2 - r_{i-1}^2). \tag{5.6.31}$$

Here, $\nu(E)$ is the measure of the area in which the point process is located, and we write division of two vectors with the meaning of element-wise division. When using a single Euclidean distance measure $d_1$, the values of $K_i$ correspond exaclty to the expected value $\mathbb{E}[y_i(x)]$ where $x$ is a Poisson point process. To apply a similar idea when using two non-Euclidean distance measures, we instead define our starting value as

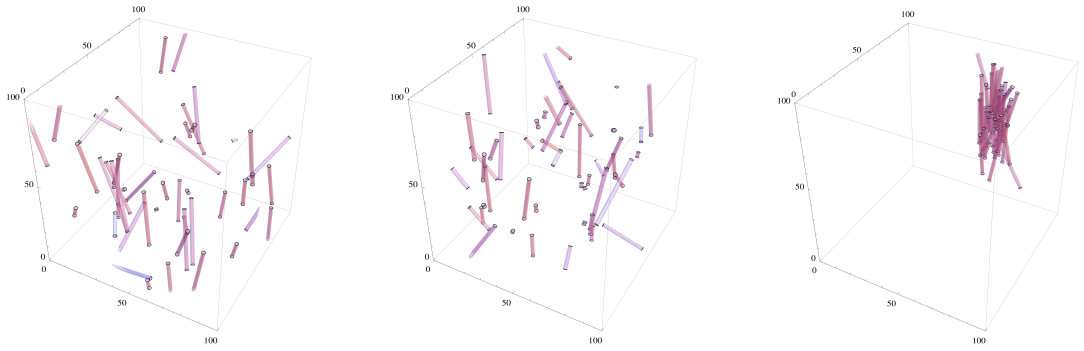
$$\zeta_0 = \frac{1}{a} \log\left(\frac{y(x)}{\overline{y}}\right), \tag{5.6.32}$$

where $\overline{y}$ is the average sufficient statistic over a Poisson-distributed MCMC chain, and $a$ is a normalizing factor chosen purely for numerical convenience to limit the initial impact of outliers in the implementation.

### 5.6.5 Stabilizing the MCMC-based Gradient

Another potential problem in using Monte Carlo root-finding in practice is that the variance of Gibbs point processes, in particular those showing attractive patterns, can be extremely high: We have found that MCMC chains of attractive point processes tend to exhibit two distinct, binary states: A “spread” pattern in which points of a realization are distributed relatively evenly over the available space, and a “clustered” pattern in which enough points have banded together that any new random point in the Markov chain that is not within the cluster is assigned a low probability through the interaction function – see figure 5.5; If a process is either repulsive or strongly attractive, it will stably converge to

a spread or clustered pattern, respectively. However for mildly attractive processes, changes between the patterns do occur, but with a very low probability. This means that the natural sufficient statistic $y(x_k)$ of the entries in the Markov chain can remain relatively stable for several hundred thousand steps or more before a switch occurs, and its values are again stable, but completely different, for a long time. As a result, the entries of $\overline{y}$ can empirically vary by $100\%$ or more even when averaging over several million steps. This obviously leads to difficulties when trying to converge to a local minimum following a gradient that is estimated via from the Markov chain; in a naive implementation, pure random fluctuation quickly overpowers the influence of the current parameter $\zeta$.



(a) "spread" pattern, $\overline{y} = (0, 8.69, 13.86, 22.14, \ldots)$

(b) "spread" pattern, $\overline{y} = (0, 8.65, 13.81, 22.14, \ldots)$

(c) "clustered" pattern, $\overline{y} = (0, 57.99, 33.53, 22.70, \ldots)$

Figure 5.5: "Spread" and "clustered" realizations of three MCMC chains after $10^6$ steps, sufficient statistic $y$ averaged over each chain. Despite the use of same parameter value $\zeta$ for all chains, the average $\overline{y}$ is wildly different for the two types of pattern.

To allow a stable convergence in a computationally feasible scope, we modify the Monte Carlo root-finding method in later applications: Instead of estimating $\mathbb{E}_\zeta[y]$ and $\mathrm{var}_\zeta[y]$ from a single Markov Chain, we average over 400 relatively short Markov chains of 10.000 steps each. To compensate for the shorter chain length, we do not initialize with a uniformly distributed (Poisson) point pattern, but always with the same point pattern – namely the data $x$ from which we wish to estimate – and use no burn-in period. Ideally, i.e. if $x$ is indeed a realization of a Gibbs density with the current value of $\zeta$, then we initialized the chain with a realization of the stationary distribution, making a burn-in unnecessary. In any other situation the chain will not necessarily reach its stationary distribution after 10.000 steps, meaning the estimations of $\mathbb{E}_\zeta[y]$ and $\mathrm{var}_\zeta[y]$ will not necessarily be close to their true values; however we found that empirically, divergence from the initialization is large enough to make a gradient-based approach feasible, while estimated values of the gradient and Hessian are stabilized enough so that random effects do not overpower the influence of the current choice of parameter $\zeta$ until it is close to its optimal value.

# Chapter 6

# Model-based Randomized Set Cover

## 6.1 Combining Gibbs Model and RSC

Recall that our overall aim is to find a subset of some given discrete segmentation candidates that best fits our model of object interaction. To minimize a Gibbs energy (or, equivalently, maximize a log-likelihood) on a discrete set of candidates $\{x_i\}$, we need to find a binary vector $z_i$ that minimizes

$$U(\{x_i\}) := z^\intercal \Theta(\{x_i\}) z, \tag{6.1.1}$$

where is $\Theta$ defined by the log-likelihood of our pairwise interaction model – see equations 5.4.1

$$\Theta_{k,l}(\{x_i\}) := \sum_{i,j} \zeta_{ij} \mathbb{1}_{[r_i, r_{i+1})}(d_1(x_k, x_l)) \mathbb{1}_{[s_j, s_{j+1})}(d_2(x_k, x_l)). \tag{6.1.2}$$

In any application that concerns segmentation of image data, we obviously want to incorporate how well each segmentation candidate fits not only the model of interaction, but the image as well; we can achieve this with the introduction of a "data term" $d_k$ weighing each single candidate, resulting in a quadratic program in the standard form of

$$U(\{x_i\}) := d^\intercal z + z^\intercal \Theta z. \tag{6.1.3}$$

In any application on larger data, a global optimization over all segmentation candidates will generally not be feasible by the size of $\Theta$ alone. One obvious solution is to use the same idea as the *Randomized Set Cover* approach, restricting our optimization to repetitions over small, random subsets. At the same time, we know that the RSC algorithm assigns a probability $p_k$ to each candidate, and updates them in such a way that the probability of optimal candidates increases in the long run (see section 2.3). We can thus forego a case-specific, manual design of a data term, and instead use the updating RSC probabilities $p_k$ as the singular term for our quadratic optimization as well. Because the RSC updates are exponential in the number of steps, we use a $\log$ scaling for the optimization:

$$d_k := -\log p_k. \tag{6.1.4}$$

Specifically, there are two obvious approaches to combining the stepwise RSC probability update with an energy minimization: one for which in each step we draw a random sample of candidates

that we use as input for both the the RSC update and the energy minimization, and one for which we update sampling probabilities *after* energy minimization, using all points that are not covered by the resulting minimizer, rather than the input. We will call these $\text{MRSC}_\text{a}$ and $\text{MRSC}_\text{b}$, respectively – see algorithm 10 and algorithm 11. The RSC update shifts weight to points that are difficult to cover with the current input; conceptually, we either find the solution to the original minimal set covering problem that best fits our interaction model (a), or try to solve a set covering problem with additional constraints (b). See figure 6.3 for an example of interaction-based segmentation: A dataset was sampled from a density that penalizes significantly different orientations for close cylinders. These cylinders where then segmented three times using $\text{MRSC}_\text{b}$ and a penalty for the same class of cylinders that was either non-existent, weak, or strong.

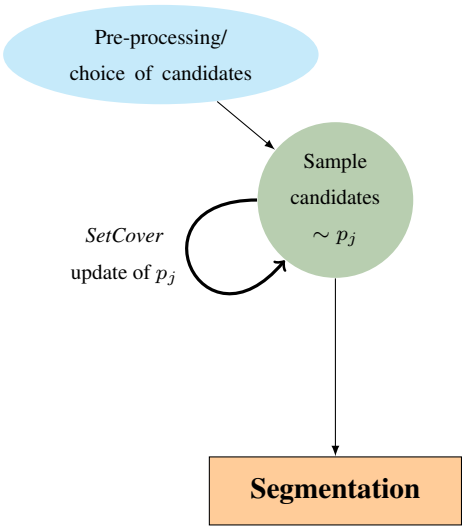



Figure 6.1: RSC

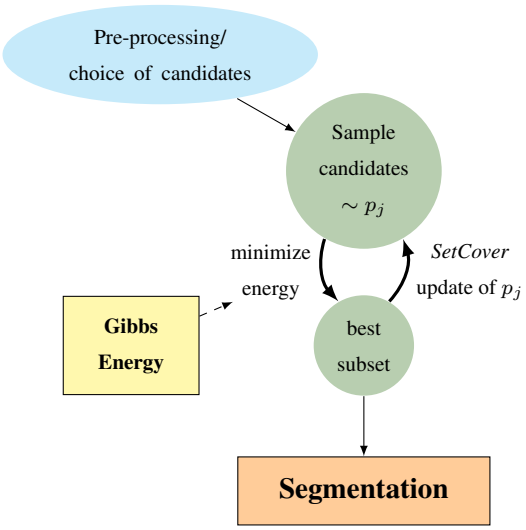



Figure 6.2: MRSC

**Algorithm 10:** Model-Based Set Covering **(a)**

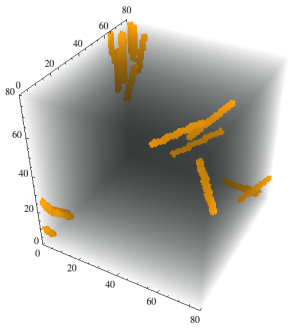
Sample $\mathcal{C} \supset \mathcal{C}_{(0)} \sim \mu_0$
**while** $\exists\, p \in P \setminus \bigcup \mathcal{C}_n$ **do**
  Sample a new set of candidates $N_{(n)} \sim \mu_n$
  Combine with old optimum: $\mathcal{C}_{(n)} = N_{(n)} \cup \mathcal{C}^*_{(n-1)}$
  Minimize energy: $\mathcal{C}^*_{(n)} = \underset{X \subseteq \mathcal{C}_{(n)}}{\operatorname{argmin}} U(X)$
  Update $\mu_n$ using the set cover probability update 12 with candidates $\mathcal{C}_{(n)}$
return $\mathcal{C}^*_{(n_{\max})}$

**Algorithm 11:** Model-Based Set Covering **(b)**

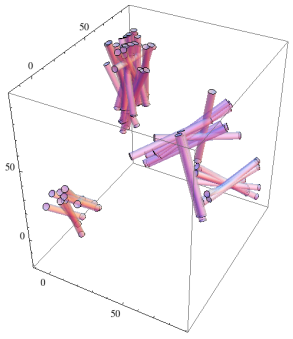
Set $\mathcal{C}^*_0 = \emptyset$.
**while** $\exists\, p \in P \setminus \bigcup \mathcal{C}^*_n$ **do**
  Sample a new set of candidates $N_{(n)} \sim \mu_n$
  Combine with old optimum: $\mathcal{C}_{(n)} = N_{(n)} \cup \mathcal{C}^*_{(n-1)}$
  Minimize energy: $\mathcal{C}^*_{(n)} = \underset{X \subseteq \mathcal{C}_{(n)}}{\operatorname{argmin}} U(X)$
  Update $\mu_n$ using the set cover probability update 12 with candidates $\mathcal{C}^*_{(n)}$
return $\mathcal{C}^*_{(n_{\max})}$

**Algorithm 12:** RSC Probability Update

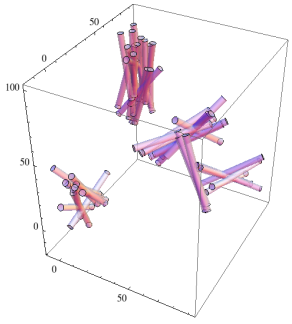
**Input:** Set of image points $P$,
collection of candidates $\mathcal{C}_n$,
a distribution $\mu_n(c)$,
parameter $\varepsilon$
**Output:** new distribution $\mu_{(n+1)}(c)$
**begin**
  set $\overline{P}_c = P \setminus \bigcup \mathcal{C}_n$
  **if** $\overline{P}_c \neq \emptyset$ **then**
    pick any point $p \in \overline{P}_c$
    **if** $\mu_n(\mathcal{C}^p) < \varepsilon$ **then**
      $\omega_{(n+1)}(c) \longleftarrow 2\mu_n(c), \quad \forall c \in \mathcal{C}^p$
      $\omega_{(n+1)}(c) \longleftarrow \mu_n(c), \quad \forall c \notin \mathcal{C}^p$
      Normalize: $\mu_{(n+1)}(c) \longleftarrow \frac{\omega_{(n+1)}(c)}{\sum_{c'} \omega_{(n+1)}(c')}, \quad \forall\, c \in \mathcal{C}$
    **else**
      $\mu_{(n+1)} \longleftarrow \mu_{(n)}$



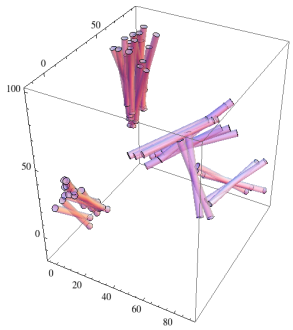
(a) Data simulated with strong interaction



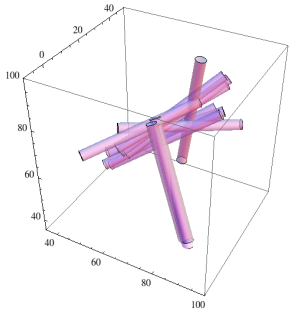
(b) Cover, no interaction



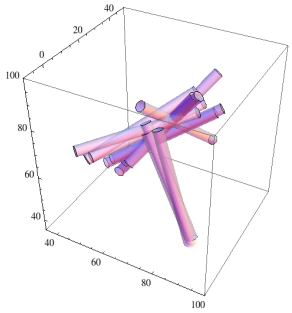
(c) Cover, weak interaction



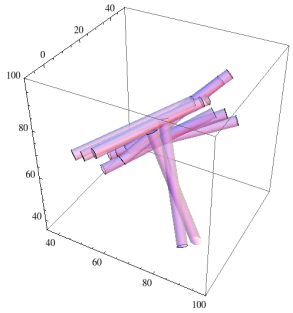
(d) Cover, strong interaction



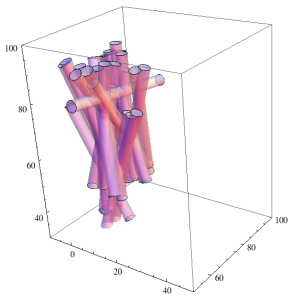
(e) No interaction



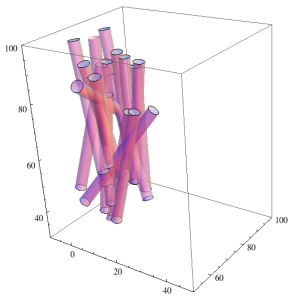
(f) Weak interaction



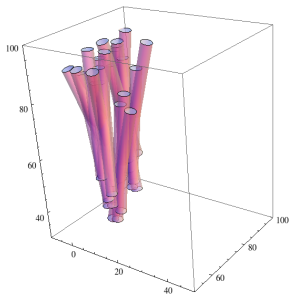
(g) Strong interaction



(h) No interaction



(i) Weak interaction



(j) Strong interaction

Figure 6.3: Graphical Example

## 6.2 Convergence

The difficulty of proving convergence is strongly dependent on which variant of the algorithm we observe. For $\text{MRSC}_\text{a}$ – algorithm 10 – the RSC update, and thus the part of the algorithm that is critical for convergence, is independent of the energy minimization, and we can re-use former results in a relatively straightforward way:

**Proposition 6.2.1.** *Let $\mathcal{R}* = \{R_1^*, \dots, R_{n^*}^*\}$ be a minimal set cover of size $n^*$.*

*If the drawn sample $N_{(n)}$ is a $\varepsilon$-net with prbability $q > 0$, then algorithm 10 must terminate after a finite number of steps.*

*Proof.* Algorithm $\text{MRSC}_\text{a}$ terminates exactly when the RSC algorithm terminates, i.e. when the randomly drawn sample $N_{(n)}$ is a full cover. Since its probability update does not depend on the result of the energy minimization, we can closely follow the proof of proposition 2.3.1 proving convergence of the original set covering algorithm.

Assume $\varepsilon = 2^{\frac{1}{an^*}} - 1$ for some $a > 1$. Further assume in each step we can find a point $p$ that is not covered by a drawn sample $N_{(n)}$ and satisfies $\mu_n(\mathcal{C}^p) < \varepsilon$, which will be true in particular if the sample is an $\varepsilon$-net, but not (yet) a full cover. Following the proof of the proposition 2.3.1, especially equations 2.3.3 and 2.3.4, we then know that in each Set Cover probability update the weight of the optimal subset $\mathcal{R}*$ must grow faster than the weight of the whole range space $\mathcal{R}$, leading to a contradiction in a finite number of steps. □

Note that as for the original RSC algorithm we can usually ensure that a drawn sample is an $\varepsilon$-net with relative ease – i.e. by random sampling a sufficient number of candidates via theorem 2.2.1.

For $\text{MRSC}_\text{b}$ – algorithm 11 – we cannot use a similar proof for two reasons: First, the local energy minimizer will not always be an $\varepsilon$-net. We can easily choose an interaction energy in such a way that any local minimum cannot contain more than one element, which will not be an $\varepsilon$-net for almost any range space. We can circumvent this problem as we we did when using the RSC algorithm with sample sizes that are smaller than theoretical limits – by empirically checking that an $\varepsilon$-light dual range for an update can be found in a sufficient percentage of steps.

The second problem is that even if the total relative weight of an optimal cover grows in each step, we cannot necessarily guarantee that it will selected as an energy minimum. However, if the interaction energy is chosen in such a way that it does not penalize an optimal cover,

$$\sum_{i \neq j} \zeta_{ij} \mathbb{1}_{[r_i, r_{i+1})}(d_1(R_i^*, R_j^*)) \mathbb{1}_{[s_j, s_{j+1})}(d_2(R_i^*, R_j^*)) = 0, \tag{6.2.1}$$

we can plausibly assume it is an increasingly likely energy minimizer: We know that the total weight of a set of optimal (minimally covering) ranges grows; If the asymptotic behavior of each single optimal range is not different from the total weight of optimal ranges, their data term becomes asymptotically large enough relative to other candidates that it is an increasingly likely part of the energy minimizer. The tendency of the RSC update to increase the weight of unlikely candidates, and decrease that of likely ones, seems to make a similar asymptotic behavior likely; however, since transitional probabilities for each update step depend on the specific local geometry of a given problem, any statements about local rather than global probabilities in weight update schemes seem to be beyond the current

state of the art. We thus restrict ourselves to empirically checking for $\varepsilon$-light dual ranges and convergence to an optimum.

## 6.3 Data Term Scaling

Even if we can ensure that a minimization of the energy 6.1.3 converges to a solution of optimal candidates asymptotically, in any application we will want to scale data and interaction term to get a set of minimizing candidates of a reasonable cardinality from the start, and in as many of the following steps as possible: if the penalties of the interaction term are very low compared to the data term, leading to a minimizer of large cardinality, the difference to a standard RSC algorithm may be negligible. One the other hand, if the penalties of the interaction term are very high compared to the data term, this will result in a minimizer of very low cardinality, leading to suboptimal partial covers for both algorithms – as well as possibly making it very difficult to update probabilities for $\text{MRSC}_\text{b}$, since we need to find a point in the complement of the small minimizer that satisfies $\mu_n(\mathcal{R}_p) < \varepsilon$, i.e. is globally covered with low probability.

Thus, in any later applications we define admissable minimal and maximal solution cardinalities $k_{\min}$ and $k_{\max}$ and use a modified data term of the form

$$d'_k := -\exp(\lambda)\left(\log p_k - \min_l(\log p_l)\right). \tag{6.3.1}$$

$\lambda$ is a scaling factor that gets increased (decreased) in each step if the cardinality of the energy minimizer falls below the minimal (above the maximal) solution cardinality until we find a suitable value via binary search. We additionally normalize the minimum data term of each set of random candidates to zero to increase stability. Note that this does not influence convergence, since any range that grows in relative weight globally will also do so relative to other members of the random sample.

# Part III

# Experimental Evaluation

# Chapter 7

# Evaluation on Synthetic Data

In our empirical evaluation of the previously introduced segmentation methods, we first compare performance on synthetic data for which we know ground truth. We distinguish approaches without an underlying model as discussed in part I of this thesis, which we compare on a relatively basic data set in section 7.2, and approaches that combine RSC with an underlying Gibbs energy as introduced in part II, which we compare on a noisy data set in section 7.3.

## 7.1 Range and Parameter Structure and Set Up

In the following chapter, the focus will be on cyclic (2D) or cylindrical (3D) ranges; we will assume a fixed radius for both of these. Thus, the space of possible ranges will generally be of the form

$$\{\text{Possible center points, i.e. image dimensions/voxels(2D/3D)}\}\times$$
$$\{\text{directions we wish to distinguish (2D)}\}\times$$
$$\{\text{cylinder lengths/circle radii we wish to distinguish}\}$$

or some subset thereof.

Cylinder lengths or circle radii will generally be uniformly discretized on an interval $[cl_{min}, cl_{max}]$;

To ensure a uniform distribution of directions in 3D, directions will be discretized on a geodesic grid – see figure 7.1.

The construction of the geodesic grid as a subdivision of an icosahedron limits the possible number of distinct directions to

$$10 * 4^k + 2, \quad k \in \mathbb{N} \tag{7.1.1}$$

on a sphere. Since direction of a cylinder is only defined up to reflection in its center point, we need only distinguish directions on a semi-sphere, limiting us to

$$2^{k+1} + 5 * 4^k + 1 \tag{7.1.2}$$

distinct directions. Note that we get slightly more than half the directions on a sphere due to wanting to include the whole "equator" of the sphere for numerical convenience. We choose $k = 3$ for a total of 337 directions on the semi-sphere in later applications.

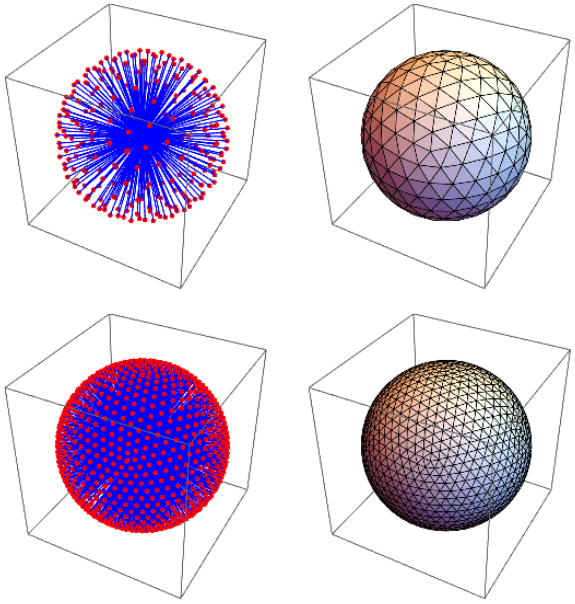


Figure 7.1: Geodesic Grid on a sphere

### 7.1.1 Area Bias

If we want to solve a minimal set cover problem for candidates of varying area, in particular if we want to use our solution as a segmentation, intuitively we wish to use a cover that consists of candidates that are both as few as possible *and* each as small as posssible – covering two small objects with one large is intuitively not a better fit.

However in the standard formulation of a minimal set cover problem, i.e. if we simply want to minimize the cardinality of a full cover, a larger set will always be a better solution than any of its subsets, as it obviously covers more points. This is reflected by the fact that for the RSC algorithm 1, the weight of a set will be doubled as least as many times as that of any of its subsets, and starting with uniform weights for any $R' \supset R$ we know that $\omega(R') \geq \omega(R)$ in all steps; thus, in any later application we need to introduce a bias to ensure that in the long run larger candidates will only be chosen for a cover by the RSC algorithm if they truly cover more image points.

Specifically, assuming uniform starting weights, the weight of each range $R$ after $n$ steps is $\omega(R) = \sum_{p \in R} 2^{n(p)}$, where $n(p)$ is the number of times each point $p$ has been chosen for a weight update. In a simplified model that simply repeatedly chooses all points sequentially for updates, we expect

$$\frac{\omega(R')}{\omega(R)} \approx 2^{\frac{n}{P}(|R'|-|R|)} \tag{7.1.3}$$

where $|R|$ is the number of image points $R$ covers, and $P$ is the number of all points – in each sequence of updating all points, $R'$ will be updated $(|R'| - |R|)$ times more. We thus modify our starting probabilities introduce with an area bias of the form $\sim m^{-|A|}$, where $|A|$ is the area of a candidate; The specific best value of $m$ is chosen heuristically in applications, though equation (7.1.3) may serve for orientation.

## 7.2 RSC Approaches without an underlying Model

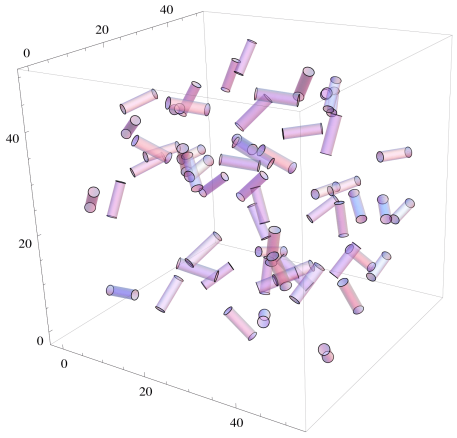



Figure 7.2: Data Set used to compare non-model versions of the RSC algorithm

We will first evaluate the approaches that do not use an underlying point process model as presented in part I of this dissertation. We choose a synthetic data volume of size $50^3$ filled with $k_{opt} = 60$ cylinders – see figure 7.2. The cardinality of the intended cover will be an input variable, as it is the main regulator in the trade-off of cover cardinality versus run time. To reduce randomness of results and calculating times, cylinder lengths will be fixed at seven pixels.

For each cover cardinality $n$, the mean over 10 repetitions will be taken to reduce randomness. We list the time taken by the algorithms, the total number of steps, the percentage of steps in which weights were updated (i.e. the $\varepsilon$-net condition was satisfied), as well as the cover cardinality after a greedy postprocessing (see section 4.1). We present values for the orginal RSC algorithm 1, two versions of our parallelized approach presented in section 3.1, an example of the hierarchical approach from section 3.2, a well as an evaluation of a probabilistic Hough transform – see section 4.6. Figure 7.3 depicts an overall graphical comparison of produced cover cardinality vs running time. The following subsections contains a more detailed discussion of each method.

### 7.2.1 RSC Algorithm

Orignal RSC as defined in algorithm 1 serves as a baseline.

| Ranges used | 701 | 561 | 422 | 279 |
|---|---|---|---|---|
| Time taken (s) | 1031 | 1187 | 1258 | 2211 |
| Steps taken | 1081 | 1141 | 1234 | 1522 |
| Percentage of update steps | 99.9 | 99.8 | 99.1 | 91.6 |
| Cardinality after greedy postprocessing | 68.9 | 65.9 | 63.7 | 60.6 |

Table 7.1: Performance of the (B/G) RSC algorithm

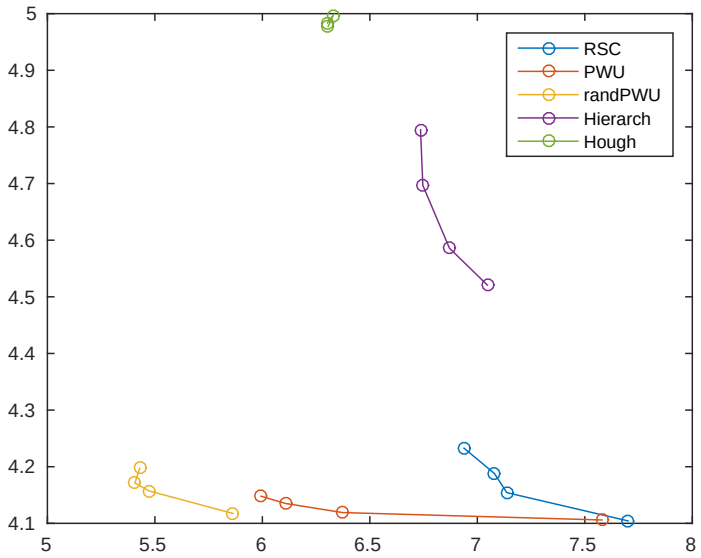



Figure 7.3: Graphical Performance Comparison: Cardinality of the cover result (y) against time taken (x), log-log-scaling (natural log). Depicted are the RSC algorithm (RSC), the approach using parallelized weight updates on a grid(PWU), the approach using parallelized weight updates on a random grid (randPWU), a hierarchical RSC approach (Hierarch) and a probabilistic Hough transform with the same number of weight updates as the RSC algorithm (Hough). The data point (6.26,6.80) of the Hough transform was not depicted, as it is too much of a (negative) outlier.

### 7.2.2 Parallelized Weight Updates

As discussed in section 3.1. Two different versions are examined: In the first version, the order in which the different grids are checked for non-emptiness is fixed: As long as there exists a point that is not covered in the first grid, we update only points in the first grid, then the next, and so forth. In the second version of the PWU algorithm, choice of update points, and by extension grid, is random. Below are the values for both versions in this order.

| Ranges used | 701 | 561 | 422 | 279 |
|---|---|---|---|---|
| Time taken (s) | 401 | 449 | 586 | 1958 |
| Steps taken | 390 | 460 | 630 | 2066 |
| Percentage of update steps | 99.2 | 95.6 | 81.8 | 31.9 |
| Cardinality after greedy postprocessing | 63.3 | 62.5 | 61.5 | 60.7 |

Table 7.2: Performance of the parallelized approach, updates on the first possible grid

As we can see, both versions are considerably faster and even supply a better quality of cover (lower cardinality) than the standard algorithm. However, the first version's tendency to pick the same points for weight-updating lowers the percentage of steps in which a point is chosen that is not already

| Ranges used | 701 | 561 | 422 | 279 |
|---|---|---|---|---|
| Time taken (s) | 228 | 223 | 238 | 352 |
| Steps taken | 223 | 232 | 255 | 361 |
| Percentage of update steps | 100.0 | 99.83 | 98.4 | 85.15 |
| Cardinality after greedy postprocessing | 66.5 | 64.8 | 63.9 | 61.4 |

Table 7.3: Performance of the parallelized approach, updates on a random grid

covered with a probability of more than $\varepsilon$. The second version is better in every aspect, especially considering that using $279$ ranges it is still faster than the first using $422$, while providing a slightly better quality. Thus, use of random grid points will be our standard way of applying PWU for further applications.

### 7.2.3 Hierarchical RSC

As discussed in section 3.2. We first generate an approximate cover using candidates that are $cl_{add} = 6$ pixels longer than the ground truth objects, then in a second phase use RSC again to find a solution using all candidates of the correct length that are subsets of the first cover.

| Ranges used | 701 | 561 | 422 | 279 |
|---|---|---|---|---|
| Total time taken (s) | 842 | 851 | 963 | 1149 |
| Time taken in first phase (s) | 835 | 843 | 955 | 1092 |
| Time taken in second phase (s) | 7.3 | 7.9 | 8.1 | 56.9 |
| Steps taken in first phase | 782 | 835 | 956 | 1135 |
| Steps taken in second phase | 896 | 1139 | 1312 | 10867 |
| Cardinality after greedy postprocessing | 120.9 | 109.6 | 98.1 | 91.9 |

Table 7.4: Performance of a Hierarchical RSC approach

Compared to the values of the original RSC algorithm, we can see the expected trade of running times for quality: While running times show an improvement over those of the original algorithm that is increasingly significant the longer both algorithms run, ranging up to a halved running time for the slowest version, the cardinality needed for a full cover via a greedy postprocessing is also shows an increase between about $100\%$ for the fastest, and $50\%$ for the slowest version. It is clearly most advantageous to use Hierarchical Set Covering when trying to solve a set cover problem using a comparatively small number of ranges, however even then the faster running times may not be worth the loss in quality.

### 7.2.4 Hough Transform

As discussed in section 4.6. Due to the form of its parameter updates, i.e. reweighting all parameters within the dual range of a point, the RSC algorithm can be seen as a variant of the generalized

Hough transform that uses randomized updates on a small subset instead of sequential updates on all image points. This makes the Probabilistic Hough Transform the most comparable version of the classical Hough transform; we compare the RSC algorithm to two versions of the Hough transform – a probabilistic Hough transform that performs the same number of steps/weight updates as the RSC algorithm, and the classical Hough transform that performs a step/weight update for each point of the image, which will usually be considerably more.

| Ranges used in original algorithm | 701 | 561 | 422 | 279 |
|---|---|---|---|---|
| Total time taken (s) | 527 | 546 | 560 | 548 |
| Percentage of points covered | 98.4 | 97.1 | 96.0 | 92.2 |
| Ranges needed for total cover | 3869 | 3884 | 3974 | 3891 |
| Cardinality after greedy postprocessing | 898.3 | 146.0 | 147.8 | 145.1 |

Table 7.5: Performance of the Probabilistic Hough transform, same number of steps/weight updates as the RSC algorithm

| | |
|---|---|
| Total time taken (s) | 550 |
| Percentage of points covered | 95.9 |
| Ranges needed for total cover | 3929 |
| Cardinality after greedy postprocessing | 145.72 |

Table 7.6: Performance of the Hough transform, using all points for weight updates

As we can see, despite updates on specific points being identical, the RSC algorithm's more intelligent choice of *where* to update via the $\varepsilon$-net condition leads to slower running times for the same number of updates, but a vast increase in quality (more than factor 2) over both a sequential update on all points (non-probabilistic Hough), and the purely random choice by the probabilistic Hough transform.

## 7.3 Model-based RSC

To examine the MRSC algorithm on a controlled data set, we simulate a small volume of cylinders with a minimal distance and a bias towards parallel orientation. Specifically, we sample a realization of cardinality $n_* = 15$ from a Gibbs density of the form

$$f_\Phi(x) := \frac{1}{c_\zeta} \exp\{U_\zeta(x)\} \tag{7.3.1}$$

$$U_\zeta(x) := \sum_{k<l} \Phi(d_1(x_k, x_l), d_2(x_k, x_l)) \tag{7.3.2}$$

$$\Phi(d_1, d_2) : = \sum_{i,j} \zeta_{ij} \mathbb{1}_{[r_i, r_{i+1})}(d_1) \mathbb{1}_{[s_j, s_{j+1})}(d_2), \tag{7.3.3}$$

where $d_1$ is the distance of the two line segments that are the middle-axis of two cylinders and $d_2$ will be the distance of normed orientation vectors modulo reflection (see section 5.4). To establish minimal distance and bias towards local parallelity, we discretize cylinder distances and relative orientation via

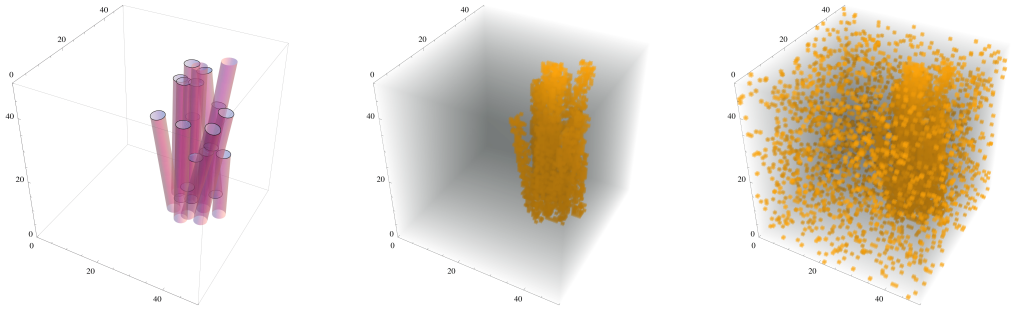
$$(r_i) = (0, 2, 10, \infty) \tag{7.3.4}$$

$$(s_j) = (0, 0.3, 1) \tag{7.3.5}$$

with a numerical bias of

$$\zeta^3_{ij} = \begin{pmatrix} -50 & 1 & 0 \\ -50 & 0 & 0 \end{pmatrix} \tag{7.3.6}$$

Here, $\zeta_{\cdot 1} = -50$ assigns an extremely small probability to all realizations in which a pair has distance $d_1$ smaller than 2 , while $\zeta_{12} = 1$ establishes a positive bias towards objects of distance smaller than 10 and similar orientation. We choose a cylinder radius of 2, meaning that for a minimal distance of 2 overlap is possible, but limited. This should reflect the fact that in real applications radius estimation is usually not perfect, and overlap cannot be completely excluded. To further increase difficulty of segmentation, we randomly delete about $50\%$ of image points of the ground truth, and add a "salt and pepper" white noise on the whole image that is 1 with $2\%$ probability – see figure 7.4.



(a) Point Process Realization/ Ideal Cover

(b) True Positives

(c) Data with Noise

Figure 7.4: Generation of Image Data with Noise

In section 6.1, we established two different versions of the MRSC algorithm: Both repeatedly minimize model energy in each step, but the $\text{MRSC}_\text{a}$ algorithm 10 performs probability updates depending on all candidates $N_{(n)}$ sampled in the current step, meaning that the probabilites produced will be extremely close to those of the original RSC algorithm 1: The only difference is that we add the energy minimizer of the previous step $\mathcal{C}^*_{(n-1)}$ to the (much more numerous) randomly sampled ranges. On the other hand, the $\text{MRSC}_\text{b}$ algorithm 11 performs probability updates based only on the energy minimizer $\mathcal{C}^*_{(n)}$ of the current step, which will usually be of much smaller cardinality – here, we randomly sample 60 cylinders, while energy minimizers are of cardinality $\approx 10$ or less.

For both versions of the MRSC algorithm, we compare optimizations with different degrees of prior knowledge: One binary version that only assumes minimal distance ($\zeta^1$), one with a bias towards minimal distance and similar orientation ($\zeta^2$), and one that uses the true density parameter $\zeta^3$ which encourages minimal distance, parallel orientation and clustering below a distance of 10.

$$\zeta_{ij}^{1} = \begin{pmatrix} -50 & 0 & 0 \\ -50 & 0 & 0 \end{pmatrix} \tag{7.3.7}$$

$$\zeta_{ij}^{2} = \begin{pmatrix} -50 & 1 & 1 \\ -50 & 0 & 0 \end{pmatrix} \tag{7.3.8}$$

$$\zeta_{ij}^{3} = \begin{pmatrix} -50 & 1 & 0 \\ -50 & 0 & 0 \end{pmatrix} \tag{7.3.9}$$

Since the energy maximization problem is in general NP-hard we approximate it using the sequential tree-reweighted message passing [Kol06] implementation of OpenGM2 [ABK12]. We set the optimal cover cardinality to the number cylinders used to produce the voxel data, $n_* = 15$; and set $\varepsilon = 2^{\frac{1}{n^*}} - 1$ (see equation 2.3.2) and randomly sample $n = 60$ cylinders for the RSC updates. Since it is not advantageous to perform an energy minimization with a completely uniform data term, we first perform $400$ steps of the original RSC algorithm and use the resulting weights $\omega_i$ as starting probabilities for all MRSC algorithms. We perform $600$ steps for each MRSC algorithm, and repeat each algorithm $10$ times to reduce randomness. To scale the data term for the non-hard-core MRSC covers (see section 6.3), we restrict energy minimizer cardinality to a minimum of $4$ and a maximum of $20$.

For each algorithm we measure the cardinality of the produced cover ("Cardinality"), the percentage of true positive voxels covered ("Covered"), as well as the ratio of true positive voxels to all voxels covered ("TP"). Since to produce our voxel data we randomly deleted about half the points of the cylindrical realization of the point process, the ideal cover/ground truth covers $100\%$ of all true positive points, but its ratio of true positive points to all is $51\%$ – the percentage of ground truth voxels remaining after random deletion.

| | Ideal Cover | hard-core ($\zeta^1$) $\text{MRSC}_\text{a}$ | Parallel ($\zeta^2$) $\text{MRSC}_\text{a}$ | True Prior ($\zeta^3$) $\text{MRSC}_\text{a}$ | hard-core ($\zeta^1$) $\text{MRSC}_\text{b}$ | Parallel ($\zeta^2$) $\text{MRSC}_\text{b}$ | True Prior ($\zeta^3$) $\text{MRSC}_\text{b}$ |
|---|---|---|---|---|---|---|---|
| Cardinality | 15 | **12.2** | 7.4 | 8.1 | 9.2 | 9.0 | 10.7 |
| Covered [%] | 100 | 32.6 | 33.5 | 34.1 | 46.3 | 46.7 | **49.3** |
| TP [%] | 51.7 | 18.8 | 30.2 | 30.6 | 34.1 | 34.8 | **35.5** |

Table 7.7: Comparison of all versions of the MRSC algorithm, averaged over 10 repititions.

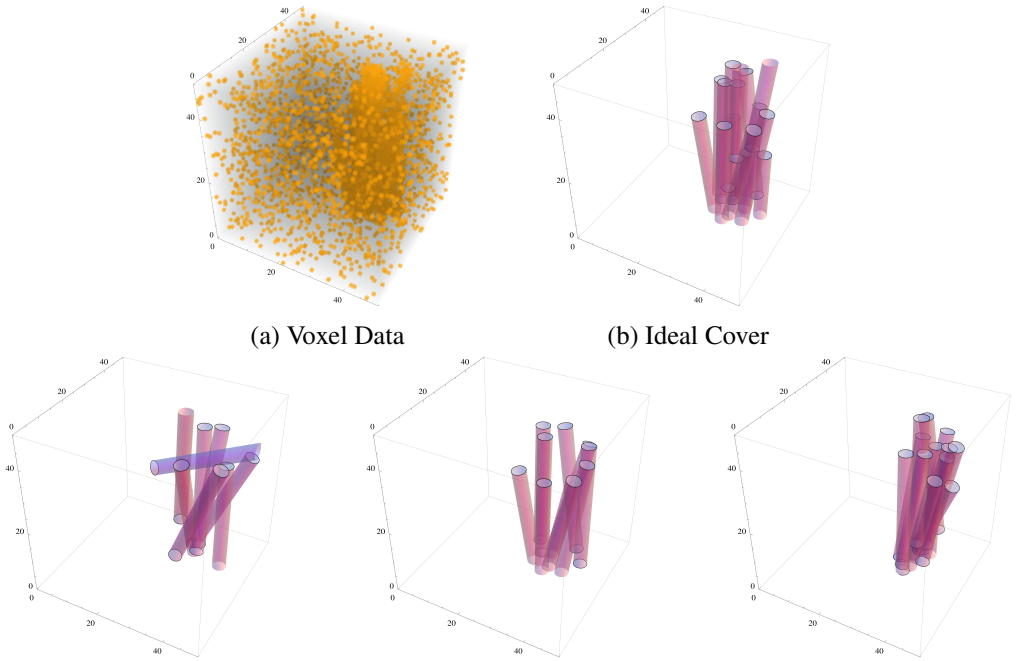
Table 7.7 depicts our measures of quality for the ground truth, as well as those produced by all versions of the MRSC algorithm. As we can see, the $\text{MRSC}_\text{b}$ algorithm that updates probabilities based on stepwise energy minimizers significantly outperforms the $\text{MRSC}_\text{a}$ for any value of $\zeta$ in almost any category. The only exception is the cover cardinality of $12.2$ produced by the hard-core $\text{MRSC}_\text{a}$ algorithm, which is closest to the true value of $15$. However since the sampled candidates are clearly far worse matches for the data, as indicated by their lowest coverage and ratio of true positives, this can hardly be seen as an overall improvement. This means we can safely assume that $\text{MRSC}_\text{b}$ outperforms a vanilla RSC algorithm with a singular model energy optimization as a postprocessing step as well, since the underlying probability updates of RSC are nearly identical to those of $\text{MRSC}_\text{a}$, while repeatedly searching for optima will outperform a single minimization. At the same time, as we would expect intuitively, the quality of the cover improves with the amount of prior knowledge for

both versions of the MRSC algorithm. We thus use the $\text{MRSC}_\text{b}$ type algorithm for applications on real data in the following sections 9.1 and 9.2. See figure 7.5 for graphical examples of the $\text{MRSC}_\text{b}$-covers with different degrees of prior knowledge.

Optimal values may depend on the goal of the segmentation: For example, if we lower the hard-core penalty to $\zeta_{1\cdot} = -10$, and set the minimum cardinality of a non-binary cover to $8$ instead of $4$ for the same data set, we achieve coverage of a higher number of data points, at the price a lower percentage of true positives in the cover – see table 7.8. Relative performance of the algorithms remains almost exactly the same as before.

| | Ideal Cover | hard-core ($\zeta^1$) $\text{MRSC}_\text{a}$ | Parallel ($\zeta^2$) $\text{MRSC}_\text{a}$ | True Prior ($\zeta^3$) $\text{MRSC}_\text{a}$ | hard-core ($\zeta^1$) $\text{MRSC}_\text{b}$ | Parallel ($\zeta^2$) $\text{MRSC}_\text{b}$ | True Prior ($\zeta^3$) $\text{MRSC}_\text{b}$ |
|---|---|---|---|---|---|---|---|
| Cardinality | 15 | 12.1 | 12.7 | 12.0 | 9.3 | 13.3 | **14.6** |
| Covered [%] | 100 | 34.1 | 43.3 | 46.1 | 40.0 | 54.4 | **56.0** |
| TP [%] | 51.7 | 19.8 | 25.41 | 28.9 | 29.8 | **32.4** | 32.1 |

Table 7.8: Comparison of all versions of the MRSC algorithm, higher minimal cover, averaged over 10 repititions.



(a) Voxel Data (b) Ideal Cover

(c) Hard-core Cover (d) Hard-core and Parallel Cover (e) True Prior Cover (hard-core, parallel, clustered)

Figure 7.5: Synthetic Data and Covers of the MRSC algorithms

# Chapter 8

# Pre- and Postprocessing Methods for Real Data

In contrast to the controlled environment of synthetic data, an algorithm that searches for a full cover of all image points can usually not be naively applied to a real data set without risking significant oversegmentation thanks to image noise. At the same time, our range space – as introduced in chapter 7.1 – can be of siginificant size even for smaller data sets if it includes any possible combination of location, direction and length of an object. Any method that can quickly and safely eliminate voxels which are likely noise, as well as unlikely object parameters, can lead to a significant speedup and improvement of any parameter-based approach for a relatively minor cost.

Lastly, even correctly recognized real objects will rarely perfectly fit the pre-defined shape of ranges used by all set cover-based approaches. Particularly organic cell shapes tend to only roughly match any parameterizable object, making additional post-processing based on an approximated minimal cover an option that can lead to a siginificant improvement of the segmentation with a minor cost. Thus, we use the following chapter to briefly discuss a selection of pre- and postprocessing methods we use for real data sets before moving on to an empirical evaluation.

## 8.1 Preprocessing Methods

Filter functions offer a quick and intuitive way to exclude certain parameters based on image data and assumptions of noise structure. In later applications, we focus on both the exclusion of spatial as well as directional paramters for cylindrical structures, and, in the former case, approximately cyclic shapes.

Unlikely centers of shapes can be relatively easily excluded using isotropic Gaussian filters; the exclusion of unlikely directions is slightly more complicated:

Although the convolution with a directed filter mask will supply information on the local direction of the neighbourhood of a voxel, this method scales badly with the number of directions we wish to distinguish. In addition, it does not use the information that in later applications most voxels will be part of relatively thin, cylindrical shapes with a single direction.

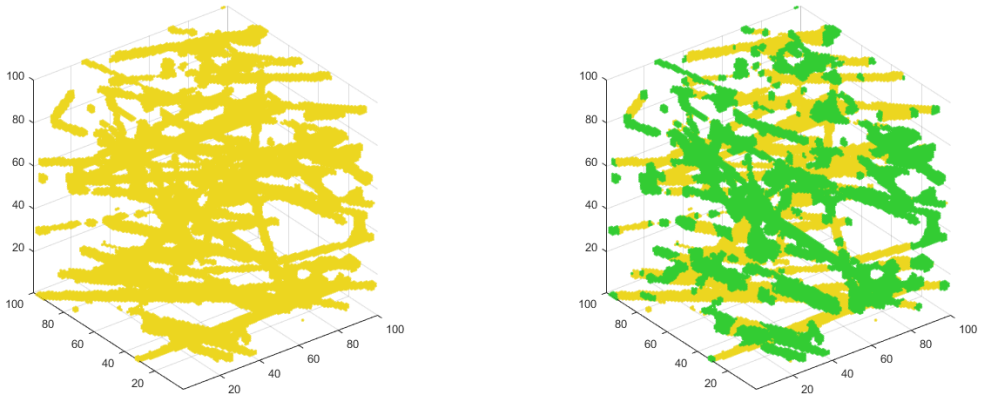



Figure 8.1: Data points (left, yellow) and an overlay with possible centers of fibers in one direction (right, green). The result of a conservative filtering based on local directions generally includes points of fibers pointing roughly in the chosen direction, as well as image points for which direction is difficult to determine, such as fiber end points.

For these thin fiber structures, the smallest eigenvalue of the structure tensor is expected to be significantly smaller than the other two, and its corresponding eigenvector indicates their main direction [FL03].

Thus, in later detection of cylindrical shapes, we will calculate the local structure tensor for each voxel. For the discrete approximation of local derivations, we use the 3D Scharr operator because of its relatively isotropic nature[SW00]. We then check if the smallest eigenvalue is indeed significantly smaller than the other two and limit possible directional parameters to a neighbourhood of the corresponding eigenvector. Both the value of the threshhold indicating "significantly smaller" and the size of the neighbourhood are determined empirically and chosen conservatively – see figure 8.1.

The following section, in particular example 8.1.1, may be useful to quantify cutoff values if the random variables describing image noise are approximately independently normally distributed, or follow any other distribution for which we can easily determine the quantiles of a linear combination of said variables.

**Definition 5** (Filter Response function)**.** Given

(i) a finite set $P \subset \mathbb{R}^l$ that will represent our picture,

(ii) a (nonnegative) greyscale function on this picture $g : P \to \mathbb{R}$,

(iii) and a (nonnegative) filter function $f : P \to \mathbb{R}$

we can define a *filter response function*

$$\mathscr{F}(g, f) := \sum_{x \in P} g(x) f(x). \tag{8.1.1}$$

In the context of set covering, it will often be reasonable to assume that the filter function is part of a family of functions that depend on a parameter $t \in T \subseteq \mathbb{R}^m$: $f(x) = F(x,t)$;

Similarly, we can often safely assume that assume that the greyscale function of the image $g(x)$ is of the form $G(x,u_1) + \ldots + G(x,u_{k_{opt}}), u_i \in U \subseteq \mathbb{R}^n$, where each component $G(x,u_i) \geq 0$ represents one of the shapes that make up the image.

For example, in the context of $2D$ cylinders, $u_i$ could be defined as a starting point, an angle, and possibly the length and/or width of the cylinder, while $G(x,u_i)$ is the indicator function of the cylinder defined by these parameters.

In this context, we define, with a slight abuse of notation:

$$\mathscr{F}(g,t) := \sum_{y \in P} g(y)F(y,t) \tag{8.1.2}$$

$$\mathscr{F}(u,t) := \sum_{y \in P} G(y,u)F(y,t) \tag{8.1.3}$$

**Remark 8.1.1.** Note that $\mathscr{F}$ as a function of either

- $g$ and $f$ or
- $g$ and $F(\cdot,t)$ for a fixed value of $t$ or
- $G(\cdot,u)$ and $F(\cdot,t)$ for fixed values of $u$ and $t$

is symmetric, linear and positive definite, and thus a valid inner product.

In particular, the *Cauchy-Schwarz inequality* applies: for an image composed of objects in the form of the filter mask, the filter response will be maximal for the true function or parameter.

In the context of image analysis, we can thus see the filter response function as a measure of the similarity of the filter mask described by the function $F(\cdot,t)$ and the image itself in the area $B := \{y\colon F(y,t) > 0\}$.

For example, if we know there will be certain shapes in the image, but we do now know their exact position, we can choose a filter mask in the form of this shape; by Cauchy-Schwarz, a normed filter response at a certain point $x$ can be viewed as the probability of the shape being located there.

If we assume that our image is made of certain shapes disturbed by a random noise, the following proposition is handy:

**Proposition 6.** *Let the greyscale function be of the form $h = g + N$, where $g$ is a fixed function and $N$ is a random noise that is assigned to every position $y \in P$ in an i.i.d. way;*

*Let $B \subseteq P$ be defined as $B := \{y\colon f(y) > 0\}$;*

*Let $Q_{fN}$ be the cumulative distribution function of $S_{fN} := \sum_{y \in B} f(y)N(y)$, and $q_1$ and $q_2$ be the $\alpha$- and $(1-\alpha)$-quantiles of $Q_{fN}$.*

*Define $S_g := \sum_{y \in B} g(y)$;*

*Then for any $x \in P$ and $t \in T$:*

$$\mathrm{P}(\sum_{y \in B} f(y)N(y) \in [q_1, q_2]) \geq 1 - 2\alpha \tag{8.1.4}$$

$$\mathrm{P}(\mathscr{F}(h,f) \in [\mathscr{F}(g,f) + q_1, \mathscr{F}(g,f) + q_2]) \geq 1 - 2\alpha \tag{8.1.5}$$

*And in the one-sided version:*

$$\mathrm{P}(\sum_{y\in B} f(y)N(y) \geq q_1) \geq 1-\alpha \tag{8.1.6}$$

$$\mathrm{P}(\mathscr{F}(h,f) \geq \mathscr{F}(g,f) + q_1) \geq 1-\alpha \tag{8.1.7}$$

*Proof.* Equations 8.1.4 and 8.1.6 are a direct consequence of the definition of a quantile; equations 8.1.5 and 8.1.7 follow directly using the definition of the filter response function $\mathscr{F}$. □

**Example 8.1.1.** If we assume that $N(y) \sim_{i.i.d.} \mathcal{N}(\mu,\sigma^2)$, then $f(y)N(y) \sim \mathcal{N}(f(y)\mu, f(y)^2\sigma^2)$ independently and thus $S_{fN} = \sum_{y\in B} f(y)N(y) \sim \mathcal{N}(S_f\mu, S_{f^2}\sigma^2)$, where $S_f := \sum_{y\in B} f(y)$ and $S_{f^2} := \sum_{y\in B} f(y)^2$;

Then we can write $S_{fN} = \sqrt{\sigma^2 S_{f^2}} X_0 + \mu S_f$ where $X_0 \sim \mathcal{N}(0,1)$;

Consequently, if $q_0$ is the $\alpha$-quantile of the standard normal distribution, $-q_0$ is its $1-\alpha$-quantile, and we can write

$$q_1 = \sqrt{S_{f^2}}\,\sigma * q_0 + \mu\, S_f \tag{8.1.8}$$

$$q_2 = -\sqrt{S_{f^2}}\,\sigma * q_0 + \mu\, S_f \tag{8.1.9}$$

as the $\alpha$- and $1-\alpha$-quantiles of $Q_{fN}$.

For practical purposes, it will normally make sense not to try out every possible filter parameter $t$. However, if we choose a filter value $t_1$, and our computation produces a certain filter response value $\mathscr{F}_1$, we can still assume that if

$$g(x) = G(x,u_1) + \ldots + G(x,u_{k_{opt}}) \tag{8.1.10}$$

and $u \in \{u_1,\ldots,u_{k_{opt}}\}$, then

$$\mathscr{F}_1 = \mathscr{F}(g,t_1) = \sum_i \mathscr{F}(u_i,t_1) \geq \mathscr{F}(u,t_1). \tag{8.1.11}$$

In other words, all parameters $u'$ that would produce response values $\mathscr{F}(u',t_1) > \mathscr{F}_1$ cannot be true parameters, and the space of valid parameters is narrowed down to

$$U_1 = \{u\colon \mathscr{F}(u,t_1) \leq \mathscr{F}_1\}. \tag{8.1.12}$$

**Example 8.1.2.** In the case of two-dimensional cylinders of a fixed length $L$ and sidelength $2r$, we could characterize a cylinder using one point $y$ and an angle $\alpha$, for example:

$$G(\cdot,(x,\alpha)) = \mathbb{1}_A,\ A = \left\{x + \vec{v}a + \vec{w}b \mid a\in[0,L], b\in[-r,r], \vec{v} = \begin{pmatrix}\cos\alpha\\ \sin\alpha\end{pmatrix}, \vec{w} = \begin{pmatrix}\sin\alpha\\ -\cos\alpha\end{pmatrix}\right\} \tag{8.1.13}$$

Similarly, to quantify response to a cylindrical filter mask of "length" $2L_f$, "width" $2W_f$, and a main direction defined by an angle $\beta$ , we can use a function of the form

$$F(y,(u,\beta)) = \exp\left(-\left|\begin{pmatrix}\frac{1}{L_f} & 0\\ 0 & \frac{1}{W_f}\end{pmatrix}\times\begin{pmatrix}\cos\beta & -\sin\beta\\ \sin\beta & \cos\beta\end{pmatrix}\times\begin{pmatrix}(y_0-u_0)\\ (y_1-u_1)\end{pmatrix}\right|^{10}\right) \tag{8.1.14}$$

$$= \exp\left(-\left|\begin{pmatrix}\frac{1}{L_f}\left(\cos\beta*(y_0-u_0)-\sin\beta(y_0-u_0)\right)\\ \frac{1}{W_f}\left(\sin\beta*(y_1-u_1)+\cos\beta(y_1-u_1)\right)\end{pmatrix}\right|^{10}\right) \tag{8.1.15}$$

Note that this filter function equally penalizes a distance of $L_f$ in the direction of its main axis, and a distance of $W_f$ in the orthogal direction. The exponent of 10 is an arbitrary parameter to control the strength of the penalty, and can be chosen by empirical convenience, though it should be even to guarantee symmetry.

In the context of set covering, we can use filters to apply a prior weighting of ranges before applying the set covering algorithm 1:

Recall that in the original algorithm we start with a uniform distribution on the ranges $\{R\}$, and are required to have some prior knowledge of the shapes we wish to cover, up to some parameters $u \in U \subseteq \mathbb{R}^n$.

Using the idea above, we exclude certain values of $u$, and thus certain ranges, possibly speeding up calculation time of the algorithm. In certain situations, e.g. if $F = G$ and a large selection of filter parameters $t_j = u_j$ is tested, we may also be able to choose a non-uniform distribution dependent on the calculated filter response values for the remaining ranges.

If we assume that the original image, is perturbed by a random noise $N(x)$, i.e. $h = g + N$, we can use proposition 6 to achieve $\alpha$-certainty for correctly excluding a range:

Again, if we assume that a certain value $u$ is a true value, i.e.

$$g(x) = G(x,u_1) + \ldots + G(x,u_{k_{opt}}) \tag{8.1.16}$$

and $u \in \left\{u_i,\ldots,u_{k_{opt}}\right\}$, then, writing $f_1 = F(\cdot,t_1)$ and $B = \{f_1 > 0\}$,

$$\mathscr{F}_1 = \mathscr{F}(h,t_1) = \mathscr{F}(g,t_1) + S_{f_1N} \tag{8.1.17}$$

$$= \sum_i \mathscr{F}(u_i,t_1) + S_{f_1N} \tag{8.1.18}$$

$$\geq \mathscr{F}(u,t_1) + S_{f_1N} \tag{8.1.19}$$

$$\geq \mathscr{F}(u,t_1) + q_1, \tag{8.1.20}$$

with probability $\geq 1-\alpha$, where $q_1$ is the $\alpha$-quantile of $S_{f_1N} = \sum_{y\in B} f_1(y)N(y)$.

Thus, we can exclude all values $u'$ that produce response values $\mathscr{F}(u',t_1) > \mathscr{F}_1 - q_1$ with high probability, narrowing the space of valid parameter values down to

$$U_1 = \left\{u\colon \mathscr{F}(u,t_1) \leq \mathscr{F}_1 - q_1\right\}. \tag{8.1.21}$$

## 8.2 Data-specific Postprocessing

While in the application of fiber segmentation the shape of a graphite or glass fiber will be close to that of a long, thin cylinder, the specific shape of grown cells can be somewhat more unpredictable; we thus briefly present two methods that can be used to better match cyclic candidates produced by an RSC algorithm to cell image data.

### 8.2.1 Estimating Linear Transformation of Circles

Both the RSC and MRSC methods work to select the best candidates from a discretized parameter space. In the case of cell images, the obvious simplest parametrization of possible cell candidates is cyclic, i.e. the combination a central point and radius. An equally intuitive first refinement of this parameterization is the more flexible elliptic shape. Using such candidates directly will however already significantly increase the parameter space, and thus running time and hardware requirements, of the algorithms: If an ellipse is defined via directional vector and central point that are discretized as finely as the cyclic centers and radius are, the size of the parameter space is squared. It can thus be easier, or even necessary, to work with cylcic candidates and refine them in a second step.

The first method we present is based on the observation that an ellipse constitutes a linear transformation of a circle. In other words, given a function $g(x)$ describing the gray values of the image of a (filled) circle, and a function $h(x)$ describing those of an ellipse, we can find a matrix $A$ and vector $c$ such that

$$h(x) = g(\tilde{A}\mathbf{x} + c). \tag{8.2.1}$$

If a matrix is quadratic, real, and has a positive determinant, which we can ensure by restriction later, singular value decomposition tells us it is of the form $R_1 * S * R_2$, where $R_1$ and $R_2$ are rotary matrices and $S$ is a scaling matrix. In the case of the transformation from circles to ellipses, one of the rotary matrices is obviously superfluous, reducing the number parameters that need to be estimated compared to a general matrix, and supplying an easy way to produce additional elliptic parameters (rotiation, two axes) from circular ones.

To determine $\tilde{A}$ and $c$, Hagege et al.[HF10] proposed a method for estimating the linear transformation of an image:

Starting with equation 8.2.1 , we can write

$$\tilde{y} = \tilde{A}x + c \Leftrightarrow x = \tilde{A}^{-1}\tilde{y} + b \Leftrightarrow x = Ay \tag{8.2.2}$$

where $b := -\tilde{A}^{-1}c$, $y := [1, \tilde{y}_1, \ldots, \tilde{y}_n]'$ and $A := [b|\tilde{A}^{-1}]$.

Their method is based on choosing a fitting set of functions $\{\mathbf{w_l}\}$ and using the relation

$$\int_{\mathbb{R}^n} x\mathbf{w_l}(h(x)) = \left|A^{-1}\right| A^{-1} \int_{\mathbb{R}^n} y\, \mathbf{w_l}(g(y)) + \left|A^{-1}\right| A^{-1}c \int_{\mathbb{R}^n} y\, \mathbf{w_l}(g(y)) \tag{8.2.3}$$

to produce a set of equations

$$\left|\tilde{A}\right| [\int_{\mathbb{R}^n} x\, \mathbf{w_1}(h(x)) \ldots \int_{\mathbb{R}^n} x\mathbf{w_p}\,(h(x))] = A[\int_{\mathbb{R}^n} \tilde{y}\, \mathbf{w_1}(g(y)) \ldots \int_{\mathbb{R}^n} \tilde{y}\, \mathbf{w_p}(g(y))], \tag{8.2.4}$$

If we define

$$\tilde{G}_p := [\int_{\mathbb{R}^n} \tilde{y}\, \mathbf{w_1}(g(y)) \dots \int_{\mathbb{R}^n} \tilde{y}\, \mathbf{w_p}\,(g(y))] \tag{8.2.5}$$

$$\tilde{H}_p := [\int_{\mathbb{R}^n} x\, \mathbf{w_1}\,(h(x)) \dots \int_{\mathbb{R}^n} x\, \mathbf{w_p}(h(x))] \tag{8.2.6}$$

this becomes

$$\left|\tilde{A}\right| \tilde{H}_p = A\, \tilde{G}_p \tag{8.2.7}$$

and can be solved for $A$ if $\tilde{G}_p$ is of full rank.

However, recall that the proposed application of the method is to estimate an approximate linear transformation between the (binary) image of a cyclic candidate and possibly equally binary image data. In this case, i.e. $\text{Image}(g) = \{0, 1\}$, the set of functions $\mathbf{w_p} : \{0, 1\} \to \mathbb{R}$ is obviously too limited to generally produce a matrix $\tilde{G}_p$ of full rank.

To include the application on binary images we can use a modified approach based on [TLSK15] and [DKF08]: use equations of the form

$$|A| \int_{F_t} w(x) = \int_{F_o} w(A^{-1}y), \tag{8.2.8}$$

where $F_t$ and $F_o$ are the support of template and output, with functions $w(x)_{p_1,\dots,p_n} := x_1^{p_1} \dots x_n^{p_n}$, resulting in a set of equations

$$\left|\tilde{A}\right| \int_{F_t} x_k^p = \sum_{i_1=0}^{p} \binom{p}{i_1} \sum_{i_2=0}^{i_1} \binom{i_1}{i_2} \dots \sum_{i_n=0}^{i_{n-1}} \binom{i_{n-1}}{i_n} q_{k1}^{p-i_1} \dots q_{kn}^{i_{n-1}-i_n} q_{k(n+1)}^{i_n} \int_{F_o} y_1^{p-i_1} y_2^{i_1-i_2} \dots y_n^{i_{n-1}-i_n}, \tag{8.2.9}$$

where $q_{ij} := (A)_{ij}$. To estimate the parameters $q_{ij}$ in a noisy case, generate the equations to a high enough order, and minimize the quadratic difference between left and right side as a function of the parameters. We can restrict this minimization to ensure we can apply singular value decomposition to the results. This method can be thought of as a matching of the theoretically predicted and empirical moments of the grayscale values of the image. See also [BF11] for a generalized approach.

### Graphical Example

See figure 8.3 for a graphical example. Cell data (black) is first approximately segmented using model-based set covering (MRSC) with cyclic candidates/ranges. In practice, using the elliptical estimation as a form of post-processing does in principle allow for a better matching segmentation without the use of the much higher-dimensional parameter space needed to directly segment a picture using ellipsoid candidates for MRSC. However, even though translations should theoretically be included in this form of postprocessing, it empirically works best only if the cyclic segmentation is very good,

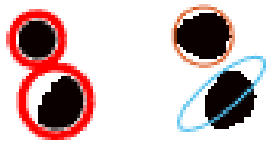


Figure 8.2: Good and bad ellipsoid matching in figure 8.3

i.e. the centers of the cyclic candidates closely match real object centers. It should thus be used with caution. See the smaller image section in figure 8.2 for an example.

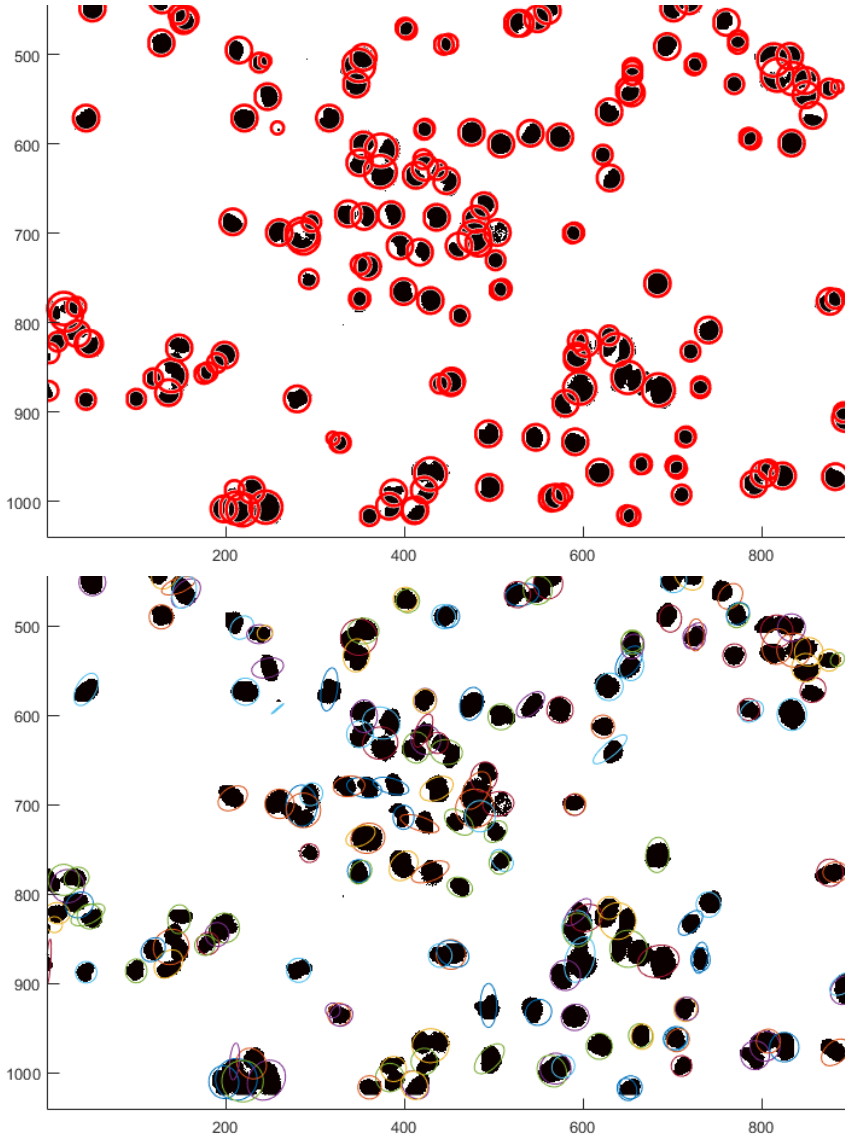



Figure 8.3: Approximate segmentation using cylcic candidates (top) and estimated ellipsoids (bottom).

### 8.2.2 Seeded Watershed Transformation

If image data can be binarized with relative ease, a seeded watershed transformation ([Soi13, chapter 9]) can provide a somewhat more flexible and stable alternative to the ellipsoid transformation described in the previous section. For this method of postprocessing the centers of the candidates output by the MRSC algorithm 11 are used as seeds for the watershed transformation on the binarised image. The result is obviously more flexible in shape compared to ellipsoid candidates; it does however depend on a good binarization of the image; this is especially true since the seeded watershed should preferably be applied to the connected components of the image seperately, as otherwise the influence of close seeds may lead to oversegmentation – see figure 8.4. It also does not take gray values into account, whereas the method of linear transformation explicitly can. Which method is preferably must thus be decided on a case-by-case basis, though the degree of gray scale fluctuation in the image may be a good indicator; for the application given in 9.1, a good binarization of the cell images could be achieved with relative ease via global thresholding, making the seeded watershed method overall preferable to an estimation of ellipsoid candidates via linear transformation – see figure 8.5.

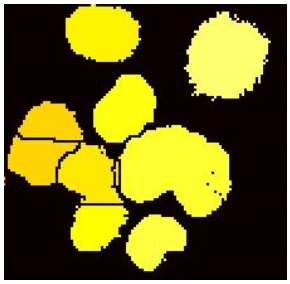
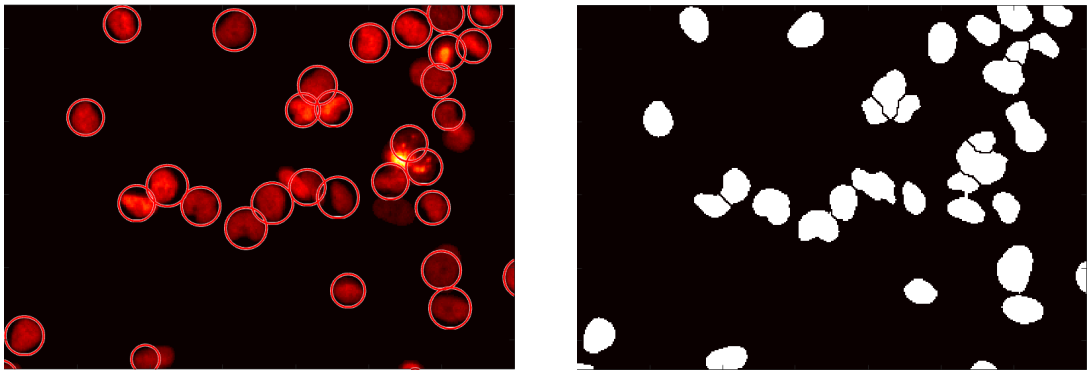


Figure 8.4: Oversegmentation for close neighbours



(a) Approximate segmentation using circular candidates

(b) Segmentation result after seeded watershed transform

Figure 8.5: Seeded watershed transform for postprocessing

# Chapter 9

# Application on Real Data

## 9.1 2D Cell Data

In a first example of applications of the MRSC algorithm on real data, we perform a segmentation of several 2D cell images; one relatively recent method of cell segmentation that is comparable to our MRSC approach is the one by Soubies/Poulain et al.[SWD15, PPSD15]: Both methods iteratively sample a random set of discrete segmentation candidates, and find the subset that best matches a Gibbs model that combines image data and object interaction via energy minimization. The main difference is that the Soubies/Poulain method samples candidates uniformly and uses a static data term for its energy model; in contrast, our MRSC approach uses the probabilities dynamically generated by the RSC update 12 as both sampling probabilities and data term. We compare both methods on several images of cell data with varying difficulty of segmentation. For further details see the paper by Markowsky et al.[MRZ+17] that this section is partially based on.

### 9.1.1 Interaction Model and Input Parameters

The Soubies/Poulain method uses a binary interaction model that punishes candidate overlap that exceeds a previously defined maximum. To establish a direct comparison, we likewise use a binary model based on a single distance for the MRSC algorithm; in our notation, interaction is defined for both methods via

$$U_\zeta(x) := \sum_k D(x_k) + \sum_{k<l} \Phi(d_1(x_k, x_l)) \tag{9.1.1}$$

$$\Phi(d_1) : = \sum_{i,j} \zeta_i \mathbb{1}_{[r_i, r_{i+1})}(d_1). \tag{9.1.2}$$

$$(r_i) = (-\infty, 0, \infty) \tag{9.1.3}$$

$$\zeta_i = (-\infty, 0). \tag{9.1.4}$$

where $D$ is the data term (dependent on the method) and any point $x_i$ of the point process represents a cyclical candidate defined by a center $p_i$ and radius $r_i$; we choose the 9 possible radii $\mathcal{R} = \{7, 8, \ldots, 15\}$ based on an inspection of typical cell sizes. In the case of cells with strongly

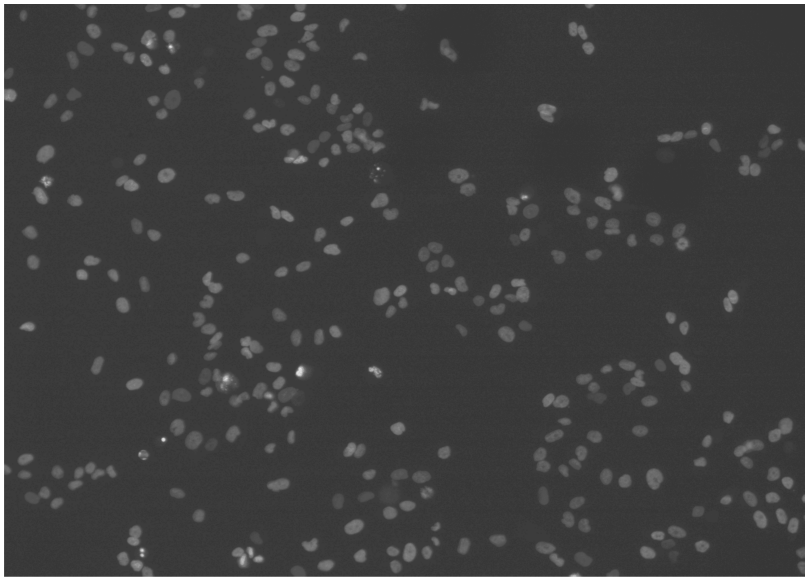


Figure 9.1: Example of a Cell Data Set

varying radii, it makes sense to make maximum admissable overlap dependent on candidate radius. We thus define our distance as a slight modification of the Euclidean distance $\|\cdot\|$:

$$d(p_i, p_j, r_i, r_j) = \|p_i - p_j\| - 0.75 * (ri + rj). \tag{9.1.5}$$

We use an isotropic Gaussian filter with conservative thresholding to eliminate all extremely improbable candidate centers as a pre-processing step for all algorithms (see section 8.1); Since this is a case in which object diameter is significantly smaller than image size, we use the parallelized approach of section 3.1 with a grid to speed up running times of RSC updates.

As a post-processing step, we perform a marker-controlled watershed segmentation on the thresholded image, using the centres of segmented candidates as seeds, to improve the local matching of candidate shape to image data (see section 8.2.2). Since the seeded watershed is prone to oversegmentation, we apply the algorithm locally on the connected components of the thresholded image, and ignore connected components of the result that are smaller than 5 pixels. We estimate the number $k$ of cells as a multiple of the connected components of the thresholded cell image: $1.4$ for easy and medium images, and $2.4$ for hard ones. Furthermore, we heuristically set the sample size $||\mathcal{C}_{(n)}|| = 3k$ (see algorithm 11) , and the starting penalty for candidates of radius $r$ to $q(r) = 4^{-r^2}$ (see the last paragraph of section 7.1.1).

### 9.1.2 Data and Performance Measures

We evaluate three different approaches: First, our proposed approach, referred to as MRSC; its starting probabilities are proportional to the output of the filter-based pre-processing multiplied by $q(r)$. Second, a version of the algorithm that uses static uniform sampling probabilities and the static data

term for circles proposed in [Des16] (MBC) with parameter $d_0 = 0.1$, and third, an approach that uses the data term of MBC multiplied by $q(r)$ as starting probabilities, but is otherwise identical to the first version (MRSC/MBC). Note that the second approach is similar to the Multiple Birth and Cut algorithm proposed in [PPSD15], with the difference of using a slightly different data term, and an interaction term that defines overlap based on distance instead of area. Additionally, in contrast to [PPSD15], the repeated energy minimization is performed on a random set of size $3k$ using the TRWS minimizer, instead of two sets of non-interacting candidates of roughly size $k$ using graph cuts, for all three versions.

We compare the three approaches with a standard segmentation method, namely Otsu thresholding [Ots79] as implemented in Fiji/ImageJ [SACF+12]. We combine Otsu thresholding with slightly different preprocessing steps consisting of local contrast enhancement using contrast limited adaptive histogram equalization (CLAHE) and anisotropic diffusion filtering in order to optimize the segmentation result. For postprocessing the Otsu thresholded images, small holes (of less than 40 square pixels) are filled if their circularity exceeds a value of 0.5. As a final postprocessing step for all approaches, a basic morphological opening operation is applied using a disk with radius of 3 pixels as structuring element.

We apply the approaches to previously acquired microscopy images of fluorescently labeled neuroblastoma cell lines (see [BHG+12] for a detailed description of the data). The images have a size of 1344 x 1024 pixels. For the evaluation, the image data is categorized into different degrees of difficulty: the images of the cell line SH-EP exhibit a small to medium cell count and little clustering (low to medium difficulty). The images of the cell line SK-N-BE(2)-C have a high cell count as well as many clustering cells and constitute more challenging data. The evaluation is performed on 18 images comprising of 6 images of each category, i.e. low, medium and high difficulty.

The performance of the approaches is evaluated with respect to manual segmentation by a human expert. The values of the confusion matrix (True Positives, TP; False Positives, FP; True Negatives, TN; and False Negatives, FN) are calculated and normalized with respect to the number of cells determined by manual segmentation. Additionally, we investigate the performance of the approaches using a selection of measures described in [TH15]. We used an overlap-based metric (Dice coefficient or F1-Measure), a distance-based metric (Hausdorff distance), and a pair-counting-based metric (Adjusted Rand Index, ARI) to assess different characteristics of the segmentation results. For the Hausdorff distance we calculate the modified Hausdorff distance as proposed in [DJ94] for the True Positives. All measures are normalized with respect to the number of cells or pixels in the manually segmented images and averaged over all images of the respective category of difficulty.

### 9.1.3 Results

On average, manual segmentation results in a total of 74 cells per image for easy, 419 cells for medium and 641 cells for hard data. The results for the different performance measures averaged over all images of the respective difficulty are summarized in table 1. Although the MBC algorithm achieves the highest number of TP, this is mainly the result of a consistent oversegmentation as evident in the far larger percentage of FP. Otsu thresholding exhibits the lowest percentage of FP and is arguably the best method with regard to the confusion matrix for easy and medium images. For difficult images the MRSC and MRSC/MBC methods perform comparably, however. In terms of FN, MBC achieves

| | easy | | | | medium | | | | difficult | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Otsu | MBC | MRSC/ MBC | MRSC | Otsu | MBC | MRSC/ MBC | MRSC | Otsu | MBC | MRSC/ MBC | MRSC |
| TP [%] | 87.9 | **91.4** | 87.8 | 88.2 | 90.3 | **92.4** | 89.3 | 89.5 | 69.8 | **79.8** | 69.5 | 71 |
| FP [%] | **1.06** | 26.1 | 4.05 | 6.4 | **1.02** | 18.2 | 2.83 | 2.92 | **4.2** | 30.5 | 4.71 | 11.9 |
| FN [%] | 12.1 | **8.63** | 12.6 | 12.2 | 9.92 | **7.61** | 11 | 10.7 | 33.5 | **20.2** | 34 | 31 |
| Dice | 0.846 | 0.859 | **0.876** | **0.876** | 0.865 | 0.871 | **0.888** | 0.887 | 0.81 | 0.8 | **0.857** | 0.826 |
| Hausdorff | 1.77 | **1.75** | 1.87 | 1.86 | 9.63 | **9.16** | 9.52 | 9.42 | **11.8** | 12 | 15.6 | 14.4 |
| ARI | 0.839 | 0.852 | **0.869** | **0.869** | 0.82 | 0.825 | **0.846** | 0.845 | 0.72 | 0.707 | **0.778** | 0.739 |

Table 9.1: Averaged values of different performance measures chosen for the segmentation approaches applied to "easy", "medium" and "difficult" cell images. MCN is the mean cell number per image.

very low values due to its tendency to oversegment while the three other methods yield almost equal results. Concerning the Hausdorff distance, MBC performs better than both MRSC and MRSC/MBC, although it is outperformed by Otsu thresholding for easy and medium images. Both variants of the proposed method based on MRSC are shown to consistently outperform Otsu thresholding as well as MBC with regard to ARI and the Dice coefficient. The improvement is more prominent for difficult images with a high cell density and a higher amount of clustering. The combined approach of MRSC/MBC yields results of equal or slightly better quality compared to the MBSC method for almost all performance measures. The MRSC/MBC method demonstrates that although better tuned starting probabilities do have a slight positive influence on the overall outcome, the Set Cover probability update has a far larger effect, and the algorithm is stable towards its initialization.

## 9.2 3D Fiber Data

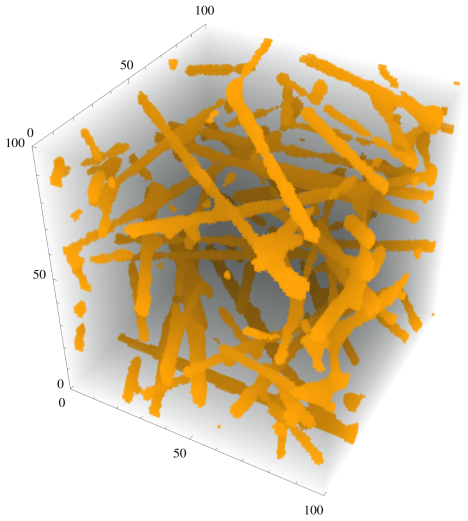
Due to the lack of both ground truth data and established measures of cover quality for 3D fiber data, as well as the significantly increased computational cost of a statistical analysis compared to 2D segmentations and model estimation, we restrict ourselves to a proof-of-concept demonstration of the previously presented methods of model estimation and image segmentation on a 3D fiber volume. To this end, we first segment the given volume using the MRSC algorithm with an underlying binary model, as in the previous section 9.1, then estimate a Gibbs model of fiber interaction from the result of the binary segmentation as presented in chapter 5 , before finally using the estimated model for a refined, non-binary MRSC segmentation of the same volume.

Fibers are parameterized as 3D cylinders with varying center, orientation and length as described in section 7.1; We distinguish 337 directions on the closed semisphere and three cylinder lengths of $20, 40$ or $60$ pixels, and use an area (here: volume) bias of $1.5^{-|A|}$. Cylinder diameter is set to a fixed value of 4 heuristically.

Improbable origin points and directions are excluded via Gaussian filters and the local Scharr operator as described in section 8.1. The large relative length of the fibers compared to the total image dimensions make the MWU approach unfeasible. We thus use the standard MRSC algorithm 11.

### 9.2.1 Hardcore Segmentation

We first perform a hardcore segmentation, i.e. a MRSC segmentation for which the underlying model only punishes pairwise distance below object diameter, as a basis for both our model estimation and second model-based segmentation. We start with weights that are uniform on all admissable parameters except for the volume bias, and let the MRSC algorithm 11 run with a sample size of $\left|N_{(n)}\right| = 300$ for 10 000 steps. This results is an approximate cover consisting of 92 cylinders, depicted in figure 9.2. We cover $29.45\%$ of object voxels, while the percentage of covered object voxels to total cover size is $41.97\%$.




(a) $100^3$ Volume of Fiber Data

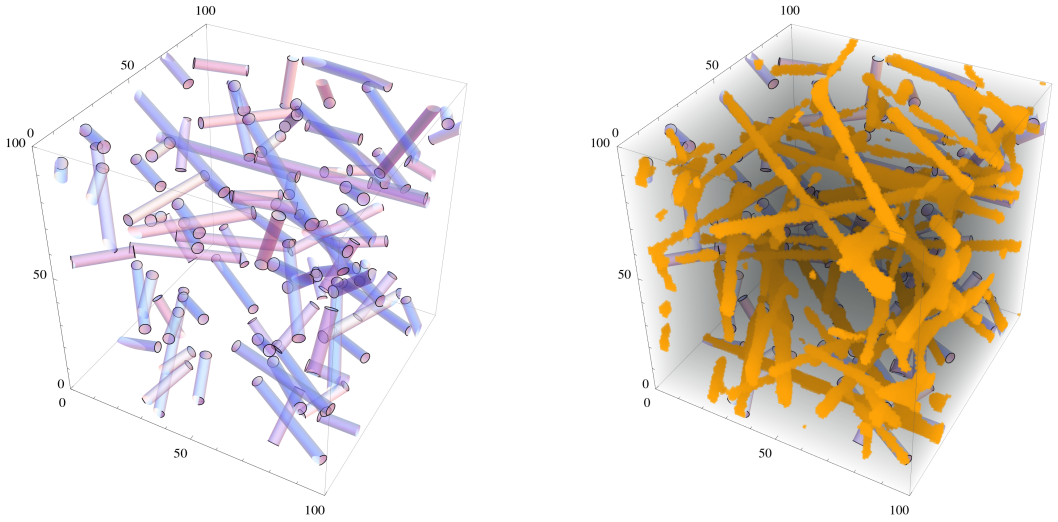



Figure 9.2: Hardcore Cover of Fiber Data

### 9.2.2 Estimation of Interaction parameters

After producing a preliminary segmentation based on a binary, hard-core interaction function, we use the Monte Carlo root-finding algorithm described in section 5.6.2 to produce a better estimate of the true interaction function of fiber objects and be able to base any following segmentation on a more refined underlying object model.

We discretize the line segment distance $d_1$ and orientation distance $d_2$ via

$$(r_i) = (0, 4, 10, 20, 30, 40, 50, 60, 70, \infty) \tag{9.2.1}$$

$$(s_j) = (0, \frac{1}{3}, \frac{2}{3}, 1). \tag{9.2.2}$$

Here, $4$ is the diameter of our fibers/cylinders, and the minimal distance enforced in the binary/hard-core-based segmentation. To avoid the problems that result from constant values and linear dependencies in the sufficient statistic – 5.6.1 – we do not include the values of $\zeta_{0,\cdot}$, which correspond to the distance interval $[r_0, r1) = [0, 2)$ below minimal distance, as well as the last value $\zeta_{9,1}$, in the Levenberg-Marquardt optimization.

As discussed in section 5.6.4, we determine our starting value via the relation of the characteristic statistic $y(x)$ of our data in relation to the average $\overline{y}$ over a uniform/Poisson MCMC chain of of length $10^5$, but choose a somewhat more conservative[1] starting value of

$$\zeta_0 = \frac{1}{20} \log(\frac{y(x)}{\overline{y}}). \tag{9.2.3}$$

Similarly, for the stabilizing factor $\lambda$ of Levenberg-Marquardt we choose a starting value of $\lambda_0 = 1/16$ and multiplying factor of $\mu = 4$, which are somewhat more conservative compared to the values of $\lambda_0 = 10^{-4}$ and $\mu = 10$ used in [HP99].

Using these parameters, we achieve a stable convergence; listed below are the values of the merit function $\chi^2 = (y(x) - \overline{y})^2$; accepted changes for which $\zeta_i$ was updated in bold.

**5133 1668 88.26 5.11 3.46 2.10** 3.90 **0.66** 2.34 **0.48** 1.65 4.40 1.12

At this point, proposed changes to $\zeta$ do not exceed $2 * 10^{-3}$, and it seems likely that the effect of random variance of $\chi^2$ is overpowering the influence of the changes in $\zeta$. We break off at

$$\zeta^* = \begin{pmatrix} -\infty & -0.150 & 0.139 & -0.045 & -0.001 & 0.067 & -0.045 & -0.018 & 0.021 \\ -\infty & 0.192 & -0.095 & 0.084 & 0.076 & -0.154 & -0.031 & 0.033 & 0.004 \\ -\infty & 0.041 & -0.085 & -0.021 & 0.057 & -0.019 & 0.006 & -0.055 & -0.001 \end{pmatrix}$$

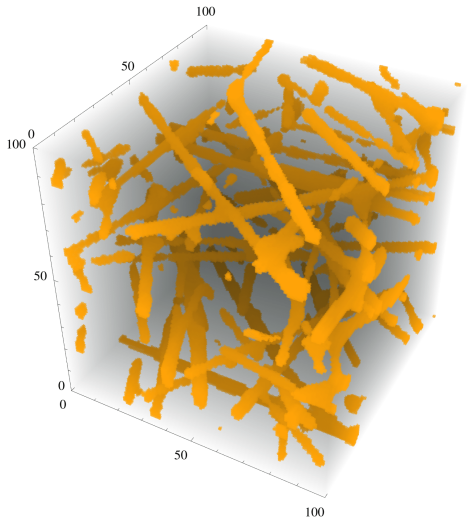
corresponding to the merit function value $\chi^2 = 0.48$. Since we can achieve a stable convergence to a low value, we forego the use of a smoothing prior in this case. For practical use in the MRSC algorithm, which works with finite values, we replace the hardcore penalty with $\zeta_{\cdot 1} = -50$.

[1] closer to uniform

### 9.2.3 Non-Hardcore Segmentation

As with the hardcore segmentation, we perform 10 000 steps with a sample size of $\left|N_{(n)}\right| = 300$, but use the (non-uniform) weights produced by the hardcore segmentation as starting weights. To scale our data term against our estimated interaction term (see section 6.3), we start with a hardcore segmentation, and rescale until the non-hardcore segmentation is of a cardinality between $80\%$ and $130\%$ of the hardcore cover. Since the cardinality of the energy minimizer can change as the data term changes, we repeat this process in every consequent step in which the cover size is below (above) the previously defined minimal (maximal) cover size.

The result is an approximate cover of 75 cylinders, depicted in figure 9.2. Despite the lower cardinality of the solution, we achieve a better segmentation, covering $34.31\%$ of object voxels, while the percentage of covered object voxels to total cover size is $54.59\%$.




(a) $100^3$ Volume of Fiber Data

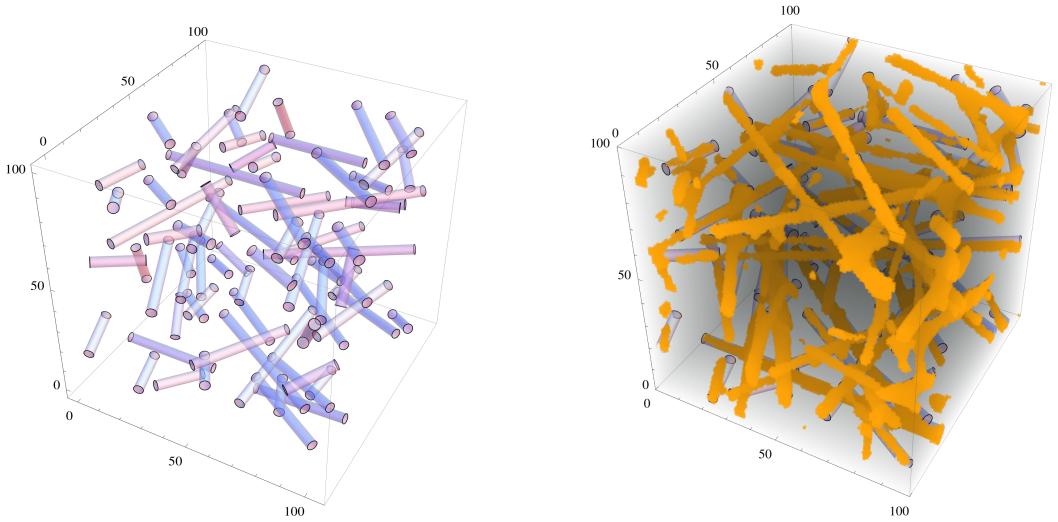
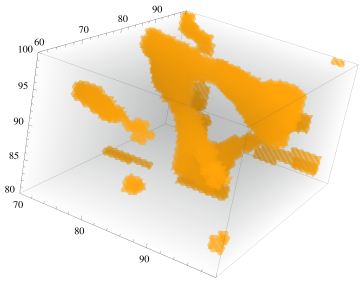



Figure 9.3: Model-based Cover of Fiber Data

# Chapter 10

# Conclusion and Outlook




Our initial motivation was the segmentation of fiber data that is clotty and noisy enough to invalidate many established approaches of image segmentation.

To this end, we first examined a randomized set cover approach that combines robustness towards noise and scalable, approximate solutions for large problems in part I. We tightened existing bounds for convergence for a generalized version of the approach, as well as introducing a new approach for its parallelization that distinctly outperforms established methods on both a theoretical and practical level. We also showed links to other comparable combinatorial problems and algorithms. Among other things, we established a method of approximately solving large linear programs as a special case of the RSC approach.

In part II of this thesis, we modified existing multiscale point processes to include several non-Euclidean distances. This proposed model of object interaction strikes a balance between an accurate and flexible representation of physical object behaviour, and the ability to simulate and estimate the model with relative ease. We additionally showed how in applications, established methods of modeling and parameter estimation can and should be altered to avoid several practical pitfalls.

The combination of the inherent robustness of the randomized RSC algorithm with a multiscale point process model as prior knowledge proposed in section 6.1 allows for the approximate segmentation of extremely difficult data. We showed how this new, model-based set cover method can be applied in several settings, including the segmentation of 2D cell and 3D fiber data, in part III. For the purpose of application, we discussed the pre- and postprocessing methods best used in different settings. We also showed how different versions of the RSC and MRSC approaches perform relative to each other, and comparable established methods, for both synthetic and real data.

The combination of the use of prior knowledge and a dynamic, probabilistic approximation of optima gives the MRSC approach a high degree of stability and a wide range of applications. However on both a theoretical and practical level, several questions remain open:

The number of candidates we sample in each step is an important input parameter for all versions of both RSC and MRSC, as it indirectly controls the trade-off of running time versus quality of the resulting cover. On a theoretical level, the sampled candidates should form an $\varepsilon$-net, or at least locally

cover points with sufficient probability. To our knowledge, estimation of cardinalities of $\varepsilon$-nets constructed via random sampling from discrete distributions $p_i$ is in current literature generally reduced to a construction for uniform distributions. However both intuitively and empirically, the cardinality of an $\varepsilon$-net sampled from a distribution for which a lot of mass is concentrated in relatively few places, as expressed by the variance or entropy, is much lower than in the uniform case, leading to a gap between theoretical bounds and practice. The RSC algorithm usually starts with uniform, or close to uniform, sampling probabilities that become increasingly concentrated with each step. Any estimation of $\varepsilon$-net sizes that takes the concentration of the underlying distribution explicitly into account might lead to the possibility of a RSC algorithm with dynamic instead of static sampling sizes, possibly enabling a sharper and more efficient convergence to an optimum, as well as a reduction of manual input.

At the same time, to our knowledge all established versions of proof of convergence for RSC in some way follow the same idea as the more general multiplicative weight method, namely comparing the growth of the summed weight of an optimal subset to the total weight of all candidates. The bounds we established in section 2.3 are tight for a sequential reweighting of optimal ranges, and thus can be expected to be close to tight in simpler geometric sitations, i.e. those in which a collection optimal ranges has little to no mutual overlap. However, they do not supply any estimation of single probabilities of optimal ranges or dual ranges for the general case. Any proof reliably bounds the probabilities of single ranges, instead of a collection of optimal ranges, could again lead to sharper estimations of convergence for more complicated geometrical situations, as well as closing the gap between plausible consideration and formal proof for the convergence of the $\text{MRSC}_\text{b}$ algorithm 11.

In section 5.6.5, we approximated a Monte Carlo estimation by averaging over many relative short Markov chains with fixed starting points to combat the slow convergence and high variance of Markov chains of weakly attractive multiscale point processes. While it heuristically led to huge improvements in the stability and convergence of gradient-based parameter estimation, a more general application to chains of high variance or slow convergence might merit a more formal, quantitative investigation of the changes in speed and quality compared to a standard MCMC approximation.

Lastly, an interesting trade-off is present in the scaling of the cardinalities of energy minimizing subsets used for the MRSC algorithm: a minimum of large cardinality will hardly take our prior model into account, while one of low cardinality might only be an $\varepsilon$-net with low probability (or for high values of $\varepsilon$). Ideally, scaling should take into account both our certainty of the estimated model and a dynamic estimation of the needed size for an $\varepsilon$-net given the current concentration of sampling probabilities. Any deliberations following this train of thought could be of interest both for applications of the MRSC algorithm, and in the more general context of a model-based, Bayesian probabilistic optimization.

In particular, the practical implications of link between RSC and the solution of linear programs established on a theoretical level in section 4.4 remain relatively unexplored. While we are immediately supplied with the use of better bounds for quality and speed of convergence, the possible transfer of parallelized or even Bayesian approaches to the context of solving linear programs offers opportunities for further research.

# Appendix A

# Randomized Set Cover and $\varepsilon$-Nets

## A.1 Detailed Bounds of the Number of Weight-Doubling Steps

**Algorithm 13:** Randomized Set Covering

**Input:** $X, \mathcal{R}, \varepsilon, n$.
**Output:** set cover $R_1, \ldots, R_n$ of $X$.
**begin**
  set $\omega(R) = \frac{1}{|\mathcal{R}|},\ \forall R \in \mathcal{R}$ `// starting weights`
  sample sets $R_1, \ldots, R_n \sim \mu = \frac{1}{\omega(\mathcal{R})}\omega$ and `// random approximate cover`
  set $\overline{X}_c = X \setminus \{x \colon x \in R_i \text{ for some } R_i\}$ `// points not covered`
  **while** $\overline{X}_c \neq \emptyset$ **do**
    pick any point $x \in \overline{X}_c$
    **if** $\omega(\mathcal{R}_x) < \varepsilon\omega(\mathcal{R})$ `// ε-condition`
    **then**
      $\omega(R) \longleftarrow 2\omega(R),\ \forall R \colon \{x\} \cap R \neq \emptyset$ `// weight update`
    sample sets $R_1, \ldots, R_n \sim \mu = \frac{1}{\omega(\mathcal{R})}\omega$ and `// repeat sampling`
    set $\overline{X}_c = X \setminus \{x \colon x \in R_i \text{ for some } R_i\}$
  return $\mathcal{R}^{(n)}$

Although several ways of constructing $\varepsilon$-nets are described in literature – see theorem 2.2.1 for the general case – they share the property that an $\varepsilon$-net is constructed with a high probability $p \in (0, 1]$.

We aim to bound the number of steps taken as tightly as possible under this assumption – substitute $p = 1$ for deterministic approaches.

**Theorem A.1.1** (Termination, Iteration Number Bound)**.** *Let $\omega^0$ denote the starting weights of algorithm 2, and let*

$$\varepsilon < \varepsilon_{\max} = 2^{\frac{1}{n^*}} - 1. \tag{A.1.1}$$

*If a hitting set $H$ of size $n^*$ exists, and condition 2.3.1 is satisfied in every step independently with a probability of at least $p$ for such a value of $\varepsilon$ then with probability one the number $N$ of iterations*

*of Algorithm 2 is finite and*

$$\mathbb{E}[N] \leq \frac{1}{1-p} N_{\mathrm{wd}}, \tag{A.1.2}$$

*where*

$$N_{\mathrm{wd}} = \lfloor \frac{a}{a-1} \lg \left( \frac{\omega^0(X)}{\omega^0_{\min} n^*} \right) n^* \rfloor + 1 \tag{A.1.3a}$$

$$a > 1 \qquad \textit{such that} \quad \varepsilon = 2^{\frac{1}{an^*}} - 1, \tag{A.1.3b}$$

$$\tag{A.1.3c}$$

$$w^0_{\min} = \min_{h \in H} w^0(h). \tag{A.1.3d}$$

*Proof.* We first focus on bounding the number of weight-doubling steps, i.e. steps in which condition 2.3.1 is satisfied, by the limit given in equation (A.1.3a). Using the same arguments as in the proof of lemma 2.3.1, we can establish the upper and lower bounds:

$$\omega^i(X) \leq (1+\varepsilon)\omega^{i-1}(X) \leq (1+\varepsilon)^i \omega^0(X) = 2^{\frac{i}{an^*}} \omega^0(X), \tag{A.1.4}$$

$$\omega^i(H) \geq \sum_{j=1}^{n^*} 2^{t_i(j)} \omega^0_{\min} \geq n^* 2^{\sum_{j=1}^{n^*} t_i(j)/n^*} \omega^0_{\min} \geq n^* \omega^0_{\min} 2^{i/n^*}. \tag{A.1.5}$$

(A.1.4) and (A.1.5) together yield

$$\omega^0_{\min} n^* 2^{\frac{i}{n^*}} \leq 2^{\frac{i}{an^*}} \omega^0(X) \qquad \Leftrightarrow \qquad \lg \omega^0_{\min} n^* + \frac{i}{n^*} \leq \frac{i}{an^*} + \lg \omega^0(X) \tag{A.1.6a}$$

$$\Leftrightarrow \qquad \frac{i}{n^*} - \frac{i}{an^*} \leq \lg \left( \frac{\omega^0(X)}{\omega^0_{\min} n^*} \right) \tag{A.1.6b}$$

$$\Leftrightarrow \qquad i \leq \frac{a}{a-1} \lg \left( \frac{\omega^0(X)}{\omega^0_{\min} n^*} \right) n^*. \tag{A.1.6c}$$

After $i = N_{\mathrm{wd}}$ weight-doubling steps, inequality (A.1.6c) is violated.

We conclude that after not more than $N_{\mathrm{wd}}$ weight-doubling steps given by (A.1.3a) the algorithm must terminate.

It remains to estimate the number of non-weight-doubling steps $N_{\overline{\mathrm{wd}}}$. To this end, we show that for every weight-doubling step $i$, the expected number $\mathbb{E}[N^i_{\overline{\mathrm{wd}}}]$ of non-weight-doubling steps until weight-doubling step $i+1$ is bounded by

$$\mathbb{E}[N^i_{\overline{\mathrm{wd}}}] \leq \frac{1}{1-p}. \tag{A.1.7}$$

Let $Q_j$ denote the random variable that indicates whether condition 2.3.1 holds in step $j$. By the assumptions of the theorem, all $Q_j$ can be bounded by a set of variables that are i.i.d. $\sim \mathsf{Ber}(p)$.

Thus, after weight-doubling step $i$, the maximal number of non-weight-doubling steps (failures of the Bernoulli variable) until condition 2.3.1 is satisfied and weights are updated again, is bounded by a geometrically distributed variable $N^i_{\overline{\text{wd}}}$, with an expected value of $\mathbb{E}[N^i_{\overline{\text{wd}}}] = \frac{p}{1-p}$; Since the number of weight-doubling steps is bounded by $N_{\text{wd}}$, it immediately follows that the total number of non-weight-doubling steps is bounded:

$$\mathbb{E}[N_{\overline{\text{wd}}}] = \mathbb{E}[\sum_{i=1}^{N_{\text{wd}}} N^i_{\overline{\text{wd}}}] = \sum_{i=1}^{N_{\text{wd}}} \mathbb{E}[N^i_{\overline{\text{wd}}}] \tag{A.1.8}$$

$$\leq \sum_{i=1}^{N_{\text{wd}}} \frac{p}{1-p} \tag{A.1.9}$$

$$= N_{\text{wd}} \frac{p}{1-p} \tag{A.1.10}$$

Consequently, the expected total number of steps is

$$\mathbb{E}[N] = \mathbb{E}[N_{\text{wd}} + N_{\overline{\text{wd}}}] \leq N_{\text{wd}} + \frac{p}{1-p} N_{\text{wd}} \tag{A.1.11}$$

which equals (A.1.2). □

## A.2 Asymptotic Behaviour of $\varepsilon_{\max}(x)$

**Lemma A.2.1.** $\varepsilon_{\max}(x) = 2^{\frac{1}{x}} - 1$ *asympotically approaches* $\frac{\ln(2)}{x}$, *i.e.*

$$\lim_{x\to\infty} \frac{\varepsilon_{\max}(x)}{(\frac{\ln(2)}{x})} = 1 \tag{A.2.1}$$

*Proof.*

$$\lim_{x\to\infty} \frac{2^{\frac{1}{x}} - 1}{(\frac{\ln(2)}{x})} = \lim_{x\to\infty} \frac{\exp(\ln(2)\frac{1}{x}) - 1}{\ln(2)\frac{1}{x}} \tag{A.2.2}$$

$$= \lim_{x\to\infty} \frac{\ln(2)(-\frac{1}{x^2})\exp(\ln(2)\frac{1}{x})}{\ln(2)(-\frac{1}{x^2})} = \lim_{x\to\infty} 2^{\frac{1}{x}} = 1 \tag{A.2.3}$$

using the rule of L'Hospital for the transition to the second line. □

# Bibliography


[ABK12] B. Andres, T. Beier, and J.H. Kappes, *Opengm: A C++ library for discrete graphical models*, CoRR **abs/1206.0111** (2012).

[AHK12] S. Arora, E. Hazan, and S. Kale, *The Multiplicative Weights Update Method: a Meta-Algorithm and Applications.*, Theory of Computing **8** (2012), no. 1, 121–164.

[AP14] P. K. Agarwal and J. Pan, *Near-linear algorithms for geometric hitting sets and set covers*, Proceedings of the thirtieth annual symposium on Computational geometry, ACM, 2014, p. 271.

[Avr03] M. Avriel, *Nonlinear programming: analysis and methods*, Courier Corporation, 2003.

[Bal87] D. Ballard, *Generalizing the hough transform to detect arbitrary shapes*, Readings in computer vision, Elsevier, 1987, pp. 714–725.

[Bes86] J. Besag, *On the statistical analysis of dirty pictures*, Journal of the Royal Statistical Society. Series B (Methodological) (1986), 259–302.

[BF11] J. Bentolila and J.M. Francos, *Combined affine geometric transformations and spatially dependent radiometric deformations: A decoupled linear estimation framework*, IEEE Transactions on Image Processing **20** (2011), no. 10, 2886–2895.

[BFS12] G.E. Blelloch, J.T. Fineman, and J. Shun, *Greedy sequential maximal independent set and matching are parallel on average*, Proceedings of the twenty-fourth annual ACM symposium on Parallelism in algorithms and architectures, ACM, 2012, pp. 308–317.

[BG95] H. Brönnimann and M. T. Goodrich, *Almost optimal set covers in finite VC-dimension*, Discrete & Computational Geometry **14** (1995), no. 4, 463–479.

[BGMR16] N. Bus, S. Garg, N. H. Mustafa, and S. Ray, *Tighter estimates for -nets for disks*, Computational Geometry **53** (2016), 27–35.

[BHG+12] R. Batra, N. Harder, S. Gogolin, N. Diessl, Z. Soons, C. Jäger-Schmidt, C. Lawerenz, R. Eils, K. Rohr, F. Westermann, et al., *Time-lapse imaging of neuroblastoma cells to determine cell fate upon gene knockdown*, PloS one **7** (2012), no. 12, e50988.

[Bjo96] A. Bjorck, *Numerical methods for least squares problems*, vol. 51, Siam, 1996.

[BM03] K. K. Berthelsen and J. Møller, *Likelihood and non-parametric bayesian mcmc inference for spatial point processes based on perfect simulation and path sampling*, Scandinavian Journal of Statistics (2003), 549–564.

[BMR15] N. Bus, N. H. Mustafa, and S. Ray, *Geometric hitting sets for disks: theory and practice*, Algorithms-ESA 2015, Springer, 2015, pp. 903–914.

[BRT15] A. Baddeley, E. Rubak, and R. Turner, *Spatial point patterns: Methodology and applications with R*, Chapman and Hall/CRC Press, London, 2015.

[BT11] J. Bien and R. Tibshirani, *Prototype Selection for Interpretable Classification*, Ann. Appl. Stat. **5** (2011), no. 4, 2403–2424.

[BVL95] A. J. Baddeley and M.N.M. Van Lieshout, *Area-interaction point processes*, Annals of the Institute of Statistical Mathematics **47** (1995), no. 4, 601–619.

[CBL06] N. Cesa-Bianchi and G. Lugosi, *Prediction, learning, and games*, Cambridge university press, 2006.

[CCHP09] C. Chekuri, K. Clarkson, and S. Har-Peled, *On the set multi-cover problem in geometric settings*.

[Cha16] T. M. Chan, *Improved deterministic algorithms for linear programming in low dimensions*, Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, Society for Industrial and Applied Mathematics, 2016, pp. 1213–1219.

[Cla88] K.L. Clarkson, *A Las Vegas algorithm for linear programming when the dimension is small*, Foundations of Computer Science, 1988., 29th Annual Symposium on, IEEE, 1988, pp. 452–456.

[Cla95] K. L. Clarkson, *Las vegas algorithms for linear and integer programming when the dimension is small*, Journal of the ACM (JACM) **42** (1995), no. 2, 488–499.

[CM96] B. Chazelle and J. Matoušek, *On linear-time deterministic algorithms for optimization problems in fixed dimension*, Journal of Algorithms **21** (1996), no. 3, 579–597.

[DA07] G. Dong and S. T. Acton, *Detection of rolling leukocytes by marked point processes*, Journal of Electronic Imaging **16** (2007), no. 3, 033013–033013.

[Der17] D. Dereudre, *Introduction to the theory of gibbs point processes*, arXiv preprint arXiv:1701.08105 (2017).

[Des16] X. Descombes, *Multiple objects detection in biological images using a marked point process framework*, Methods (2016).

[DFG+94] P.J. Diggle, T. Fiksel, P. Grabarnik, Y. Ogata, D. Stoyan, and M. Tanemura, *On parameter estimation for pairwise interaction point processes*, International Statistical Review/Revue Internationale de Statistique (1994), 99–117.

[DGS87] P.J. Diggle, D.J. Gates, and A. Stibbard, *A nonparametric estimator for pairwise-interaction point processes*, Biometrika **74** (1987), no. 4, 763–770.

[DJ94] M. P. Dubuisson and A. K. Jain, *A modified hausdorff distance for object matching*, Pattern Recognition, 1994. Vol. 1 - Conference A: Computer Vision & Image Processing., Proc. of the 12th IAPR International Conference on, vol. 1, Oct 1994, pp. 566–568 vol.1.

[DKF08] C. Domokos, Z. Kato, and J. M. Francos, *Parametric estimation of affine deformations of binary images*, 2008 IEEE International Conference on Acoustics, Speech and Signal Processing, IEEE, 2008, pp. 889–892.

[Elb16] K. Elbassioni, *Finding small hitting sets in infinite range spaces of bounded VC-dimension*, arXiv preprint arXiv:1610.03812 (2016).

[ERS05] G. Even, D. Rawitz, and S. M. Shahar, *Hitting sets when the VC-dimension is small*, Information Processing Letters **95** (2005), no. 2, 358–362.

[FL03] J.-J. Fernández and S. Li, *An improved algorithm for anisotropic nonlinear diffusion for denoising cryo-tomograms*, Journal of structural biology **144** (2003), no. 1, 152–161.

[FS95] Y. Freund and R. Schapire, *A desicion-theoretic generalization of on-line learning and an application to boosting*, Computational learning theory, Springer, 1995, pp. 23–37.

[GDGF10] A. Gelfand, P. Diggle, P. Guttorp, and M. Fuentes, *Handbook of spatial statistics*, CRC press, 2010.

[HF10] R. Hagege and J. M. Francos, *Parametric estimation of affine transformations: An exact linear solution*, Journal of Mathematical Imaging and Vision **37** (2010), no. 1, 1–16.

[HP99] J. Heikkinen and A. Penttinen, *Bayesian smoothing in the estimation of the pair potential function of gibbs point processes*, Bernoulli (1999), 1119–1136.

[HP11] S. Har-Peled, *Geometric Approximation Algorithms*, AMS, 2011.

[IK88] J. Illingworth and J. Kittler, *A survey of the hough transform*, Computer vision, graphics, and image processing **44** (1988), no. 1, 87–116.

[JD92] A. K. Jain and M.-P. Dubuisson, *Segmentation of x-ray and c-scan images of fiber reinforced composite materials*, Pattern Recognition **25** (1992), no. 3, 257–270.

[Kar72] R. Karp, *Reducibility among combinatorial problems*, Springer, 1972.

[KEB91] N. Kiryati, Y. Eldar, and A. M. Bruckstein, *A probabilistic Hough transform*, Pattern recognition **24** (1991), no. 4, 303–316.

[Ken98] W. S. Kendall, *Perfect simulation for the area-interaction point process*, Probability towards 2000, Springer, 1998, pp. 218–234.

[KGS15] A. Khan, S. Gould, and M. Salzmann, *A linear chain markov model for detection and localization of cells in early stage embryo development*, Proc. IEEE Winter Conference on Applications of Computer Vision 2015, IEEE, 2015, pp. 526–533.

[KHXO95] H. Kälviäinen, P. Hirvonen, L. Xu, and E. Oja, *Probabilistic and non-probabilistic Hough transforms: overview and comparisons*, Image and vision computing **13** (1995), no. 4, 239–252.

[KKA00] N. Kiryati, H. Kälviäinen, and S. Alaoutinen, *Randomized or probabilistic Hough transform: unified performance evaluation*, Pattern Recognition Letters **21** (2000), no. 13, 1157–1164.

[KM00] W.S. Kendall and J. Møller, *Perfect simulation using dominating processes on ordered spaces, with application to locally stable point processes*, Advances in Applied Probability (2000), 844–865.

[KMP17] A. Kupavskii, N. Mustafa, and J. Pach, *Near-optimal lower bounds for epsilon-nets for half-spaces and low complexity set systems*, 2017.

[Kol06] V. Kolmogorov, *Convergent tree-reweighted message passing for energy minimization*, IEEE Transactions on Pattern Analysis and Machine Intelligence **28** (2006), no. 10, 1568–1583 (English).

[KPW92] J. Komlós, J. Pach, and G. Woeginger, *Almost tight bounds for $\epsilon$-Nets*, Discrete & Computational Geometry **7** (1992), no. 1, 163–173.

[LB96] M.N.M. van Lieshout and A.J. Baddeley, *A nonparametric measure of spatial interaction in point patterns*, Statistica Neerlandica **50** (1996), no. 3, 344–361.

[Lit88] N. Littlestone, *Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm*, Machine learning **2** (1988), no. 4, 285–318.

[Lub86] M. Luby, *A simple parallel algorithm for the maximal independent set problem*, SIAM journal on computing **15** (1986), no. 4, 1036–1053.

[Mar63] D. Marquardt, *An algorithm for least-squares estimation of nonlinear parameters*, Journal of the society for Industrial and Applied Mathematics **11** (1963), no. 2, 431–441.

[Mat02] J. Matousek, *Lectures on Discrete Geometry*, Springer, 2002.

[MDG16] N. Mustafa, K. Dutta, and A. Ghosh, *A simple proof of optimal epsilon nets*.

[Møl99] J. Møller, *Markov chain monte carlo and spatial point processes*.

[MR95] R. Motwani and P. Raghavan, *Randomized algorithms*, 1. publ. ed., Cambridge University Press, Cambridge [u.a.], 1995 (eng).

[MRZ+17] P. Markowsky, S. Reith, T. Zuber, R. König, K. Rohr, and C. Schnörr, *Segmentation of cell structures using model-based set covering with iterative reweighting*, Biomedical Imaging (ISBI 2017), 2017 IEEE 14th International Symposium on, IEEE, 2017, pp. 392–396.

[MV17] N. H. Mustafa and K. Varadarajan, *Epsilon-approximations and epsilon-nets*, arXiv preprint arXiv:1702.03676 (2017).

[MW04] J. Moller and R. P. Waagepetersen, *Statistical inference and simulation for spatial point processes*, CRC Press, 2004.

[MZ17] P. Markowsky and T. Zuber, *Parallelizing $\varepsilon$-net-based reweighting with tight bounds*, 2017.

[NG09] F. Nielsen and V. Garcia, *Statistical exponential families: A digest with flash cards*, arXiv preprint arXiv:0911.4863 (2009).

[NS08] A. Noga and J. Spencer, *The Probabilistic Method*, Wiley, 2008.

[Ots79] N. Otsu, *A threshold selection method from gray-level histograms*, IEEE Transactions on Systems, Man, and Cybernetics **9** (1979), no. 1, 62–66.

[PPSD15] E. Poulain, S. Prigent, E.l Soubies, and X. Descombes, *Cells detection using segmentation competition*, 2015 IEEE 12th International Symposium on Biomedical Imaging (ISBI), IEEE, 2015, pp. 1208–1211.

[PR08] E. Pyrga and S. Ray, *New existence proofs $\varepsilon$-nets*, Proceedings of the twenty-fourth annual symposium on Computational geometry, ACM, 2008, pp. 199–207.

[PST95] S. A. Plotkin, D. B. Shmoys, and E. Tardos, *Fast approximation algorithms for fractional packing and covering problems*, Mathematics of Operations Research **20** (1995), no. 2, 257–301.

[Rip77] B. Ripley, *Modelling spatial patterns*, Journal of the Royal Statistical Society. Series B (Methodological) (1977), 172–212.

[Rip91] ———, *Statistical inference for spatial processes*, Cambridge university press, 1991.

[RRSS17] T. Rajala, C. Redenbach, A. Särkkä, and M. Sormani, *A review on anisotropy analysis of spatial point patterns*, arXiv preprint arXiv:1712.01634 (2017).

[SACF+12] J. Schindelin, I. Arganda-Carreras, E. Frise, V. Kaynig, M. Longair, T. Pietzsch, S. Preibisch, C. Rueden, S. Saalfeld, B. Schmid, et al., *Fiji: an open-source platform for biological-image analysis*, Nature methods **9** (2012), no. 7, 676–682.

[SE02] P. Schneider and D. H. Eberly, *Geometric tools for computer graphics*, Morgan Kaufmann, 2002.

[Soi13] P. Soille, *Morphologische bildverarbeitung: Grundlagen, methoden, anwendung*, Springer-Verlag, 2013.

[SW00] H. Scharr and J. Weickert, *An anisotropic diffusion algorithm with optimized rotation invariance*, Mustererkennung 2000, Springer, 2000, pp. 460–467.

[SWD15] E. Soubies, P. Weiss, and X. Descombes, *Graph cut based segmentation of predefined shapes: Applications to biological imaging*, Pattern Recognition Applications and Methods, Springer, 2015, pp. 153–170.

[TH15] A. A. Taha and A. Hanbury, *Metrics for evaluating 3d medical image segmentation: analysis, selection, and tool*, BMC Medical Imaging **15** (2015), no. 1, 29.

[TLSK15] A. Tanács, J. Lindblad, N. Sladoje, and Z. Kato, *Estimation of linear deformations of 2d and 3d fuzzy objects*, Pattern Recognition **48** (2015), no. 4, 1391–1403.

[Var10] K. Varadarajan, *Weighted geometric set cover via quasi-uniform sampling*, Proceedings of the forty-second ACM symposium on Theory of computing, ACM, 2010, pp. 641–648.

[Vaz01] V. V. Vazirani, *Approximation algorithms*, Springer, Berlin ; Heidelberg [u.a.], 2001 (eng), Auf dem Titelblatt falsche Schreibweise: Appromixation algorithms.

[WR02] D. Walsh and A. E. Raftery, *Accurate and efficient curve detection in images: the importance sampling Hough transform*, Pattern Recognition **35** (2002), no. 7, 1421–1431.

[YNM+15] Q. Yang, A. Nofsinger, J. McPeek, J. Phinney, and R. Knuesel, *A complete solution to the set covering problem*, Proceedings of the International Conference on Scientific Computing (CSC), The Steering Committee of The World Congress in Computer Science, Computer Engineering and Applied Computing (WorldComp), 2015, p. 36.

[You95] N. E. Young, *Randomized rounding without solving the linear program*, SODA, vol. 95, 1995, pp. 170–178.

[YTL92] R. Yip, P. Tam, and D. Leung, *Modification of hough transform for circles and ellipses detection using a 2-dimensional array*, Pattern recognition **25** (1992), no. 9, 1007–1022.