DEPARTMENT OF COMPUTER SCIENCE
SERIES OF PUBLICATIONS A
REPORT A-2011-3

Simulation and graph mining tools for improving gene mapping efficiency

Petteri Hintsanen

To be presented, with the permission of the Faculty of Science of the University of Helsinki, for public criticism in Auditorium XIV (University Main Building, Unioninkatu 34) on 30 September 2011 at twelve o'clock noon.

UNIVERSITY OF HELSINKI
FINLAND

Supervisor
Hannu Toivonen, University of Helsinki, Finland

Pre-examiners
Tapio Salakoski, University of Turku, Finland
Janne Nikkilä, University of Helsinki, Finland

Opponent
Michael Berthold, Universität Konstanz, Germany

Custos
Hannu Toivonen, University of Helsinki, Finland

Contact information

Department of Computer Science
P.O. Box 68 (Gustaf Hällströmin katu 2b)
FI-00014 University of Helsinki
Finland

Email address: postmaster@cs.helsinki.fi
URL: http://www.cs.Helsinki.fi/
Telephone: +358 9 1911, telefax: +358 9 191 51120

ISSN 1238-8645
ISBN 978-952-10-7139-3 (paperback)
ISBN 978-952-10-7140-9 (PDF)
Computing Reviews (1998) Classification: G.2.2, G.3, H.2.5, H.2.8, J.3
Helsinki 2011
Helsinki University Print

Simulation and graph mining tools for improving gene mapping efficiency

Petteri Hintsanen

Department of Computer Science
P.O. Box 68, FI-00014 University of Helsinki, Finland
petteri.hintsanen@iki.fi
http://iki.fi/petterih

PhD Thesis, Series of Publications A, Report A-2011-3
Helsinki, September 2011, 136 pages
ISSN 1238-8645
ISBN 978-952-10-7139-3 (paperback)
ISBN 978-952-10-7140-9 (PDF)

Abstract

Gene mapping is a systematic search for genes that affect observable characteristics of an organism. In this thesis we offer computational tools to improve the efficiency of (disease) gene-mapping efforts. In the first part of the thesis we propose an efficient simulation procedure for generating realistic genetical data from isolated populations. Simulated data is useful for evaluating hypothesised gene-mapping study designs and computational analysis tools. As an example of such evaluation, we demonstrate how a population-based study design can be a powerful alternative to traditional family-based designs in association-based gene-mapping projects.

In the second part of the thesis we consider a prioritisation of a (typically large) set of putative disease-associated genes acquired from an initial gene-mapping analysis. Prioritisation is necessary to be able to focus on the most promising candidates. We show how to harness the current biomedical knowledge for the prioritisation task by integrating various publicly available biological databases into a weighted biological graph. We then demonstrate how to find and evaluate connections between entities, such as genes and diseases, from this unified schema by graph mining techniques.

Finally, in the last part of the thesis, we define the concept of reliable subgraph and the corresponding subgraph extraction problem. Reliable subgraphs concisely describe strong and independent connections between two given vertices in a ran-

dom graph, and hence they are especially useful for visualising such connections. We propose novel algorithms for extracting reliable subgraphs from large random graphs.

The efficiency and scalability of the proposed graph mining methods are backed by extensive experiments on real data. While our application focus is in genetics, the concepts and algorithms can be applied to other domains as well. We demonstrate this generality by considering coauthor graphs in addition to biological graphs in the experiments.

Computing Reviews (1998) Categories and Subject Descriptors:
G.2.2 Graph algorithms
G.3 Probabilistic algorithms
H.2.5 Data translation
H.2.8 Data mining
J.3 Life and medical sciences

General Terms:
Algorithms, Experimentation

Additional Key Words and Phrases:
Bioinformatics, Databases, Data mining, Graphs, Simulation

Acknowledgements

I would like to thank my supervisor Hannu Toivonen for his everlasting guidance throughout my research struggles. Many thanks to all members of the now-defunct Biomine research group and its similarly defunct predecessors. In particular, many substantial intellectual and tangible hæritages of Lauri Eronen and Petteri Sevon have been the most indispensable for this thesis and possibly otherwise as well. Other pundits whose legacies have been, directly or indirectly, woven into this text include Kimmo Kulovesi, Atte Hinkka, Melissa Kasari, Päivi Onkamo and probably dozens of others whose contributions I unfortunately cannot recall.

Populus simulator package, upon which we hacked the simulator of Chapter 2, was originally developed by Vesa Ollikainen. My thanks to him for sharing his work. Further thanks to Tarja Laitinen for supplying us with the asthma data used in Chapter 3. An implementation of EATDT was kindly provided by David J. Cutler of Johns Hopkins University School of Medicine.

This research has been supported by, at least, Helsinki Institute for Information Technology and its former Basic Research Unit (HIIT BRU, nowadays simply HIIT); the Finnish Funding Agency for Technology and Innovation (Tekes); GeneOS Ltd; Jurilab Ltd; Biocomputing Platforms Ltd; Graduate school in Computational Biology, Bioinformatics and Biometry (ComBi), and its later incarnation Finnish Doctoral Programme in Computational Sciences (FICS); Algorithmic Data Analysis (Algodan) Centre of Excellence of the Academy of Finland (Grant 118653); and the Department of Computer Science at the University of Helsinki. I might even have received a fistful of euros from the European Commission under the 7th Framework Programme FP7-ICT-2007-C FET-Open, contract BISON-211898. While this list grimly exposes the scattered and woefully unstable nature of research funding, I am sincerely grateful for all support that let me focus on my musings without significant financial challenges.

Thanks to the free software movement and the capable IT people at the department for providing the hardware and software bits for this endeavour. Finally, thanks to Stiga Games Ltd for table ice hockey games that were instrumental in cooling off after fastidious scientific debates and other daily engagements.

Contents

Chapter 1

Introduction

The aim of this thesis is to offer computational tools to improve the efficiency of (disease) gene-mapping efforts. *Gene mapping* is a systematic search for genes that affect observable characteristics of an organism (*phenotype*). A well-known example is disease gene mapping where the objective is to find a gene or genes that put an individual into greater risk to develop a certain disease. Knowledge of disease gene locations and variants is useful, for instance, in diagnostics, screening and drug development. From the reader we assume an elementary knowledge of genetics; a short introduction is given in Section 1.1 and a summary of genetics terms can be found in the glossary.

The biomedical role of the contributions of this dissertation are best characterised by viewing a gene-mapping project as a somewhat abstract and streamlined "pipeline" with two major phases: an *initial analysis phase* and a *refined analysis phase*. Both phases have multiple steps as illustrated in Figure 1.1. The objective of the initial analysis phase is to find a set of regions from the genome that are statistically linked to the phenotype. Known genes in these regions form the set of *putative* disease susceptibility (DS) genes. In the refined analysis phase the putative genes are prioritised and their connections to each other and possibly other DS genes are evaluated. In this thesis we propose simulation and graph mining tools to improve the efficiency of some of these steps.

During the study design the investigators have to decide between family-based or population-based study. This choice affects sampling and applicable computational tools. Furthermore, one must perform power analysis to decide sufficient sample sizes so that the possible discovered associations will be statistically significant. In Chapter 2 we propose a novel simulation procedure to assist in making these choices. With simulated data sets researchers can assess the potential of the hypothesised study designs and computational analysis tools before embarking on the actual and expensive sampling. Simulated data sets with controlled parameters are also useful in the development and evaluation of novel gene-mapping tools.

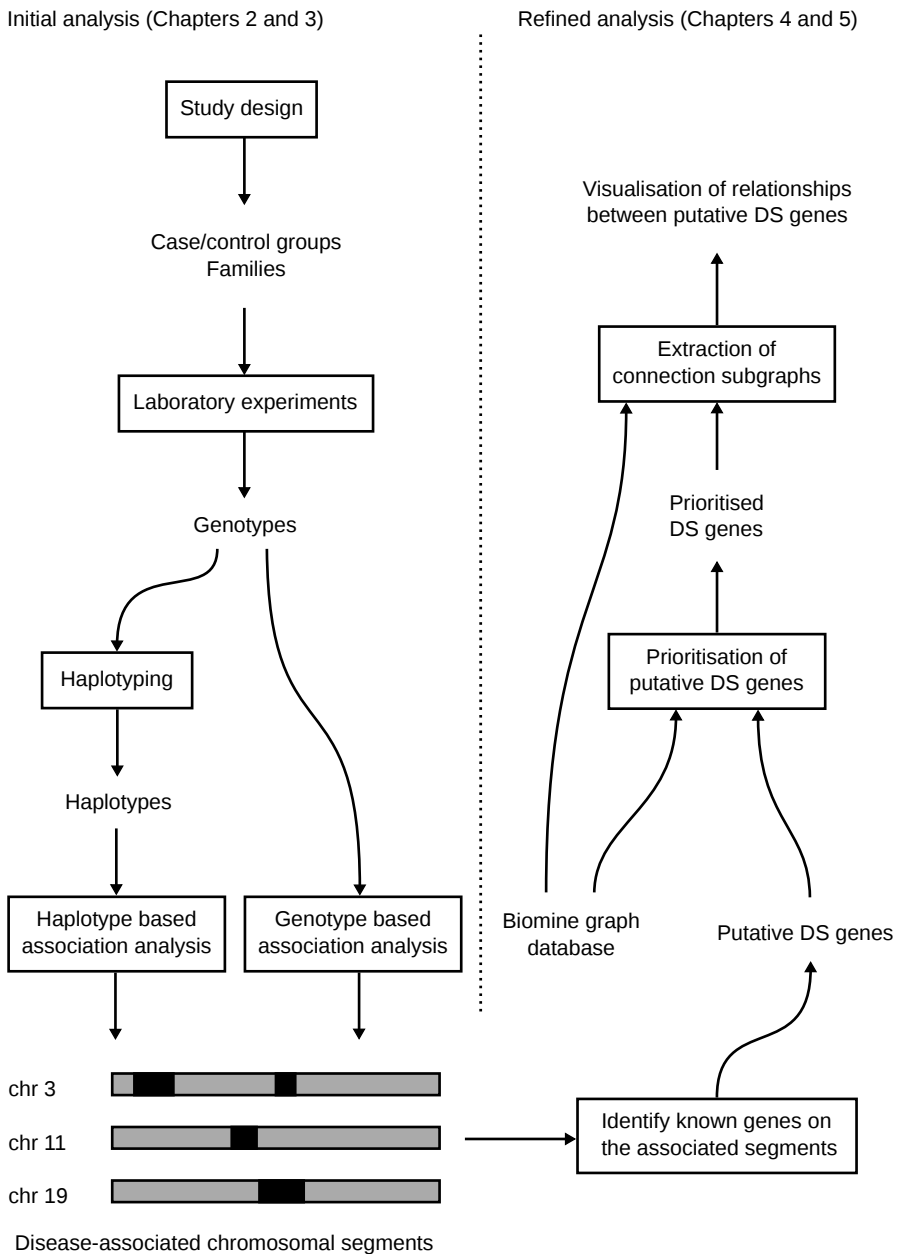


Figure 1.1: Overview of a simplified disease gene-mapping project. Boxes denote processes and actions that manipulate input data and produce output (arrows). The left column depicts the initial analysis phase where statistically significant candidate markers are sought. These markers are identified with known gene locations and further analysed in the refined analysis phase on the right.

Our proposed approach solves both computational and conceptual issues in simulation by combining two unrelated but complementary simulation paradigms into a novel two-phase procedure.

In Chapter 3 we argue, based on extensive experimental evaluation using the simulator of Chapter 2, that a population-based association study design combined with a population-based haplotyping can be more powerful than a family-based one. This is an interesting result because it can save a considerable amount of effort in sampling and genotyping ("Laboratory experiments" in Figure 1.1) which are probably among the most expensive steps in the whole gene-mapping pipeline.

The initial analysis phase typically produces a large set of putative genes that have a statistically significant link to the phenotype under study. These putative genes are likely to include false positives due to the large number of initially tested genes. It is beneficial to exclude as many false positives as possible before proceeding with the refined analysis. Of course, this requires more information about the putative genes than the genetical data which was used in the first place.

In Chapter 4 we present *Biomine*: a framework for expressing different interlinked biological databases as a large graph that allows analysis and ranking of putative genes. Public biological databases such as PubMed [112] and OMIM [100] contain vast amounts of useful background information for assessing the strengths of the links between putative genes and phenotypes (or other biological entities). Many of these databases are interlinked and form a network of current biological, genetical and medical knowledge. Accessing this network can be cumbersome as models and automated methods for network analysis have been proposed only recently.

We show how to weigh edges (relations between biological concepts) as a function of their reliability, relevance and rarity (informativeness). Another distinguishing feature of Biomine is the probabilistic interpretation of edge weights: a given edge is considered to "exist" with a certain probability and to "not exist" otherwise. This interpretation lets us apply random graph techniques, in addition to conventional weighted graph techniques, for mining the graphs. One application of Biomine is the detection of gene–phenotype and other kinds of links [116]. The framework also admits ranking of putative genes so that further efforts can be targeted to the most promising candidates. We will show how to discover links efficiently and assess their strength and statistical significance.

In Chapter 5 we consider *reliable subgraphs*. Reliable subgraphs can be helpful when assessing the current knowledge about the putative genes and their relations to the phenotype, as they display in a concise form the most probable and relatively independent connections between a putative DS gene and the phenotype. Reliable subgraph extraction is closely related to information search or discovery: given a large graph and two or more concepts of interest, we would like

to obtain a small number of other concepts that connect the interesting objects as well as possible in the graph. Such subgraphs contain the most relevant objects from the original (large) graph for the user to focus on. Small subgraphs are also viable for visual inspection.

We give a definition of a reliable subgraph and the corresponding reliable subgraph extraction problem in Chapter 5. We also propose efficient and scalable algorithms for extracting subgraphs from random graphs [54, 56, 57]. Most of the algorithms are based on heuristics due to the computational complexity of the subgraph extraction problem. The validity of the algorithms is established with extensive experimentation.

This introductory chapter is intentionally short. Since the chapters of this dissertation address separate problems of a gene-mapping project, each chapter begins with a substantial introductory section. We will next give an overview of the genetic concepts used and the contributions of this dissertation.

1.1 Genetic concepts

(This section is directly based on previous work [125] and included here with permission[1]. The material has been revised for this thesis.)

Genome is a collection of an organism's hereditary information. The human genome is organised into 23 different *chromosomes* present in every cell as two homologous copies (Figure 1.2A): one from the mother and another from the father. A chromosome is a single, giant DNA molecule consisting of millions of consecutive pairs of nitrogenous bases, A-T (adenine and thymine) and C-G (cytosine and guanine), which form the well-known double helix structure with a four-letter alphabet. Most of the DNA has no known functional relevance; only a minority of DNA is estimated to be genes or their regulatory factors (Figure 1.2B).

The order of the bases and genes is the same from individual to individual with only minimal variation: one recent study estimated a 99.5 % similarity between two human genomes [84]. This variation inside the genome is utilised as genetic *markers* (Figure 1.2C). The alternative forms of markers, *alleles*, can be readily distinguished from each other using standard laboratory methods (*genotyping*). Alleles can be used to compare individuals or populations and to estimate the co-occurrence of a disease with a certain combination of marker alleles. *Haplotype* is a string of alleles in an individual chromosome. Haplotypes are sparse, economic representations of chromosomes whose focus and density is set by the *marker map* used in the study. In *genome scans* the marker map covers the whole genome or a full chromosome, while in *fine mapping* studies the markers are more densely located in a candidate area for a disease-susceptibility gene.

[1] Courtesy of Päivi Onkamo, Department of Biosciences, University of Helsinki.

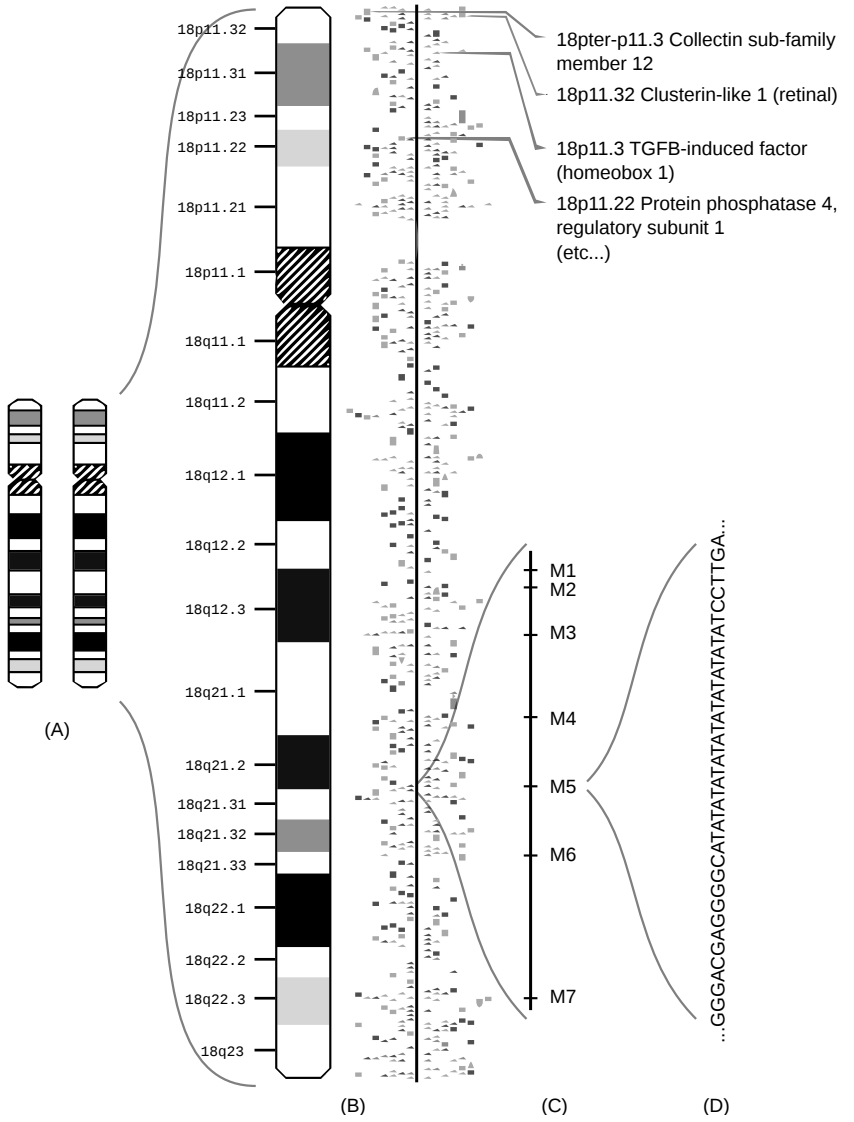


Figure 1.2: (A) A homologous pair of human chromosome 18. (B) Enlargement of a chromosome view from NCBI GenBank: the annotated human chromosome 18. Cytogenetic locations are given on the left, and the known and predicted genes are shown with dots along the vertical line on the right. In the upper right corner are examples of names of the genes as they appear in the NCBI site. (C) A small section of chromosome showing some marker locations, or *loci* (denoted by M1–M7). The alleles at M1–M7 constitute a haplotype. (D) Enlargement from (C): a stretch of DNA sequence including an 11-repeat allele of locus M5 flanked by a unique DNA sequence.

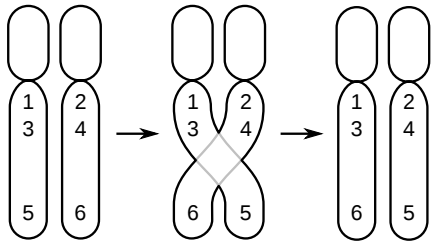


Figure 1.3: An example of crossing-over. *Left:* Two homologous chromosomes have been duplicated in *meiosis*. Here only two of the four are shown for simplicity. *Middle:* The lower ends of the chromosomal arms are exchanged (*crossing-over*). *Right:* The resulting daughter chromosomes (*recombinants*) are transmitted to different gametes.

Recombination is an elementary phenomenon in genetics: a pair of homologous chromosomes (represented by haplotypes in a gene mapping study) exchange genetic material in the gamete production process. As a result, the chromosome transmitted from a parent to an offspring is not an exact copy of either parental chromosomes but a mosaic of them. See Figure 1.3 for a simple example of a recombination event with one crossing-over. Consequently, recombination maintains variation between individuals in each generation. At large scales, such as those typically used in gene mapping, the probability of recombination is approximately constant along the chromosome and the number of recombinations between two locations, or *loci*, correlates well with the physical distance between the loci. Recombination is the key factor in gene mapping: since it fragments haplotypes, the genealogies of different loci in the genome are different, and this helps to localise genes.

In *association mapping* correlations between the disease or trait (phenotype) and marker alleles are sought from a sample of affected and healthy individuals (Figure 1.4). It is assumed that disease mutations derive from one ancestral chromosome where a single mutation occurred a long time ago. Such mutations are said to be *identical by descent*, or IBD, while alleles that are chemically identical but cannot be traced to a common ancestor are *identical by state*, or IBS. As generations have passed the disease mutation has been transmitted onward and recurrent recombinations have narrowed the ancestral DNA segment around the mutation. The length of the preserved segment is proportional to the amount of time elapsed since the introduction of the mutation. With sufficiently dense marker map around the mutation we can observe a statistically overrepresented haplotype segment in affected individuals compared to the unaffected in the current population. Formally, there is *linkage disequilibrium* (LD) between the actual disease gene and the surrounding markers, and this LD can be used to infer the

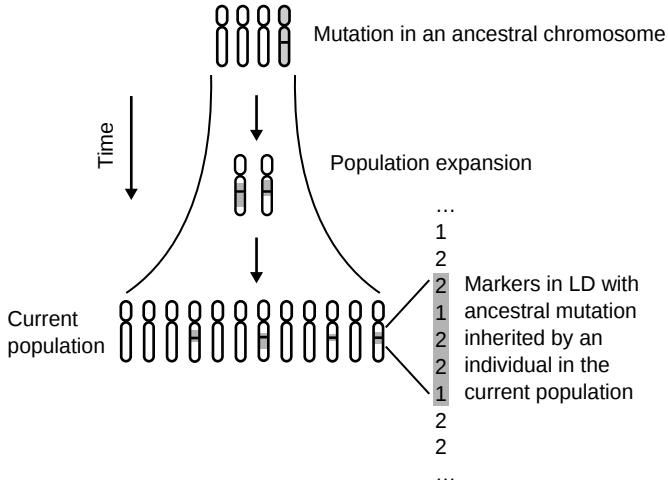


Figure 1.4: Gene mapping by association analysis. The disease mutation has originated in a common ancestor, marked by a horizontal line in the rightmost ancestral chromosome. In the course of generations, consecutive recombinations narrow down the segments of conserved ancestral DNA around the disease mutation (shaded in light grey). Only short conserved segments are remaining in the current population, but genotyping a dense map of markers along the chromosome from affected and unaffected individuals and comparing the resulting haplotypes would reveal the area of increased sharing in the disease-associated haplotypes.

possible disease-susceptibility loci.

1.2 Contributions

We summarise the main contributions of this thesis below. Most of the text is based on original research articles and manuscripts by the author and other contributors. All material has been thoroughly revised to be self-contained with more details and related work. The author of this thesis (later simply *the author*) assumes all responsibility for the correctness of the revised material. We also describe the author's contributions to the research.

In Chapter 2 we propose a novel and efficient two-phase simulation procedure and its implementation. The new procedure is useful for simulating realistic case–control haplotype data from isolated populations. Chapter 2 is based on a cooperative work with Petteri Sevon: the approximated coalescent model and the concept of two-phase simulator were originally introduced by him [114]. The author implemented a complete two-phase simulator by extending the *Populus* simulator [101] and combining it with the approximated coalescent simulator. The

author also wrote a manuscript that forms the basis of Chapter 2. The manuscript itself has not been published.

In Chapter 3 we present the results of extensive experimental comparisons between two study designs and associated haplotyping and gene-mapping techniques. Based on these results, we argue that certain population-based study designs are superior to family-based (trio) designs. Chapter 3 is based on a research article by the author and others [55]. The author simulated the data sets using the simulation tools from Chapter 2 and ran all experiments. The author also contributed substantially to the writing of the article.

In Chapter 4 we show how multiple biological databases can be integrated to form a large biological graph representing current biomedical knowledge. We also demonstrate how to weigh the edges of the graph and give a novel probabilistic interpretation for the weights. The usefulness of the graph model is established by experiments on Alzheimer genes and protein interactions. Chapter 4 is based on a research article by Sevon and others [116]. The author contributed to the design and implementation of the graph database *Biomine*, and to the writing of the article.

In Chapter 5 we formulate a novel subgraph extraction problem for random graphs. We briefly discuss the complexity of the problem and give several algorithms for solving the problem. We establish the validity of the algorithms by extensive experiments. The chapter is based on three research papers by the author and contributors [54, 56, 57]. The first article was compiled solely by the author, while the second is a joint work with Hannu Toivonen. Petteri Sevon invented the idea of the Monte Carlo algorithm (Section 5.5) of the third article; the author collaborated with Hannu Toivonen to design a complete algorithm and improved the idea by adding the cut sampling algorithm and lookahead optimisation. Otherwise, the author is mostly responsible for the algorithms, implementations, test setups and experiments of Chapter 5. The author also wrote the majority of the articles.

Chapter 2

Simulation techniques for marker data

Although larger amounts of marker data are becoming publicly available, researchers searching genes for complex traits face difficult problems with both amount and structure of data. In the study design phase of our simplified gene-mapping pipeline (Figure 1.1), assessing the statistical power to localise disease-susceptibility (DS) genes in a given study setting usually requires a large number of independent case–control sets or families depending on the type of the planned study. Developing or testing new methods or statistics for localisation requires data sets with known properties such as DS mutation loci and allele frequencies. Evaluating the behaviour of a certain method benefits from controlled experiments where few parameters of the test data are varied and others held constant.

Real data sets with known genetic effects and population models are limited both in availability and quantity for large-scale experiments. Simulated data addresses all of the issues above: with a simulator arbitrary amounts of data with specified properties can be easily generated. It is crucial that the simulated data resembles real-world data sets as faithfully as possible to ensure the validity of the obtained results.

In this chapter we present a two-phase simulation procedure for generating realistic haplotype data in isolated populations. The procedure is a combination of a coalescent-based simulation and a forward-time simulation. These methods are discussed in Sections 2.1 and 2.2. Briefly stated, coalescent models are established tools for generating recombination and mutation history for a sample of random chromosomes. Since case–control samples and trio pedigrees are not random samples from a population, but conditioned on the phenotype, the models are generally unsuitable for generating such data. In contrast, a forward-time simulator generates a complete pedigree and recombination history for the entire population. After specifying the founder haplotypes and a disease model, haplotypes and phenotypes for all individuals (including ancestors) can be deduced and acquiring case–control samples or trio pedigrees is straightforward. However,

the generation of founder haplotypes with realistic genealogical history remains a problem in the forward-time simulation paradigm. We solve the drawbacks of both paradigms by combining them into a two-phase procedure as described in Section 2.3.

2.1 Coalescent simulation

The stochastic process known as *the coalescent* is a well established, versatile tool for analysing DNA sequence polymorphism data [73, 121, 122]. In addition to rich inference potential [120], the coalescent is also useful for simulating artificial genetical data [59, 139].

Coalescent simulation is about generating a possible genealogy or genetical history for a sample of *sequences* (chromosomes, for example). It turns out that, under a few assumptions about the population, all sequences in the sample are descendants of a single, ancient ancestral sequence. The output of the simulation is a possible history for the sampled sequences, represented as a tree or graph depending on the specific model. Next, we briefly outline the general theory of the process following a treatise given by Nordborg [99].

2.1.1 Model overview

We begin with the simplest possible case: a constant-sized population of N haploid organisms that reproduce according to the Wright–Fisher model of neutral evolution [38]. Haploid organisms reproduce by cloning themselves so that each organism has only one parent and recombination cannot occur. Model neutrality means that genetic variation does not interfere with reproductive success; that is, all organisms are equally likely to reproduce. Generations are assumed to be discrete and, because the population size does not change, each generation consists of exactly N organisms. We can think that in each generation a parent is randomly and independently chosen with replacement for each N offspring, and the possible genetic variation (mutations) can be handled separately.

Let us suppose that we have a sample of sequences from this population. To construct a genealogical history for these sequences we trace ancestral lineages for each sequence back in time starting from the present generation. During backtracking the lineages coalesce together whenever two or more separate lineages have the same ancestor. Eventually the number of lineages reaches one when the most recent common ancestor (MRCA) for the sample has been found. The result is a genealogical tree for the sample: sampled sequences are in the leaves and the inner vertices are ancestral sequences. The root of the tree contains the MRCA of the sample. Figure 2.1 depicts a partial genealogical tree for a small sample of sequences.

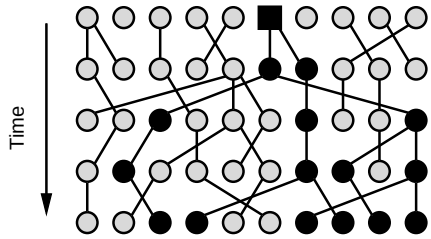


Figure 2.1: A part of a genealogical tree for a sample of 10 sequences (filled circles in the bottom row) showing coalescence events in the last five generations. The MRCA (black box) for six sample sequences has been already found. Descendants of that MRCA are coloured black; these descendants form a subtree of the complete genealogical tree for the 10 sampled sequences.

To determine coalescence times consider two lineages from a given generation. The probability that the lineages coalesce (that is, they choose the same parent) when going one generation back in time is $1/N$, and the probability that they remain distinct is $1-(1/N)$. Since generations are independent, the coalescence time follows the geometric distribution with parameter $1/N$. Consequently, the probability that two lineages are separate more than t generations into the past is $1-(1/N)^t$, and the expected coalescence time is N generations. This probability can be approximated with a continuous-time *diffusion approximation* [96]. In the approximation we measure time in units of N generations: one time unit τ corresponds to N generations. Let $\mathbf{T}$ denote the time when the two lineages coalesce. Then the probability that the two lineages remain distinct for τ units, $\tau > 0$, is

$$\Pr(\mathbf{T} > \tau) = \left(1 - \frac{1}{N}\right)^{N\tau} \underset{N\to\infty}{\longrightarrow} \mathrm{e}^{-\tau}.$$

The limit obtained is known as *diffusion limit*, and we see that in this limit $\Pr(\mathbf{T} \leq \tau) = 1 - \mathrm{e}^{-\lambda\tau}$ where $\lambda = 1$. Thus $\mathbf{T}$ is exponentially distributed with parameter 1, and with relatively large values of N we can approximate coalescence times by generating exponential variates.

Consider now k ancestral lineages. The probability that the lineages do not coalesce when going one generation back in time is

$$\begin{aligned}\prod_{i=1}^{k-1}\left(\frac{N-i}{N}\right) &= \prod_{i=1}^{k-1}\left(1-\frac{i}{N}\right) \\ &= 1 - \sum_{i=1}^{k-1}\frac{i}{2N} + O\left(\frac{1}{N^2}\right) = 1 - \frac{\binom{k}{2}}{N} + O\left(\frac{1}{N^2}\right)\end{aligned}$$

where $O(1/N^2)$ represents all terms which are divided by N^2 or any higher power

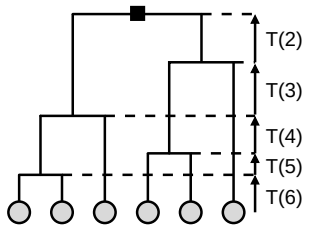


Figure 2.2: A genealogy tree for a sample of six sequences. Each waiting time $\mathbf{T}_k$ (marked in the figure as $\mathrm{T}(k)$) between two coalescence events is an exponentially distributed random variable with parameter $\binom{k}{2}$. The topmost coalescence event (black box) represents the MRCA of the sample.

of N. Denote the time between the coalescing events where the size of the genealogy goes from k to $k-1$ by $\mathbf{T}_k$. As above, for any $\tau > 0$, we have

$$\begin{aligned} \Pr(\mathbf{T}_k > \tau) &= \left(\prod_{i=1}^{k-1} \left(1 - \frac{i}{N} \right) \right)^{N\tau} \\ &= \left(1 - \frac{\binom{k}{2}}{N} + O\left(\frac{1}{N^2} \right) \right)^{N\tau} \xrightarrow[N\to\infty]{} \exp\left[-\binom{k}{2}\tau \right]. \end{aligned}$$

Again, we see that $\mathbf{T}_k \sim \mathrm{Exp}(\binom{k}{2})$ in the diffusion limit. These approximations, which basically define the stochastic process known as "the coalescent", make a very efficient simulation tool. To generate a genealogy for a sample of n sequences under the Wright–Fisher model one just needs to generate n exponential random variates and a random bifurcating topology instead of simulating lineages on a discrete time scale. See Figure 2.2 for an example of a complete coalescent tree for a sample of six sequences.

Probably the most significant aspect of the coalescent is its versatility. For now we have assumed the Wright–Fisher model and clonal haploid organisms. However, the coalescent approximates or, more precisely, is a limiting process for a wide range of neutral models [92]. For example, varying population sizes, diploid organisms, separate sexes and overlapping generations can all be handled by simply rescaling time appropriately.

Recombination cannot be handled with the (standard) coalescent, since under recombination different parts of a sequence can originate from different ancestors and the genealogy can no longer be represented with a simple bifurcating tree. Fortunately it is relatively straightforward to incorporate recombination into the coalescent as described below.

2.1.2 Coalescent with recombination

Coalescent with recombination should be used if the population is diploid and the simulated sequences are not extremely short [59]. Compared to the standard coalescent, where each sequence has a single ancestral lineage towards its ancestors, a recombination event in a lineage effectively breaks it into two distinct lineages when going backward in time [49, 50]. One of these lineages carries ancestral material to the left of the recombination point $\mathbf{X} \sim \mathrm{U}(0,1)$ and the other one to the right of $\mathbf{X}$. (Here we assume that the sequences have unit length and the recombinations are uniformly distributed along the sequence.) These two lineages are then treated as separate ancestral lineages that recombine and coalesce independently.

To model recombinations within the coalescent we assume a diploid population of size N with $2N$ sequences and measure time in units of $2N$ generations. Let r be the *recombination rate* per generation (i.e. the expected number of recombinations in one sequence per generation), and set $\rho = 4Nr$ as the *scaled recombination rate*. In this model, k distinct lineages coalesce with rate $\binom{k}{2}$, as before, and recombine with rate $k\rho/2$ [59, 139]. Recombination event splits a random lineage into two, and the new lineages coalescence and recombine independently when going back in time. Simultaneous recombination and coalesce events have negligible probabilities and are ignored.

We can think coalescent with recombination as a birth-death process, where lineages "born" (recombine) with rate $k\rho/2$ and "die" (coalesce) with rate $\binom{k}{2}$ when there are k lineages [50]. Since the coalescence rate is $\Theta(k^2)$ and the recombination rate is $\Theta(k)$, the number of distinct lineages will eventually reach one when going back in time. We call this "ultimate ancestor" the *grand most recent common ancestor* or GMRCA. The result of the process is no longer a genealogy tree but a graph known as the *ancestral recombination graph* (ARG). Figure 2.3 gives an example of a simple ARG for two sequences.

For each fixed point in a sequence (a base pair, for instance) the ARG contains an embedded genealogical tree of the sample with its own "local" MRCA. In general, each recombination-free interval on the sampled sequences has a single genealogy tree in the ARG. This property is useful when simulating allele data (see Section 2.1.3). We can extract such tree for a fixed point $p \in [0,1]$ in the sequence simply by following lineages back in time in the ARG starting from the sample sequences. When we encounter a recombination vertex we choose the lineage which traces ancestral material on the side of the recombination point where p resides. Coalescence vertices are extracted as such. When the number of lineages hits one we have found the MRCA and the complete genealogy tree. Note that this MRCA is not necessarily the same as the GMRCA (see Figure 2.3).

Simulating ARGs with the coalescent with recombination is analogous to simulating a genealogy with the standard coalescent: we trace lineages back in time,

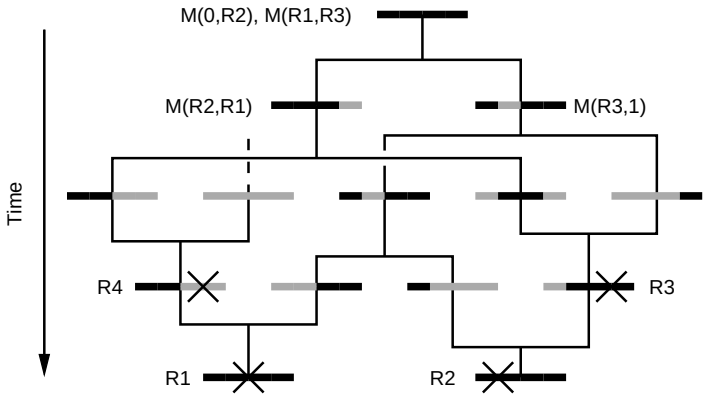


Figure 2.3: Ancestral recombination graph for two sequences (bottom row). Solid black segments denote ancestral material and dark grey segments denote non-ancestral material. Four recombinations have occurred. They are marked with crosses and R$x \in [0,1]$ denotes the location of the corresponding crossover. Most recent common ancestor for a (recombination-free) segment $[a,b]$ is denoted by M(a,b). The topmost sequence is the grand most recent ancestor: it contains all ancestral material for both sequences in the bottom row. R4 creates a lineage with only non-ancestral material which can be ignored.

generate exponential variates according to the (independent) coalescence and recombination processes, and update the ARG accordingly until we find the GMRCA. There are a few possible optimisations. First, we do not need to trace lineages containing only non-ancestral material (such lineages can occur as depicted in Figure 2.3). Second, if we are simulating allele data it is sufficient to find all local MRCAs for each marker locus instead of the whole ARG including the GMRCA [59].

2.1.3 Simulation of neutral mutations

Given a genealogical tree of a sample, the simulation of selectively neutral alleles (mutations) for the sample is relatively straightforward using "mutation-dropping". First, fixed alleles are assigned to the MRCA by deciding that each allele is wild-type, for instance. Next, starting from the MRCA, we follow each lineage *forward* in time and assign the same alleles to each offspring unless a mutation occurs in the lineage.

Mutations are assigned according to a chosen (neutral) mutation model. For example, in the *infinite sites model* of Kimura [72] mutation events within a single lineage occur independently with rate u in each generation. Each mutation event introduces a new allele into a random locus on a sequence. This model can be

simulated with a Poisson process with rate $\theta/2$, where $\theta = 4Nu$ is the *scaled mutation rate* (cf. scaled recombination rate) [49]. For a branch with length τ the number of mutations $\mathbf{X}$ is distributed as $\mathbf{X} \sim \text{Poisson}(\tau\theta/2)$, and their locations are uniformly distributed along the branch. All sequences below the mutation point carry the mutated allele at the (random) mutation locus while others share the same allele with the MRCA.

Mutations in ARGs can be handled in a similar manner by simulating mutations on the branches of the ARG. Since ARGs contain embedded genealogical trees with their respective MRCAs we can use similar "mutation-dropping" strategy as above by assigning fixed alleles to the MRCAs and following lineages to the sample sequences. However, only those mutations affect the alleles of the MRCA that occur in the recombination-free segment represented by the current tree.

2.1.4 Limitations

Despite its flexibility the coalescent is not ideally suitable for simulating realistic samples of (human) genotype data with complex diseases. This is due to several reasons. One is that the coalescent is based on estimating genealogies for sequences of a *random* sample from a population. Case–control and trio samples are not random: they have an ascertainment bias towards affected individuals with the disease-susceptibility mutation(s). Wang and Rannala have recently proposed coalescent models with recombination and ascertainment, but their models are restricted to one dominant DS mutation [134, 135].

Another problem is that variable and unknown selection pressures are difficult to handle with the coalescent [141, 102]. (Recall that the genealogical and mutation processes were handled separately.) Although weak selection with single or multiple loci can be incorporated into the coalescent [97, 77], selection based on multiple, linked DS loci (as is the case with complex human diseases) is complicated [40]. However, this is an area of active research [41, 23].

As demonstrated in Section 2.1.1, simulation with the standard coalescent is computationally extremely efficient. But simulations with the coalescent with recombination become increasingly demanding when the sequence length (which is proportional to the recombination rate) grows. For human chromosome-wide sequences the exact model becomes intractable and approximations have to be used [114, 91, 89]. These approximations may limit the usefulness of the model to directly simulate case–control or trio data.

Finally, coalescent models often make some delicate assumptions about the population parameters [102, 132]. For example, the assumption that the effective population size N is significantly larger than the sample size might not be warranted in case–control studies where sample sizes can be in thousands. The

practical significance of these failed assumptions may be limited, though [43].

2.2 Forward-time simulation

Forward-time simulation paradigm is conceptually simple. As its name suggests, the simulation starts from some *initial population* and proceeds forward in time generation by generation until the present generation, or *current population*, has been reached. In each generation the whole population is subjected to the specified genetical and demographical changes such as mating, recombination, selection and migration to produce a new generation. Some implementations allow overlapping generations as well, but the basic principle is the same.

Essentially the complete population history is simulated, and along the way as much information as needed can be carried along. For example, a complete pedigree can be constructed as well as complete genotypes for the simulated individuals. Based on this information the individuals can be divided into cases ("affected") or controls ("healthy") according to the specified disease model. Samples can be drawn from arbitrary generations, though usually we are interested in samples from the last few generations or the current population.

In contrast to the mathematically rigorous coalescent process and its variants the forward-time simulation paradigm does not have any specific theoretical base or constraints. Almost arbitrarily complex demographical, genealogical and environmental processes and sampling scenarios can be simulated. The downside is that only relatively young and small populations can be simulated as the memory and computation time requirements are proportional to the complexity of the simulated phenomena.

This freedom of discipline means also that there are no established standards for forward-time simulations. Most of the existing forward-time simulation methods can be seen as independent solutions for varying scenarios with no common theoretical base. Forward-time simulators SIMLINK [9], SLINK [137] and SIMLA [8] can be used to simulate pedigree data commonly utilised in parametric linkage analysis. Population genotype data simulators include POPSIM [51], EASYPOP [7], Populus [101], simuPOP [103] and genomeSIM [28]. All of these simulators have their own assumptions about population and disease models and their respective parametrisation. Some are also more general than others. For example, Populus is designed for simulating isolated populations, while simuPOP is probably the most flexible simulator environment currently available.

Peng and others note the following difficulties when designing forward-time simulations [102]:

1. *Population initialisation.* How to determine the specific alleles for the individuals in the initial population?

2. *Introduction of the disease-susceptibility mutation.* It is non-trivial to decide which ancestor(s) should introduce a DS mutation into the genealogy. With multiple, interlinked mutations the problem is even more difficult.

3. *Parameter control.* How to control population parameters in the present population? Disease prevalence and mutation frequencies, for example, should be close to the specified values to enable fair comparison of the simulated samples, but genetic drift makes it difficult to accurately control these parameters.

Because of these difficulties, forward-time simulated populations tend to have high variances in linkage disequilibrium and disease outcomes [141, 15].

The forward-time paradigm offers, however, solutions to many problems with coalescent-based methods. It seems intriguing to combine the two approaches to get the benefits of both. We next describe a two-phase simulation procedure implementing that idea.

2.3 Two-phase simulation

In this section we describe an efficient two-phase procedure for simulating realistic case–control or trio samples from isolated populations. The simulation outline has been given by Sevon [114]. We present a somewhat more detailed and extended description here.

The target population is a relatively small *population isolate* (10^5–10^6 individuals) that has gone through a genetic bottleneck in recent history (15–20 generations ago) and rapidly expanded since the bottleneck to its present size. Before the bottleneck we assume a static population with an effective size of 10^3–10^4 individuals. One or more individuals in the bottleneck, or *founder population*, carry a DS mutation or mutations. The size of the founder population is assumed to be small (100–1,000 individuals). We focus on a genome-wide setting where chromosome-wide sequences are used.

Isolated populations are useful in association studies since they are expected to exhibit more linkage disequilibrium than admixed and outbred populations, and they have other favourable properties as well [140, 71, 118, 107, 30]. Examples of small isolates include Kainuu [80] and Sardinian [123] subpopulations.

Sample sizes in association studies are generally large [109]: from tens to hundreds or even thousands of individuals [119]. Coalescent-based methods are clearly infeasible for simulating chromosome-wide sequences of such samples. In contrast, forward-time simulation is appropriate especially for the population expansion phase. Even though the population grows exponentially, the small number of founders and the short expansion time should pose no significant difficulties for

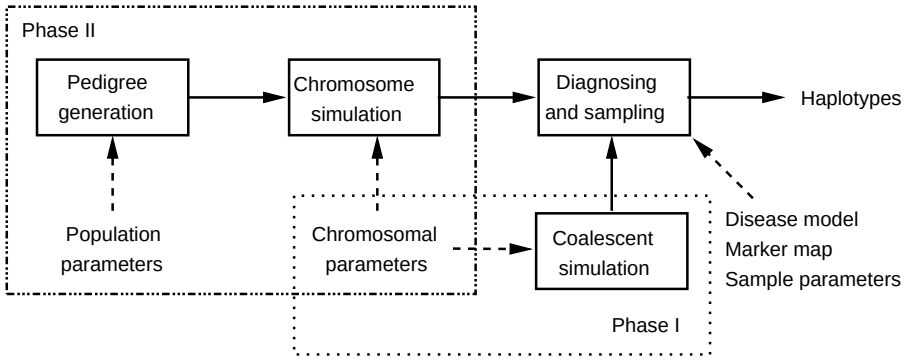


Figure 2.4: Outline of the simulation procedure. Solid arrows denote data flow and dotted arrows parameters given by the user. The two major simulation phases and their corresponding parameters (see Table 2.1) are enclosed in dotted rectangles.

the simulation. But simulating the complete population history forward in time (including the history of the founder population) suffers from increasing complexity and the simulation parameters may be difficult to specify when simulating long histories. The initialisation of the founder sequences remains an open question as well.

To maximise the correctness of the simulation, we combine the coalescent and forward-time simulation to a two-phase simulation procedure. Figure 2.4 gives an outline of the process and data flow in the simulation.

In the first phase the founder sequences are simulated using any feasible coalescent method and mutation model. The coalescent process ensures that the founder sequences are connected by a genealogy and their marker alleles reflect natural patterns of linkage disequilibrium. The mutation model produces a set of all possible polymorphic markers and DS mutations. Using coalescent process becomes tractable since the sizes of the founder and static populations are relatively small. We also give an approximated variant of the coalescent suitable for simulating chromosome-wide sequences.

In the second phase the target population isolate is simulated with a forward-time simulator. We simplify the process by assuming that the simulation is not influenced by individual genotypes. Although this makes simulation of selection impossible, the assumption allows us to choose disease-susceptibility mutation loci *after* the whole population has been simulated. Furthermore, as the allele frequencies are known for all simulated individuals, we can choose DS mutations such that the mutated allele frequencies are as close as possible to the desired values. Fixing the mutation loci also fixes the ancestors that introduce the respective mutations into the population.

Chromosomal parameters	
Recombination rate per generation	r
Mutation rate per generation	u
Population parameters	
Static population size in individuals	N
Founder population size in individuals	n_f
Current population size in individuals	n_c
Number of generations	g
Disease model	
Disease-susceptibility mutation frequency	f_{dm}
Disease-susceptibility mutation penetrance	$\Pr(\text{D}^+\|\text{M}^+)$
Disease prevalence	$\Pr(\text{D}^+)$
Marker map	
Number of markers	$\bar{m}$
Minimum marker allele frequency	f_{min}

Table 2.1: Simulation parameters mentioned in the text. The grouping of the parameters corresponds to the grouping in Figure 2.4.

After the population has been simulated we divide the simulated individuals into cases and controls according to the given disease model and observed allele frequencies. A suitable marker map is chosen from the set of mutations simulated in the first phase. Finally, we perform sampling and generate haplotype data for the sampled individuals.

We discuss the simulation process in detail below. For simplicity, we focus on exponentially growing isolates with random mating and no population substructure, a single dominant mutation disease model, and a marker map with a fixed number of equidistant markers. The simulation parameters for this restricted case are given in Table 2.1.

2.3.1 Phase I: Coalescent simulation

We begin the first phase of the simulation by simulating a genealogy for the founder population with a suitable coalescent model such as Hudson's coalescent [59, 60]. We denote the founder population size by n_f and the static population size before the genetic bottleneck by N, both in individuals. To simulate a genealogy we fix the coalescent parameters as follows: the sample size is $2n_f$ sequences, the effective population size is $2N$, the scaled recombination rate is $\rho = 4Nr$ and time is measured in $2N$ generations to compensate for the diploid setting (see Section 2.1.2). Different coalescent models can be used to simulate subpopulations, migration or expanding founder populations. We denote the set of $2n_f$ founder sequences by F. The founder sequences are randomly paired to

form diploid founder individuals.

Exact simulation of the coalescent with recombination becomes computationally demanding if the length of the simulated sequence is in tens of millions of base pairs or more. To allow reasonably fast simulation of chromosome-wide sequences we use an approximate form, originally proposed by Sevon [114], of the spatial algorithm by Wiuf and Hein [139].

The spatial algorithm simulates the sequences from left to right instead of simulating the genealogy backward in time. It works roughly as follows. First, a genealogical tree is simulated for the left endpoint of the sequences according to the standard coalescent model. Next, the distance to the next recombination event (locus) to the right is drawn from the exponential distribution with parameter $\lambda\rho$, where λ is the total length of all branches in the current genealogy and ρ is the scaled recombination rate (see Section 2.1.2). A corresponding recombination vertex is created at a random position in the current genealogy, and the new lineage is then coalesced back to the genealogy; the waiting time to the coalescence is exponentially distributed with rate k in the epoch of k lineages. New recombination loci and vertices are generated until the entire sequence is covered.

In the approximated version of the spatial algorithm we assume a first degree Markov property for the ARG as we move across the sequence: when a new recombination vertex is created we discard the lineage carrying ancestral material to the left of the recombination point and coalesce the right lineage as usual [114]. This reduces the ancestral recombination graph to a current local genealogy tree at all times. A similar strategy for approximating the coalescent has been independently proposed by McVean and Cardin [91] and slightly improved by Marjoram and Wall [89]. According to our empirical comparisons with the Hudson's implementation [60] of the exact model, the approximation is very close to correct in terms of linkage disequilibrium (r^2 statistic), allele frequencies and inter-marker distances (Figure 2.5). Independent experiments suggest similar behaviour [91, 89]. The approximation is significantly faster: our prototype implementation has $O(2n_f\mathbf{r})$ time complexity where $\mathbf{r}$ is the number of recombinations occurred during the simulation.

After the genealogy has been formed any desired neutral mutation model can be applied to the ARG produced by the coalescent (see Section 2.1.3). These mutations form the set of available polymorphisms (markers and DS mutations) for the simulation. For simplicity, we consider here only the infinite-sites model of Kimura [72] for simulating single nucleotide polymorphisms (SNPs), and we set the scaled mutation rate to $\theta = 4Nu$ (see Section 2.1.3). We denote the set of SNPs by $M = \{(l_1, F_1), \ldots, (l_k, F_k)\}$ where $l_i \in [0,1]$ is the locus of the polymorphism i and $F_i \subset F$ is the set of founder sequences which carry the mutated allele. A subset of these SNPs are used later as markers and DS mutations.

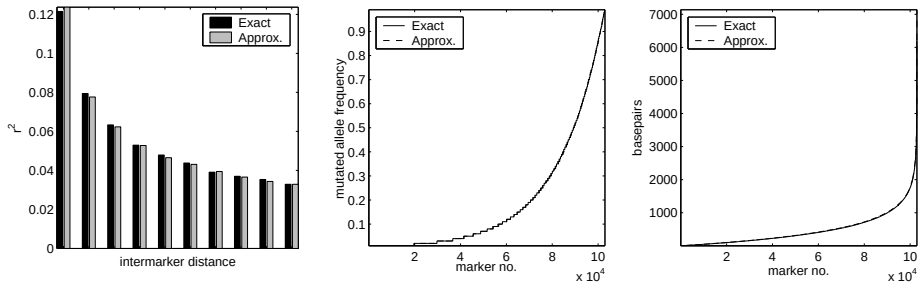


Figure 2.5: Statistics between coalescence models. In all tests a single 1 Mb sequence was simulated (effective population size 10,000, sample size 100, mutation and recombination rate 10^{-8} per base pair per generation). *Left:* Mean r^2 values between markers at most 50 kb apart from each other. First pair of bars shows values for markers which are 0–5 kb apart from each other, second pair for markers 5–10 kb apart and so on. Mean values have been calculated from 10 independent replicate data sets. *Middle:* Allele frequencies for each marker, sorted by frequency. *Right:* Inter-marker distances, sorted by distance. Allele frequencies and distances have been calculated from 50 independent replicate data sets.

2.3.2 Phase II: Forward-time simulation

In the second phase a rapidly growing population isolate with n_c individuals in the current population is simulated forward in time. This simulation process is similar to that of Populus simulator by Ollikainen [101]: we first generate a pedigree tree for the isolate and then simulate recombinations in the tree.

In the *pedigree tree generation* we sequentially simulate g distinct generations G_i, $0 \leq i \leq g$, starting from the founder generation G_0 simulated in the first phase. Each generation G_i consists of a fixed number of individuals, where $|G_i|$ depends on the chosen population growth model such that $|G_0| = n_f$ and $|G_g| = n_c$. Arbitrary population growth and migration models can be used as long as they do not depend on the genotypes.

Algorithm 1 implements an exponential growth model with random mating. The individuals in the founder generation G_0 are first divided into males ♂ and females ♀ uniformly at random. A new generation G_{i+1} is simulated from generation G_i as follows. First we generate $|G_i| \cdot (n_c/n_f)^{1/g}$ new individuals that form G_{i+1}. Next we form couples C by randomly sampling a male and a female from sets ♂ and ♀ without replacement until either ♂ or ♀ becomes empty. Then we choose parents for each individual in the generation $i+1$ uniformly at random from C. This is equivalent of drawing the number of offspring for each couple from the binomial distribution $\mathrm{Bin}(|G_{i+1}|, 1/|G_i|)$. Finally, each individual in generation $i+1$ is assigned a gender uniformly at random, thereby forming

new sets ♂ and ♀. Algorithm 1 can be straightforwardly implemented to run in $O(n_c g)$ time.

Algorithm 1 Pedigree generator

Input: Number of generations g, current population size n_c, founder population size n_f
Output: A tuple (generation, id number, father id, mother id, gender) for each individual

```
popsize ← f
grrate ← (n_c/n_f)^(1/g) {calculate growth rate}
for i ← 1 to n_f do {simulate founders}
    gender ← random({0,1})
    if gender = 0 then
        append(♂, i) {♂ is an array of male ids}
    else
        append(♀, i) {♀ is an array of female ids}
    print 0, i, 0, 0, gender
id ← f + 1
for i ← 1 to g do {simulate generation i}
    C ← ∅
    shuffle(♂), shuffle(♀) {shuffle arrays ♂ and ♀ in random}
    for j ← 1 to min{|♂|, |♀|} do {generate couples C (parents)}
        C ← C ∪ {(♂[j], ♀[j])}
    ♂ ← ∅, ♀ ← ∅
    popsize ← popsize · grrate
    for j ← 1 to round(popsize) do {choose parents}
        (father, mother) ← random(C) {pick a random couple from C with replacement}
        gender ← random({0,1})
        if gender = 0 then
            append(♂, id)
        else
            append(♀, id)
        print i, id, father, mother, gender
        id ← id + 1
```

Chromosome simulation concludes the population simulation phase by simulating meioses in the generated pedigree. We keep track of sequence fragments for each simulated (non-founder) individual i. A *sequence fragment* is a tuple (i, p, l_s, l_e, f) where i is an individual and p is the phase of the fragment; that is, whether the fragment is a part of paternal or maternal chromosome. Each fragment represents a contiguous ancestral chromosomal segment, where no crossovers have occurred between loci l_s and l_e, that has been inherited identical by descent from the founder sequence $f \in F$. Each individual has several fragments that completely describe the ancestral sources for the chromosomes of that particular individual. In other words, all chromosomes in the isolate are "mosaics": compositions of sequence fragments ultimately inherited from the founder se-

quences [128]. We denote the set of all segment fragments for the current population by S.

Whenever two individuals mate and produce an offspring, the sequences of the offspring are formed from their parents' sequences by mendelian inheritance. In meiosis, each crossover splits into two the sequence fragment where it occurs. (See Ollikainen [101] for details.) Any meaningful recombination model can be used—for example, a hotspot model if haplotype blocks [44, 133] are to be simulated. We have adopted the chromosome simulator implementation from Populus [101]. This component has $O(n_c g)$ time complexity.

2.3.3 Diagnosing and sampling

Here we combine outputs from the coalescent and forward-time simulators. The marker map $\mathcal{M} = \{m_1, \ldots, m_{\bar{m}}\} \subset M$ is selected from the set of SNPs such that each marker $m \in \mathcal{M}$ has a minimum allele frequency $f_m \geq f_{\min}$ in the current population. We choose the mutation $d = \arg\min_{m \in M \setminus \mathcal{M}} |f_m - f_{\mathrm{dm}}|$ as the disease-susceptibility (DS) mutation. After $\mathcal{M}$ and d have been chosen, all individuals in the population are diagnosed based on their genotypes at the DS locus and the specified disease model. Finally, a sample is ascertained and the haplotypes for the sampled individuals are produced.

The allele frequencies f_m are calculated using sequence fragments S. A trivial algorithm for counting the allele frequencies requires $\Omega(n_c|M| + |S|)$ time. This is inefficient, since typically we have $n_c \approx 10^5$–10^6 and $|M| \approx 10^4$–10^5. We utilise an improved algorithm by Sevon [114] with time complexity $O(|S| \log |S| + \mathbf{m} \log \mathbf{m})$, where $\mathbf{m} = \sum_{m \in M} |m_2|$ is the (random) number of mutated alleles in founder sequences. The improved algorithm (Algorithm 2) operates on a sequence of the segments sorted by their endpoints.

Markers $\mathcal{M}$ and DS mutation d are selected during the allele frequency counting. Algorithms 2–4 try to pick $\bar{m}$ approximately equidistant markers with minimum allele frequency $f_{\min}$. The first and last SNPs satisfying the frequency constraint are chosen to be the first and last markers. Remaining markers are chosen with the following scheme. The length of the sequence is divided by $\bar{m} - 1$ to get equidistant marker sites. SNPs from the coalescence simulator are mapped to these sites by choosing for each marker site the closest SNPs satisfying the frequency constraint. All SNPs that are not used as markers are considered as possible candidates for the disease mutation. Out of these, the candidate which has the closest (mutated) allele frequency to f_{dm} is chosen. If there are several such candidates, one of those is randomly chosen. The marker selection does not increase the asymptotic time complexity of the frequency counting algorithm (Algorithm 2).

After marker selection individuals are divided into affected and healthy by

Algorithm 2 Marker map and disease mutation selection

Input: Founder sequences F, sequence fragments S, mutations M, minimum marker allele frequency $f_{\min}$, number of markers $\bar{m}$, disease mutation frequency f_{dm}
Output: Marker map $\mathcal{M}$, disease mutation d

```
site ← 0 {site is the next equidistant marker site (locus)}
prev ← ∅, d ← ∅ {create dummy markers prev and d}
Construct min-heap H_S from all segments s ∈ S with segment endpoints s_4 as keys
Construct min-heap H_M from M with mutation loci as keys
Set count[f] ← 0 for all f ∈ F
for all {s ∈ S : s_3 = 0} do {populate founder sequence counters}
    count[s_5] ← count[s_5] + 1
prev ← ∅, mut ← pop(H_M), s ← pop(H_S)
while s ≠ ∅ do
    while mut ≠ ∅ and mut_1 < s_4 do {mut_1 is the locus of mut}
        f_mut ← Σ_{f∈mut_2} count[f]/|S| {calculate the frequency of mut in the population}
        if f_min ≤ f_mut ≤ 1 − f_min then {mut is a feasible marker}
            if mut_1 ≥ site then {site has been reached}
                choose mut or prev {see Algorithm 3}
            else {replace prev by mut}
                check if prev is a suitable disease mutation {see Algorithm 4}
                prev ← mut
        else {mut can be a feasible disease mutation}
            check if mut is a suitable disease mutation {see Algorithm 4}
        mut ← pop(H_M)
    count[s_5] ← count[s_5] − 1
    s ← pop(H_S)
return M, d
```

Algorithm 3 Choose marker

Input: Potential marker mutations *prev* and *mut* (*prev* can be null)
Output: $\mathcal{M}$ contains a new marker (either *prev* or *mut*), *prev* and *site* are updated

```
if prev = ∅ or |site − mut_1| < |site − prev_1| then {choose mut}
    M ← M ∪ {mut}
    check if prev is a suitable disease mutation {see Algorithm 4}
    prev ← ∅
else {choose prev}
    M ← M ∪ {prev}
    prev ← mut
site ← site + 1/(|m̄| − 1) {calculate the next marker site (locus)}
```

Algorithm 4 Check if a mutation is suitable DS mutation

Input: Potential disease mutation *mut*
Output: DS mutation d and counter n are updated

```
if mut ≠ ∅ and |f_mut − f_dm| ≤ |f_d − f_dm| then
    if f_mut = f_d then {mut and d are equally good candidates}
        n ← n + 1 {n is the number of equally good candidates seen}
        Choose random r ∈ [0, 1)
        if r < 1/n then
            d ← mut
    else
        n ← 1
        d ← mut
```

looking at their genotypes in the disease mutation locus. Consider a dominant mutation disease model with two parameters: penetrance $\Pr(\mathrm{D}^+|\mathrm{M}^+)$ and prevalence $\Pr(\mathrm{D}^+)$. If an individual carries the disease allele (notation M^+), the probability of being affected (notation D^+) is the penetrance $\Pr(\mathrm{D}^+|\mathrm{M}^+)$. The prevalence $\Pr(\mathrm{D}^+)$ is the probability of being affected regardless of the genotype at the DS locus. Phenocopy probability $\Pr(\mathrm{D}^+|\mathrm{M}^-)$ is the probability of being affected without carrying the DS mutation (notation M^-). In this case

$$\Pr(\mathrm{D}^+|\mathrm{M}^-) = \frac{\Pr(\mathrm{D}^+) - \Pr(\mathrm{D}^+, \mathrm{M}^+)}{1 - f_{\mathrm{dm}}}$$

where $\Pr(\mathrm{D}^+, \mathrm{M}^+)$ is the frequency of affected DS mutation carriers. Classifying the simulated individuals as affected or healthy is straightforward by applying the penetrance and phenocopy probabilities.

Various kinds of samples can be ascertained after diagnosing. Our prototype sampler supports four sample types: (1) pure random sample without respect to the phenotype, (2) sample containing only affected individuals, (3) sample with affected and healthy individuals and (4) sample with affected and random individuals. Mode (1) is useful when phenotype data is not needed, for example when testing haplotyping accuracy. Mode (2) is suitable for generating trio pedigrees with affected offspring which are useful for testing linkage-based localisation methods and trio haplotyping (see Chapter 3). The sampler treats each affected offspring equally likely to be sampled, so families with several affected children will be overrepresented (the classical ascertainment bias [16]). This is desirable as then the expected frequency of the disease mutation allele in the cases is higher. Finally, modes (3) and (4) can be used to simulate case–control data for population-based studies.

In the last step, haplotypes are generated for the sampled individuals using a similar procedure as in the allele frequency counting: instead of counting frequen-

cies we output alleles for the selected markers by referring to the founder alleles. This can be done in $O(|S|\log|S| + n\bar{m})$ time where n is the sample size.

The total time complexity of the simulation depends heavily on the simulation parameters and is subject to random variation. With our prototype implementation, a complete simulation of a medium-sized isolate (500 founders expand to 100,000 individuals in 20 generations) with 450 markers over a 100 Mbp sequence takes about 45 seconds on a 3.16 GHz Intel Core 2 Duo PC.

2.4 Conclusions

We have introduced a new two-phase simulation procedure for generating realistic haplotype data in isolated populations. The procedure combines the most expedient features of two simulation paradigms and allows realistic and efficient simulations of reasonably large populations and long marker maps. Our implementation has been tested and applied in algorithm development, method comparison and study design [114, 55].

The raised interest in association studies in recent years has introduced new simulation approaches for simulating case–control data [82, 86, 134, 103, 141]. Some of these are feature-wise on par with our two-phase procedure. The combination of coalescent and forward-time simulation is, to the best of our knowledge, still unique. It is an interesting future work to compare these new approaches with our simulator.

We have also discussed an approximate coalescence model with recombination. A similar model has been independently proposed recently [91, 89]. The approximation is computationally much less demanding than the exact model: it enables an efficient simulation of sequences spanning several millions of base-pairs. The model and our implementation are not restricted to isolated populations and can be useful in other applications.

The simulation procedure is extensible as well. For example, immigration can be incorporated into the procedure as follows: in each generation a certain amount of immigrants enter the population and pair up with resident individuals. Haplotypes for the immigrants can be simulated with the coalescent: either by using a variant of the model supporting subpopulations (for residents and immigrants) or by performing an independent simulation for the immigrants. It is possible to simulate pedigrees with various demographical parameters such as non-exponential growth, overlapping generations, varying death risk and population substructures [101]. Known pedigrees and alleles can be used instead of generated pedigrees and founders. The chromosome simulator can be extended to support recombination hotspots or more complex interference models.

Our prototype implementation does not have the aforementioned extensions,

but the modular design of the simulator makes it relatively easy to implement them. It is also fairly straightforward to implement more complex disease models, marker selection and sampling. We leave these improvements for future work. Our prototype implementation and the supporting documentation are available on request.

Chapter 3

Study designs in association analysis

Gene mapping studies can be based on case–control groups or families (Figure 1.1). This choice has a profound effect on the study: besides fixing possible sampling strategies it defines the range of suitable (or even required) computational analysis tools. In this chapter we apply the simulation techniques from Chapter 2 to generate data sets that mimic two common study designs for association-based gene mapping: a traditional family-based setting and a more recent population-based setting. We then empirically compare the gene-mapping power and accuracy on these data sets.

The motivation for our study stems from the fact that gene-mapping efforts have been criticised for their modest success at finding and replicating findings of disease-susceptibility genes [138]. While linkage methods are popular in trying to elucidate the genetic basis of complex traits, they have inherent limitations in detecting genes of modest-to-moderate effect [109]. Association analysis is a good alternative: it has a better resolution and, consequently, capability to utilise high density genotype data produced by high-throughput genotyping techniques [18]. Further, gene-mapping studies that only use triad or case–control data from epidemiological cohorts greatly reduce the sampling effort compared to linkage analyses based on large families.

The objective in association analysis is simple: detect statistical association between one or more genetic polymorphisms and a trait (phenotype) that might be some quantitative characteristic, a discrete attribute or disease [24]. However, association analysis is not panacea by being sensitive to the chosen study design, implementation and interpretation [52]. With inadequate design, false positives are likely and such "positive" associations are not replicable [138]. In terms of the gene-mapping pipeline (Figure 1.1), false positives inflate the number of putative DS genes and consequently turn the refined analysis phase cumbersome. Study design pitfalls include too small sample sizes, population stratification, poorly matched control group, multiple testing and over-interpretation of results [17, 63,

52].

Association studies can be based on either haplotypes or genotypes (Figure 1.1). Haplotype-based analysis is an area of increasing interest [2, 95, 142] with a growing number of computational methods [90, 124, 86, 116, 136]. Using haplotypes instead of single loci for measuring association has been observed to be more powerful in some cases [2], or even required to map rare variants [86]. Parental genotypes have been traditionally used to infer haplotypes from genotypes (*trio-based haplotyping*) and thus adding a considerable burden of recruiting and genotyping parents as well. With late-onset diseases this requirement can be critical as parents may not be available for genotyping at all. Trio-based haplotyping is also susceptible for errors as haplotypes cannot be inferred unambiguously in general. As an alternative approach, *population-based haplotyping* provides a possibility for haplotype-based studies where parental genotypes are not needed. Population-based haplotyping has recently received a considerable research attention, and many novel haplotyping algorithms have been introduced [35, 113, 14, 108].

Association studies are typically built around case–control groups or families (Figure 1.1). By *case–control study* we mean a design where both cases and controls are sampled independently from a population. By *family study* we mean a design where families with affected offspring are sampled, and these offspring are used as cases. Control genotypes are formed by deducing the non-transmitted parental haplotypes using trio haplotyping; these are called *pseudo-controls*. (Unaffected siblings can also be utilised.) The major difference between the designs is the formation of the control group: in case–control studies controls are separate individuals, while in family studies pseudo-controls are used.

The use of pseudo-controls or siblings makes family-based studies robust against population stratification [37]. By contrast, case–control studies are sensitive to population stratification and other confounding effects if controls are not carefully matched against cases. But case–control studies combined with population-based haplotyping can present an advantage since parental genotypes of sampled individuals are not required. The spared genotyping resources can be used to recruit more cases and controls. This increases sample sizes and consequently adds more power to the study.

We present a simulation study [55] where we compared experimentally the two aforementioned strategies for study design and haplotyping in association analysis: (1) a traditional family-based setting where trios are ascertained using affected children as probands, and the non-transmitted haplotypes of the parents are used as controls or additional cases depending on the phenotype of the parent; and (2) a pure population-based case–control setting where cases are ascertained as above, but independent healthy control individuals are sampled from

the population, and where the haplotypes are estimated for all individuals with a population-based statistical method. We compared the power to detect a presence of a disease-susceptibility gene and the accuracy to locate the gene. The methods we used are computationally efficient and therefore valid alternatives for analysis of large data sets. The population model is an exponentially and rapidly growing founder population similar to ones successfully used in gene mapping studies.

3.1 Methods

We used simulated data sets since they allow power analysis using a large number of replicate data sets with controlled parameter values. Different sampling and haplotyping methods (Section 3.1.2) and eventually a haplotype association mapping algorithm (Section 3.1.3) were applied to each of these replicates in turn. The power to detect the disease-susceptibility gene was then analysed as well as the mapping accuracy. By controlling each of the parameters separately we were able to analyse their effects in isolation. A recent asthma data set was used to confirm the results with real data (Section 3.2.4).

The compared settings differ in three major aspects (Table 3.1):

1. *Sample ascertainment:* An affected child was used as a proband in all sampling designs. In the trio design the non-transmitted (pseudo) haplotypes of parents were used as controls or additional cases depending on the phenotype of the parent. In the case–control design healthy individuals were sampled from the population as controls, while in the case–random design random individuals were sampled regardless of their disease status.

2. *Haplotyping:* Haplotypes can be partially deduced when trios are available. In trio-based designs we considered two extreme options: (1) a simple method where all ambiguous alleles are marked unknown, and (2) the true haplotypes (known from the simulations) as the best possible case where all ambiguous alleles are successfully estimated. In the case–control and case–random designs a statistical or combinatorial estimation method must be used. We also considered randomly phased haplotypes and maximally wrongly phased haplotypes for comparison.

3. *Sample size:* The default sample size was 500 individuals. In the case–control and case–random designs we used 250 cases and 250 control or random individuals whose haplotypes were then estimated. In the trio design we could afford, assuming equal genotyping costs, $500/3 \approx 167$ trios. From these trios we obtained haplotypes for the 167 probands plus 167 pseudo-individuals from the non-transmitted haplotypes giving 334 individuals in total.

	Sample ascertainment and effective sample sizes					
	Case–control		Case–random		Trios	
Haplotyping	500	334	500	334	500*	334*
Population-based	+	−	−	−	N/A	N/A
Trio-based	−	−	−	−	−	+
True haplotypes	−	−	−	−	−	+
Randomly phased	−	−	−	−	−	−
Maximally wrong	−	−	−	−	−	−

Legend: $*$ = numbers of genotyped individuals were 750 and 500; + = primary alternatives to be compared; − = settings used to analyse effects of different factors some of them unrealistic; N/A = not available

Table 3.1: Tested alternatives of sample ascertainment, haplotyping methods and sample sizes.

For association analysis we used haplotype association with all haplotypes between 1 and 10 markers of length. This simplicity was a deliberate choice: the method easily scales up to very large data sets, and it is also relatively powerful as will be shown by comparisons to exhaustive allelic transmission disequilibrium test (EATDT) [86].

3.1.1 Simulation

We used a two-phase simulation procedure to mimic a founder population (see Section 2.3). In the first phase we simulated the founder haplotypes using a coalescent model with recombination and infinite sites mutation model. This ensured a realistic polymorphism structure and realistic allele frequency distributions for the founder chromosomes. In the second phase the final population was simulated forward in time to obtain a larger population with a realistic recombination history.

The population model is an exponentially and rapidly growing founder population starting from 100 founder individuals randomly chosen from the coalescence simulation. The final size, 100,000 individuals, is reached in 15 generations. This approximately corresponds to a recently founded subpopulation living in isolation, such as Kainuu region in North-Eastern Finland [79].

The marker map consists of SNPs separated by approximately 33 kb from each other corresponding to the average density of a genome-wide 100k microarray SNP chip. We simulated 451 markers which is equivalent to a sequence of 15,000 kb (15 cM). The SNPs were chosen to have minor allele frequency at least 0.05. As a result of picking markers based on these two criteria, the mean minor allele frequency was 0.21 and the mean distance between markers was 33 kb (with 2.63 kb standard deviation on average).

	Disease model		
Parameter	Common	Intermediate	Rare
Susceptibility allele frequency $\Pr(M)$	.2000	.1000	.0100
Carrier frequency in population $\Pr(M^+)$	.3600	.1900	.0199
Penetrance $\Pr(D^+\|M^+)$	.2000	.3000	.4000
Prevalence $\Pr(D^+)$	.1000	.1400	.0500
Phenocopies $\Pr(M^-\|D^+)$	.2800	.5929	.8408
Penetrance for non-carriers $\Pr(D^+\|M^-)$	.0438	.1025	.0429

Legend: D = affected; M = disease-susceptibility allele; M^+ = susceptibility allele carrier (genotype MM or MW, where W is a wild type allele); M^- = susceptibility allele non-carrier (genotype WW)

Table 3.2: Disease model parameters.

Three disease models were designed to resemble interesting and challenging cases (Table 3.2). *Common* is a common disease variant with susceptibility allele frequency of 20 % in the population and low penetrance (20 %). It corresponds to a typical common complex disease, such as asthma or diabetes, in human populations. *Rare* is a model with low susceptibility allele frequency (1 %) and low prevalence (5 %) in the population with 40 % penetrance, in accordance to the "rare variant, rare disease" hypothesis. This corresponds to a typical inherited disease studied since the late 1990's. Finally, we created an *intermediate* disease model with medium penetrance (30 %) and susceptibility allele frequency (10 %) between those of the rare and common. All these models result in relatively difficult mapping problems where differences between methods and approaches can be more easily observed than with easier models.

After diagnosing the simulated individuals using the disease model, we ascertained samples of cases, random individuals and healthy controls (corresponding to the columns of Table 3.1). Siblings were not allowed in samples. To minimise the random effects caused by random sampling of individuals, the overlap of the samples for different strategies was maximised for a given replicate: the set of cases was identical in all strategies, and the healthy individuals of the case–random design were a subset of the controls of the case–control design. This held for both sample sizes. Further, the smaller sample was a subset of the larger one. Since we report results over 100 independent simulations in each setting, we believe the random effects are well controlled. We do not handle population stratification issues in this study so we did not simulate any population substructure or other confounding effects. Trio designs are generally robust to stratification effects, while in case–control studies false positives resulting from population substructure can be reduced by genomic control [25].

3.1.2 Haplotyping

Trio-based inference of haplotypes was done simply by deducing the phase of each marker from the genotypes of the child and the parents. If all members of the trio were heterozygous at a marker, the respective alleles of the child and the pseudo-haplotype were denoted unknown. This caused a part of the data to be missing. More complex methods for haplotype inference would estimate the phases of these markers and eventually improve the quality of the haplotypes. We also used the best possible case, true haplotypes, to approximate an upper limit for trio-based haplotyping.

For population-based statistical reconstruction of haplotypes we used *HaploRec* [34, 35] that is targeted especially for large numbers of relatively sparsely spaced markers. HaploRec assumes Hardy–Weinberg equilibrium: the probability of a haplotype pair is modelled as the product of the probabilities of the two individual haplotypes. The probability of a haplotype is broken into a product of conditional probabilities of individual alleles with each allele conditional on a varying number of its immediate neighbours. When computing the probability of haplotype H with length l, the distribution of alleles at marker i is estimated by conditioning on the longest observed haplotype fragment that (1) matches haplotype H and ends at marker $i-1$, and (2) has an estimated relative frequency of at least 0.2 %:

$$\Pr(H) = \Pr\big(H(1)\big) \prod_{i=2,\ldots,l} \Pr\big(H(i)|H(s_i, i-1)\big) \tag{3.1}$$

where $H(i)$ is the allele at marker i, $H(i,j)$ is the haplotype fragment covering markers $i,\ldots,j$, and $s_i = \min\{s|\Pr\big(H(s,i-1)\big) \geq 0.002\}$. Using a frequency threshold is motivated by the fact that a long haplotype fragment is likely to be shared by several individuals only if it is inherited from the same ancestor, and thus is useful in estimating haplotypes.

Given parameter estimates for the variable-order Markov model, the haplotypes are reconstructed by choosing the phases such that the resulting pair of haplotypes has the maximal product of probabilities. As in practise neither the model parameters or the haplotypes are known in advance, the original HaploRec algorithm was adapted to apply an EM-like algorithm for simultaneously learning the model and reconstructing the haplotypes. The algorithm starts with an uniform model and alternates between steps of reconstructing the haplotypes and estimating the model parameters.

3.1.3 Association analysis

Allelic association was estimated by predicting the disease-susceptibility locus to be at the marker that has the highest value for the χ^2 test statistic. For haplo-

type association each haplotype of length 1 to 10 markers was evaluated with the χ^2 test. The disease-susceptibility gene was predicted to reside in the middle point of the best haplotype. In case of a draw one of the best markers or haplotypes was chosen at random.

The p-value of the best allele or haplotype was computed for each data set by a permutation test: the disease-association statuses of the original haplotypes were randomly shuffled 9,000 times, the χ^2 values were recalculated each time and the best χ^2 value of each permutation was used as the test statistic. The procedure calculates a corrected p-value against multiple testing of haplotypes [117].

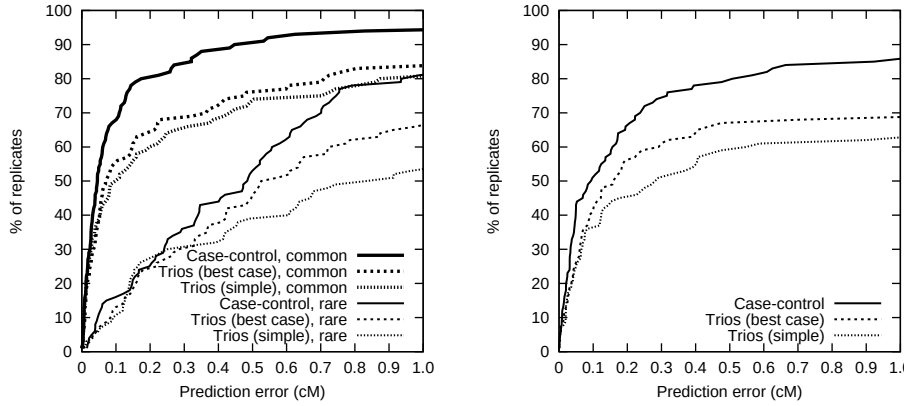
We estimated the statistical power to detect the disease-susceptibility gene at significance level $\alpha = 0.05$. Because the simulated chromosomes were only 15 cM long—approximately $1/187$ of the human genome—we adjusted the p-values for genome-wide analysis by Šidák-correction, essentially assuming that the genome consists of 187 independent 15 cM blocks. The genome-wide p-value p^* was estimated by $p^* = 1 - (1-p)^{187}$, where p is the p-value obtained for the simulated 15 cM region by permutation tests.

In addition to the statistical power to detect the disease-susceptibility gene, the haplotype association gives a point estimate for the DS locus. The results over 100 simulation replicates form a sample of prediction errors. The accuracy of association analysis can be then visualised as a cumulative distribution function of absolute prediction error.

Throughout the results the maximum prediction error shown will be 1,000 kb (1 cM in the simulated data), as prediction errors over 1 cM to either side of the true disease-susceptibility gene location are not particularly interesting for fine resolution mapping. With the population and disease models used the methods are reasonably accurate so that most of the interesting differences lie within this range. With our disease models, detecting the presence of a gene is more difficult than locating it and, correspondingly, the powers are much less than one. However, this design is intended to bring out differences between the methods.

3.2 Results

A direct comparison of the case–control and trio designs, under the assumption of equal genotyping costs, shows that the case–control design is more accurate for association analysis by a large margin across the three different disease models (Figures 3.1a and 3.1b). Even in the best possible case, with true haplotypes, the accuracy of the trio design was inferior to the case–control designs. The differences in the statistical power to detect the gene are even more striking (Table 3.3): the powers are 0.72 vs. 0.07 (rare), 0.48 vs. 0.08 (intermediate) and 0.76 vs. 0.30 (common) for the case–control and trio designs, respectively. Using the true hap-



(a) Common and rare models

(b) Intermediate model

Figure 3.1: Prediction errors. We assume equal genotyping costs, so the case–control sample contains 500 individuals and the trio-based sample has 167 trios. Case–control samples were haplotyped with HaploRec. "Best case" refers to true haplotypes and "simple" refers to simple trio-based haplotyping.

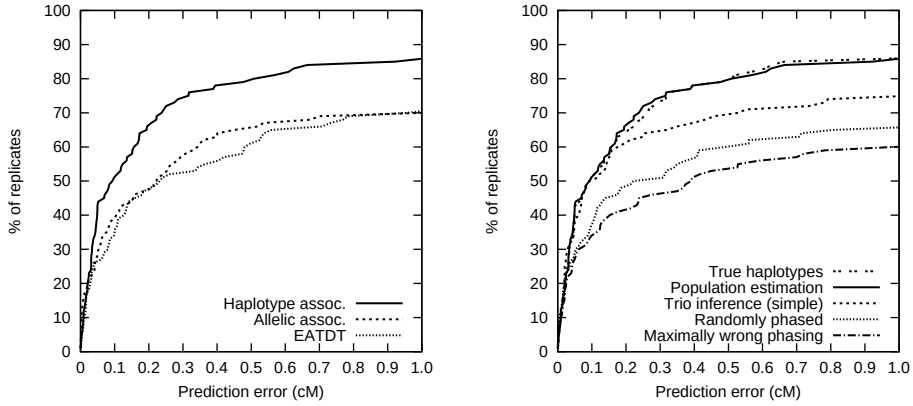
lotypes did not significantly improve powers in the trio designs.

To validate the use of the simple haplotype association mapping method in this study we compared the results to those obtained with allelic association and EATDT (Exhaustive Allelic Transmission Disequilibrium Test, applicable to trio-based data only) [86]. The haplotype association method used in this study is on par or even outperforms allelic association, EATDT (Figure 3.2a and Table 3.3, last two lines), HPM [124] and TreeDT [117] (results not shown) indicating that it is powerful and accurate enough for measuring the differences between different study designs and haplotyping approaches.

We next perform a detailed experimental analysis of the differences between the case–control and trio-based approaches. The effects caused by sample size, sample ascertainment and haplotyping method are isolated by studying each of them separately. For illustration, we only show prediction accuracy curves for the intermediate disease model (the most difficult one). The power and prediction results for all three disease models are summarised in Table 3.3.

3.2.1 Sample size

The most obvious cause for the performance differences above is the sample size: with equal genotyping costs the effective sample size in the trio-based approach is



(a) Association mapping method

(b) Haplotyping method

Figure 3.2: Effect of methods on prediction error. Disease model is intermediate with case–control sample of size 500 except for EATDT which operated on 500 trios. HaploRec was used for haplotyping in (a).

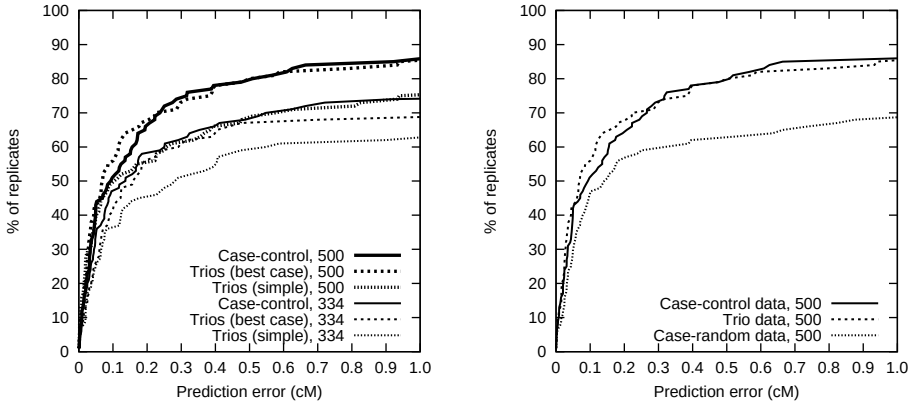
only two thirds of that in the population-based approach. In the case of power this is expected, as the sample size is usually the most critical single factor affecting study power. However, the difference in sample size only explains a part of the difference (Figure 3.3a, Table 3.3).

3.2.2 Sample ascertainment

In the trio-based approach the non-transmitted chromosomes of the parents are used as additional data: as a pseudo-control if the parent is healthy, or as a pseudo-case if the parent is diseased. This procedure often results in somewhat unbalanced numbers of cases and controls, but this does not seem to have much effect on the mapping accuracy (results not shown).

To test the effect of sample ascertainment we conducted experiments where the sample size and haplotyping method were fixed, and only the origin of controls varied. In particular, population-based controls were haplotyped using their parents (trio-based haplotyping) in some experiments. According to the results, there is virtually no difference in the prediction accuracies between the two major sample ascertainment methods (Figure 3.3b). In terms of the power to detect a gene population-based controls tend to be more powerful than trio-based data (Table 3.3).

Sampling random individuals instead of healthy ones has a clear negative ef-



(a) Sample size

(b) Sample ascertainment

Figure 3.3: Effect of sampling on prediction error. Case–control samples were haplotyped with HaploRec in (a) and known haplotypes were used in (b).

fect. For the most difficult disease model, the intermediate one, the effect is comparable to the effect of sample size. For easier models the effect is smaller. A trio-based strategy where all non-transmitted haplotypes are labelled as controls yields similar results compared to sampling random individuals (results not shown).

3.2.3 Haplotyping method

The final aspect in which the case–control and trio-based approaches differ is the haplotyping method. Population-based reconstruction is based on estimated phases which can contain errors, especially with the long maps used in this study (451 markers, 15 cM). The simple trio-based approach we used cannot always resolve the phase of a marker; in such cases we marked the alleles as unknown. The true haplotypes used in our experiments represent the best case scenario for haplotyping, and they can be viewed as an upper bound for the performance of any trio-haplotyping method.

Controlled experiments, where other factors (sample size and sample ascertainment in particular) are constant, show that the mapping results obtained with population-based haplotyping are virtually identical to those obtained with the true haplotypes across all tests (Figure 3.2b, Table 3.3). In other words the (few) phasing errors did not affect mapping power or accuracy. The simple trio-based haplotyping is clearly inferior. This is explained by the amount of missing data since all inferred phases have to be correct. Trio-based haplotyping resulted in

Rare	Sample ascertainment and effective sample sizes					
	Case–control		Case–random		Trios	
Haplotyping	500	334	500	334	500*	334*
Estimated	**.72 (.81)**	.22 (.68)	.56 (.78)	.16 (.57)	N/A	N/A
Trio-based	.37 (.69)	.05 (.55)	.31 (.64)	.05 (.48)	.33 (.62)	**.07 (.53)**
Known	.74 (.79)	.17 (.68)	.56 (.75)	.11 (.59)	.62 (.74)	**.10 (.66)**
Random	.07 (.53)	.01 (.44)	.07 (.45)	.02 (.40)	.05 (.52)	.01 (.39)
Worst case	.08 (.45)	.01 (.29)	.05 (.37)	.01 (.27)	.03 (.48)	.00 (.30)
Allelic assoc.	*.06 (.47)*	.01 (.38)	.05 (.48)	.02 (.39)	.04 (.51)	.01 (.33)
EATDT	N/A	N/A	N/A	N/A	.49 (.84)	*.13 (.62)*

Intermediate	Sample ascertainment and effective sample sizes					
	Case–control		Case–random		Trios	
Haplotyping	500	334	500	334	500	334
Estimated	**.48 (.85)**	.21 (.74)	.20 (.68)	.08 (.52)	N/A	N/A
Trio-based	.39 (.74)	.13 (.56)	.17 (.61)	.06 (.43)	.25 (.75)	**.08 (.62)**
Known	.46 (.85)	.21 (.70)	.19 (.68)	.07 (.53)	.34 (.85)	**.11 (.68)**
Random	.22 (.65)	.06 (.52)	.13 (.57)	.04 (.38)	.06 (.48)	.02 (.35)
Worst case	.20 (.60)	.09 (.45)	.08 (.50)	.04 (.29)	.10 (.47)	.02 (.39)
Allelic assoc.	*.24 (.69)*	.07 (.55)	.08 (.56)	.06 (.38)	.06 (.62)	.04 (.47)
EATDT	N/A	N/A	N/A	N/A	.15 (.70)	*.07 (.47)*

Common	Sample ascertainment and effective sample sizes					
	Case–control		Case–random		Trios	
Haplotyping	500	334	500	334	500*	334*
Estimated	**.76 (.94)**	.47 (.87)	.59 (.95)	.25 (.84)	N/A	N/A
Trio-based	.72 (.94)	.46 (.82)	.54 (.91)	.30 (.75)	.61 (.95)	**.30 (.80)**
Known	.77 (.95)	.47 (.88)	.60 (.97)	.27 (.83)	.62 (.94)	**.29 (.83)**
Random	.61 (.85)	.35 (.73)	.39 (.80)	.20 (.69)	.34 (.83)	.19 (.68)
Worst case	.50 (.79)	.30 (.67)	.35 (.75)	.20 (.67)	.35 (.77)	.08 (.64)
Allelic assoc.	*.57 (.90)*	.33 (.83)	.42 (.87)	.27 (.73)	.36 (.82)	.19 (.70)
EATDT	N/A	N/A	N/A	N/A	.47 (.86)	*.23 (.79)*

Legend: * = numbers of genotyped individuals were 750 and 500; N/A = not available
Haplotyping methods: "Estimated" = population-based estimation with HaploRec [35]; "Trio-based" = trio-based haplotype inference; "Known" = Known (true) haplotypes; "Random" = Randomly phased haplotypes; "Worst case" = Maximally wrongly phased haplotypes

Table 3.3: The power to detect the disease-susceptibility gene using different sample ascertainment methods, haplotyping methods and sample sizes for rare, intermediate and common mutation models. Numbers in parenthesis are fractions of predictions for which prediction error was less than 1 cM. Results from allelic association and EATDT [86] are included for comparison; their primary alternatives are printed in italics.

5.4 % missing alleles in our simulated data sets on average. In the best case a sophisticated trio-based haplotyping method would give mapping accuracy equal to the true and the estimated haplotypes (Figure 3.3a). However, the power of the trio design to detect a gene remains inferior to the case–control design even in this case (Table 3.3).

3.2.4 Asthma data

We demonstrate the power of the case–control design and population-based haplotyping on real asthma data consisting of 194 small families [80]. Our goal is to compare different approaches with this real data, not to reproduce the original results of Laitinen et al. The subjects were genotyped at 91 microsatellite and 64 SNP markers spanning a 20 cM region in chromosome 7. The original genotyping process had been iterative: new markers were genotyped from the regions where the intermediate analyses showed strongest associations. In the final stage a 133 kb risk-conferring haplotype was identified [80].

To mimic the case–control study design we used families with an affected and a healthy parent constituting an effectively independent case–control pair. For the trio-based approaches we randomly sampled one child from each family. We sampled at most one such trio per family; at the end, we got 93 trios from the available 194 families.

All markers with at least 20 % of genotypes missing were rejected. The remaining marker map consisted of 73 microsatellite and 15 SNP markers. We haplotyped the case–control data set of affected and healthy parents using HaploRec. For comparison we inferred the haplotypes of a random subsample of 62 trios (two thirds of 93) utilising the genotypes of the children. Additionally, we subsampled the haplotypes of the parents in these 62 trios from the set of 93 case–control pairs haplotyped using HaploRec. This data set represents our best estimate of the haplotypes in the trios.

With the case–control sample there is a strong association peak within the correct 133-kb region, but also a false positive about 500 kb to the right (Figure 3.4a). The trio-based setting does not show any associations within the correct region, but two false positives instead. Increasing the sample size to include all 93 trios did not produce different results (data not shown). The results suggest that the case–control approach is feasible even with relatively small samples in fine mapping.

3.3 Conclusions

We reported on simulation experiments where we compared sample ascertainment strategies and related haplotyping approaches in association mapping studies. We

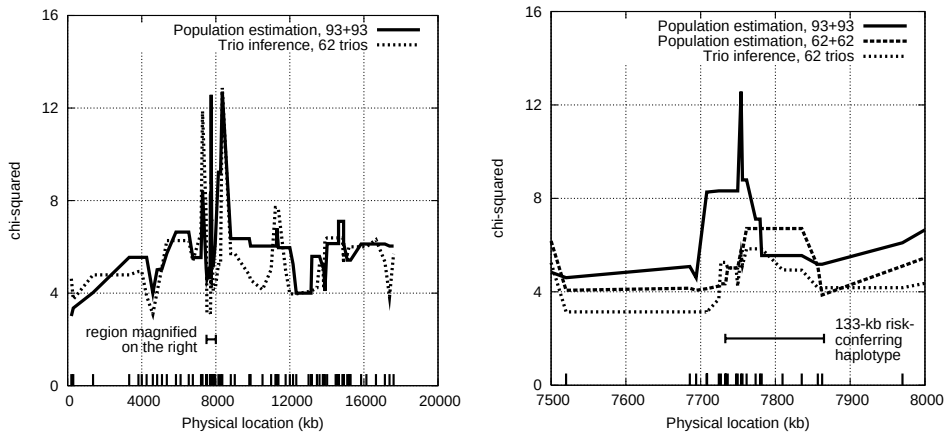


Figure 3.4: Haplotype association in asthma data with the population-based and trio-based approaches. (a) Entire 20 cM region. (b) 500 kb region containing the identified 133-kb haplotype (7733–7865 kb). The curves show the highest χ^2 value for each marker from the set of all haplotypes with 1–10 markers spanning over the marker. The locations are reported relative to an arbitrarily chosen origin. The bars at the bottom show the locations of the markers.

considered two main alternatives: ascertainment of family trios with an affected child from which it is easy to partially infer the haplotypes, and ascertainment of a case–control sample of unrelated individuals for which the haplotypes were estimated statistically. We conducted the mapping step using haplotype association analysis, a simple but powerful and efficient method. Case–control samples were haplotyped with HaploRec [35]. Both haplotype association and HaploRec scale up to large amounts of markers and individuals and are suitable for high-throughput association studies. Finally, we isolated and experimentally analysed the effects of three separate factors: sample size, sample ascertainment method and accuracy of haplotyping.

For an equal number of genotyped individuals the effective sample size in the trio-based approach is only two thirds of that of the case–control design. According to our experiments, this difference has a major effect on the mapping power and accuracy as expected. With equal sample sizes the case–control setting is more powerful than the trio-based design, but they are roughly equally accurate. The case–random design was clearly inferior, but this depends on the disease model and, in particular, the susceptibility allele frequency. Trios have the benefit that they produce controls that are well paired with cases for population substructure and other factors that are not uniform over the population. Ascer-

tainment of matching controls from the population is more difficult, but genomic control can be used to correct for stratification [25].

Our experiments suggest that controls should be ascertained from unaffected individuals to maximise the power and accuracy of the study. In a population-based study it can be possible to ascertain healthy controls for this purpose. However, random population controls are often used since their ascertainment is easier especially if they are stratified to match the cases. In our experiments using random population controls led to weaker power, but the mapping accuracy was less affected.

Surprisingly, haplotypes inferred with a population-based statistical method were found out to be as powerful as the corresponding true haplotypes. This is in contrast to the result of Morris and others who concluded that statistically inferred haplotypes are inferior to the true haplotypes and genotype-based approaches should be preferred instead [94]. There are several differences between our studies that can partially explain the results. First, the results of Morris et al. are based on a shorter map (950 kb interval and 20 SNPs vs. 15,000 kb and 451 SNPs in our study). Second, Morris et al. used their own COLDMAP software for mapping. We could not evaluate the effect of this choice since, according to Morris et al. themselves, COLDMAP is not feasible for data sets of the size in our study. Third, they used SNPHAP[1] by David Clayton to estimate haplotypes. SNPHAP is not well suited for long maps with recombinations and is likely a suboptimal choice here (we could not evaluate it in our study since it often stopped without haplotyping all individuals). Finally, Morris et al. applied different simulation techniques.

Despite some contrasts with the conclusions of Morris et al. [94] we do agree with many of their points. In particular, mapping results based on inferred haplotypes are likely to be optimistic in terms of confidence or credibility intervals due to exaggeration of linkage disequilibrium.

With the trio-based haplotyping the only source of uncertainty are similar heterozygotes whose alleles were marked unknown by our simple method. This resulted in a poorer mapping performance compared to the population-based estimates. An obvious recommendation is to use population-based techniques to augment trio-based inference when trios are available to avoid missing alleles. However, our experiments with the true haplotypes suggest that even the best possible haplotyping method would give the same mapping accuracy than the population-based haplotyping, and the power to detect the gene would remain inferior to the case–control design due to the smaller effective sample size.

It is important to test the utility of the case–control study design in real life

[1] `http://www-gene.cimr.cam.ac.uk/clayton/software/snphap.txt` (referred on 25 February 2011)

experiments to verify that it does not suffer from unexpected effects from genotyping errors, missing data or varying disease models. The effect of more elaborate association methods (e.g. those proposed by Purcell et al. [105]) to the power and mapping accuracy should be investigated, although our comparisons to EATDT suggest that the simple haplotype association with a population-based haplotyping is a powerful option. We leave these issues for future research.

In summary we suggest that case–control study designs, preferably with familial cases and reliably trait-excluded controls, with efficient population-based haplotyping method may serve as powerful starting points for genetic association analyses. With case–control study there is no obligatory need to genotype families. This is beneficial as the total cost of the gene mapping pipeline (Figure 1.1) can be significantly reduced, assuming equal genotyping costs. Also, the refined analysis of putative DS genes becomes easier as the number of false positives is reduced. Finally, it seems that a relatively simple but efficient haplotype association can be sufficiently powerful in high-throughput analysis.

HaploRec software for population-based reconstruction of haplotypes and the simulated data sets are available on the WWW[2].

[2] `http://www.cs.helsinki.fi/group/genetics` (referred on 25 February 2011)

Chapter 4

Link discovery in biological graphs

In this chapter we focus on the refined analysis phase of the gene mapping pipeline (Figure 1.1). We consider specifically the first step of the refined analysis: how to *prioritise* putative disease-susceptibility (DS) genes so that further efforts can be focused on the most promising candidates? One approach is to look at what is already known about the putative DS genes and see how they relate to each other and to the phenotype under study. This might reveal evidence for the hypothesised association or facilitate a more detailed hypothesis about the mechanisms of the relationship.

More generally, many domains in contemporary data mining—such as social networking, collaboration (or affiliation) networks and World Wide Web—involve *relational data*. Such data is usually *heterogeneous*: entities and relations between them have different types and attributes. *Graphs* are natural and useful models for representing relational and heterogeneous data. We focus on *biological graphs* whose entities (vertices) come from the biological or biomedical domain [11, 116, 48]. Entities in biological graphs can include concrete biological concepts such as genes, proteins and tissues, but also abstract concepts such as biological processes, phenotypes and scientific articles. Relations (edges) between these entities correspond to real-world phenomena such as "a gene codes for a protein" or "an article refers to a phenotype".

We investigate link discovery in biological graphs with the primary aim of prioritising putative DS genes. We use term *link* loosely to refer to any (direct or indirect) connection between two vertices in a graph. Recall that the initial analysis phase may have produced a large set of putative genes linked to the (disease) phenotype. Before moving on with the refined analysis the investigators compare the putative genes based on what is known about them in the public databases and literature. This way false positives can be detected and the efforts and resources can be concentrated on the most promising candidates. Due to the lack of automated methods the work is mostly done by manually browsing the databases. This

is a slow and laborious process which necessarily limits the extent and coverage of the search. In this chapter we describe methods for (partial) automation of the prioritising task by focusing on gene–phenotype links and their relative strengths. Methods for automated discovery and analysis of connections between a putative gene and a phenotype have only recently started to emerge [127, 58, 104, 74].

Biological graphs can be built from publicly available biological databases. Converting (relational) biomedical knowledge to a graph form is conceptually simple though not straightforward. For instance, how to map different biological concepts and their attributes into the graph and how to weight edges is nontrivial. In Sections 4.1 and 4.2 we consider these issues in the context of Biomine, a relatively large biological graph. In Sections 4.3 and 4.4 we review some proposed link goodness measures and consider the evaluation of link significance. We conclude the chapter in Section 4.5 by re-enacting experimental results by Sevon et al. [116] that demonstrate the effectiveness of the chosen graph model and two link goodness measures that are suitable for Biomine graphs.

4.1 Biomine database

As a concrete example of modelling relational biological data we use *Biomine*: a large index of various interlinked public biological databases. Biomine offers a uniform view to these databases by representing their contents as a large, heterogeneous random graph. Vertices in this graph represent entities (records) in the original databases, and edges represent their annotated relationships (cross-references between records). Edges have weights that are interpreted as probabilities. Preliminary version of Biomine has been described by Sevon et al [116]. In this section we take a brief look at the core components of Biomine: its data model and source databases. Edge weighting is considered separately in Section 4.2.

4.1.1 Data model

The choice of *data representation*, or *data model*, is important in link mining [47]. Consider, for instance, a simple bibliography database which indexes authors, articles and journals. One natural representation would map entities as separate vertex types by creating a vertex for each author, article and journal. These vertices would be then connected by edges according to the authorship and containment relations. This representation emphasises journal-centric view on data as journal vertices would likely be major large-degree hubs in the resulting graph. A more condensed representation could include only author vertices connected by edges (or hyperedges) if the corresponding authors have coauthored at least one article. Article and journal information could be included as edge attributes. This

representation resembles social network, and it might be useful if we are mostly interested in coauthor relations.

With multiple heterogeneous biological databases the number of possible data models becomes immense. To facilitate wide applicability, the core Biomine data model is deliberately simple: all source database records are represented as vertices in an undirected, labelled and weighted multigraph $G = (V, E)$. The elements of the vertex set V are biological entities such as genes, proteins and biological processes as well as more general objects like article abstracts. They are labelled by a type, such as *gene* or *protein*, from set T_v. We denote the vertex type mapping by $t_v : V \mapsto T_v$.

Edge multiset $E \subset [V]^2$ consists of unordered vertex pairs $\{u, v\}$. As with vertices, edges have labels from edge type set T_e, and we denote this mapping by $t_e : E \mapsto T_e$. Edge types depict annotated relations between vertices, such as *codes for* (e.g., gene *codes for* protein) or *refers to* (e.g., article *refers to* gene). Each edge has a source database where the corresponding relation resides. We denote this source database mapping by $s : E \mapsto \mathcal{D}$ where $\mathcal{D}$ is the set of source databases.

For a given graph $G = (V, E)$, we refer to its vertex set V by $V(G)$ and its edge set E by $E(G)$. We define a *path* in a slightly non-standard manner as a sequence of consecutive edges (instead of vertices). Finally, we denote the set of neighbouring vertices of v by $N(v) = \{u \in V : \{v, u\} \in E\}$.

Table 4.1 lists the vertex types used in Biomine; similarly, Table 4.2 lists the edge types. Some representative examples of typed edges are given in Table 4.3. All tables refer to the Biomine database built on 4 June 2010.

4.1.2 Source databases

Biomine consists of several interlinked, publicly available *source databases*. Each database provides different kinds of entities and relations to Biomine, some overlapping. We briefly review the main features of the source databases below.

NCBI's *Entrez Gene* [87, 112] provides gene entries for different organisms. Currently, Biomine contains five model organisms: human, mouse, rat, fruit fly and nematode (*Caenorhabditis elegans*). Genes are connected to their protein products and other homologous genes (similar genes in different organisms). Homology relations come from an another Entrez database *HomoloGene* [112].

UniProt [22] is the main source of protein-related information. Its core elements are proteins, pathways and tissues. These elements form vertices in the graph. Manually annotated and reviewed proteins are in *Swiss-Prot* subdatabase, while *TrEBML* subdatabase contains automatically annotated and nonreviewed proteins. UniProt contains many relations, such as protein interactions and expressions, and classifications into protein families and pathways.

Type	Primary source database	Amount	Mean degree
Active site	InterPro	89	95.82
Allelic variant	OMIM	19,455	1.44
Article	PubMed	532,675	3.98
Binding site	InterPro	62	111.18
Biological process	GO	19,539	32.47
Cellular component	GO	2,856	122.18
Compound	KEGG	15,879	0.55
Conserved site	InterPro	575	58.29
Domain	InterPro	5,515	69.55
Drug	KEGG	8,846	0.69
Enzyme	KEGG	5,095	10.15
Family	InterPro	12,718	10.61
Gene	Entrez Gene	192,893	18.60
Gene/Phenotype	OMIM	343	82.35
Genomic context	Entrez Gene	11,825	18.68
Glycan	KEGG	2,519	0.92
Homolog group	HomoloGene	25,780	3.18
Molecular function	GO	9,529	49.07
Ortholog group	KEGG	13,067	3.81
Pathway	UniProt	1,875	37.10
Phenotype	OMIM	6,559	16.95
PTM	InterPro	16	82.88
Protein	UniProt	275,292	29.58
Region	InterPro	1,441	20.14
Repeat	InterPro	255	94.84
Tissue	UniProt	1,317	189.10
		total 1,166,020	14.84

Table 4.1: Biomine vertex types T_v, primary source database for each type, and the total amount and mean degrees of corresponding vertices.

Type	Source databases	Amount
affects	Entrez Gene	5,077
belongs to	Entrez Gene, HomoloGene, KEGG, STRING, Swiss-Prot, TrEMBL	689,026
codes for	Entrez Gene, KEGG, STRING	174,480
contains	Swiss-Prot, TrEMBL	454,553
functionally associated to	STRING	2,916,286
has	Entrez Gene, InterPro, KEGG, OMIM, Swiss-Prot, TrEMBL	464,369
has synonym	Entrez Gene	1,666
interacts with	Entrez Gene, Swiss-Prot, TrEMBL	97,361
is a	GO, InterPro, KEGG	51,483
is expressed in	Swiss-Prot, TrEMBL	234,153
is found in	Entrez Gene, InterPro, KEGG, Swiss-Prot, TrEMBL	337,542
is homologous to	HomoloGene	259,390
is located in	Entrez Gene, OMIM	144,495
is part of	GO, InterPro, OMIM	54,196
is related to	GO, HomoloGene, KEGG, OMIM, Swiss-Prot, TrEMBL	25,414
overlaps	OMIM	8,199
participates in	Entrez Gene, InterPro, KEGG, Swiss-Prot, TrEMBL, UniProt	605,237
refers to	Entrez Gene, KEGG, OMIM, Swiss-Prot, TrEMBL	2,216,614
subsumes	Entrez Gene, KEGG, STRING, Swiss-Prot, TrEMBL	140,555
targets	KEGG	4,885
		total 8,884,981

Table 4.2: Biomine edge types T_e and amount of edges of each type.

Edge	Source database	Amount
Gene *codes for* Protein	STRING	5,948
Protein *belongs to* Family	Swiss-Prot	30,651
Family *participates in* Biological process	InterPro	5,274
Biological process *is related to* Tissue	GO	13,103
Protein *is expressed in* Tissue	Swiss-Prot	176,034
Enzyme *subsumes* Protein	TrEMBL	7,907
Gene *codes for* Enzyme	KEGG	14,195

Table 4.3: Some examples of Biomine edge types, their source databases and the amount of corresponding edges. Observe that a sequence of such edges would constitute a gene–gene-path in the graph.

InterPro [62] is another protein-related database. It indexes protein families and structural elements (domains, regions, sites, etc.), and it has hierarchies for these elements. The third protein database, STRING [65], contains known and predicted protein–protein interactions. The interactions include direct (physical) and indirect (functional) associations. STRING also contains clusters of orthologous groups (COGs) and their interactions, with mappings between proteins and COGs.

Gene Ontology (GO) aims to provide a controlled vocabulary for genes and gene products [21]. Its core domains are cellular components, biological processes and molecular functions. The ontology is structured as a directed acyclic graph, and each term has defined relationships to one or more other terms in the same domain and sometimes to other domains. This graph is a subgraph of Biomine, and the term vertices are referred to by other databases such as Entrez Gene and UniProt.

Online Mendelian Inheritance in Man (OMIM) is a catalogue of human genes and genetic disorders [100]. It is the main source of phenotype information in Biomine: most of the OMIM entries are *Phenotype* vertices. The database also contains descriptions of allelic variants, gene locations and a large number of references to biomedical literature.

PubMed [112] is a freely accessible online database of biomedical journal citations and abstracts with approximately 20 million entries. Many biological databases (such as UniProt and OMIM) contain references to PubMed entries, for example to index articles where a particular gene or phenotype is mentioned. In Biomine these cross-referenced PubMed entries are *Article* vertices.

Kyoto Encyclopedia of Genes and Genomes (KEGG) is a large, integrated database resource consisting of 16 main databases broadly categorised into systems information, genomic information and chemical information [66]. Biomine uses a subset of KEGG: its pathway, gene, drug, orthology, compound and glycan databases.

Each source database has its own schema for arranging and formatting data. Schemata are generally mutually incompatible and contain more information than we are interested in. Raw data files are therefore preprocessed into a simpler, uniform "intermediate" schema by dedicated database-specific parser programs before the actual integration. The resulting files are essentially lists of typed edges, vertex attributes and synonym mappings. These files are then imported into a relational database to form a large graph. During the importing process synonyms, invalid references and other anomalies are resolved. The complete conversion and importing process is complicated and out of the scope of this thesis.

4.2 Edge goodness in Biomine

One of the goals of Biomine is to allow discovery and evaluation of links between vertices specified by the user. To rank paths or assess the significance of a connection between two vertices we need a measure for edge goodness. Edges sometimes have natural weights in the source databases. For example, a homology between two proteins could have a value denoting the degree of sequence similarity. Biomine extends such domain-specific static weighting by considering edge weight, or *goodness*, as a function of three factors:

1. *Reliability.* How confident are we that the relation (and consequently the edge) really exists? How reliable is the data source, how reliable is the method used to produce or predict the edge and how strong or probable is the connection estimated to be in the data source?

2. *Relevance.* How relevant is the edge with respect to the query? We assume that the investigator can give query-specific weights for vertex and/or edge types according to his or her subjective opinions of the importance of each type for the query at hand.

3. *Rarity.* How rare and informative is the edge? As an extreme example, an article [46] that refers to over 18,000 human and mouse genes is not likely to be relevant for a specific gene whereas an article that only refers to few genes is much more likely to be informative. In Biomine edge rarity is directly related to the degrees of its incident vertices.

A distinguishing feature of Biomine is the probabilistic interpretation of the above factors: an edge $e \in E$ is considered to be reliable with probability $r(e)$, relevant with probability $q(e)$ and rare (or *informative*) with probability $d(e)$. These factors are combined to a single probability $g(e)$ so that e is an existing and potentially useful relation if e is at the same time reliable, relevant and informative. In other words, edges are random: e "exists" or "is true" with probability $g(e)$, or "does not exist" or "is not true" with probability $1 - g(e)$. With the probabilistic interpretation G is a random graph that naturally models the uncertainty in the source data and the query-specific relevance. We next give definitions for r, q and d, and we combine them into one goodness g.

Reliability $r(e)$ of an edge $e \in E$ is defined as a product of two (independent) reliabilities: a database reliability $r_d : \mathcal{D} \mapsto [0,1]$ and a relation (edge) reliability $r_r : E \mapsto [0,1]$. The database reliability r_d is given by the user, and the interpretation of r_d is the degree of belief the user has for a relation being correctly annotated in the corresponding database. For example, manually curated Swiss-Prot database could be given a perfect reliability by letting $r_d(\text{Swiss-Prot}) = 1.0$,

while computer-annotated TrEMBL database could be assumed to be less precise by letting $r_d(\text{TrEMBL}) = 0.75$. Relation reliability r_r comes from the source database instead: if there is a separate confidence value c associated to e (that reflects similarity or homology score, for example), we let $r_r(e) = c$, where c is scaled between 0 and 1 if needed. Otherwise we let $r_r(e) = 1$. The interpretation of $r_r(e)$ is the confidence of the data source itself on the relation represented by e.

We define the *edge reliability* $r : E \mapsto [0,1]$ by treating the reliabilities r_d and r_r as probabilities of independent events:

$$r(e) = r_d\big(s(e)\big) \cdot r_r(e) \tag{4.1}$$

where $s(e)$ is the source database of e. The interpretation of $r(e)$ is that e is reliable if both the database (as a whole) and the annotation are considered reliable.

Relevance $q(e)$ of an edge $e = \{u,v\} \in E$ is the degree of belief that e represents a relevant connection between vertices u and v with respect to the current query. Edge relevance is analogous to edge reliability r but, in contrary to the static database-related reliability, relevance is query-specific.

Relevance values may be sometimes easier to give in terms of vertex types instead of edge types. Hence Biomine uses two relevance functions: $q_v : T_v \mapsto [0,1]$ for vertex types and $q_e : T_e \mapsto [0,1]$ for edge types. Both q_v and q_e are given by the user. A practical implementation could have a default configuration for both q_v and q_e, so only few adjustments would be needed for a typical query.

As in (4.1), relevance values q_v and q_e are treated as probabilities of independent events. The *edge relevance* $q : E \mapsto [0,1]$ is

$$q(e) = q_e\big(t_e(e)\big) \cdot \sqrt{q_v\big(t_v(u)\big)} \cdot \sqrt{q_v\big(t_v(v)\big)} \tag{4.2}$$

where $e = \{u,v\} \in E$, and t_e and t_v are the edge and vertex type mappings as above. Vertex relevance coefficient $\sqrt{q_v\big(t_v(x)\big)}$ in (4.2) decomposes the vertex type specific relevance $q_v\big(t_v(x)\big)$ of vertex x for each of its adjacent edges. As path relevance will be later defined as a product of edge relevance values this gives the desired outcome: the relevance of any path visiting a vertex of type τ is multiplied by $q(\tau)$.

We want to give lower scores for paths that visit vertices with high degrees: the higher the degree of vertex $v \in V$ the less likely it is that any two neighbours of v actually have an interesting (non-random) connection through v. For example, the aforementioned article [46] that refers to 18,000 genes is likely to connect trivially every gene vertex to every other gene vertex. Hence we define *rarity* d_v : $V \mapsto [0,1]$ first for vertices. Rarity $d_v(v)$ represents the probability that any two edges incident on v are related to each other and represent a meaningful path; the higher the rarity, the more informative v is.

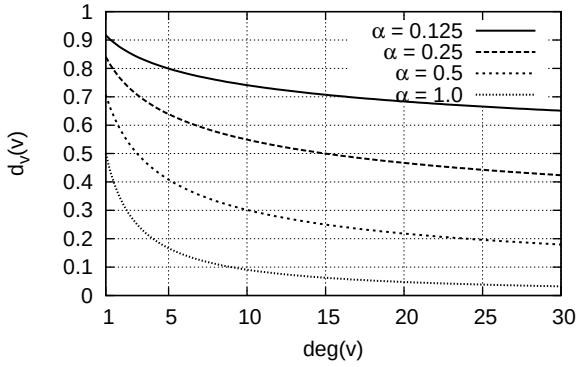


Figure 4.1: Rarity function d_v for various values of α.

The following ad hoc formula is used as a basis for rarity:

$$d_v(v) = \frac{1}{(\deg(v)+1)^{\alpha}} \tag{4.3}$$

where $0 \leq \alpha \leq 1$ is a *penalising parameter*. It determines how steeply d_v decreases as a function of vertex degree. Figure 4.1 illustrates d_v for some values of α and $\deg(v)$. With $\alpha = 0$ we have $d_v(v) \equiv 1$ so that all vertices are considered equally informative. With $\alpha = 1$ we have $d_v(v) = (\deg(v)+1)^{-1}$ and $d_v(v)$ has the following probabilistic interpretation. Consider a random walker who, at any vertex, is equally likely to follow any edge or stop at the vertex. Given a path with vertices $v_1, v_2, \ldots, v_k$, $v_i \in V$, rarity $d_v(v_i)$ is the probability that the walker who has so far traversed vertices $v_1, \ldots, v_i$ will next stay on the path and visit vertex v_{i+1}.

The simple formula (4.3) can be too inflexible in practise. Take for example PLA2G7: a widely studied asthma gene that has been referred in 97 articles. Because of these article links (4.3) would penalise PLA2G7 vertex severely. However, it has only one interaction link and it participates in three biological processes, so PLA2G7 could be informative when the investigator is mostly interested in gene–gene interactions or biological processes. Another issue is that (4.3) does not consider varying vertex degree distributions: some graphs may have a relatively large fraction of vertices with high degrees ("scale-free" or "power law" graphs for instance). In Biomine, vertex degrees vary wildly between different vertex types (see Table 4.1) but α is independent of vertex types. This causes unreasonable penalisation for some large-degree vertex types such as GO terms.

To allow more flexibility in degree penalising we replace the single constant α and vertex degree function deg with vertex-type and edge-type specific functions $\alpha : T_v \mapsto [0,1]$ and $\deg : V \times T_e \mapsto \mathbb{N}$ (that is, $\deg(v,\tau)$ denotes the num-

ber of edges of type τ adjacent to v). Now the vertex *rarity* $d_v : V \times T_e \mapsto [0,1]$ for vertex $v \in V$ is

$$d_v(v,\tau) = \frac{1}{(\deg(v,\tau)+1)^{\alpha(t_v(v))}}. \tag{4.4}$$

As with relevance (4.2), the rarity values are decomposed into edge-specific coefficients. The edge *rarity* $d : E \mapsto [0,1]$ becomes

$$d(e) = \sqrt{d_v\big(u,t_e(e)\big)} \cdot \sqrt{d_v\big(v,t_e(e)\big)} = \big[d_v\big(u,t(e)\big) \cdot d_v\big(v,t(e)\big)\big]^{-1/2} \tag{4.5}$$

where $e = \{u,v\} \in E$.

Now that we have defined all the components of edge goodness, the goodness $g : E \mapsto [0,1]$ itself is simply a product of those factors:

$$g(e) = r(e) \cdot q(e) \cdot d(e)$$

where $r(e)$, $q(e)$ and $d(e)$ are the reliability (4.1), relevance (4.2) and rarity (4.5) of an edge $e \in E$. Under the assumptions that $r(e)$, $q(e)$ and $d(e)$ are probabilities for mutually independent necessary conditions for the edge and that edges are independent of each other, the goodness $g(e)$ is the probability that e exists. We remark that these assumptions of independence are strong, and in some cases they are arguably unrealistic. However, independence allows us to calculate path and subgraph probabilities easily—we return to these in Section 4.3 and Chapter 5.

4.3 Link goodness measures

Link is a more general concept of connection than a simple relation (edge) between two vertices s and t. Links are useful since they can be used to model indirect, weak or otherwise non-trivial connections. Simple paths are intuitive and useful links, but shared neighbourhoods, connected subgraphs and random walks can also be used to represent links. To discover or predict links, assess their strengths or analyse statistical significances of links we need a measure for link goodness in addition to edge goodness.

We next give a short review of some link goodness measures proposed in the literature. They are presented in the order of increasing generality; more general measures utilise more information for determining the strength of a link. The discussion is not restricted to Biomine graphs, so $G = (V,E)$ refers to an arbitrary directed or undirected graph below. See Liben-Nowell and Kleinberg [85] for an experimental evaluation of many of these measures for link prediction.

4.3.1 Path and neighbourhood level

The shortest s–t-path P is a simple but efficient link type. Its length $w(P)$ is a natural measure for link strength:

$$g_s(s,t) = \min_{P \in \mathcal{P}} w(P) = \min_{P \in \mathcal{P}} \sum_{e \in P} w(e) \tag{4.6}$$

where $w(e)$ is the length (weight) of an edge $e \in E$, and $\mathcal{P}$ is the set of all s–t-paths in G. This measure is easy and efficient to calculate by any shortest path algorithm.

For random graphs, where edge "lengths" are probabilities, (4.6) does not make much sense. However, if edges are independent of each other, like in Biomine graphs, path "length" or goodness follows in a natural way. Let $P = (e_1, \ldots, e_k)$, $e_i \in E$, be a path in G. The *path goodness* $g_p : \mathcal{P} \mapsto [0,1]$ is

$$g_p(P) = \prod_{e \in P} g(e). \tag{4.7}$$

With the interpretation that $g(e)$ is the probability that edge e exists (Section 4.2) the path goodness $g_p(P)$ is the probability that the whole path P exists in a realisation H of G. A *realisation* of G is a non-random subgraph $H \subset G$ where each edge of G has been randomly and independently decided according to the corresponding edge probabilities (see Section 5.1.1).

With path goodness g_p the shortest path corresponds to the most probable, or *best* path. By combining (4.6) and (4.7) we get

$$g_b(s,t) = \max_{P \in \mathcal{P}} g_p(P) = \max_{P \in \mathcal{P}} \prod_{e \in P} g(e). \tag{4.8}$$

Again, any shortest path algorithm can be applied to find most probable paths by using edge weights $w(e) = -\log\big(g(e)\big)$. Let P be the shortest path found with weight $w(P)$. Then

$$w(P) = \sum_{e \in P} -\log\big(g(e)\big) = -\log\big(\prod_{e \in P} g(e)\big) = -\log\big(g_p(P)\big)$$

and since the logarithm function is strictly increasing and $w(P)$ is minimised, $g_p(P)$ is maximised.

Overlapping vertex neighbourhoods may indicate indirect similarity or proximity (consider, for instance, shared friends in social networks or shared colleagues in collaboration networks). The number of overlapping neighbours is the simplest measure in this context:

$$g_n(s,t) = |N(s) \cap N(t)|, \tag{4.9}$$

where $N(v)$ is the set of neighbours of v. This measure has been observed to positively correlate with future collaboration probability in coauthor networks [98]. The normalised form of (4.9)

$$g_J(s,t) = \frac{|N(s) \cap N(t)|}{|N(s) \cup N(t)|} \tag{4.10}$$

is the well-known *Jaccard index*. Adamic and Adar have proposed [1] a modification of (4.10) that rewards vertex pairs that share neighbours with low degrees:

$$g_A(s,t) = \sum_{u \in N(s) \cap N(t)} \frac{1}{\log |N(u)|}.$$

This measure gives more weight for pairs in small and independent cliques.

All neighbourhood level measures are based on the intersection of neighbour sets between two vertices s and t. More shared neighbours result in greater proximity. Direct (s,t) edges and links between neighbours do not contribute to the proximity. Neighbourhood level measures are thus probably more useful as complementary than standalone proximity measures.

4.3.2 Subgraph level

The goodness of a single s–t-path, as in (4.6) and (4.8), is not necessarily a good measure of the strength of the link between vertices s and t. For example, a link consisting of several parallel paths could be considered to be stronger than a single path even if all of the parallel paths are weak. *Connection subgraphs* take this into account by evaluating connected subgraphs, which can be thought to be a set of paths, containing s and t. Specifically, a connection subgraph between s and t is a connected subgraph $H \subset G$, of a given size, such that $\{s,t\} \subset V(H)$. Subgraph H can be, for example, a set of k shortest paths for some fixed k or it can be chosen to maximise a given connection subgraph goodness function [39].

Faloutsos et al. view G as an electrical network of resistors [39]. They propose an algorithm that extracts a fixed size subgraph H which maximises total delivered current over the subnetwork from s to t when s is assigned a potential of +1 volt and t is grounded (0 volts). Total delivered current has a random walk interpretation [75]. At first, let us define transition probabilities

$$p(u,v) = \frac{g(u,v)}{\sum_{w \in N(u)} g(u,w)} \tag{4.11}$$

for each $(u,v) \in E$. Next, let p_{esc} denote the escape probability according to (4.11) from s to t; i.e. the probability that a random walker starting from s will reach t

before returning to s. The *effective conductance* between s and t is now

$$g_{\mathrm{EC}}(s,t) = \sum_{u \in N(s)} g(s,u) \cdot p_{\mathrm{esc}} \tag{4.12}$$

which is the expected number of "successful escapes" when the number of escape attempts is $\sum_{u \in N(s)} g(s,u)$ [27].

Effective conductance is an appealing link goodness measure, and it has been used to measure centrality in networks [12]. However, it does not penalise uninformative vertices that have large degrees (cf. (4.4)). Faloutsos et al. dodge this by introducing a global grounded "sink" vertex that is connected to all vertices $v \in V$ with conductance proportional to $\sum_{u \in N(v)} g(v,u)$. As pointed out by Koren et al. [75], this introduces a counterintuitive size bias where the link goodness can decrease if the connection subgraph is enlarged. They propose a modified version of (4.12) titled cycle-free effective conductance (CFEC):

$$g_{\mathrm{CFEC}}(s,t) = \sum_{u \in N(s)} g(s,u) \cdot p_{\text{cf-esc}}(s,t) = \sum_{u \in N(s)} g(s,u) \cdot \sum_{P \in \mathcal{P}} \Pr(P) \tag{4.13}$$

where $p_{\text{cf-esc}}$ is the escape probability restricted to cycle-free random walks (walks that are simple s–t-paths), $\mathcal{P}$ is the set of all simple s–t-paths in G, and $\Pr(P)$ is the random walk probability along a path P. CFEC has two desirable properties: it is monotonically increasing as a function of graph size, and a relatively small connection subgraph consisting of the most probable simple s–t-paths is usually enough to approximate $g_{\mathrm{CFEC}}(s,t)$ [75].

4.3.3 Graph level

A link goodness measure can utilise the topology of the whole graph G. Most measures on this scale are based on random walks like (4.12) and (4.13), although a measure proposed by Katz [70] considers sets of s–t-paths such that

$$g_K(s,t) = \sum_{l=1}^{\infty} \beta^l |\mathcal{P}_l|$$

where $\mathcal{P}_l$ is the set of all s–t-paths of length l. Parameter $\beta > 0$ controls the effect of longer paths to the goodness.

Random walk models typically consider a single walker w starting from s or two walkers w_1 and w_2 with one starting from s and the other from t. Walkers traverse G randomly with transition probabilities (4.11). *Hitting time* $H(s,t)$ considers the expected number of steps w has to take to reach t [19]. Its symmetric variant is *commute time* $C(s,t) = H(s,t) + H(t,s)$. Both can be readily used as distance measures, and they have been used as link goodness (proximity) measures as well [85].

SimRank by Jeh and Widow [64] is based on a recursive definition

$$g_{\mathrm{SR}}(s,t) = \begin{cases} 0 & \text{if } N(s) = \emptyset \text{ or } N(t) = \emptyset, \\ 1 & \text{if } s = t, \\ C/\big(|N(s)||N(t)|\big) \cdot \sum_{\substack{u \in N(s) \\ v \in N(t)}} s(u,v) & \text{otherwise} \end{cases}$$

where $C \in [0,1]$ is a constant, and $N(v)$ is the set of neighbours of vertex v as before. SimRank also has a random walk interpretation: $g_{\mathrm{SR}}(s,t)$ corresponds to the expected value of $C^{\mathbf{t}}$ where $\mathbf{t}$ is the time (number of steps) when walkers w_1 and w_2 first meet [64].

Liben-Nowell and Kleinberg [85] proposed a *rooted PageRank* measure for link goodness based on well-known *PageRank* measure [13]. In rooted PageRank the random walker w returns to s with probability α in every step, or it continues the walk with probability $1-\alpha$. The measure is the steady state (stationary) probability of t.

With random graphs $g_b(s,t)$ is the probability that the best path exists in a realisation of G. A more appropriate measure could be the probability that at least one path exists between s and t. This measure is closely related to the theory of *network reliability* [20], and the desired measure

$$g_R(s,t) = \Pr(H : H \subset G, H \text{ contains an } s\text{–}t\text{-path}) \tag{4.14}$$

is the *two-terminal network reliability* of G with terminals s and t. (The connected parties are called terminals in the reliability literature.) We will return to the network reliability in Chapter 5, but for now it is sufficient to note that it can be readily used as a link goodness measure on random graphs.

4.4 Estimation of link significance

We eventually want to measure how strongly two given vertices s and t are related in graph G. Link goodness measures, such as those discussed in Section 4.3, allow ranking of links but their values may be difficult to put into perspective. For example, assume we have $f(s,t) = 0.4$ for some goodness measure f. Is this particular value of f high or low? This obviously depends on the data and the specific instances of s and t.

We can estimate the statistical significance of the link by using the goodness value $f(s,t)$ as a test statistic. Returning to the previous example this tells us how likely it is to obtain a link with goodness 0.4 or better by chance. There are multiple meaningful null hypotheses:

N1. Vertices s and t of types $\tau_s \in T_v$ and $\tau_t \in T_v$ are not more strongly connected than randomly chosen vertices s' and t' of types τ_s and τ_t.

N2. Vertex s of type $\tau \in T_v$ is not more strongly connected to vertex t than a randomly chosen vertex s' of type τ.

N3. Vertices s and t are not more strongly connected in the given graph G than in random graph H with edge weights $w' : E(H) \mapsto \mathbb{R}$ generated by model $\mathcal{H}$ similar to the (unknown) model which generated G and w.

The last null hypothesis N3 is clearly the most complicated one: it is not easy to come up with a model $\mathcal{H}$ that generates random graphs which are sufficiently similar to the observed graph. The choice from the first two null hypotheses depends on what we are testing. In a symmetrical case, for example when testing the significance of connection between two candidate genes, N1 is appropriate. If the roles of the vertices are asymmetric, as in testing for the connection from a set of candidate genes to a single phenotype, N2 should be used. In the experiments (Section 4.5) we apply N1 to assessment of gene–gene links and N2 to assessment of gene–phenotype links.

Under null hypothesis N1 we can estimate p-value for the test statistic $f(s,t)$ by randomly sampling N pairs of vertices (s',t') from V. Let us denote the sample by $S = \{(s_1,t_1),\ldots,(s_N,t_N)\}$. To obtain an empirical null distribution we compute the value of test statistic $f(s_i,t_i)$ for each $(s_i,t_i) \in S$, and let $S_+ = \{(s_i,t_i) \in S : f(s_i,t_i) \geq f(s,t)\}$. Then the estimated p-value $\tilde{p}$ is simply

$$\tilde{p} = \frac{|S_+|}{N}. \tag{4.15}$$

The same procedure can be used under null hypothesis N2 by sampling single vertices $S = \{t_1,\ldots,t_n\}$ and letting $S_+ = \{t_i \in S : f(s,t_i) \geq f(s,t)\}$.

Because vertices of the same type may have wildly varying degrees one should sample vertices s' and t' that have degrees similar to s and t, respectively. If several hypotheses are tested (several candidate genes, for example), the resulting p-values should be adjusted accordingly to account for multiple testing.

4.5 Experiments

We demonstrate the use of link goodness on Biomine graphs with two examples by Sevon and others [116]. We focus on two measures that are specifically targeted for random graphs: the probability g_b (4.8) of the best (or the most probable) path between vertices and the two-terminal reliability g_R (4.14) computed from the connection subgraph induced by k best paths. As discussed above, g_b is a simple choice for link goodness on random graphs, and g_R generalises g_b to consider a set of paths simultaneously.

In the first experiment we selected a handful of known Alzheimer disease genes and estimated the significance of the gene–phenotype link for each gene.

In the second experiment we evaluated the significance of links between genes whose protein products are known to interact. Both experiments were performed using a preliminary version of Biomine consisting of NCBI's Entrez databases[1] and a similar edge weighting scheme to the one described in Section 4.2. See Sevon et al. [116] for details.

Test design is not straightforward: for all classified genes there are trivial links in the graph. For example, the OMIM entry for the disease refers directly to the candidate gene. We considered s–t-paths with three edges or less "trivial", and we did not take such short paths into account when measuring link goodness values $g_b(s,t)$ and $g_R(s,t)$. The ideal solution would be to use only edges that are annotated prior to publication of the gene–disease association and the addition of respective edges into the graph, but it can be difficult to obtain the state of all databases at an earlier date. See Langohr [81] and Eronen et al. [36] on exploiting temporally different versions of Biomine.

To simplify the experimental setting and to avoid introducing subjective bias we assumed that all edges have the same product $r(e) \cdot q(e)$ of reliability and relevance. We also used a single α value for each vertex type as in (4.3). Consequently the goodness of a path or subgraph depends only on the topology of the graph and parameters α and rq.

4.5.1 Gene–phenotype link

We chose a set S of ten known human susceptibility genes for Alzheimer disease:

$$S = \{\text{APP}, \text{PSEN1}, \text{AD5}, \text{AD6}, \text{AD9}, \text{AD7}, \text{COL25A1}, \text{APOE}, \text{PSEN2}, \text{AD6}\}.$$

These identifiers were queried from Entrez Gene database[2] with query term *Alzheimer*. For the phenotype vertex t we chose AD from the OMIM database [100]. AD is a phenotype description of Alzheimer disease: it contains trivial links to all known Alzheimer genes as well as a large number of references to literature on the disease. To assess the statistical significance of s–t-link for each susceptibility gene $s \in S$, we randomly sampled 100 genes R_s from the set of all human genes that had similar degrees to s. The goodness values $g_b(s',t)$ and $g_R(s',t)$, $s' \in R_s$, constitute our empirical null distributions for each s.

To calculate link goodness values we enumerated the best 100 acyclic paths $\mathcal{P}$ with at most 6 edges from each gene vertex $s \in S \cup S'$ to the phenotype vertex t. For two candidate genes, COL25A1 and AD9, no paths to Alzheimer disease were found. Next we removed paths all paths shorter than three edges from $\mathcal{P}$ (recall that paths with three or fewer edges were considered "trivial"). We used the

[1] `http://www.ncbi.nlm.nih.gov/sites/gquery` (referred on 25 February 2011)

[2] `http://www.ncbi.nlm.nih.gov/gene/` (referred on 25 February 2011)

	Best path		Reliability	
s	$\tilde{p}$	$g_b(s,t)$	$\tilde{p}$	$g_R(s,t)$
AD7	$< .01$	.024	.01	.153
APOE	$< .01$	.184	.01	.876
APP	.02	.123	.01	.719
AD8	$< .01$	.119	$< .01$	.262
PSEN1	.04	.103	.01	.963
PSEN2	$< .01$	.153	$< .01$	.993
AD6	$< .01$	.033	$< .01$	.336
AD5	.01	.040	.01	.238

Table 4.4: Link strengths and p-values for Alzheimer disease with parameters $\alpha = 0.25$ and $rq = 0.8$.

goodness value of the best of the remaining paths and the two-terminal network reliability of the graph induced by the remaining paths as test statistics. Two-terminal network reliability was estimated using crude Monte Carlo algorithm with 100,000 iterations (see Algorithm 5 in Chapter 5). We then estimated two p-values using (4.15), one for the best path goodness and another for the connection subgraph goodness, for each $s \in S$.

We experimented with parameter values

$$(\alpha, rq) \in \{0.125, 0.25, 0.5, 1.0\} \times \{0.2, 0.4, 0.6, 0.8, 1.0\}.$$

The estimated p-values $\tilde{p}$ and values of the test statistics for each gene $s \in S$ with $\alpha = 0.25$ and $rq = 0.8$ are given in Table 4.4. The probabilities of best paths and connection subgraphs vary across genes, as expected, and they are not alone sufficient indicators of the strength of a link as discussed in Section 4.4. The estimated p-values $\tilde{p}$ are more useful here. In this test they are consistently small, and in many cases none of the 100 randomised data sets produced equally high goodness values so that $\tilde{p} < 0.01$. It is also difficult to claim that connection subgraphs are more powerful link indicators than best paths.

The goodness values also vary with the values of the two parameters of our test. However, comparable p-values were obtained for all combinations of parameter values except for $\alpha = 1$. Mean estimated p-values for all combinations are given in Table 4.5. This can be seen as an indication of the stability of the measures with respect to the parameters, but it also shows that the links are very strong and rather obvious even though all short paths were removed.

4.5.2 Protein interactions

In the second and more challenging experiment we evaluated the strength of link between APP gene and five genes whose protein products interact with the APP

	$\alpha = .125$		$\alpha = .250$		$\alpha = .500$		$\alpha = 1.000$	
rq	BP	Rel	BP	Rel	BP	Rel	BP	Rel
.2	.0100	.0075	.0175	.0063	.0200	.1325	.0438	.3813
.4	.0088	.0075	.0150	.0075	.0163	.0075	.0300	.3813
.6	.0088	.0063	.0088	.0075	.0263	.0100	.0063	.1338
.8	.0088	.0063	.0088	.0063	.0075	.0075	.0138	.0075
1.0	.0088	.0200	.0088	.0063	.0088	.0075	.0238	.0088

Table 4.5: Mean p-values for all combinations of parameter values. BP and Rel denote the test statistics: in BP (best path) $g_b(s,t)$ was used, and in Rel (reliability) $g_R(s,t)$ was used.

	Best path		Reliability	
Gene	$\tilde{p}$	$g_b(s,t)$	$\tilde{p}$	$g_R(s,t)$
HADH7	$< .01$	.159	.01	.917
APBA1	$< .01$	.137	$< .01$	.998
CHRNA7	.17	.058	.52	.359
APOA1	.56	.041	.51	.530
SHC1	.15	.118	.07	.937

Table 4.6: Interactions with APP ($\alpha = 0.25, rq = 0.8$)

protein: HADH2, APBA1, CHRNA7, APOA1 and SHC1. The interactions were obtained from the IntAct-database [4]. The experiments were carried out the same way as with Alzheimer disease except that we used the symmetric null hypothesis N1 (that is, both s and t were randomised). In the results two genes show significant linkage to APP (Table 4.6). The other three genes do not get significant p-values despite relatively high values of the test statistics compared to the Alzheimer experiment. This result suggests that pairs of genes are generally strongly connected. A possible remedy is to give higher relevance coefficients for interaction-related edge types. It is also possible that the simple edge weighting we used is not sufficient to distinguish the potential interaction-related paths between the pairs of genes in these cases.

4.6 Conclusions

We described a simple data model for representing relational biological data as labelled graphs. Such graphs can be constructed from many biological databases as demonstrated by our biological graph database Biomine. We also introduced the idea of assigning probabilities to the edges. The probabilities are derived from three factors: reliability, relevance, and rarity (informativeness). Due to the simplicity of the data model, integration of data is relatively straightforward and

the only essential requirement is referential integrity between the data sources. We believe that the probabilistic interpretation for edge and link strength is natural and intuitive for investigators, especially when the data is intrinsically unreliable.

We discussed measures and methods for assessing the strength and significance of a link between a pair of vertices in a graph. The probabilistic interpretation of edge goodness enables us to harness random graph techniques, such as most probable path and reliability, for measuring link goodness.

We demonstrated the use of two link goodness measures for evaluating the strength of gene–phenotype link on Biomine graphs using a set of known Alzheimer genes and Alzheimer phenotype. Both measures gave low p-values for the known genes. This indicates that the measures would have successfully identified the correct candidate genes for Alzheimer disease among a random set of genes, except for two genes for which no link was found.

In the second experiment we evaluated the strength of the link between APP and five other genes whose protein products are known to interact with the APP protein, again on Biomine graphs. The results suggest that, although two of the genes showed significant linkage to APP, the simplistic experimental setup using a single relevance value for all edge types is not optimal. See Eronen et al. [36] for an evaluation of relevance coefficients specified by an expert.

Our results indicate that link discovery with multiple integrated data sources can be feasible approach for prioritising putative disease-susceptibility genes as envisioned in Chapter 1 (Figure 1.1). The use of abstract, labelled graphs as data representation has a number of tradeoffs though. On one side, it is a generic format, it is easy to convert data into it and there is a large body of known results and algorithms for graphs. The downside is that information may be lost in transformation, the vertex or edge types may be too different to be really used in the same graph, and without built-in knowledge about particular biological concepts, mechanisms and phenomena, specific discoveries about them cannot be made. It seems that several different approaches on different levels of detail and integration are needed, and that they complement rather than compete with each other.

The data model combined with source databases and edge weighting define the core Biomine database. In the refined analysis phase (Figure 1.1) the investigator is probably interested to see what is known about the putative DS genes (how they relate to the phenotype and each other, for instance) in addition to ranking them. To actually query the database for such information we need a query language that lets the investigator specify interesting connection types. Earlier suggestions for query languages include regular expressions [78] and context-free grammars (CFGs) [93]. Biomine provides a query engine which currently supports three query types for extracting subgraphs: a neighbourhood query, a best-path based connection subgraph query, and a CFG-based connection sub-

graph query [115]. *Reliable subgraphs* provide another, well-founded approach for extracting small subgraphs that contain strong and independent connections between vertices of interest and which are suitable for information retrieval and visualisation. We will focus on reliable subgraphs in Chapter 5.

Chapter 5

Reliable subgraphs

In this chapter we focus on the last step of our gene mapping pipeline (Figure 1.1). The investigator has a prioritised set of putative disease-susceptibility genes, and she wants to discover how they are related to the disease phenotype to find possible evidence for the hypothesis, or to discover more detailed hypotheses about the mechanisms of the relationship. The existing biological knowledge can be conveniently represented as a random graph of biological concepts and their relations as described in Chapter 4, and this graph can be then used to assess the relationships between putative DS genes and the phenotype. The problem can be formulated as a task of finding a small connected subgraph that contains the most relevant connections between the prioritised DS genes and the phenotype [39].

Another application example is the problem of identifying a subnetwork (or community) that connects two given persons of a large network, for example social or collaboration network. Figure 5.1 gives an example of such collaboration subnetwork between two researchers extracted from DBLP computer science authorship network. Applications of subnetworks include analysis and description of potential collaborations, discovery of hidden relationships for instance in criminology, and description of possible viral effects between given individuals.

The search task can be also seen as information retrieval: given some concepts ("search terms") return other concepts and relations that are maximally relevant to connecting the given pair of individuals. For random graphs a natural choice for the relevance of a subgraph is the probability (*reliability*) that the search terms are connected in the subgraph [54, 106]. Relevant subgraphs are then more likely to contain strong edges, shorter paths, and independent paths, but edges or paths that add little to the connectivity of the search terms are not likely to be included, reflecting the notion of relevance. Since the definition favours robust subgraphs and thus alternative paths, the results have an implicit bias towards graphs with more variety and less redundancy; this is also a desirable feature for information search and data mining. Small, reliable subgraphs can be visualised, and they can

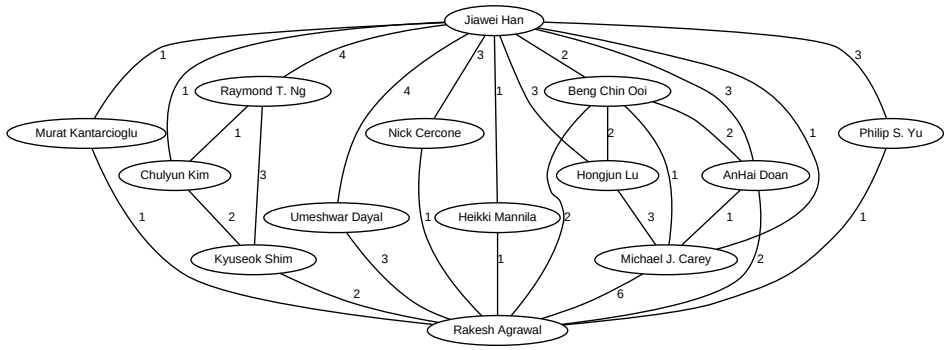


Figure 5.1: An excerpt from a reliable connection subgraph between Rakesh Agrawal and Jiawei Han. The original subgraph has 61 vertices (14 authors and 47 articles) connected by 100 edges. Here we show only the author vertices of that graph; they are connected by an edge if the corresponding authors have at least one common article neighbour (that is, they have coauthored at least one article) in the original graph. Edge labels denote the number of coauthored articles.

be used as compact representations of large graphs for network analysis methods that do not scale well.

In this chapter we discuss the aforementioned search problem on random graphs formalised as the *most reliable subgraph problem* [54]. We first give the problem definition and explain some associated concepts in Section 5.1. Efficient polynomial-time algorithms are unlikely to exist for the problem, but it can be solved efficiently on restricted cases as described in Section 5.2. For the general case we review few heuristic algorithms in Sections 5.3, 5.4 and 5.5 following our previous work [54, 56, 57]. Experimental evaluation of these heuristics is given in Section 5.6.

5.1 Basic concepts

In this section we formally define the problem of finding the most reliable subgraph from a given random graph. We begin by introducing our random graph model, the concept of network reliability and finally the subgraph extraction problem. We also describe a simple Monte Carlo approximation algorithm for estimating network reliability which is used extensively in later sections.

5.1.1 Random graph model

Let $G = (V, E)$ be a graph with a vertex set V and an edge set E. A graph can be either *directed* or *undirected*. If G is directed, the edge set $E \subset V \times V$ consists of

ordered vertex pairs. Otherwise G is undirected and the edges are 2-subsets of E: $E \subset [V]^2$. However, we make a distinction between them only when necessary as most of the following applies equally to directed and undirected graphs.

We denote the set of vertices of a graph G by $V(G)$ and the set of edges by $E(G)$. The number of vertices is denoted by $|G|$ and the number of edges by $\|G\|$. The union between two graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ is a new graph $H = G_1 \cup G_2 = (V_1 \cup V_2, E_1 \cup E_2)$. Other set operations for graphs are defined analogously. For notational convenience we treat paths and edges as graphs in set operations. This makes it notationally easy to, for instance, add a path P to a graph G by writing simply $G \cup P$.

To consider reliability, or any probabilities on graphs in general, we need to define a random graph model. In this chapter we use a standard Erdős–Rényi-like random graph model where edges are considered random. Given a graph $G = (V, E)$ and probabilities p_e for each $e \in E$, we assume that each edge $e \in E$ exists with probability p_e or does not exist with probability $1 - p_e$. Edges are assumed to be independent. The random outcomes, or *realisations*, of G are subgraphs $H = (V, F) \subset G$, $F \subset E$, that can be generated by deciding each edge $e \in E$ in turn by flipping a suitably biased coin. The probability of obtaining a fixed subgraph $H = (V, F)$ is then

$$\Pr(H) = \prod_{e \in F} p_e \prod_{e \in E \setminus F} (1 - p_e).$$

In addition to fixed subgraphs it is meaningful to speak about the probability of observing a subset of edges. For instance, let $P \subset E$ be a path (a sequence of edges) in G. The probability that P exists in a realisation of G is simply the product of the probabilities of its edges (cf. (4.7)):

$$\Pr(P) = \Pr(H : P \in H) = \prod_{e \in P} p_e \prod_{e \notin P} \big(p_e + (1 - p_e)\big) = \prod_{e \in P} p_e. \tag{5.1}$$

The existence of edges $e \notin P$ is irrelevant here so their probabilities do not affect $\Pr(P)$.

Our model differs from the classical Erdős–Rényi model by allowing varying edge probabilities. In Erdős–Rényi model edges "appear" between any pair of vertices with a fixed probability p. In our model probabilities p_e depend on e, and $p_e = 0$ for all $e \notin E$. Diestel [26], among others, gives a rigorous treatment of random graphs.

5.1.2 Network reliability

Now that we have defined our random graph model we can move on to reliability. Let G be an undirected random graph with n vertices and m edges, and let $U \subset V$ be a set of k *terminal vertices*. We review the six classical reliability measures

as given by Colbourn [20]. First, *k-terminal reliability* $R_k(G)$ is defined as the probability that each of the k terminals in U can communicate in G; equivalently, $R_k(G)$ is the probability that there exists a path between all pairs of terminals. When $k = 2$ this measure is referred to as *two-terminal network reliability* $R_2(G)$, while the case $k = n$ is known as *all-terminal network reliability* $R_A(G)$. We omit explicit references to U as they are clear from the context.

These measures have natural counterparts for directed random graphs. One vertex $s \in U$ is chosen as the *source vertex* and the rest of the vertices $U \setminus \{s\}$ are *target vertices*. The directed version of $R_k(G)$ is *s,T-connectedness* $C_k(G)$; it is the probability that there exists a directed path from s to all target vertices. When $k = 2$ this measure is called *s,t-connectedness* $C_2(G)$. Finally, the directed analogue of R_A is known as *reachability* $C_A(G)$.

Reliability $R(G)$ of a random graph G can be generally defined as the probability

$$R(G) = \sum_{F \subset E} \phi(F) \cdot \Big(\prod_{e \in F} p_e \prod_{e \in E \setminus F} (1 - p_e) \Big) \tag{5.2}$$

where $\phi : \mathcal{P}(E) \mapsto \{0,1\}$ is an indicator function defined for subsets $F \subset E$. Function ϕ, or *reliability function*, distinguishes working network states from failed network states, and its definition depends on the chosen reliability measure. Equation (5.2) sums the probabilities of all network states that are considered functional (or *operational*) according to ϕ. For example, by letting $\phi(F) = 1$ if and only if s and t are connected by at least one path in graph $H = (V, F)$, (5.2) sums the probabilities of all states where s and t are connected in G and we have $R(G) = R_2(G)$.

Exact computation of $R(G)$ for general random graphs is difficult. A trivial brute-force application of (5.2) would need $\Omega(2^m)$ time. Graph simplification and specialised combinatorial algorithms can cut the required time significantly [20, 10], but no polynomial-time algorithm is known for computing the reliability in general case. The existence of such algorithm is unlikely since reliability problems are #P-complete [131]. The complexity class #P corresponds to counting problems where the output is the number of accepting computation paths of a non-deterministic polynomial-time (NP) Turing machine [130]. For example, consider NP-complete SATISFIABILITY (SAT) decision problem that asks whether a given boolean formula has a satisfying truth assignment or not. Its corresponding #P-complete counting problem is #SAT that asks how many satisfying truth assignments the given boolean formula has. Since counting is obviously at least as difficult as deciding we have NP $\subset$ #P.

5.1.3 Crude Monte Carlo method

Monte Carlo sampling is an efficient approximation method for many computationally complex problems [68] including network reliability [42]. Since we use

Monte Carlo sampling extensively we briefly review its basic principle here. In the *crude Monte Carlo method* (CMC) for estimating network reliability we draw N independent samples (realisations) G_i, $1 \leq i \leq N$, from the random graph G. Each realisation is generated by simulating the existence of each edge of G randomly and independently. To estimate reliability $R = R(G)$ according to (5.2) we count the amount N_C of those realisations G_i for which $\phi\big(E(G_i)\big) = 1$. Then $\tilde{R} = N_C/N$ is an unbiased estimator for R with variance $R \cdot (1-R)/N$ [32]. See Algorithm 5 for a pseudo-code description of a basic CMC. The algorithm can be straightforwardly implemented to run in $O(Nm)$ time.

Algorithm 5 Crude Monte Carlo (CMC)

Input: Random graph $G = (V,E)$, reliability function $\phi : \mathcal{P}(E) \mapsto \{0,1\}$, integer N
Output: Estimate $\tilde{R}$ for $R(G)$ according to (5.2)

```
1:  N_C ← 0
2:  for i = 1 to N do {generate realisations G_i}
3:      F = ∅ {F is the edge set of G_i}
4:      for all e = (u,v) ∈ E do
5:          Choose random p ∈ [0,1)
6:          if p < p_e then
7:              F ← F ∪ {e}
8:      if φ(F) = 1 then
9:          N_C ← N_C + 1
10: return R̃ = N_C/N
```

The method is called "crude" because the estimator's relative standard error

$$\mathrm{re}_{\tilde{R}} = \frac{\sqrt{\mathrm{Var}(\tilde{R})}}{E(\tilde{R})} = \frac{\sqrt{R(1-R)/N}}{R} = \sqrt{\frac{R(1-R)}{R^2 N}} = \sqrt{\frac{1-R}{R}} \cdot \frac{1}{\sqrt{N}}$$

grows without bound when $R \to 0$. The same result holds for estimating failure probability $1-R$ in very reliable networks. Many sophisticated Monte Carlo methods have been proposed to reduce the variance of the estimator [42, 32, 61] including a FPRAS [67]. Since our primary purpose is not to estimate reliability as accurately as possible but rather to extract a "reliable enough" subgraph, we confine ourselves to the simple CMC.

5.1.4 The most reliable subgraph problem

The objective in the *most reliable subgraph problem* (MRSP) is to find the most reliable subgraph of G with at most K edges [54]. As before, let $G = (V,E)$ be a random graph with n vertices and m edges, and let $U \subset V$ be a set of k terminal vertices where $2 \leq k \leq n$. Let $R \in \{R_2, R_k, R_A, C_2, C_k, C_A\}$ be the corresponding reliability measure with respect to G and U, and let $K \in \mathbb{N}$ with $0 \leq K \leq m$. The

objective is to find a subgraph $H^* \subset G$ with at most K edges such that $R(H^*) \geq R(H)$ for all subgraphs $H \subset G$ with $\|H\| \leq K$:

$$H^* = \operatorname*{arg\,max}_{\substack{H \subset G \\ \|H\| \leq K}} R(H). \tag{5.3}$$

Note that in the previous work on the MRSP [54, 56], the objective was to *remove* K edges from G to get a subgraph H. Obviously this does not change the problem, but is simply a matter of parametrisation.

We introduced the problem itself recently [54] and as such it has not been researched a lot. However, the MRSP is closely related to the connection subgraph problem of Faloutsos and others [39] (see Section 4.3). The connection subgraph problem has been since extended to handle multiple terminals, or *query nodes*, by Tong and Faloutsos (center-piece subgraphs [126]), and Koren and others (proximity graphs [75], Section 4.3). The underlying models and optimisation problems in these settings are based on random walks and hence different from the MRSP.

Kroese and others have independently considered a very similar and slightly more general network planning problem [76]. In their graph model the edge set $E = V \times V$ consists of all possible edges between vertices V, and each edge $e \in E$ has an associated *cost* c_e in addition to probability p_e. Given a set of terminals $U \subset V$ the objective is to choose a reliable subnetwork $H^* \subset G$ that connects the vertices in U and is restricted to a given budget B:

$$H^* = \operatorname*{arg\,max}_{\substack{H \subset G \\ c(H) \leq B}} R(H), \text{ where } c(H) = \sum_{e \in E(H)} c_e. \tag{5.4}$$

By comparing (5.3) and (5.4) it is easy to see that the MRSP on a given random graph $G = (V, E)$ is a special case of this network planning problem. Let each edge $e \in E$ have a unit cost $c_e = 1$, and let other edges $e \in (V \times V) \setminus E$ have an arbitrary large cost $c_e > K$. Then the MRSP is equivalent to the network planning problem with budget K.

Given the fact that exact calculations of R_2, R_k, R_A and C_2, C_k, C_A are #P-complete problems, it is not surprising that the MRSP is likely to be computationally hard as well. The problem does not ask for the value of $R(H)$ for the chosen R and an optimal subgraph H. Despite this, the k-terminal undirected MRSP is NP-hard.

Theorem 1. *MRSP with $R = R_k$ is NP-hard.*

Proof. We give a polynomial time reduction from the NP-complete STEINER TREE problem [45] to the MRSP. Let (G, U, B) be an instance of the STEINER

TREE where $G = (V,E)$ is a graph with positive edge weights, $U \subset V$ is a set of terminals and $B \in \mathbb{N}$ is a bound for the size of the tree.

Without a loss of generality we assume that all edge weights are equal to 1. We transform G into a probabilistic graph $H = (V,E)$ by letting $p_e = 1/2$ for each $e \in E$. Next we find the smallest (that is, having the least number of vertices and edges) optimal subgraph $H^* \subset H$ connecting the terminals, by solving the MRSP for $K = k-1,\ldots,m$ and checking the results in polynomial time. Obviously H^* is a tree; it is also a minimal Steiner tree. Assume to the contrary that there exists a minimal Steiner tree T such that $\|T\| < \|H^*\|$. By construction, we have $R_k(T) = 1/2^{\|T\|} > 1/2^{\|H^*\|} = R_k(H^*)$ which contradicts the optimality of H^* since T is also a subgraph of H connecting the vertices in U.

To complete the reduction we simply check if $\|H^*\| \leq B$ holds. □

The complexity of cases where $R \in \{R_2, R_A\}$ remains open, but we conjecture that they are also NP-hard. The directed variants of the problem are probably hard too, since the directed reliability problems are as hard as the corresponding undirected problems [6].

The inherent difficulty of the problem is mostly due to two factors. First is the combinatorial factor: there number of possible paths (subgraphs H) is exponential. In this respect the problem is similar to classical combinatorial problems like TRAVELLING SALESMAN [45]. Another factor is the hardness of the reliability calculation: it is difficult to even evaluate the reliability $R(H)$ of a given solution subgraph H yet to optimise it. Brute force approaches are therefore inefficient even for relatively small input graphs.

In the following sections we review few algorithms for solving the MRSP. We limit the discussion to two-terminal case, that is $U = \{s,t\}$, although some of the algorithms have natural extensions to the k-terminal case as well. We give guidelines for such extensions where available. First we consider a special case where the input graph is restricted and hence an efficient algorithm can be constructed (Section 5.2). Then we proceed to the general case (Sections 5.3, 5.4 and 5.5) where we use iterative and greedy approaches to limit the search space of possible solutions. We simplify the reliability calculations by MC approximation or restricting the set of possible solutions to graphs for which the reliability can be easily calculated. Most of the algorithms are computationally efficient and scalable for large scale graph mining—see Section 5.6 for experimental evaluation.

5.2 Algorithm for series-parallel graphs

It is most likely that there is no efficient algorithm for solving the MRSP in a general case. However, we can construct a polynomial-time algorithm for solving the two-terminal MRSP in an important special case where the input graph is

series-parallel. The original algorithm was proposed by Hintsanen [54] and it works by removing edges from the original graph (thus being in line with the original definition of the MRSP). We give here a slightly modified variant that builds a subgraph with at most K edges according to (5.3).

The class of series-parallel graphs is usually defined using series and parallel composition rules [129]. For our purposes the following equivalent definition is better. An undirected random graph G with specified terminals s and t is *series-parallel* if it can be reduced into a single edge $\{s,t\}$ by repeatedly applying the following reductions:

- *Series reduction*: If G has a vertex $v \notin \{s,t\}$ of degree two, v and its adjacent edges $e = \{u,v\}$ and $f = \{v,w\}$ can be replaced with a single edge $g = \{u,w\}$ with $p_g = p_e p_f$.
- *Parallel reduction*: If G has two parallel edges $e = \{u,v\}$ and $f = \{u,v\}$, they can be replaced with a single edge $g = \{u,v\}$ with $p_g = 1 - (1 - p_e)(1 - p_f)$.

The specific sequence of reductions is irrelevant: if reductions are applied in any order until no reduction is possible, the result is the single edge $\{s,t\}$ [129]. *Directed* series-parallel graphs can be defined analogously: series reduction can be applied when both in-degree and out-degree of v is one, and parallel reduction can be applied when both of the parallel edges have the same direction.

Let us introduce some terminology and notation before we describe the algorithm. For an arbitrary edge set $F \subset E$ let $G[F]$ be the subgraph edge-induced by F. We denote the set of original edges (that is, edges in E) reduced into an edge e by $S(e)$; specifically $f \in S(e)$, $f \in E$, if f occurs in the sequence of series-parallel reductions that produced e. Initially we let $S(e) = \{e\}$ for each $e \in E$.

Let $e = \{u,v\}$ be an edge, either from E or obtained by a sequence of reductions. An i-edge subset $S(e,i) \subset S(e)$ is said to be an *optimal solution for* $G[S(e)]$ if $G[S(e,i)]$ is the most reliable subgraph of $G[S(e)]$ with i edges and terminals u and v. In other words $G[S(e,i)]$ is a solution to the MRSP for $G[S(e)]$ with $K = i$ and $U = \{u,v\}$. Let $R_S(e,i) = R_2(G[S(e,i)])$ be the reliability of an optimal solution $S(e,i)$.

The iterative definition of series-parallel graphs suggests an iterative, dynamic programming algorithm for solving the MRSP given that an optimal solution can be constructed from optimal solutions to smaller subgraphs. The following lemma states that this is indeed the case:

Lemma 1. *Let e and f be two edges in series or parallel, and let g be the edge produced by the reduction of e and f. If $S(e,i)$ and $S(f,i)$ are the optimal solutions for $G[S(e)]$ and $G[S(f)]$, where $0 \leq i \leq K$, then optimal solutions $S(g,i)$ can be formed in $O(K^2)$ time for all i.*

Proof. Let i be fixed. Since $S(g) = S(e) \cup S(f)$ and $S(e) \cap S(f) = \emptyset$, an optimal solution $S(g,i)$ has j edges in $S(e)$ and $i-j$ edges in $S(f)$ for some $0 \leq j \leq i$. If e and f are in series, we have $R_S(g,i) = R_S(e,j) \cdot R_S(f,i-j)$. Otherwise e and f are parallel and we have $R_S(g,i) = 1 - \big(1 - R_S(e,j)\big) \cdot \big(1 - R_S(f,i-j)\big)$. An optimal solution $S(g,i)$ can be found by simply enumerating all i possible combinations of edge assignments and choosing one which maximises $R_S(g,i)$:

$$k = \operatorname*{arg\,max}_{0 \leq j \leq i} \begin{cases} R_S(e,j) \cdot R_S(f,i-j) & \text{if } e \text{ and } f \text{ are in series} \\ 1 - \big(1 - R_S(e,j)\big) \cdot \big(1 - R_S(f,i-j)\big) & \text{if } e \text{ and } f \text{ are parallel} \end{cases}$$

$$S(g,i) = S(e,k) \cup S(f,i-k) \ .$$

The solution can be found in $O(i)$ time. By repeating the procedure for all i, $0 \leq i \leq K$, we obtain the solutions $S(g,i)$ in $O(K^2)$ time. □

To solve the MRSP for a series-parallel graph G, we repeatedly apply series and parallel reductions until the graph is reduced into a single edge. At first we let $S(e,0) = \emptyset$, $R_S(e,0) = 0$, $S(e,1) = \{e\}$, and $R_S(e,1) = p_e$ for each $e \in E$. This establishes an invariant: each e has optimal solutions $S(e,i)$ for $G[S(e)]$ where $i = 0, \ldots, \min\{|S(e)|, K\}$. We maintain the invariant by keeping track of optimal solutions $S(e,i)$ and their reliabilities for each edge e. The invariant, with the definition of series-parallel graphs, guarantees that in the end we have an optimal solution to the MRSP for G. See Algorithm 6 for a concise pseudo-code description.

At every iteration we identify a pair $\{e,f\}$ of reducible edges. This can be done in constant time by suitably augmenting the graph data structure. These edges are replaced with a new edge g. By Lemma 1 it is straightforward to form optimal solutions $S(g,i)$ and maintain the invariant. Since each reduction effectively removes one edge from G, after $m-1$ iterations only a single edge e remains and $S(e,K)$ contains an optimal solution for G. We have established the following theorem:

Theorem 2. *Let $G = (V,E)$ be a series-parallel random graph with m edges. The MRSP on G with $R \in \{R_2, C_2\}$ can be solved in $O(K^2 m)$ time, where $1 \leq K \leq m$.*

Note that the time complexity of the algorithm depends on K. In practical applications K is usually significantly smaller than m so this is favourable. However, if $K > m/2$ it is beneficial to use the original algorithm [54] which works in a top-down fashion by removing edges one by one, but it has a decreasing time complexity $O\big((m-K)^2 m\big)$ with respect to K.

Algorithm 6 Algorithm for series-parallel graphs with two terminals

Input: Random series-parallel graph $G = (V,E)$, terminals $\{s,t\} \subset V$, integer K
Output: Subgraph $H \subset G$ such that $\|H\| \leq K$ and $R_2(H)$ is maximal

for all $e \in E$ **do** {initialisation}
  $S(e,0) \leftarrow \emptyset$, $S(e,1) \leftarrow \{e\}$
  $R_S(e,0) \leftarrow 0$, $R_S(e,1) \leftarrow p_e$
  for all $i \in \{2,\ldots,K\}$ **do**
    $S(e,i) \leftarrow \emptyset$, $R_S(e,i) \leftarrow \emptyset$
while $|E| > 1$ **do**
  Find two reducible edges e and f from G
  Reduce e and f into g (and add g to E)
  for all $i \in \{0,\ldots,K\}$ **do** {see Lemma 1}
    $M \leftarrow -\infty$
    for all $j \in \{0,\ldots,i\}$ **do**
      if e and f are in series **then**
        $r \leftarrow R_S(e,i) \cdot R_S(f,i-j)$
      else
        $r \leftarrow 1 - \big((1 - R_S(e,i)\big) \cdot \big(1 - R_S(f,i-j)\big)$
      if $r > M$ **then**
        $k \leftarrow j$, $M \leftarrow r$
    $R_S(g,i) \leftarrow r$
    $S(g,i) = S(e,k) \cup S(f,i-k)$
return $G[S(E,K)]$ {E has only one edge at this point}

5.3 Simple algorithms for general graphs

The set of series-parallel graphs is a very restricted class of graphs, and in general graphs are lot more complex. Unfortunately, as suggested in Section 5.1, the computational effort required for an exact solution quickly becomes excessive. It is most likely that one must content with approximate or heuristic solutions.

In this section we introduce two algorithms for solving the MRSP on general graphs. Both algorithms are based on simple greedy strategies. The first one prunes the graph one edge at a time until the graph has shrunk enough. The second one incrementally adds paths one at a time to an initially empty graph.

5.3.1 Removing less critical edges

Recall from Section 5.1 that the reliability $R \in \{R_2, R_k, R_A, C_2, C_k, C_A\}$ of a given random graph G can be efficiently estimated by a Monte Carlo simulation (CMC, Algorithm 5). Hence, in theory, one could solve the MRSP in a brute-force manner by considering all possible K-edge subgraphs of G, estimating the reliability of each such subgraph and choosing the best one. The algorithm would run in $\Theta(\binom{m}{K}Nm)$ time which is clearly intractable even for modestly sized graphs.

However, we can use CMC to estimate $R(G-e)$, for each $e \in E$, where $G-e$ denotes G with an edge e removed. Edges with large values of $R(G-e)$ are likely less critical, so we use those values to guide a greedy heuristic. Algorithm 7 gives a pseudo-code description of the heuristic entitled *Monte Carlo Pruning* (MCP) as it removes edges one by one from the input graph G. The algorithm first estimates $R(G-e)$ for all $e \in E$. It then iteratively removes $m-K$ edges with the highest $R(G-e)$ values. If there are edges with a non-terminal endpoint of degree one at the beginning of an iteration, we remove those edges first. Such edges can be safely removed since they do not occur on any acyclic path between the terminals.

MCP can be implemented in a straightforward manner to run in $O(Nm^2+m+(m-K)\log m) = O(Nm^2)$ time where N is the amount of MC iterations. MCP works on directed and undirected graphs with two or more terminals since any valid reliability function can be used in the Monte Carlo estimation (Algorithm 5).

Although MCP is feasible for small graphs with up to hundreds of edges, the dominating Nm^2 factor in the asymptotic running time is unacceptable for large graphs with thousands of edges or more. Another problem in MCP is its sensitivity to small differences between $R(G-e)$ values for different edges e. In large graphs most of the true $R(G-e)$ values tend to be close to $R(G)$, and the accuracy of the corresponding estimates is not sufficient to separate edges. In such cases MCP removes edges almost by random. This deficiency can be avoided to some extent by re-estimating $R(G-e)$ values at each iteration or some intervals [56, 106]. We

Algorithm 7 Monte Carlo Pruning (MCP)

Input: Random graph $G = (V, E)$, terminals $\{s, t\} \subset V$, integers K and N
Output: Subgraph $H \subset G$ such that $\|H\| \leq K$

for all $e \in E$ **do**
  $C[e] \leftarrow$ MC-estimate of $R(G - e)$ with N iterations {See Algorithm 5}
Sort table C into descending order
$H \leftarrow G$
while $\|H\| > K$ **do**
  while there exists a vertex $v \in H$, $v \notin \{s, t\}$ with degree 1 **do**
    Remove v and its adjacent edge from H and C
  if $\|H\| \leq K$ **then**
    return H
  Remove the first element (edge) e from C
  Remove e from H
return H

	$e \in G_i$	$e \notin G_i$
$\phi(G_i) = 1$	C_{11}	C_{12}
$\phi(G_i) = 0$	C_{21}	C_{22}

Table 5.1: Monte Carlo contingency table. Different C values are the numbers of realisations G_i where the corresponding event has been observed.

will return to this issue in our experiments.

The efficiency of Algorithm 7 can be improved to $O(Nm)$ by modifying the Monte Carlo sampling algorithm (Algorithm 5) to check the existence of each $e \in E$ in each Monte Carlo realisation. Then a contingency table like Table 5.1 can be formed in $O(Nm)$ time, from which we can readily estimate $R(G - e) \approx C_{12}/(C_{12} + C_{22})$. This optimisation comes with a cost as the effective number of MC iterations for estimating $R(G - e)$ depends on p_e. The accuracy of the estimation is the same as with Algorithm 5 using $N \cdot (1 - p_e)$ iterations. Another problem is that if $p_e = 1$ (or very close to 1), C_{12} and C_{22} are both zero and the estimator is undefined. One possible solution is to separately estimate $R(G - e)$ for each such edge with Algorithm 5.

5.3.2 Collecting most probable paths

The next algorithm, *Best Paths Incremental* (BPI, Algorithm 8), is based on a simple idea: find a set of the most probable, or *best*, paths between the terminal vertices s and t, and let them span a subgraph. BPI adds best paths to an initially empty solution subgraph until it has at least K edges. To have exactly K edges BPI calls MCP to remove the possible excess edges. This is a somewhat arbitrary deci-

sion: MCP might affect the subgraph adversely, especially for larger values of K, due to the aforementioned problems. Optionally one can accept a slightly larger subgraph or modify the algorithm to stop until all suitable candidate paths (up to some limit) have been considered. We used MCP to make the results comparable in our experiments.

Algorithm 8 Best Paths Incremental (BPI)

Input: Random graph $G = (V,E)$, terminals $\{s,t\} \subset V$, integers K, k_0, and k_1
Output: Subgraph $H \subset G$ such that $\|H\| \leq K$

```
1: H ← ∅
2: k ← k_0 {the initial number of best paths looked for}
3: while ||H|| < K do
4:     Let P be the set of k most probable s–t-paths sorted in descending order of probability
5:     while ||H|| < K and P ≠ ∅ do
6:         Remove the first element (path) P from P
7:         Add P to H
8:     k ← k · k_1
9: return MCP(H, K)
```

The number of paths needed to span a subgraph of the desired size depends on G. We used initially $k_0 = 2 \cdot K$ best paths. This number was chosen experimentally and it usually gave a sufficient number of paths. If there are not enough paths to span a subgraph, the algorithm restarts with a larger number $k \cdot k_1$ of paths. We set $k_1 = 2$ in our experiments.

Best paths can be found by any k shortest paths algorithm (see Section 4.3). A multitude of polynomial-time algorithms for finding k shortest paths have been proposed in the literature [33, 53, 111]. For our experiments we implemented a straightforward extension to Dijkstra's algorithm for finding k shortest simple paths between two vertices s and t. Instead of maintaining a single shortest s–v-path for each vertex $v \in V$ we keep a record of k shortest s–v-paths. The time complexity of our implementation is $O\big((k^2n^2 + knm)\log(kn)\big)$. This could be improved to $O\big(kn(m + n\log n)\big)$ by Lawler's algorithm [83]. The total time complexity of the BPI (excluding the last MCP call) is bounded by $O\big(K(k^2n^2 + knm)\log(kn)\big)$, assuming that the chosen number of paths k is sufficient to span a subgraph of desired size.

Note that the algorithm adds paths blindly in the sense that their effect on the reliability is not evaluated. We show how to implement a greedy variant in Section 5.6.6. The greedy variant produces more reliable subgraphs but has some additional performance cost.

BPI can handle directed and undirected graphs without modification. Of course the shortest path finding algorithm must be chosen accordingly. The situa-

tion is trickier with more than two terminals as the notion of shortest path becomes complicated. With R_A (that is, all vertices are terminals) one could look for a sequence of spanning trees, or rooted arborescences in directed cases, ordered by decreasing probability. These could be then substituted for the best paths in the algorithm. With R_k the equivalent of best paths are Steiner trees. Finding optimal Steiner trees is generally NP-hard, though by heuristics or approximations [110] the problem may become tractable.

5.4 Constructing series-parallel subgraphs

As mentioned in Section 5.1.4, part of the difficulty of the MRSP is due to the hardness of reliability calculation. But the cost of evaluating reliability can be greatly reduced by considering series-parallel graphs. The next algorithm does exactly that: it constructs a series-parallel subgraph H in a greedy and iterative manner. Here, in contrary to the definition of series-parallel graphs in Section 5.2, we use the following recursive definition using composition rules [129]:

1. An undirected random graph with two vertices s and t joined by a single edge is series-parallel with terminals s and t.

2. If G_1 and G_2 are undirected series-parallel graphs with terminals $\{s_1,t_1\}$ and $\{s_2,t_2\}$, then so is multigraph $H = G_1 \cup G_2$ constructed by one of the following operations:

   (a) Identify t_1 with s_2, and let $\{s_1,t_2\}$ be the terminals of H (series composition).

   (b) Identify s_1 with s_2 and t_1 with t_2, and let $\{s_1,t_1\}$ be the terminals of H (parallel composition).

As with reduction rules in Section 5.2, *directed* series-parallel graphs can be defined analogously. Series composition can be used if and only if $t_1 = s_2$, and parallel composition can be used if and only if $s_1 = s_2$ and $t_1 = t_2$.

5.4.1 Algorithm overview

The subgraph H is first initialised to a single s–t-path. It can be, for example, the most probable path according to (5.1). Then H is trivially series-parallel. Next, we expand H by adding augmenting paths. We say that a u–v-path P in G is an *augmenting path* if and only if $H \cap P = \{u,v\}$. In other words, augmenting paths are paths between two vertices of H that visit vertices only in $G \setminus H$ except for their endpoints.

Since we restrict H to be series-parallel, we cannot use every possible augmenting path but only those paths P for which $H \cup P$ is series-parallel. We call such paths *valid* and their endpoints *connectable*. Let $\mathcal{P}$ be a set of valid augmenting paths of H. As $H \cup P$ is series-parallel we can efficiently decide which path P^* gives the maximum increase in reliability:

$$P^* = \arg\max_{P \in \mathcal{P}} R(H \cup P).$$

We will show later in Sections 5.4.2 and 5.4.3 how to find valid augmenting paths.

The basic *Series-Parallel Augmentation* algorithm (SPA, Algorithm 9) greedily adds the best valid augmenting path P^* to H until the required number of edges has been reached or there are no valid paths available. Finally it calls MCP to remove the possible excess edges. As with the BPI algorithm, the same reservations about the necessity of MCP apply here. We used MCP to keep results comparable in our experiments, but this is not necessarily needed in practical applications.

Algorithm 9 Series-Parallel Augmentation (SPA)

Input: Random graph $G = (V, E)$, terminals $\{s, t\} \subset V$, integer K
Output: Subgraph $H \subset G$ such that $\|H\| \leq K$

```
1:  Let H be the most probable s–t-path in G
2:  while ||H|| < K do
3:      Let C be the set of connectable vertex pairs
4:      if C = ∅ then
5:          return H
6:      r_max ← 0
7:      for all (u, v) ∈ C do
8:          Let P be the most probable valid augmenting path between u and v
9:          r ← R(H ∪ P)
10:         if r > r_max then
11:             r_max ← r
12:             P* ← P
13:     Add P* to H
14: return MCP(H, K)
```

A problem with series-parallel graphs is that they can be too restricted to produce good subgraphs. To alleviate this problem we can produce several different smaller series-parallel graphs and output their union which is not necessarily series-parallel. For example, consider a case where the first few iterations in Algorithm 9 quickly grow the reliability of the subgraph H_1 but the rest of the iterations give only a marginal increase to the reliability of H_1. In such case it could be beneficial to stop the algorithm and start over using a different initial solution $P \not\subset H_1$ to produce another subgraph H_2. Finally, we simply combine the solutions to get the final result $H = H_1 \cup H_2$.

We next modify Algorithm 9 to implement these ideas (Algorithm 10). First, building a series-parallel graph H is stopped as soon as $R(H \cup P^*) < C \cdot R(H)$ for a given constant $C \geq 1$ where H is the result subgraph under construction and P^* is the optimal augmenting path. Second, a new initial solution $P \not\subset H$ is chosen and a new series-parallel subgraph H' is grown. The choice $P \not\subset H$ guarantees that the new subgraph is not included in the current partial solution. This ensures an increase in the size of the result, introduces variance and likely leads to a non-series-parallel solution. We always choose the most probable such path P from the set $\mathcal{P}$ of the k most probable s–t-paths in G.

Algorithm 10 Series-Parallel Augmentation with multiple initial solutions

Input: Random graph $G = (V,E)$, terminals $\{s,t\} \subset V$, integers K, k_o and k_1, real number $C \geq 1$
Output: Subgraph $H \subset G$ such that $\|H\| \leq K$

1: $H \leftarrow \emptyset$
2: $k \leftarrow k_0$ {the initial number of most probable paths used as initial solutions}
3: Let $\mathcal{P}$ be the set of k most probable s–t-paths
4: **while** $\|H\| < K$ **do**
5: Let $P \in \mathcal{P}$ be the most probable path such that $P \not\subset H$
6: **if** $P = \emptyset$ **then** {increase k and restart}
7: $k \leftarrow k \cdot k_1$
8: Let $\mathcal{P}$ be the set of k most probable s–t-paths
9: **go to** 4
10: $H' \leftarrow P$
11: **while** $\|H'\| < K$ **do**
12: Let $\mathcal{C}$ be the set of connectable vertex pairs in H'
13: **if** $\mathcal{C} = \emptyset$ **then**
14: **go to** 25 {break from loop}
15: $r_{\max} \leftarrow 0$
16: **for all** $(u,v) \in \mathcal{C}$ **do**
17: Let P be the most probable valid augmenting path between u and v
18: $r \leftarrow R(H' \cup P)$
19: **if** $r > r_{\max}$ **then**
20: $r_{\max} \leftarrow r$
21: $P^* \leftarrow P$
22: **if** $r_{\max} < C \cdot R(H')$ **then**
23: **go to** 25 {break from loop}
24: Add P^* to H'
25: Add H' to H
26: **return** MCP(H, K)

Algorithm 10 can be also used with $C = 1$. This ensures that H will eventually have K edges. Algorithm 9 does not warrant that: for example, when K is close to n a single (although as large as possible) series-parallel graph is likely to have

less than K edges. We used Algorithm 10 with $C = 1$ as default SPA in our experiments.

The number of initial solutions needed to construct a subgraph of the desired size depends on the structure of G and the chosen constant C. More initial solutions are needed with larger values of C. We used initially $k_0 = K$ most probable (or best) paths. This value was chosen experimentally as with BPI. Algorithm 10 is restarted with a larger number $k \cdot k_1$ of initial solution paths if there are not enough paths to construct a subgraph with the desired size. In our experiments, we set $k_1 = 2$.

There are a few non-trivial implementation issues in Algorithms 9 and 10. Next, we will show how connectable vertex pairs can be found efficiently (line 3 or 12), how to find the most probable valid augmenting path (line 8 or 17) and how to calculate the reliability of a series-parallel graph (line 9 or 18). We also discuss the total time complexity of the SPA algorithms.

5.4.2 Finding connectable vertex pairs

Following the recursive definition of series-parallel graphs, a series-parallel subgraph H can be conveniently represented with a *decomposition tree* [129]. A decomposition tree T of G is a binary tree where each leaf vertex represents an edge of G. Inner vertices represent composition operations (series or parallel) and have always two children. If an inner vertex represents a series composition we call it *S-vertex*, otherwise it is *P-vertex*. For each vertex $v \in T$ the subtree rooted at v represents a series-parallel subgraph $G(v) \subset G$ with two terminals denoted by $\tau(v)$. If v is a leaf vertex, the subgraph is simply the edge stored at v and its terminals are the endpoints of the edge. At each inner vertex u with children v and w the associated composition operation joins the two subgraphs $G(v)$ and $G(w)$, and the terminals of $G(u) = G(v) \cup G(w)$ are determined by the composition operation from the terminals of $G(v)$ and $G(w)$. See Figure 5.2 for an example of a decomposition tree.

To quickly discover connectable vertex pairs we use a *compressed* decomposition tree T_c that differs from a regular decomposition tree in two ways:

1. An inner vertex $v \in T_c$ has two *or more* children. The composition operation of v is applied to all subgraphs $G(c_i)$ in succession where c_i are the children of v enumerated in order from left to right.

2. If a non-root vertex $u \in T_c$ is S-vertex (or P-vertex), then $\pi(u)$ is P-vertex (S-vertex) where $\pi(u)$ is the parent vertex of u.

We can easily convert a regular decomposition tree T to a compressed decomposition tree T_c by recursively combining $\{u, \pi(u)\}$ pairs of inner vertices having

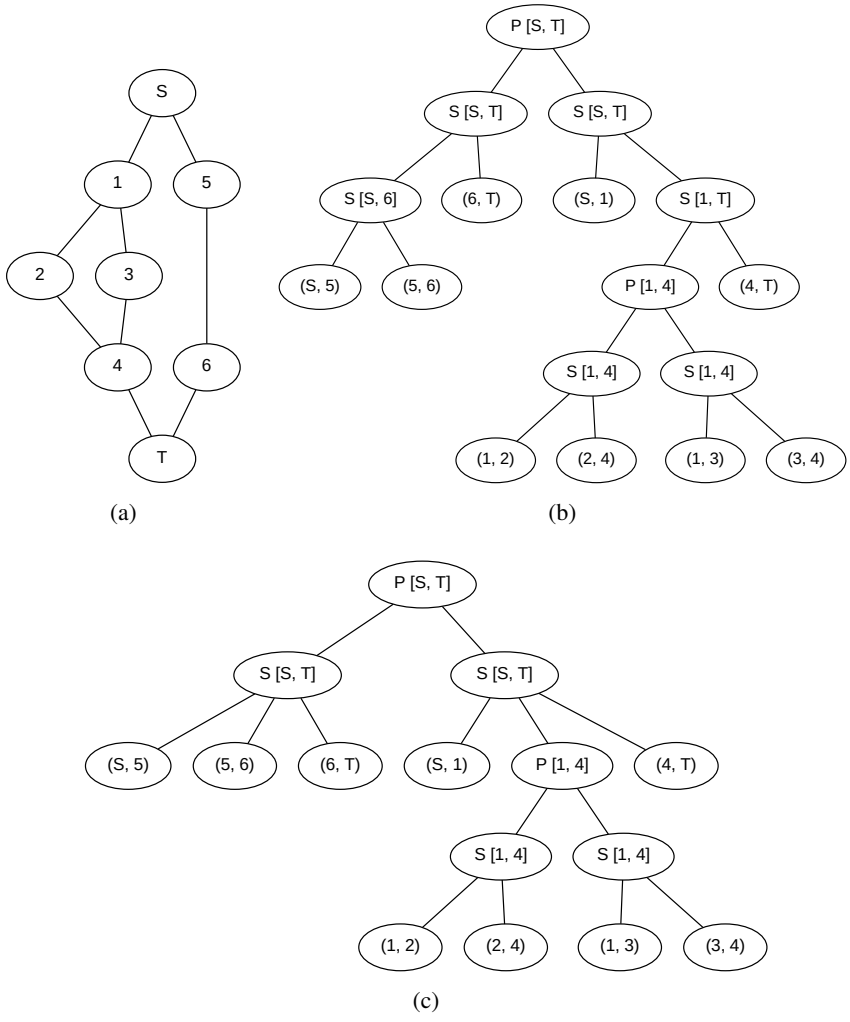


Figure 5.2: A series-parallel graph (a), its decomposition tree (b) and its compressed decomposition tree (c). Leaf vertices represent single edges, and inner vertices represent series decompositions (S) or parallel decompositions (P). Terminals of subgraphs rooted at inner vertices are enclosed in brackets.

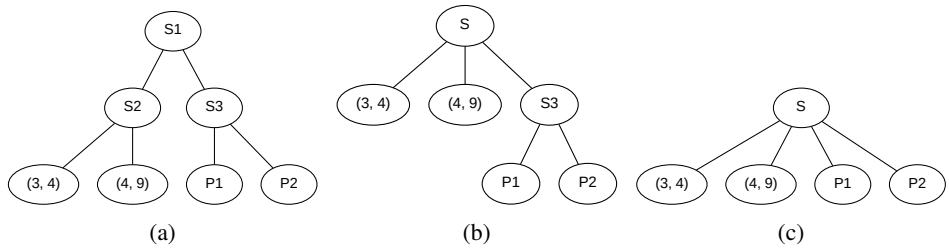


Figure 5.3: Compression of a (partial) decomposition tree. Root S-vertex S1 has two children of type S (a). We compress the tree by replacing S2 with its child edges (3, 4) and (4, 9) in (b), and S3 with its child compositions P1 and P2 in (c).

the same composition operation. Let us denote the children of a vertex u by $\lambda(u)$. Let $u \in T$ be an inner vertex and suppose that a non-leaf child $v \in \lambda(u)$ has the same type (S or P) as u. We replace v with its children and repeat this for all children of u having the same type. After repeating this procedure for all inner vertices $u \in T$ we have a compressed decomposition tree T_c. See Figures 5.2 and 5.3 for examples.

The relationship between the terminals in a compressed decomposition tree and connectable vertex pairs is characterised by the following lemma:

Lemma 2. *Let G be a probabilistic graph, T be a compressed decomposition tree of G and $\{u,v\} \subset V$ be two connectable vertices. Then there exists a vertex $x \in T$ such that*

$$\{u,v\} = \tau(x) \tag{5.5}$$

or

$$\{u,v\} \subset \bigcup_{c \in \lambda(x)} \tau(c). \tag{5.6}$$

Proof. If $(u,v) \in E$, (5.5) holds trivially. Thus we assume $(u,v) \notin E$. Consequently x is an inner vertex. Now assume the contrary of (5.6):

$$\forall x \in T : \{u,v\} \not\subset \bigcup_{c \in \lambda(x)} \tau(c). \tag{5.7}$$

Let y be the highest vertex in T such that $u \in \tau(y)$. Similarly let z be the highest vertex in T such that $v \in \tau(z)$. Finally, let $x \in T$ be the most recent common ancestor of y and z. We separate two cases.

First suppose that $x \neq y$ and $x \neq z$. Since T is compressed and y and z are as high in T as possible, y and z are P-vertices. Furthermore, they lie in different subtrees rooted at two children of x. Denote these children by c_1 and c_2. If x is

P-vertex, then connecting u and v connects $G(c_1)$ and $G(c_2)$ so that $G(x)$ contains an embedded Wheatstone bridge. Thus $G(x)$ is no longer series-parallel [29] which is a contradiction. Conversely, if x is S-vertex, then c_1 and c_2 are P-vertices connecting subgraphs formed by series compositions. If u and v are connected by an augmenting path, it does not pass through any vertex in $\tau(c_1)$. In such case $G(x)$ is not series-parallel, which is again a contradiction.

In the second case we have $x = y$ or $x = z$. Let us assume $x = y$ (the case $x = z$ is identical). Since T is compressed, z is P-vertex and $\pi(z)$ is S-vertex connecting two or more parallel compositions. Furthermore, $\tau(\pi(z)) = \{a,b\}$ does not contain v, because z is as high as possible in T, so all s–v-paths go through a or b. By (5.7), $a \neq u$ and $b \neq u$. Hence if we connect u and v by an augmenting path, we bypass a or b. Then $G(y)$ is no longer series-parallel: a contradiction. □

The significance of Lemma 2 is that the search for connectable vertex pairs in a graph reduces to a simple recursion in the corresponding compressed decomposition tree. This is summarised in Theorem 3.

Theorem 3. *Let G be a series-parallel graph, and let T be a compressed decomposition tree of G. Then the vertex pair collection*

$$\mathcal{C} = \Big\{ \{u,v\} \subset \bigcup_{c \in \lambda(x)} \tau(c) \cup \tau(x) : x \in T \Big\} \tag{5.8}$$

is the set of all connectable vertex pairs of G.

Proof. By Lemma 2, all connectable vertex pairs of G are contained in $\mathcal{C}$. We now claim that $\mathcal{C}$ contains only connectable vertex pairs. To see this consider an arbitrary vertex pair $\{u,v\} \in \mathcal{C}$, and let $x \in T$ be the corresponding vertex in (5.8).

Suppose first that x is a leaf vertex representing a single edge $e \in E$. The endpoints of e can be connected with an augmenting path P. Clearly P forms a new series composition which, in turn, forms a new parallel composition with e, and $G \cup P$ is series-parallel. Hence u and v are connectable.

Now assume that x is an internal vertex. Since T is compressed there exist child vertices y and z such that $u \in \tau(y)$ and $v \in \tau(z)$. Let P be an augmenting path connecting u and v. Again, P forms a new series composition S. Assume first that x is P-vertex. Then $\tau(c)$ are identical for all $c \in \lambda(x)$, and S is a new series component to the existing parallel composition of x. Conversely, if x is S-vertex, we have a new parallel composition consisting of S together with a new series composition of $y, c_1, \ldots, c_k, z$ where c_i are the children of x between y and z. In both cases $G \cup P$ is series-parallel, and thus u and v are connectable. □

We can implement line 3 of Algorithm 9 efficiently by maintaining a separate compressed decomposition tree T for the solution subgraph H. Then all possible

connectable vertex pairs can be found by recursively traversing T and forming the vertex pairs according to (5.8). This can be done in $O(\|H\|^2)$ time: although the size of T is $O(\|H\|)$, in the worst case almost every vertex pair is connectable (for instance when H consists of a single s–t-path). Finally, after expanding H with the optimal augmenting path we need to update the compressed decomposition tree. This can be done in constant time.

5.4.3 Finding the most probable augmenting path

After forming the collection of connectable vertex pairs we need to find the most probable augmenting path for each connectable vertex pair. Let $\{u,v\} \subset H$ be such a pair and denote $G \setminus H$ by G_S. A straightforward way to proceed is to attach u and v into G_S using only edges from a set $F = \{(x,y) : (x,y) \in E, x = u \vee x = v, y \in G_S\}$. The set F contains edges with one endpoint at u or v and another in G_S. The attachment can be done in $O(m)$ time, and the most probable augmenting path connecting u and v can be then sought from the resulting graph. The running time is $O(m) + O\big((\|G_S\| + |G_S|)\log|G_S|\big)$ with a standard implementation of Dijkstra's algorithm using priority queues. After repeating the procedure for every connectable vertex pair the total running time is $O\big(\|H\|^2(m + (\|G_S\| + |G_S|)\log|G_S|)\big)$.

A few optimisations can be readily done to speed up the implementation. First, the set of neighbour vertices of v, for each $v \in H$ in G_S, can be separately maintained to facilitate rapid attachment of v to G_S. To minimise relatively expensive shortest path computations we can cache all connectable vertex pairs for which there was no augmenting path found. Such pairs can be skipped in later iterations since successive graphs G_S are strictly decreasing. Finally, if we use a single-source shortest paths algorithm (such as Dijkstra's algorithm), we can also cache the shortest paths from $u \in H$ to all $w \in G_S$ and consult these caches when finding the most probable augmenting paths between u and v during the same iteration. This effectively halves the number of shortest path computations per iteration.

5.4.4 Reliability calculation and overall complexity

The reliability of a series-parallel graph G is easy to calculate with the corresponding (regular or compressed) decomposition tree T. We perform a post-order traversal in T and in each vertex $v \in T$ the reliability of the corresponding subgraph is calculated from the reliabilities of its children:

$$R\big(G(v)\big) = \begin{cases} p_e & \text{if } v \text{ is leaf} \\ \prod_{c \in \lambda(v)} R\big(G(c)\big) & \text{if } v \text{ is S-vertex} \\ 1 - \prod_{c \in \lambda(v)} \big[1 - R\big(G(c)\big)\big] & \text{if } v \text{ is P-vertex} \end{cases} \tag{5.9}$$

Since the size of the decomposition tree is $O(\|H\|)$ we can implement line 9 of Algorithm 9 to work in $O(\|H \cup P\|)$ time. As an additional optimisation, reliability calculation can be combined to the recursive search of connectable vertex pairs (Section 5.4.2).

By substituting upper bounds $O(\|H\|) = O(K)$ and $O(\|H \cup P\|) = O(m)$ to the observations made in the previous sections we can formulate a very rough upper bound $O\big(K^3(m + (n+m)\log n)\big)$ for the total time complexity of the "basic" SPA (Algorithm 9). This bound is not tight, but it demonstrates the significant effect of K on the running time. In typical applications K is small and thus also the cubic factor is small.

5.5 Monte Carlo algorithm

Our last algorithm for solving the MRSP is based on Monte Carlo sampling. The algorithm, called *Path Covering* (PC), builds the result subgraph H incrementally by adding s–t-paths one by one to an initially empty subgraph. In this respect the approach is similar to the aforementioned SPA and BPI. PC uses two separate phases, a *path sampling* phase and a *subgraph construction* phase, to limit the search space of possible solutions and to ease the evaluation of subgraph reliabilities.

In the path sampling phase PC gathers a relatively small set $\mathcal{C}$ of *candidate paths* from the set of all s–t-paths in G. Then, in the subgraph construction phase, PC aims to choose an optimal subset $\mathcal{P}$ of the candidate paths in $\mathcal{C}$, according to the edge budget K, and returns the subgraph $G(\mathcal{P})$ induced by paths in $\mathcal{P}$. We say that a set of paths $\mathcal{P}$ *induces* a graph $G(\mathcal{P}) = (V, E)$, where $V = \{V(P) : P \in \mathcal{P}\}$ and $E = \{e \in P : P \in \mathcal{P}\}$. Here $V(P)$ denotes the vertices of a path P.

Both phases address the same general problem: choose a subset $\mathcal{P}$ of available s–t-paths to induce a reliable subgraph. Furthermore, both phases use a similar strategy to achieve this goal by iteratively and greedily maximising $\Pr(\mathcal{P}) = \Pr(\bigvee_{P \in \mathcal{P}} P)$. Here $\Pr(P_1 \vee \cdots \vee P_k)$ denotes the probability that at least one of the random events "path P_i exists" occurs, and $\Pr(P_1 \wedge \cdots \wedge P_k)$ denotes the probability that all of the random events "path P_i exists" occur. The main difference between the phases is that the path sampling phase scales to large inputs with exponentially many paths, while the subgraph construction phase produces a better optimised subgraph $G(\mathcal{P})$ with a larger computational cost per path. We give detailed descriptions of the two phases below.

5.5.1 Path sampling phase

In the first phase of Path Covering we use an iterative strategy to construct the set $\mathcal{C}$ of candidate paths efficiently. The most probable (according to (5.1)) s–t-

path is used as the initial candidate path. Then we augment $\mathcal{C}$ in each iteration with a path P such that $\Pr(\mathcal{C} \vee P)$ is approximately maximised. Let $\overline{\mathcal{C}}$ denote an event where none of the paths in $\mathcal{C}$ exists. Since

$$\Pr(\mathcal{C} \vee P) = \Pr\big(\mathcal{C} \vee (\overline{\mathcal{C}} \wedge P)\big) = \Pr(\mathcal{C}) + \Pr(\overline{\mathcal{C}} \wedge P)$$

we are looking for the most probable s–t-path P under the condition that all current paths in $\mathcal{C}$ fail. Observe that finding P which maximises $\Pr(\overline{\mathcal{C}} \wedge P)$ would require evaluation of all cut events $\overline{\mathcal{C}}$, or sets of graph realisations where none of the paths in $\mathcal{C}$ do exist. Since this is generally infeasible we use a crude but efficient Monte Carlo procedure to simulate a *single* realisation where a cut event $\overline{\mathcal{C}}$ occurs, and then we find the most probable path P from this realisation. As before, we randomly realise edges in each iteration according to their probabilities: an edge e is decided to exist with probability p_e and to not exist otherwise. A cut event $\overline{\mathcal{C}}$ has occurred if every candidate path has at least one edge that does not exist. Then we search and add a new s–t-path to $\mathcal{C}$, if one exists, and proceed to the next iteration.

Algorithm 11 Path sampling algorithm

Input: Random graph $G = (V, E)$, terminals $\{s, t\} \subset V$, number of candidate paths N
Output: Set $\mathcal{C}$ of s–t-paths

```
 1: C ← the most probable s–t-path from G
 2: while |C| < N do
 3:     Set all e ∈ E as "undecided"
 4:     for all P ∈ C do
 5:         for all e ∈ P do
 6:             if e has not been decided then
 7:                 Decide e as successful with probability p_e, failed otherwise
 8:             if e has failed then
 9:                 continue from line 4 with next P
10:         go to 3 {P exists}
11:     Find the most probable acyclic s–t-path P from G (decide edges as necessary)
12:     if P ≠ ∅ then
13:         C ← C ∪ {P}
14: return C
```

Algorithm 11 implements the path sampling phase. It is mostly looking for a cut event $\overline{\mathcal{C}}$ in each of its iterations. It realises edges during the process only when needed and applies the following two rules to avoid unnecessary work. (1) When checking if a path exists the rest of the path can be ignored as soon as a failed edge is encountered (line 9). (2) When an existing path has been found the remaining paths can be ignored (line 10). When a cut $\overline{\mathcal{C}}$ does occur we find a new candidate path among the non-failed edges (line 11). Any shortest path algorithm can be used (see Section 4.3). As an additional optimisation, edge decisions can

be integrated into the path search and carried out whenever an undecided edge is encountered. The algorithm stops when it has sampled N candidate paths where N is given by the user. The choice of N is nontrivial. We will return to this issue in Section 5.6.

A possible problem in the path sampling algorithm is that cut events $\overline{C}$ are rarely found when $\Pr(\mathcal{C})$ is high. In such cases the algorithm could require unacceptably many iterations before stopping. A more effective approach would be to draw random realisations of G directly under the condition that no path in $\mathcal{C}$ exists. Unfortunately, given the potential dependencies between paths in $\mathcal{C}$ it is difficult to do this exactly.

As an alternative algorithmic variant consider the following approximation where we deliberately "fail" edges of candidate paths until all candidate paths have been broken. First, edges are realised until all paths in $\mathcal{C}$ have been decided, even if some paths are found to exist. Then, if some paths exist, we iteratively and greedily fail the edge e which intersects the largest number of existing paths in $\mathcal{C}$ until no paths in $\mathcal{C}$ exist. If there are more than one such edge, we choose the one with the smallest probability p_e. This modification is implemented by removing line 10 of the path sampling algorithm (Algorithm 11) and adding the cut sampler (Algorithm 12) just before line 11 after the **for** loops.

Algorithm 12 Cut sampler

1: $\mathcal{F} \leftarrow \{P \in \mathcal{C} : P \text{ exists}\}$
2: **while** $\mathcal{F} \neq \emptyset$ **do**
3: $\quad E(\mathcal{F}) \leftarrow \{e \in P : P \in \mathcal{F}\}$
4: $\quad e^* \leftarrow \arg\max_{e \in E(\mathcal{F})} |\{P \in \mathcal{F} : e \in P\}|$
5: $\quad$ Re-decide e^* as failed
6: $\quad \mathcal{F} \leftarrow \{P \in \mathcal{F} : e^* \notin P\}$

5.5.2 Subgraph construction phase

In the second phase of Path Covering we take the set $\mathcal{C}$ of candidate paths generated in the first phase, choose a subset $\mathcal{P} \subset \mathcal{C}$ having at most K unique edges in total and return the subgraph $G(\mathcal{P}) \subset G$ induced by them. The objective is to choose a set $\mathcal{P}$ of paths that maximises the reliability $R\big(G(\mathcal{P})\big)$.

Exhaustive search and evaluation of all feasible subsets is intractable even though the number of candidate paths $\mathcal{C}$ is assumed to be relatively small. We relax the problem by maximising the probability $\Pr(\mathcal{P}) = \Pr(\bigvee_{P \in \mathcal{P}} P)$ instead. It is a lower bound of $R\big(G(\mathcal{P})\big)$ and is easier to evaluate, but it still requires exponential time in the worst case. Therefore we resort to Monte Carlo approximation of probabilities. This choice also allows us to cast the path selection task as a set

cover problem. We first describe a basic algorithm and then an improved version with a special lookahead optimisation.

Basic path selection algorithm

In the basic path selection algorithm (Algorithm 13) we first draw N random realisations G_i of the whole graph $G(\mathcal{C})$ induced by the candidate paths (line 3). Let $C(P) = \{i : P \in G_i\}$ be the *cover set* of each path $P \in \mathcal{C}$. Each cover set $C(P)$ contains the indexes of those Monte Carlo realisations where P did exist (line 5). Given $\mathcal{P} = \{P_1, \ldots, P_k\}$ the cover sets can be used to estimate $\Pr(\mathcal{P}) \approx |C(P_1) \cup \cdots \cup C(P_k)|/N$ in $O(kN)$ time. This is a substantial improvement over $\Theta(2^k)$ time required for exact computation.

Algorithm 13 Basic path selection algorithm

Input: Set $\mathcal{C}$ of s–t-paths, integer K
Output: A reliable subgraph $H \subset G(\mathcal{C})$ with at most K edges
1: $\mathcal{P} \leftarrow \emptyset$
2: Remove all paths with more than K edges from $\mathcal{C}$
3: Generate N realisations G_i of $G(\mathcal{C})$
4: **for all** $P \in \mathcal{C}$ **do**
5: $\quad C(P) \leftarrow \{i : P \in G_i\}$
6: **while** $K > 0$ **and** $\mathcal{C} \neq \emptyset$ **do**
7: $\quad P^* \leftarrow \arg\max_{P \in \mathcal{C}} \hat{s}(P)$ {see (5.11)}
8: $\quad$ **if** $\hat{s}(P^*) = 0$ **then**
9: $\quad\quad$ **go to** 5
10: $\quad K \leftarrow K - w(P^*)$
11: $\quad$ Add P^* to $\mathcal{P}$ and remove it from $\mathcal{C}$
12: $\quad$ Remove all paths P from $\mathcal{C}$ for which $w(P) > K$
13: **return** $H \leftarrow G(\mathcal{P})$

With cover sets the path selection problem reduces to an instance of a specialised SET COVER problem—hence the name Path Covering—where the goal is to choose a set of paths $\mathcal{P}$ such that $|\bigcup_{P \in \mathcal{P}} C(P)|$ is maximised and $\|G(\mathcal{P})\| \leq K$. This problem differs from the ordinary SET COVER in three ways: it does not require the entire universe (the set of all positive realisations) to be covered, it is weighted (via budget K) and the weights are dynamic (different choices of paths affect the cost of individual paths).

To solve the path selection problem we use a greedy approach where we add one path at a time to an initially empty $\mathcal{P}$ (lines 6–12). We always choose the best possible addition from $\mathcal{C}$ until the budget K has been exhausted. Here "best possible" means the one adding most Monte Carlo realisations to the cover per edge added to $\mathcal{P}$. Formally, the *cost* $w(P)$ of a path P is the number of new edges

added to the solution subgraph $G(\mathcal{P})$:

$$w(P) = \|P \setminus G(\mathcal{P})\|.$$

The *score* $s(P)$ of a path is defined as the ratio of the improvement in probability over its cost:

$$s(P) = \frac{\Pr(\mathcal{P} \cup \{P\}) - \Pr(\mathcal{P})}{w(P)}. \tag{5.10}$$

Note that both s and w can vary in each iteration. With cover sets the score function (5.10) has an estimate

$$\hat{s}(P) = \frac{|C(P) \setminus C(\mathcal{P})|}{w(P)}, \text{ where } C(\mathcal{P}) = \bigcup_{P \in \mathcal{P}} C(P). \tag{5.11}$$

Algorithm 13 uses (5.11) to choose the best possible addition (line 7) and calculates it for each $P \in \mathcal{C}$ in every iteration.

During successive iterations the enumerator in (5.11) approaches zero as the proportion of realisations covered by paths in $\mathcal{P}$ increases. Eventually, though rarely, all realisations may become covered so that $\hat{s}(P) = 0$ and the choice of the remaining paths becomes arbitrary. In these situations our implementation "restarts" by considering all realisations uncovered (line 9) and tries to recover them with additional paths from $\mathcal{C}$ as before. We also remove paths that are too expensive to be added to $\mathcal{P}$ (line 12).

Improved path selection algorithm

In an extreme case the cost $w(P)$ of a path P becomes zero if all of its edges have already been included in the solution. As an additional optimisation to Algorithm 13 we implement a lookahead to take advantage of such situations: if the addition of path P would make another path $Q \in \mathcal{C}$ completely included in $G(\mathcal{P})$, the cover set $C(P)$ is extended with the cover set $C(Q)$. More formally, we say that P *dominates* Q if an inclusion of P into $\mathcal{P}$ implies $Q \in G(\mathcal{P})$, and we denote this relation by $P \succ Q$. Relation $\succ$ is clearly reflexive and transitive. An improved estimate of (5.11) with dominating paths is thus

$$\hat{s}(P) = \frac{|C_E(P) \setminus C(\mathcal{P})|}{w(P)}, \text{ where } C_E(P) = \bigcup_{\substack{Q \in \mathcal{C} \\ P \succ Q}} C(Q). \tag{5.12}$$

In (5.12) we assume that all paths dominated by a path P are removed from $\mathcal{C}$ after adding P to $\mathcal{P}$.

Algorithm 14 describes the improved path selection algorithm. The basic principle is the same as in Algorithm 13, but the score estimate is based on (5.12)

instead of (5.11). Again we choose the best possible addition P^* by calculating (5.12) for each $P \in \mathcal{C}$ in every iteration and add P^* to the solution path set $\mathcal{P}$. Hence it is important to efficiently find out path dominance relations that contribute to (5.12).

We implement efficient score calculation by explicitly storing dominance relations between paths including transitively dominated paths. This makes it easy to remove dominated paths and estimate scores for candidate paths. Initially there are no dominance relations since all paths in $\mathcal{C}$ are acyclic. A path P can dominate or become dominated by another path Q if and only if $E(P) \cap E(Q) \neq \emptyset$. Therefore we update dominance relations in every iteration for all paths that edge-intersect the selected path P^*. Note also that if some path becomes dominated by another path, this relation will hold until the end of the algorithm.

Algorithm 14 uses several mappings for each $P \in \mathcal{C}$. First, $d(P)$ contains all paths that are dominated by P, and initially we let $d(P) = \emptyset$ for all P. Next, $n(P)$ denotes the set of new edges introduced by P to the solution subgraph $G(\mathcal{P})$; consequently $|n(P)| = w(P)$. Initially we have $n(P) = E(P)$ for all $P \in \mathcal{C}$. Finally, edge-path mapping $p(e)$ denotes the set of paths that contain e. Score calculation works in three passes (lines 21–40). It begins by updating all cover sets for the remaining paths and checks for too expensive paths. Dominance relations are updated in the second pass for all paths that edge-intersect P^*. In the final stage, after cover and dominance relations are correct, path scores are re-estimated for the remaining paths. Mappings d, n and p are updated during the score calculation.

5.6 Experiments

In this section we present an experimental evaluation of the heuristics MCP, BPI, SPA and PC on general random graphs. Our primary objective is to establish scalability and quality of the results. SPA and especially PC have many parameters whose effects are also evaluated. We also consider *MCPre*, a variant of MCP, which periodically re-estimates relevance values $R(G-e)$ during the execution of the algorithm to improve the accuracy of the estimates (see Section 5.3.1).

We use two data sources: Biomine from Section 4.1 and *DBLP*[1] bibliography database. All of our test data sets have been used in previous work [56, 57], although we do a bit more data preprocessing in this study. Biomine data sets represent a biological application domain, while DBLP data sets model a social network. The data sets are discussed in detail below.

The algorithms have been implemented in a mixture of Python and C. Input

[1] `http://www.informatik.uni-trier.de/{\textasciitilde}ley/db/` (referred on 25 February 2011)

Algorithm 14 Improved path selection algorithm

Input: Set $\mathcal{C}$ of s–t-paths, integer K
Output: A reliable subgraph $H \subset G(\mathcal{C})$ with at most K edges

$\mathcal{P} \leftarrow \emptyset$
Remove all paths with more than K edges from $\mathcal{C}$
Generate N realisations G_i of $G(\mathcal{C})$
for all $e \in E$ **do**
  $p(e) \leftarrow \emptyset$
for all $P \in \mathcal{C}$ **do**
  $C(P) \leftarrow \{i : P \in G_i\}$
  $d(P) \leftarrow \emptyset$ $\{d(P)$ is the set of paths dominated by $P\}$
  $n(P) \leftarrow E(P)$ {at each iteration, $n(P) = E(P) \setminus E\big(G(\mathcal{P})\big)$}
  $\hat{s}(P) \leftarrow |C(P)|/|n(P)|$
  for all $e \in P$ **do**
    $p(e) \leftarrow p(e) \cup \{P\}$
while $K > 0$ **and** $\mathcal{C} \neq \emptyset$ **do**
  $P^* \leftarrow \arg\max_{P \in \mathcal{C}} \hat{s}(P)$ {see (5.12)}
  if $\hat{s}(P^*) = 0$ **then**
    for all $P \in \mathcal{C}$ **do** {reset cover sets}
      $C(P) \leftarrow \{i : P \in G_i\}$
    go to 13
  $K \leftarrow K - w(P^*)$
  Add P^* to $\mathcal{P}$
  $\mathcal{R} \leftarrow \{P^*\} \cup d(P^*)$ {$\mathcal{R}$ contains paths to be removed}
  for all $P \in \mathcal{C} \setminus \mathcal{R}$ **do** {update cover sets and remove too expensive paths}
    $C(P) \leftarrow C(P) \setminus C(P^*)$
    if $|n(P)| > K$ **then**
      $\mathcal{R} \leftarrow \mathcal{R} \cup \{P\}$
  $\mathcal{C} \leftarrow \mathcal{C} \setminus \mathcal{R}$
  for all $P \in \{Q \in \mathcal{C} : n(Q) \cap n(P^*) \neq \emptyset\}$ **do** {update dominance relations}
    $n(P) \leftarrow n(P) \setminus n(P^*)$
    $d(P) \leftarrow d(P) \setminus \mathcal{R}$
    $D \leftarrow \emptyset$ {D is the set of paths that dominate P}
    for all $e \in n(P)$ **do** {construct D iteratively}
      $p(e) \leftarrow p(e) \setminus \mathcal{R}$
      if $D = \emptyset$ **then**
        $D \leftarrow p(e)$
      else
        $D \leftarrow D \cap p(e)$
    for all $Q \in D$ **do**
      $d(Q) \leftarrow d(Q) \cup \{P\}$
  for all $P \in \mathcal{C}$ **do** {update score estimates}
    $\hat{s}(P) \leftarrow (|C(P)| + |\bigcup_{Q \in d(P)} C(Q)|)/|n(P)|$
return $H \leftarrow G(\mathcal{P})$

and output graphs were simple text files. For performance analysis we have measured the total elapsed (wall clock) time in seconds. All tests have been run on *Ukko* computing cluster[2] running GNU/Linux operating system, with no competing processes during the tests. Each node in the cluster is equipped with a four-core Intel Xeon E5540 CPU running at 2.53 GHz. We did not implement any explicit parallelism, so only one CPU core was used in each run.

5.6.1 Data sets

All our data sets are subgraphs from the complete Biomine graph or the complete DBLP graph. We use subgraphs because the methods can not, at their current form, handle millions of vertices and edges that would be required if complete graphs were to be used. Both Biomine and DBLP graphs are stored in a relational database management system from which we query the input graphs.

The input graphs were retrieved with *Crawler* that is the subgraph query component of Biomine. Crawler is currently undocumented. Given an input graph G, terminals s and t, and a vertex budget N it works roughly as follows. First Crawler finds a large set of best (most probable) s–t-paths from G, and it records for each encountered vertex v the probability $p(v)$ of the best path between the terminals s and t that contains v:

$$p(v) = \max\{\Pr(P) : v \in P, P \text{ is an } s\text{–}t\text{-path}\}.$$

Then Crawler sequentially picks vertices with the largest $p(v)$ values. Crawler can be parametrised to fetch a fixed number of vertices from G or all vertices with $p(v) \geq p$ for a given $p \in [0,1]$. Finally, Crawler returns the subgraph induced by the chosen vertices.

All graphs were preprocessed in two steps. First, if there were any parallel edges $e_i = (u,v)$, $1 \leq i \leq k$ ($k \geq 2$), they were reduced into a single edge (u,v) that was assigned a probability

$$p(u,v) = 1 - \prod_{1 \leq i \leq k} \big(1 - p(e_i)\big).$$

Second, any edges (u,v) with $p(u,v) = 1$ were removed by contracting them. Such edges are common in Biomine graphs and they pose a problem for MCP (see Section 5.3.1). Any parallel edges resulting from edge contractions were removed in the same way as above. Both of these steps removed about 10 % of edges, but they did not alter the reliability of the graphs.

[2]The cluster is located at the Department of Computer Science, University of Helsinki.

Biological graphs

Our biological data source is Biomine (Section 4.1). For the experiments we designed three different scenarios, and then we extracted a few data sets for each scenario (Table 5.2):

- Scenario 1 (Dys): The user is interested in a known relationship between entities. The test case is built around a dyslexia gene vertex (Entrez gene id 6091) and a dyslexia phenotype vertex (OMIM id 127700). We extracted graphs of different sizes for this pair of vertices to test the methods on graphs with different sizes.

- Scenario 2 (Conn): The user is interested in a putative relationship between entities. For each data set in this scenario we selected two random vertices with a shortest connecting path of exactly three edges, and then we extracted a subgraph using these two vertices as query vertices.

- Scenario 3 (Gene): The user is interested in entities with no known connection. For each data set in this scenario we selected two random gene vertices both having degree 10, and then we extracted a subgraph using these two vertices as query vertices.

With Dys graphs we used Crawler probability threshold $p = 0.01$. The whole Biomine graph was used as the input graph G for Crawler. For the scalability test sets (Dys1–7) we used different strategies. Dys1 and Dys2 were queried with different search radii (probability threshold p) to get graphs with different sizes. Dys3–7 graphs were extracted from a large graph of 13,828 edges by randomly removing roughly 2,000 edges at a time, so that every new smaller graph is a subgraph of the previous one, i.e. DYS3 $\subset$ DYS4 $\subset$ DYS5 $\subset$ DYS6 $\subset$ DYS7.

Coauthor graphs

Our coauthorship data source is DBLP: a large computer science bibliography index for publications from journals, conference proceedings, workshops and the like. We view DBLP as a social network with authors and publications as vertices. Each author is connected to articles that he or she has written. As a result we got a bipartite graph with 2,076,911 author and publication vertices connected by 3,293,211 edges. Edge weights were assigned as in Biomine (see Section 4.2) with parameters $rq = 0.8$ and $\alpha = 0.05$. Crawler was used to extract subgraphs from this bipartite graph.

As with Dys graphs we based DBLP graphs on three different scenarios each with two terminal vertices. Rakesh Agrawal (177 links to publications in our graph) and Jiawei Han (388 links) work on similar subfields of computer science

Graph	Terminals	$\|V\|$	$\|E\|$	$\|E_P\|$	R
Dys1	Entrez gene 6091, OMIM 127700	64	146	121	.9180
Dys2	Entrez gene 6091, OMIM 127700	452	1,113	1,000	.9939
Dys3	Entrez gene 6091, OMIM 127700	1,775	3,818	3,391	.0809
Dys4	Entrez gene 6091, OMIM 127700	2,482	5,778	5,075	.3201
Dys5	Entrez gene 6091, OMIM 127700	3,142	7,835	6,926	.9197
Dys6	Entrez gene 6091, OMIM 127700	3,692	9,856	8,694	.9976
Dys7	Entrez gene 6091, OMIM 127700	4,163	11,861	10,412	.9998
Conn1	PubMed 11960552, Entrez gene 2719	487	853	808	.9972
Conn2	GO 51605, Entrez gene 286971	989	2,727	2,496	1.0000
Conn3	Entrez gene 181788, Entrez gene 169026	591	1,800	1,770	.9958
Conn4	UniProt P84751, STRING ENSP297185	908	4,504	4,412	.6028
Conn5	UniProt Q8C3X4, UniProt Q921W0	1,172	3,513	3,298	1.0000
Gene1	Entrez gene 284467, Entrez gene 26231	16,811	51,672	47,813	.6819
Gene2	Entrez gene 389874, Entrez gene 80067	3,274	10,655	8,799	.4800
Gene3	Entrez gene 54758, Entrez gene 389874	13,875	41,377	37,961	.8118
Gene4	Entrez gene 55222, Entrez gene 652968	3,815	10,082	9,170	.2452
Gene5	Entrez gene 728731, Entrez gene 79645	3,781	10,922	9,999	.5258

Legend: $|V|$ = number of vertices; $|E|$ = number of edges in the unprocessed graph from Biomine database; $|E_P|$ = number of edges in the corresponding preprocessed graph with no parallel edges and edges with probability 1; R = reliability

Table 5.2: Biomine graphs used in the experiments.

Graph	Terminals	$\|V\|$	$\|E\|$	R
DBLP1	Rakesh Agrawal, Jiawei Han	502	1,021	1.0000
DBLP2	Rakesh Agrawal, Jiawei Han	1,000	2,182	1.0000
DBLP3	Rakesh Agrawal, Jiawei Han	2,501	5,532	1.0000
DBLP4	Rakesh Agrawal, Jiawei Han	5,002	11,229	1.0000
DBLP5	Rakesh Agrawal, Jiawei Han	10,002	22,987	1.0000
DBLP6	Donald E. Knuth, Edsger W. Dijkstra	504	869	0.9868
DBLP7	Donald E. Knuth, Edsger W. Dijkstra	1,002	1,826	0.9959
DBLP8	Donald E. Knuth, Edsger W. Dijkstra	2,501	4,598	0.9990
DBLP9	Donald E. Knuth, Edsger W. Dijkstra	5,002	9,985	0.9991
DBLP10	Donald E. Knuth, Edsger W. Dijkstra	10,001	21,022	0.9997
DBLP11	Heikki Mannila, Mark de Berg	501	977	1.0000
DBLP12	Heikki Mannila, Mark de Berg	1,000	2,150	1.0000
DBLP13	Heikki Mannila, Mark de Berg	2,501	5,653	1.0000
DBLP14	Heikki Mannila, Mark de Berg	5,000	10,995	1.0000
DBLP15	Heikki Mannila, Mark de Berg	10,000	22,766	1.0000

Legend: $|V|$ = number of vertices; $|E|$ = number of edges; R = reliability

Table 5.3: DBLP coauthor graphs used in the experiments.

and represent strongly connected terminals. Heikki Mannila (171 links) and Mark de Berg (157 links) in turn work on different subfields, and they likely are less strongly connected in the graph than Agrawal and Han. Finally, Donald E. Knuth (99 links) and Edsger W. Dijkstra (67 links) are prominent yet unrelated authors.

We extracted four graphs for each pair of authors (Table 5.3). The largest graphs (DBLP5, DBLP10 and DBLP15) have approximately 10,000 vertices and 20,000 edges, and they were extracted with Crawler parametrised with the number of vertices using the whole DBLP as an input graph. Smaller graphs (DBLP1–4, DBLP5–9, and DBLP10–14) are sequentially extracted subgraphs from the larger graphs with approximately 500, 1,000, 2,500 and 5,000 vertices. For example, DBLP4 is a subgraph of DBLP5 extracted with Crawler using DBLP5 as the input graph, DBLP3 is a subgraph of DBLP4, and so on.

One should note that this extraction strategy, while conceptually similar, produces very different graphs than the one used with Dys3–7 graphs. Dys graphs were generated by randomly removing edges from larger graphs, whereas DBLP graphs were extracted by Crawler. Since Crawler tries to preserve best paths between the query vertices we can expect DBLP graphs to have much larger subgraph reliabilities. This property will indeed become obvious in Section 5.6.3.

5.6.2 Default parameters

The terminal vertices for each input graph are specified in Tables 5.2 and 5.3. Our emphasis is on extracting small subgraphs ideally suitable for interactive exploration, so the subgraphs should have at most dozens of edges. In most of our tests we vary the result subgraph size from 10 to 250 edges to study the effect of the subgraph size on reliability and running time. To study the effect of input graph size and other parameters we extract a subgraph with a fixed size of 50 edges (Biomine graphs) or 80 edges (DBLP graphs).

The last step of the BPI and SPA algorithms involves pruning the few excess edges by the MCP algorithm. For this we used 1 million iterations, which should be a safe choice without a significant computational overhead for an extracted graph of at most 250 edges.

For SPA we used Algorithm 10 with $C = 1$: we did not use alternative initial solutions except in the tests devoted to their effects. However, Algorithm 10 with $C = 1$ *does* use more than one initial solution if required. (Recall that one initial solution might not be enough to get a large enough subgraph.)

MCP was run with 1 million Monte Carlo iterations unless otherwise stated. For MCPre we used different re-estimation strategies depending on the case. The parameters are always specified with the results: the notation 10000/50 indicates that 10,000 MC iterations were performed every 50 edge removals to re-estimate the edge relevance values.

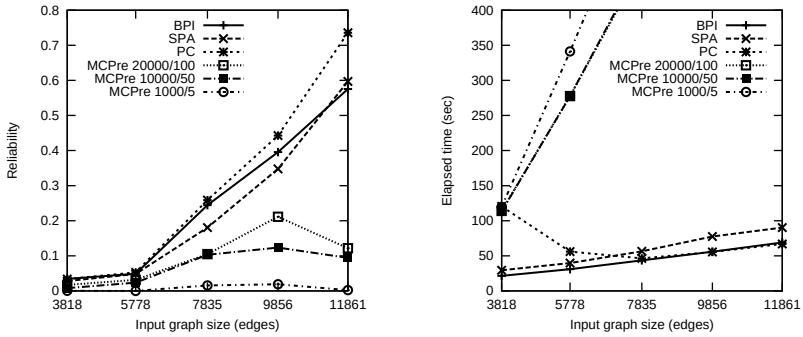
PC was run with the following parameters. In the first phase we produced $2 \cdot K$ candidate paths. In the second phase 10,000 iterations were done. The cut sampling algorithm (Algorithm 12) turned out to work well and was used in all tests. We justify these default choices in Section 5.6.6.

To control random variation we report averages over 50 independent test runs. All reported reliability values (relative and absolute) are estimates calculated with crude Monte Carlo method using 1 million iterations. For the sake of brevity most of the results we show are representative samples.

5.6.3 Scalability

We first study the scalability of the methods in terms of the reliability of the produced subgraphs and the running times. For the experiments we used Dys3–Dys7 and all DBLP graphs (see Tables 5.2 and 5.3). Using BPI, SPA, MCP and MCPre we extracted a subgraph of 50 edges from each Dys graph and a subgraph of 80 edges from each DBLP graph.

With Dys3–7 graphs BPI, SPA and PC produce similar reliabilities approximately in the range 0.04–0.75 (Figure 5.4a). Smaller input graphs result in smaller reliabilities in the extracted subgraphs, since the original graphs have smaller reli-



(a) (b)

Figure 5.4: Reliabilities (a) and running times (b) on Dys3–7 graphs with a fixed subgraph size (50 edges). Input graph sizes are given on x-axis.

abilities themselves due to random removal of edges while generating the graphs. MCPre produces relatively poor results with reliabilities always below 0.25. MCP without re-estimation fails completely, and in these tests it never returned a connected subgraph (and thus does not appear in the figure). Actually, in all our tests involving Dys graphs with over 1,000 edges all variants of MCP either failed to connect the terminals or produced clearly inferior subgraphs compared to BPI and SPA.

As mentioned in Section 5.3.1, the poor performance of MCP and to some extent of MCPre is because the true $R(G-e)$ values approach zero when the number of edges grows and the accuracy of the corresponding estimates is not sufficient to separate edges. Consequently MCP removes edges almost by random. MCPre partially avoids this deficiency by re-estimating $R(G-e)$ values at some intervals, but the effect seems to be weak.

The computational scalability of BPI, SPA and PC is good (Figure 5.4b). Running times are in the range 40–120 seconds. The Monte Carlo based alternatives MCP and MCPre are clearly slower: their running times are between 110 and 1,300 seconds. These running times depend on the frequency of re-estimation and the number of MC iterations used. However, even with these high computational costs they were not able to produce reliable results.

DBLP graphs give similar results although there are couple of differences. First, the subgraph reliability over all input graphs is almost constant as depicted in Figures 5.5a, 5.5b and 5.5c. The result suggests that the sequential subgraph extraction with Crawler preserves reliable subgraphs. Second, on DBLP11–15 graphs BPI is significantly weaker than other methods falling even below MCPre on smaller input graphs. This is due to the fact that most of the best (most prob-

able) paths form a small cut of three edges in these graphs. Consequently BPI does not gather many independent connections between the terminals and the reliability stays low. PC, on the other hand, performs consistently well on all DBLP graphs. MCPre fares again worse, and it is not robust as its performance tends to decrease with larger input graphs.

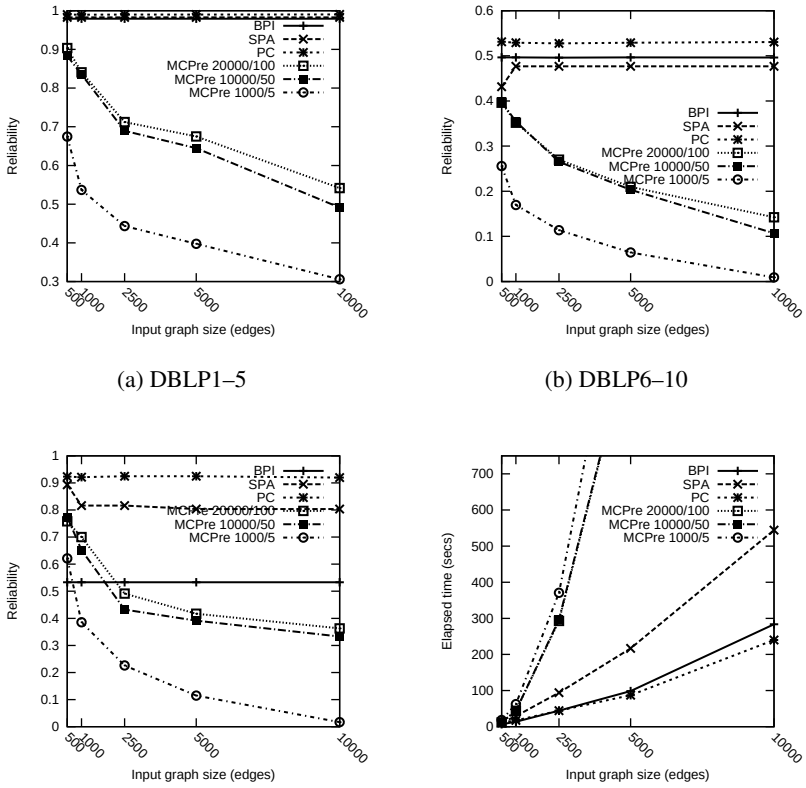
Figure 5.5d is a representative depiction of running times for BPI, SPA and PC on DBLP1–5 graphs. Elapsed times are reasonable, ranging from few seconds on the smallest graphs to about 550 seconds on the largest graphs. As with Dys graphs, MCPre becomes quickly infeasible when input graphs become larger.

Based on the results, BPI, SPA and especially PC clearly outperform the simple MCP and MCPre methods. This was the case in terms of the quality of the results and the running time. The quality/time tradeoff can be adjusted by tuning the parameters, but improving one will further degrade the other one. It does not seem plausible that MCP methods could be useful in practise for large input graphs. In contrast, BPI, SPA and PC are feasible when applied to input graphs with thousands of edges. We emphasise that the reported running times are, for the sake of comparability, from unoptimised implementations. With our optimised implementation of the PC algorithm, for example, the largest input graphs with 10,000 vertices and 20,000 edges can be handled in less than 10 seconds (results not shown).

5.6.4 Performance on small data sets

While our emphasis is on scalability and large input graphs, it is also useful to test the methods on a small data set. (1) For small data sets it is feasible for MCPre to re-estimate edge relevance values after each edge removal. This is an interesting reference for analysing the quality of the results of BPI, SPA and PC even if the running time of MCPre may be too large for practical applications. (2) We proposed to use MCP to fine tune the results of BPI and SPA to the requested size, and we are interested in studying how well MCP works in this setting. From now on we report *relative reliabilities*: how large the subgraph reliability is in proportion to the (static) input graph reliability. This is useful for several reasons: the input graph reliability is the absolute upper bound for the subgraph reliability, thus giving an idea of the optimality of heuristics; normalised reliabilities lets us compare results between different input graphs; and we can observe how relatively small subgraphs can be used to capture a large proportion of reliability.

The performance of all methods is satisfactory when the task is to remove edges from a small graph Dys1 (141 edges). The best reliabilities are obtained by PC and MCPre where edge relevance values were re-estimated with 100,000 MC iterations after each edge removal (Figure 5.6a). However, this quality comes at a substantial computational cost. The running times of MCPre 100000/1 (Fig-



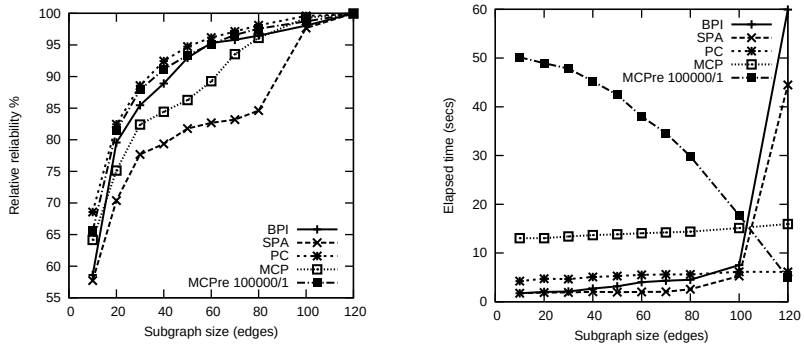
(a) DBLP1–5

(b) DBLP6–10

(c) DBLP11–15

(d) DBLP1–5

Figure 5.5: Subgraph reliabilities on DBLP1–5 (a) DBLP6–10 (b), and DBLP11–15 (c) graphs with a fixed subgraph size of 80 edges. Running times on DBLP1–5 graphs are given in (d), results on other DBLP graphs were analogous. Input graph sizes are given on *x*-axis.



(a) (b)

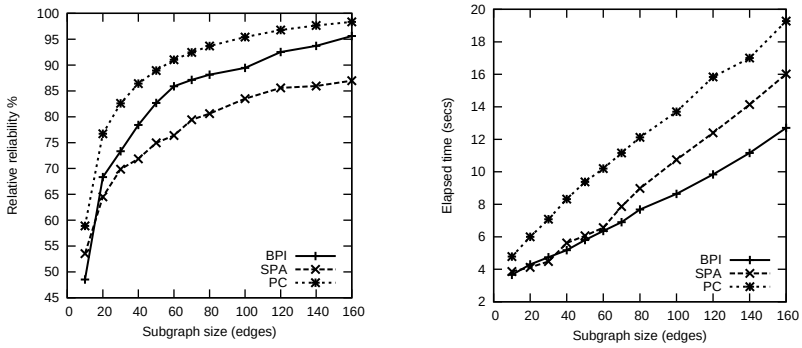
Figure 5.6: Reliabilities (a) and running times (b) for the small input graph Dys1 with 141 edges.

ure 5.6b) illustrate its dependence on the number of edges removed: a property that makes the method unsuitable for large graphs.

BPI and PC have a consistent performance both in terms of reliability of the result and computation time. SPA fares worse than BPI and PC, but it is still able to produce relatively reliable subgraphs. The peculiar sharp rise on running time with BPI and SPA depicted at the right end of Figure 5.6b is due to the exponentially growing need for best paths: with BPI such paths span the solution subgraph (Section 5.3.2), and with SPA they are used to initialise a (possibly partial) solution (Section 5.4). This behaviour is especially likely to occur when the subgraph size is close to the input graph size. Finally, the results indicate that MCP is useful on small graphs when relatively few edges are removed. This is good news for BPI and SPA that use MCP to adjust their initial result to the required size.

5.6.5 Subgraph reliability

We now take a closer look at the performance of BPI, SPA and PC. The optimal reliability is not known so we have to base the analysis mostly on comparisons of the relative performances. We know, however, that the result has to be close to optimal if the reliability is close to the (estimated) reliability of the input graph. In such cases the relative subgraph reliability is close to 100 %. This is achieved for very reliable graphs Dys2 (Figure 5.7a); Conn3 and Conn5 (Figures 5.8a and 5.8b); and DBLP2 (Figure 5.8e). In these cases the original graphs with over 1,000 edges can be reduced to circa 40 edges while maintaining a reliability of at least 85 %.



(a) (b)

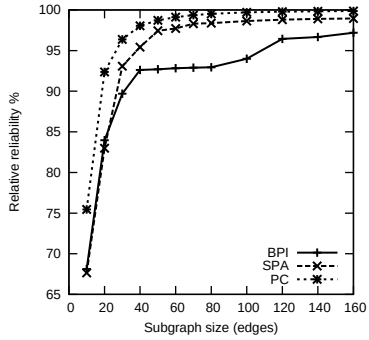
Figure 5.7: Subgraph reliabilities (a) and running times (b) for the medium-size input graph Dys2 with 1,113 edges.

In all test cases PC gave consistently superior results. It attained approximately 90 % or better relative reliability in the largest subgraphs. BPI and SPA were not as uniform in their performance. For the Conn and Dys2 graphs BPI tends to give better results with a small but a clear margin (Figures 5.7a and 5.8b, Figure 5.8a is an exceptional case). In these cases SPA with $C = 1.1$, 1.2 or 1.3 gives practically equal results to BPI (results not shown). With large values of C, SPA reduces to BPI: the series-parallel components consist of a single path each. This may happen already with the values of C we used, especially if there are very strong connections between the query vertices.

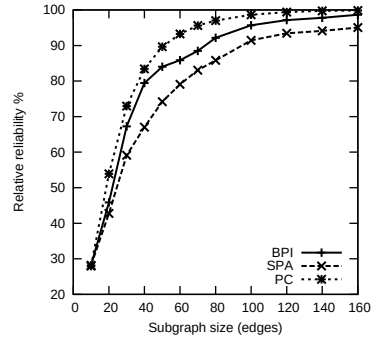
For the random gene pairs (Gene1–Gene5) the situation is completely the opposite: in most cases SPA outperforms BPI by a large margin (Figure 5.8c gives an example). The relatively poor performance of BPI is probably due to strong correlation between the best paths with distantly connected vertices, and BPI fails to acquire many independent connections in such cases.

In terms of running times, PC and BPI are the most efficient algorithms with virtually the same running times (see Figures 5.9c and 5.9d for representative examples). There are exceptions, however. With some graphs BPI needs to gather many paths when larger subgraphs are sought for (Figure 5.9a). Figure 5.9b represents a case where PC failed to rapidly sample paths on its first phase. In this particular case the reason were small cuts in the input graph. Such situation is difficult for Algorithm 11 even with cut sampler (Algorithm 12).

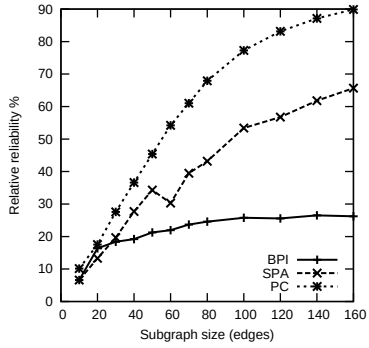
From Figure 5.8e we can see that a subgraph with 100 edges almost surely connects Han and Agrawal. Interestingly, there are only 12 author vertices between them in the extracted subgraphs on average, and a total of 47 distinct in-between authors over all 50 independent runs. Figure 5.1 is a simplified excerpt



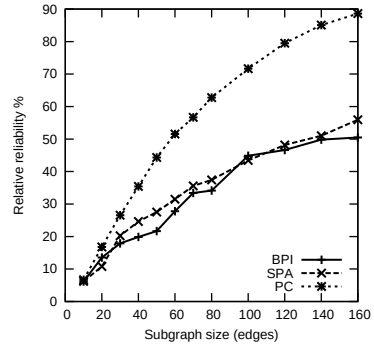
(a) Conn3



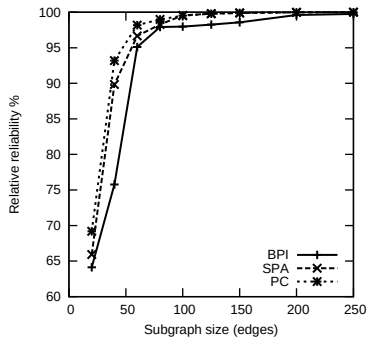
(b) Conn5



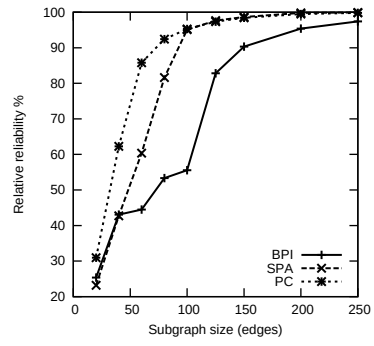
(c) Gene2



(d) Gene3

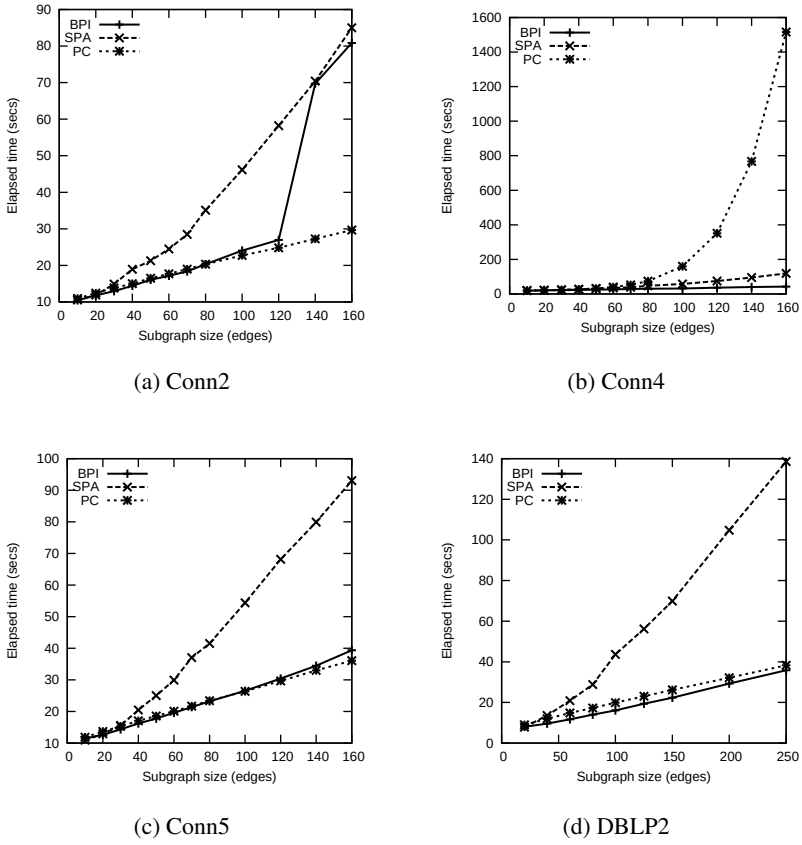


(e) DBLP2



(f) DBLP12

Figure 5.8: Subgraph reliabilities on Conn3 (a), Conn5 (b), Gene2 (c), Gene3 (d), DBLP2 (e), and DBLP12 (f) graphs.



(a) Conn2

(b) Conn4

(c) Conn5

(d) DBLP2

Figure 5.9: Running times on Conn2 (a), Conn4 (b), Conn5 (c) and DBLP2 (d) graphs.

of one such subgraph with 14 author vertices, where vertices are connected if the corresponding authors have coauthored at least one article. On the contrary, Mannila and de Berg (Figure 5.8f), and especially Knuth and Dijkstra (results not shown), are less strongly connected.

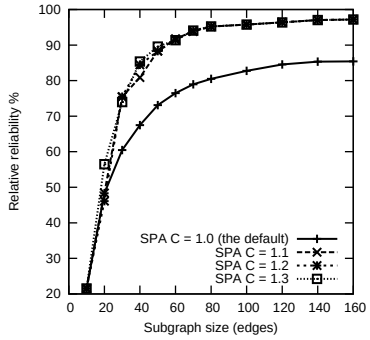
PC is the most robust algorithm across the tests we performed. Its subgraphs have consistently superior reliability and the running times are usually on par with the fastest methods. Overall, PC can be regarded as a safe and the best choice for most subgraph extraction tasks. The difference between SPA and BPI is not that clear: with distantly connected vertex pairs BPI tends to give inferior results than SPA, but with more strongly connected pairs BPI is usually clearly superior. SPA seems to be slightly more robust though, but also noticeably slower than BPI.

5.6.6 Algorithmic variants and parameters

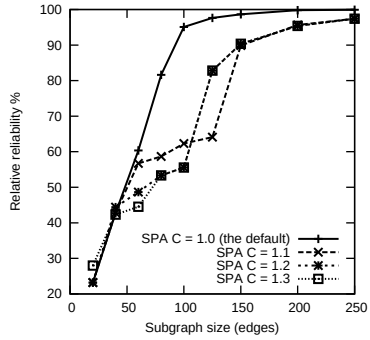
We next study variants of the SPA algorithm. It has the option of building several different series-parallel graphs from different starting points and using their union as the solution. By construction (Algorithm 10) this union is unlikely to be series-parallel, and thus it is less restricted. Recall that the use of several initial solutions is governed by parameter $C > 1$, although even in the case $C = 1$ Algorithm 10 may use more than one initial solution if the single series-parallel subgraph is not large enough. As soon as the addition of any path fragment does not improve the reliability of the solution by factor C at least the algorithm starts to build a new series-parallel graph.

We experimented with values $C \in \{1.1, 1.2, 1.3\}$ and compared them against SPA with $C = 1$. The observation is that $C > 1$, especially $C = 1.1$, can significantly improve the quality of the results (three out of five Conn graphs, one out of five Gene graphs, DYS2 graph). However, the quality may also remain the same (Conn4, DBLP7 graphs) or even decrease (all other 25 cases). See Figures 5.10a and 5.10b for two examples. In any case the use of several initial solutions results in longer running times with $C = 1.1$ being the most expensive of the tested variants (Figures 5.10c and 5.10d). The reason is that a lower value of C results in fewer but longer constructions of series-parallel components than larger values. Our conclusion is that multiple starts with $C = 1.1$ may be useful to improve the quality of the results, but since the results may also deteriorate and there is a substantial increase in running time, SPA with $C = 1$ is a safer choice.

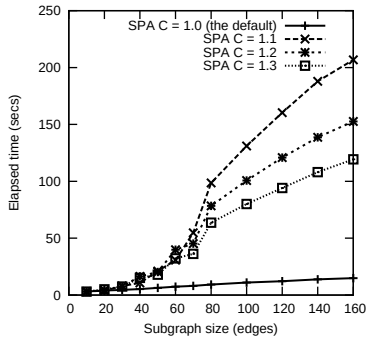
The PC algorithm has multiple parameters and algorithmic variants. We begin by evaluating the cut sampling algorithm (Algorithm 12) which approximates the basic crude Monte Carlo sampling (Algorithm 11). In all experiments the approximation did not have a significant adverse effect on the reliability of the result subgraph, regardless of the number of paths sampled (results not shown). The cut sampler is useful when a large number of paths are sampled or when the



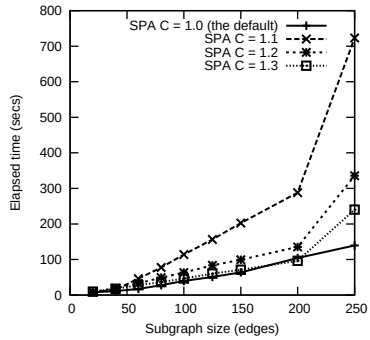
(a) Conn1



(b) DBLP12

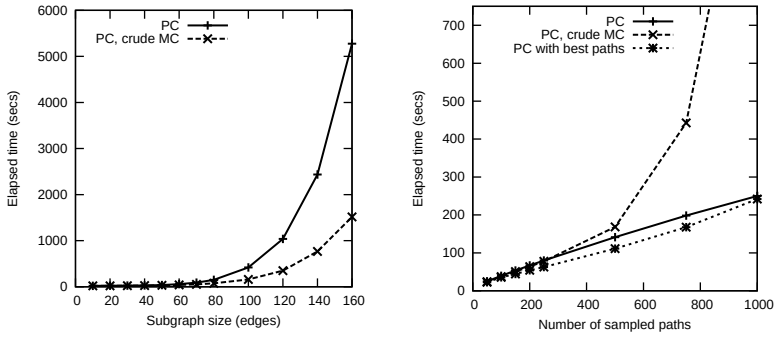


(c) Conn1



(d) DBLP12

Figure 5.10: Relative subgraph reliabilities and running times on Conn1 and DBLP12 graphs with different values of C in SPA algorithm.



(a) Conn4 (b) DBLP12

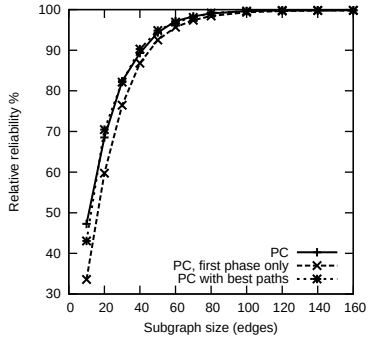
Figure 5.11: Cut sampler (PC) against crude MC sampler on Conn4 and DBLP12 graphs.

input graph is difficult for the crude MC sampler. Two examples are given in Figure 5.11. Note that Conn4 is difficult even for cut sampler (cf. Figure 5.9b). On DBLP12 graph the cut sampler has a linear time complexity, while the crude MC sampler becomes infeasible when more than 750 paths are needed.

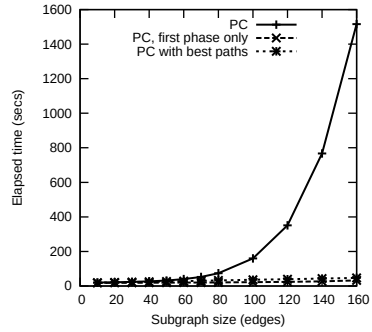
It should be noted that the basic sampler has a decent performance when the number of sampled paths is less than a couple of hundred. This is usually enough when small subgraphs are extracted, as we will see below. However, we found that the cut sampler is in any case almost as fast as the crude sampler even when the basic sampler performs well (results not shown). Thus it can be regarded as a safe choice.

Both phases of PC try to choose an optimal set of paths, but the first phase is constrained by the very large number of possible paths. The additional benefit of using a second phase to fine tune the result on DBLP and Conn graphs is consistent but not huge (Figure 5.12a). With Gene graphs the second phase adds considerably to the result quality (Figure 5.12c). It seems that when the terminals are less strongly connected, the effect of the second phase is larger. However, this improvement comes with a clear additional cost in computation time (Figures 5.12b and 5.12d). Apparently the first phase can be used alone as an approximate but extremely efficient algorithm when the terminal vertices are tightly connected. This is especially evident in Figure 5.12b which shows that even the previously problematic Conn4 graph can be easily handled with practically no loss in reliability.

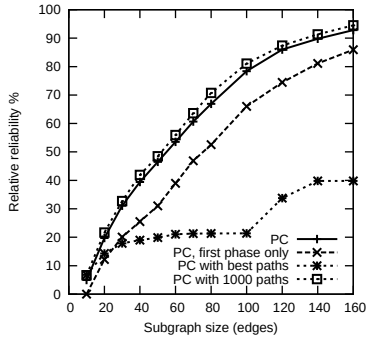
An alternative to the first phase would be to find the k best s–t-paths instead of k sampled paths. With best paths the PC algorithm becomes a greedy variant of BPI where the effect of each possible addition to the reliability is evaluated. In



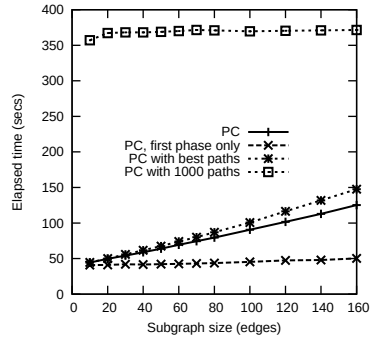
(a) Conn4



(b) Conn4

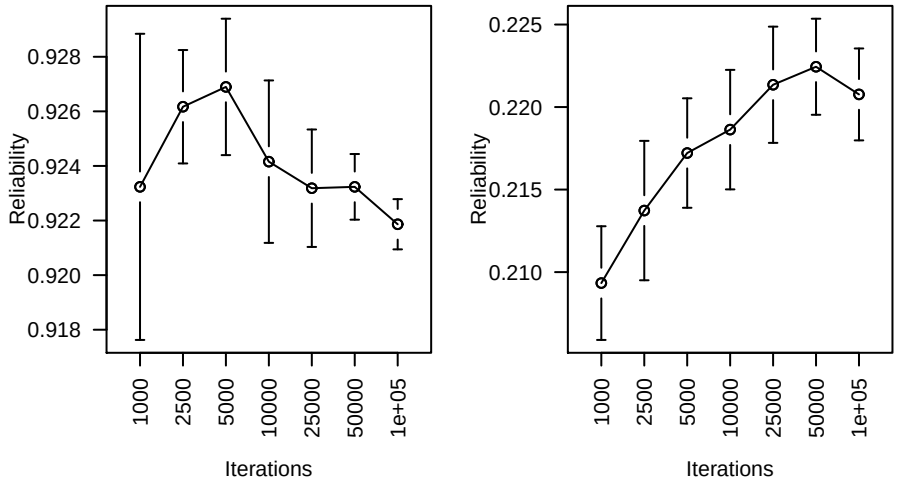


(c) Gene5



(d) Gene5

Figure 5.12: PC variants on Conn4 (a), (b) and Gene5 (c), (d) graphs.



(a) DBLP12 (b) Gene2

Figure 5.13: Estimated reliability means and 95 % confidence intervals with varying number of iterations in the second (path selection) phase of PC.

some cases using k best paths works equally well, but it can also perform significantly worse. See Figures 5.12a and 5.12c for examples. Finding a large number of best paths can become intractable without a specialised algorithm. For example, our (arguably simple) implementation ran out of memory when more than 1,000 paths were sought.

Both phases of PC have a parameter with which to tune the reliability/time trade-off: for the first phase, it is the number of sampled paths; for the second phase, it is the number of Monte Carlo samples (realisations). According to our results, sampling $2 \cdot K$ paths seems a reasonable compromise between efficiency and quality. Sampling 1,000 paths instead of 40–500 for subgraphs of 20–250 edges produces only marginally better subgraphs (Figure 5.12c) with a significant increase in running time (Figure 5.12d).

For the second phase of PC the default number of 10,000 MC iterations used in our experiments seems to be large enough to produce accurate estimates for path score calculations. Using smaller numbers of iterations can result in both smaller reliabilities and larger variances, but the differences are relatively small or negligible. Figure 5.13 gives two examples. The depicted mean reliabilities and 95 % confidence intervals were estimated from a sample of 50 independent PC runs with the same data and parameters. Subgraph size K was fixed to 80, and the number of second phase iterations varied between 1000 and 100,000. Sample means, sample standard deviations and the t-distribution were used to infer the confidence intervals.

5.7 Conclusions

As more and more domains of interest are best described as interlinked heterogeneous objects we can expect graphs to become data models of choice in many situations involving multi-relational data [47]. Weighted and especially random graphs can be used to naturally represent irregular, uncertain or probabilistic relationships.

Discovery of indirect links between vertices of a graph is a central task in graph mining. Non-trivial, indirect links in random graphs can become particularly laborious to discover and analyse due to their potentially myriad dependencies. Focusing on small, robust subgraphs with many independent and non-redundant connections between the vertices of interest can greatly simplify the situation. Reliable subgraphs can be expected to have these features.

We addressed the most reliable subgraph extraction problem (MRSP) [54]. It is a natural formulation of connection subgraph problem for random graphs, where the objective is to extract a small subgraph that connects the given vertices as strongly as possible. The MRSP is an inherently difficult problem: the search space of possible solutions is large, and the evaluation of the objective function (reliability) is difficult. It is unlikely that efficient exact algorithms for the MRSP can be formulated so approximations must be used in practise.

We reviewed few recently proposed algorithms for the MRSP [54, 56, 57]. The algorithms are iterative and greedy: they start from some initial solution and expand that (partial) solution towards a complete solution. The local search strategies for choosing the best addition into the partial solution differ, and so do the techniques for evaluating the quality of solutions. Table 5.4 summarises the heuristics used in the algorithms.

Finally, we demonstrated the usefulness of the algorithms with experimental results on graphs derived from a biological graph database Biomine and a computer science bibliography database DBLP. The results suggest that the current algorithms are able to efficiently find strong and independent connections using only a fraction of edges from the original input graphs. These subgraphs are useful for visualisation of large graphs, and they can be used as inputs for computationally demanding network analysis methods that do not scale well. The visualisation aspect is important in the last step of the gene mapping pipeline (Figure 1.1). With reliable subgraphs the investigator can find and evaluate the most relevant connections between her DS genes and the phenotype, and consequently evaluate possible hypotheses of functional relationship between them or even form new hypotheses.

Efficient methods should be developed in the future to support multiple terminal vertices—an important generalisation of the two-terminal problem we have considered. We are in the process of extending the Monte Carlo framework to han-

Algorithm	Local search strategy	Evaluation strategy
MCP	Remove the least critical edge at each iteration	MC estimation at the start (MCP), or MC estimation at specified intervals (MCPre)
BPI	Choose the k-th best path at k-th iteration	None
SPA	Choose the best "valid" path at each iteration	Explicit evaluation (solutions are series-parallel graphs)
PC	Choose the best path from a subset of all s–t-paths	MC estimation, view the problem as a variant of SET COVER

Table 5.4: Comparison of heuristics used in the algorithms. "Local search strategy" explains how the algorithm chooses the best possible addition at each iteration. "Evaluation strategy" describes how the quality of an addition is evaluated.

dle multiple terminal vertices, and the necessary modifications have been sketched by Kasari and others [69]. Future experiments include systematic tests to find out robust parameter values that perform reliably over wide range of input graphs and query nodes and extensive comparisons with related algorithms.

Another interesting research topic is the generalisation of connection subgraphs: how to define and find, in an unsupervised manner, strongly connected subsets, or clusters of terminals? Although this question has gained some research attention [126], the usefulness of reliability in this setting is an open question.

Chapter 6

Conclusions

Researchers in life sciences, such as biology and medicine, are nowadays faced with overwhelming amounts of raw data. Modern high-throughput technology allows many magnitudes larger experiments than before resulting in larger and more informative samples. Data analysis becomes more demanding as the size of the data grows, and sophisticated analysis methods that are both scalable and suitable for such multi-featured data are required.

On the other hand, investigators *must* gather vast amounts of data to tackle genetics of complex diseases. A typical disease gene mapping project starts with a genome-wide scan where several thousands of markers are genotyped from both affected and healthy individuals. At first, an initial analysis is performed to find a smaller set of candidate genes that show a statistical linkage to the disease. It is likely that only a small subset of these genes has a real, functional effect on the disease. As investigation of all candidate genes is infeasible due to limited resources, further analysis has to be focused on the most promising candidate genes.

In this thesis, we collected the aforementioned problems into a simplified gene-mapping pipeline (Figure 1.1), and we proposed computational tools to improve the efficiency of this pipeline. Our focus was on association studies, where genes are mapped using samples from general population instead of utilising large family pedigrees as in linkage studies. We argue that these tools and techniques can bring more information (e.g. localisation power, accuracy, refined hypotheses) for the same investment, thus improving the efficiency of the gene-mapping effort.

In Chapter 2 we proposed a novel two-phase simulation procedure. It is useful in the study design phase of the pipeline: with simulated data sets, one can assess the power and localisation accuracy of a hypothesised study design and the chosen computational tools. We demonstrated this in Chapter 3 by reviewing the results of our empirical comparison between two study designs for association-

based gene mapping [55]. The results suggest that a population-based study combined with statistical haplotyping can be much more powerful than the traditional family-based (trio) design, assuming equal genotyping costs. In other words, increasing the sample sizes by genotyping more cases and controls instead of parental genotypes, as would be required in a trio-based study, one can achieve better statistical power to detect associations for the same genotyping cost.

Simulation tools are indispensable in method development as well. Our prototype simulator has been used in haplotyping and gene-mapping research [35, 88]. Although there are more recent simulators available which are feature-wise on a par with or beyond our prototype implementation [102], the proposed two-phase simulation model is still unique. It could be beneficial to implement the model within some recent simulator.

For the refined analysis phase of the pipeline we introduced the Biomine graph database (Chapter 4). Biomine is effectively a large index of publicly available biological databases [116]. By representing the current biomedical knowledge as a graph we can harness graph mining techniques to rank candidate disease-susceptibility genes. Viewing biological concepts and their relations in a graph form is not a new idea [11, 116, 48], but the probabilistic interpretation of edges in Biomine is likely unique. This interpretation is intuitive as it reflects the implicitly uncertain nature of the biomedical knowledge, and it lets us use random graph concepts to model links and connections as we did in Chapters 4 and 5.

As demonstrated by the experiments in Chapter 4, the Biomine graph model can be used to assess strengths and statistical significances of gene–gene and gene–phenotype links. Further experiments suggest that Biomine can even be used to predict gene–phenotype links [81, 36]. While these results are promising they also show that the edge weighting should be adjusted to reflect the specific needs of the applications. For example, when gene–phenotype links are sought, the ranking and prediction accuracy can be improved by tuning the edge weighting parameters [36]. These parameters could be learned automatically in a supervised manner for different applications. Another issue is how to choose the most suitable definition for links and their strengths. Again, it might be that several different approaches on different levels of detail and integration are needed, and that they complement rather than compete with each other.

Finally, in Chapter 5, we focused on reliable subgraphs. Reliable subgraphs are related to information search or discovery from graphs: the user initiates a query by specifying some search terms, and she wishes to obtain a subgraph containing concepts and relationships that are related to the search terms. Reliable subgraphs are exploratory and useful in the very last steps in the gene-mapping pipeline, where the investigator already has a small, ranked set of putative disease-susceptibility genes or a novel hypothesis that a particular gene might have an

effect on the phenotype she is studying. The "search engine" we envision would now allow the scientist to search for relevant information connecting the gene and the phenotype. The Biomine database combined with reliable subgraph extraction [57, 69], context-free grammar [115] or some other query engine forms one such biological search engine.

We formulated the subgraph extraction problem as the most reliable subgraph problem (MRSP), and we reviewed some recent algorithms for solving the problem [54, 56, 57]. The algorithms are restricted to two query vertices, or terminals. In applications, such as gene mapping, connections between multiple terminals are common [126]. We are currently focusing on algorithms to handle such cases [69]. Another interesting future research direction is to modify the MRSP to incorporate information about the connections into the problem. In its current form, the reliability does not care about the "type" of the connection but only about the probability of its existence. To emphasise certain kinds of connections it could be beneficial to limit the types of allowed connections, for example by restricting the allowed paths to have only a certain length or certain types of edges. Especially in k-terminal cases this is highly nontrivial. Possible solutions include probabilistic context-free grammars [115] and probabilistic logic programming [106].

The usefulness of the proposed tools in real-life gene-mapping projects is yet to be established. Note that our abstract gene-mapping pipeline (Figure 1.1) is idealised and each real gene-mapping project is unique. Thus it is probably not useful to try to fit a particular project into this idealised form, but use the individual tools separately where applicable instead. Especially the gene–phenotype link mining and reliable subgraphs could be utilised in this manner.

Although we focused primarily on gene mapping in this thesis, the applicability of link mining and reliable subgraph techniques (Chapters 4 and 5) is clearly not limited to genetics. Indeed, exactly the same machinery and algorithms were applied both to Biomine and DBLP bibliography databases in the experiments of Chapter 5. This experience suggests that extending the proposed frameworks to new domains is viable and the applications are yet emerging.

Genetics glossary

This glossary is directly based on previous work [125] and included here with permission[1]. The material has been revised and expanded for this thesis.

Allele	An alternative form of a gene or a marker. See also *Mutation*.
Association study	Association studies try to find correlation of presence of a disease or a trait with presence of certain marker alleles at the population level. Compare with *Linkage study*.
Base pair (bp)	A pair of complementary nitrogenous bases (adenine and thymine or guanine and cytosine) in a DNA molecule. Also, the unit of measurement for DNA sequences (for example 200 bp).
Case–control study (population study)	A study design where two unrelated samples are used. Other sample consists of *cases* (e.g. individuals with a disease phenotype) and another one consists of *controls* (individuals without a disease phenotype). Compare with *Family-based study*.
Chromosome	A single DNA molecule containing genes (and markers) in linear order. In humans 23 pairs of chromosomes, each pair containing one chromosome from each parent, carry the entire genetic code.
Coalescent	A statistical model for a genealogy of a sample of sequences (e.g. chromosomes).
Complex disease, complex trait	A disease or trait whose phenotype is affected by many genes and environmental factors. See also *Disease model*.
Crossing over, crossover	The interchange of segments between pairing homologous chromosomes during meiosis.

[1] Courtesy of Päivi Onkamo, Department of Biosciences, University of Helsinki.

Diploid	Diploid cells have two homologous copies of each chromosome. One of the copies is inherited from the mother and the another from the father. Diploid organisms, such as humans, have diploid cells. Compare with *Haploid*.
Disease gene mapping (localisation)	The act of mapping a disease-susceptibility gene to a specific location on some chromosome. See also *Association study* and *Linkage study*.
Disease model	The number of genes, environmental factors and interactions which affect the disease susceptibility for a certain disease. Disease with genetic contribution may be monogenic (Mendelian one-gene disease), oligogenic where just a few genes are involved, or polygenic with several genes with weak effects each, for example.
Disease-susceptibility (DS) gene	A gene which is suspected to be linked to a disease phenotype. Carrying a certain allele or alleles of a disease-susceptibility gene increases the statistical probability to develop that disease. See also *Genetic association*.
Family-based study	A study design where samples consist of familial entities (from trios to complete pedigrees). Linkage studies are family-based. Compare with *Association study*.
Founder population	A relatively small (100–1,000) set of individuals that formed a new population in history.
Gene	Basic element of heredity that determines traits by coding for proteins.
Gene mapping	The act of mapping genes to specific locations on chromosomes. See also *Disease gene mapping*.
Genetic association	Correlation of presence of a disease or a trait with presence of certain marker allele(s) (or alleles at genes) observed at the population level.
Genetic drift	Change in the frequency of an allele in a population over time due to random sampling.
Genome	A complete collection of organism's hereditary information.

Genotype	The particular alleles present at a specified locus in an individual.
Haploid	Haploid cells have only a single copy of each chromosome. Asexually reproducing organisms and the gametes of sexually reproducing (diploid) organisms are haploids. Compare with *Diploid*.
Haplotype	A string of alleles from genes or markers which are located closely together on the same chromosome and which tend to be inherited together. See also *Linkage*.
Haplotyping	The act of inferring parental origins (phase) of haplotypes.
Identity by descent (IBD)	Two chemically identical alleles are identical by descent when they have been inherited from a common ancestor. See also *Identity by state*.
Identity by state (IBS)	Any two copies of an allele which are chemically identical. They are not necessarily inherited from the same ancestor. See also *Identity by descent*.
Isolated population	A relatively young, usually geographically distant population formed by a set of founders in a recent history which has not been significantly affected by immigration from another population(s). Isolated population has generally less genetic variation than the general population from which it originated (*founder effect*). See also *Founder population*.
Linkage	The tendency of genes in proximity of each other to be inherited together. The closer the loci, the greater the probability that they will be inherited together.
Linkage disequilibrium (LD)	A non-random association between alleles at separate loci. Alleles that occur together at population level more often than can be accounted for by chance are said to be in linkage disequilibrium. Linkage disequilibrium often indicates that the loci are physically close to each other on the chromosome.
Linkage study	Linkage studies try to find correlation of presence of a disease or a trait with presence of certain marker alleles at the family (or pedigree) level. Compare with *Association study*.

Locus (plural loci)	The specific site of a particular gene or marker on its chromosome.
Marker	A gene or a stretch of non-coding DNA sequence whose alternative forms (alleles) can be reliably detected by genotyping technologies.
Marker map	The set of markers chosen for some particular mapping study.
Meiosis	Cell division that produces reproductive cells in sexually reproducing organisms. The nucleus divides into four nuclei each containing half the chromosome number (leading to gametes in animals and spores in plants).
Mendelian trait	A trait that is regulated by a single locus and shows a simple Mendelian inheritance pattern. See also *Disease model*.
Morgan (M), centiMorgan (cM)	A length of a chromosomal segment in which an average of 1 crossover occurs per generation. One centiMorgan (cM) is $1/100$ M. In humans 1 cM is equivalent, on average, to 1 million base pairs of DNA.
Mutation	A heritable change in DNA sequence. Possible changes include single nucleotide polymorphisms (SNPs), short tandem repeats (STRs), deletions and inversions. A particular mutation is recognised as an allele.
Pedigree	A family tree diagram which shows the genetic history of a particular, often multigenerational, family.
Penetrance	The probability of expressing a trait (e.g. a disease) when carrying a particular variant of an associated gene (e.g. a disease-predisposing mutation).
Phase	The parental origin of a haplotype or chromosome.
Phenocopy	An individual expressing a phenotype with environmental instead of genetic basis.
Phenotype	The observable and measurable characteristics of an organism, e.g. presence of a disease, which may or may not be genetic. See also *Penetrance* and *Phenocopy*.
Population	A group of organisms of the same species relatively isolated from other groups of the same species.

Population model	A mathematical model of population and its characteristics that vary over time such as size, expansion and/or decline rates, migration and substructures (subpopulations).
Prevalence	The proportion of affected individuals in the given population at a given time.
Proband	An individual with a genetic disorder or other specific phenotype when this leads to the investigation of the individual's family.
Pseudo-control	Hypothetical control individual whose genotype consists of non-inherited parts of parental genotypes.
Recombination	The process by which offspring derive a combination of genes (or markers) different from that of either parent. Occurs by crossing over during meiosis.
Selection	A process in which individual organisms that possess favourable traits are more likely to survive and reproduce.
Single nucleotide polymorphism (SNP)	An allele that differs only in a single base pair.
Trio, triplet	An offspring and the parents (family trio).
Wild type allele	A hypothetical "standard", or "normal" allele at a given locus. Contrast with a "non-standard", or "mutant" allele. See also *Mutation*.

References

[1] Lada A. Adamic and Eytan Adar. Friends and neighbors on the Web. *Social Networks*, 25(3):211–230, July 2003.

[2] Joshua Akey, Li Jin, and Momiao Xiong. Haplotypes vs single marker linkage disequilibrium tests: what do we gain? *European Journal of Human Genetics*, 9:291–300, 2001.

[3] Russ B. Altman, A. Keith Dunker, Lawrence Hunter, Tiffany A. Jung, and Teri E. Klein, editors. *Pacific Symposium on Biocomputing 2004*, Hawaii, USA, January 2004. World Scientific.

[4] B. Aranda, P. Achuthan, Y. Alam-Faruque, I. Armean, A. Bridge, C. Derow, M. Feuermann, A. T. Ghanbarian, S. Kerrien, J. Khadake, J. Kerssemakers, C. Leroy, M. Menden, M. Michaut, L. Montecchi-Palazzi, S. N. Neuhauser, S. Orchard, V. Perreau, B. Roechert, K. van Eijk, and H. Hermjakob. The IntAct molecular interaction database in 2010. *Nucleic Acids Research*, 38:D525–531, January 2010.

[5] D. J. Balding, M. Bishop, and C. Cannings, editors. *Handbook of Statistical Genetics*, volume 2. John Wiley and Sons, Chichester, UK, second edition, 2003.

[6] Michael O. Ball. Complexity of network reliability computations. *Networks*, 10(2):153–165, 1980.

[7] Francois Balloux. EASYPOP (version 1.7): a computer program for population genetics simulations. *Journal of Heredity*, 92(3):301–302, 2001.

[8] M. P. Bass, E. R. Martin, and E. R. Hauser. Pedigree generation for analysis of genetic linkage and association. In Altman et al. [3], pages 93–103.

[9] M. Boehnke. Estimating the power of a proposed linkage study: a practical computer simulation approach. *American Journal of Human Genetics*, 39(4):513–527, October 1986.

[10] F. T. Boesch, A. Satyanarayana, and C. L. Suffel. A survey of some network reliability analysis and synthesis results. *Networks*, 54(2):99–107, September 2009.

[11] Catherine Bounsaythip, Erno Lindfors, Peddinti V. Gopalacharyulu, Jaakko Hollmén, and Matej Orešič. Network-based representation of biological data for enabling context-based mining. In Catherine Bounsaythip, Jaakko Hollmén, Samuel Kaski, and Matej Orešič, editors, *Proceedings of KR-BIO'05, International Symposium on Knowledge Representation in Bioinformatics*, pages 1–6, Espoo, Finland, June 2005. Helsinki University of Technology, Laboratory of Computer and Information Science.

[12] Ulrik Brandes and Daniel Fleischer. Centrality measures based on current flow. In Volker Diekert and Bruno Durand, editors, *STACS 2005*, volume 3404 of *Lecture Notes in Computer Science*, pages 533–544, Stuttgart, Germany, February 2005. Springer.

[13] Sergey Brin and Lawrence Page. The anatomy of a large-scale hypertextual web search engine. *Computer Networks and ISDN Systems*, 30(1–7):107–117, April 1998.

[14] Sharon R. Browning and Brian L. Browning. Rapid and accurate haplotype phasing and missing-data inference for whole-genome association studies by use of localized haplotype clustering. *American Journal of Human Genetics*, 81(5):1084–1097, November 2007.

[15] Francesc Calafell, Elena L. Grigorenko, Alexei A. Chikanian, and Kenneth K. Kidd. Haplotype evolution and linkage disequilibrium: a simulation study. *Human heredity*, 51(1–2):85–96, 2001.

[16] C. Cannings and E. A. Thompson. Ascertainment in the sequential sampling of pedigrees. *Clinical Genetics*, 12(4):208–212, October 1977.

[17] Lon R. Cardon and John I. Bell. Association study designs for complex diseases. *Nature Reviews Genetics*, 2:91–99, February 2001.

[18] Christopher S. Carlson, Michael A. Eberle, Leonid Kruglyak, and Deborah A. Nickerson. Mapping complex disease loci in whole-genome association studies. *Nature*, 429:446–452, May 2004.

[19] Haiyan Chen and Fuji Zhang. The expected hitting times for finite Markov chains. *Linear Algebra and its Applications*, 428(11–12):2730–2749, June 2008.

[20] Charles J. Colbourn. *The Combinatorics of Network Reliability*. Oxford University Press, 1987.

[21] The Gene Ontology Consortium. Gene ontology: tool for the unification of biology. *Nature Genetics*, 25(1):25–29, May 2000.

[22] The UniProt Consortium. The universal protein resource (UniProt) in 2010. *Nucleic Acids Research*, 38:D142–D148, January 2010.

[23] Graham Coop and Robert C. Griffiths. Ancestral inference on gene trees under selection. *Theoretical Population Biology*, 66(3):219–232, November 2004.

[24] Heather J. Cordell and David G. Clayton. Genetic association studies. *Lancet*, 366(9491):1121–1131, September 2005.

[25] B. Devlin, K. Roeder, and L. Wasserman. Genomic control, a new approach to genetic-based association studies. *Theoretical Population Biology*, 60(3):155–166, November 2001.

[26] Reinhard Diestel. *Graph Theory*, volume 173 of *Graduate Texts in Mathematics*. Springer, Berlin, second edition, 2000.

[27] Peter G. Doyle and J. Laurie Snell. Random walks and electric networks, January 2000. `http://arxiv.org/abs/math.PR/0001057`.

[28] Scott M. Dudek, Alison A. Motsinger, Digna R. Velez, Scott M. Williams, and Marylyn D. Ritchie. Data simulation software for whole-genome association and other studies in human genetics. In Russ B. Altman, Tiffany Murray, Teri E. Klein, A. Keith Dunker, and Lawrence Hunter, editors, *Pacific Symposium on Biocomputing 2006*, volume 11, pages 499–510, Hawaii, USA, January 2006. World Scientific.

[29] R. J. Duffin. Topology of series-parallel networks. *Journal of Mathematical Analysis and Applications*, 10:303–318, 1965.

[30] Elisabet Einarsdottir, Inez Egerbladh, Lars Beckman, Dan Holmberg, and Stefan A. Escher. The genetic population structure of northern Sweden and its implications for mapping genetic diseases. *Hereditas*, 144(5):171–180, October 2007.

[31] Tina Eliassi-Rad, Lyle H. Ungar, Mark Craven, and Dimitrios Gunopulos, editors. *Proceedings of the Twelfth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Philadelphia, USA, 2006. ACM Press.

[32] T. Elperin, I. Gertsbakh, and M. Lomonosov. Estimation of network reliability using graph evolution models. *IEEE Transactions on Reliability*, 40(5):572–581, December 1991.

[33] D. Eppstein. Finding the k shortest paths. *SIAM Journal on Computing*, 28:652–673, 1998.

[34] Lauri Eronen, Floris Geerts, and Hannu Toivonen. A Markov chain approach to reconstruction of long haplotypes. In Altman et al. [3], pages 104–115.

[35] Lauri Eronen, Floris Geerts, and Hannu Toivonen. HaploRec: Efficient and accurate large-scale reconstruction of haplotypes. *BMC Bioinformatics*, 7:542, 2006.

[36] Lauri Eronen and Hannu Toivonen. Biomine: Predicting links between biological entities using network models of heterogeneous databases. Manuscript.

[37] Evangelos Evangelou, Thomas A. Trikalinos, Georgia Salanti, and John P. A. Ioannadis. Family-based versus unrelated case–control designs for genetic associations. *PLoS Genetics*, 2:e123, 2006.

[38] W. J. Ewens. *Mathematical Population Genetics*. Springer, Berlin, 1979.

[39] Christos Faloutsos, Kevin S. McCurley, and Andrew Tomkins. Fast discovery of connection subgraphs. In Won Kim, Ron Kohavi, Johannes Gehrke, and William DuMouchel, editors, *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 118–127, Seattle, USA, August 2004. ACM Press.

[40] Paul Fearnhead. Ancestral processes for non-neutral models of complex diseases. *Theoretical Population Biology*, 63(2):115–130, March 2003.

[41] Paul Fearnhead. Perfect simulation from nonneutral population genetic models: variable population size and subdivision. *Genetics*, 174(3):1397–1406, 2006.

[42] George S. Fishman. A comparison of four Monte Carlo methods for estimating the probability of s-t connectedness. *IEEE Transactions on Reliability*, R-35:145–155, 1986.

[43] Yun-Xin Fu. Exact coalescent for the Wright–Fisher model. *Theoretical Population Biology*, 69(4):385–394, June 2006.

[44] Stacey B. Gabriel, Stephen F. Schaffner, Huy Nguyen, Jamie M. Moore, Jessica Roy, Brendan Blumenstiel, John Higgins, Matthew DeFelice, Amy Lochner, Maura Faggart, Shau Neen Liu-Cordero, Charles Rotimi, Adebowale Adeyemo, Richard Cooper, Ryk Ward, Eric S. Lander, Mark J. Daly, and David Altshuler. The structure of haplotype blocks in the human genome. *Science*, 21(5576):2225–2229, June 2002.

[45] Michael R. Garey and David S. Johnson. *Computers and Intractability: a Guide to the Theory of NP-Completeness*. W. H. Freeman and Company, New York, 1979.

[46] Daniela S. Gerhard et al. The status, quality, and expansion of the NIH full-length cDNA project: The Mammalian Gene Collection (MGC). *Genome Research*, 14(10B):2121–2127, October 2004. Full list of authors is available at `http://dx.doi.org/10.1101/gr.2596504`.

[47] Lise Getoor and Christopher P. Diehl. Link mining: a survey. *SIGKDD Explorations newsletter*, 7(2):3–12, December 2005.

[48] Peddinti V. Gopalacharyulu, Erno Lindfors, Jarkko Miettinen, Catherine Bounsaythip, and Matej Orešič. An integrative approach for biological data mining and visualisation. *International Journal of Data Mining and Bioinformatics*, 2(1):54–77, 2008.

[49] Robert C. Griffiths and Paul Marjoram. Ancestral inference from samples of DNA sequences with recombination. *Journal of Computational Biology*, 3(4):479–502, 1996.

[50] Robert C. Griffiths and Paul Marjoram. An ancestral recombination graph. In Peter J. Donnelly and Simon Tavaré, editors, *Progress in Population Genetics and Human Evolution*, volume 87, pages 257–270. Springer, Berlin, 1997.

[51] Jochen Hampe, Thomas Wienker, Stefan Schreiber, and Peter Nürnberg. POPSIM: a general population simulator program. *Bioinformatics*, 14(5):458–464, 1998.

[52] Andrew T. Hattersley and Mark I. McCarthy. What makes a good association study? *Lancet*, 366(9493):1315–1323, October 2005.

[53] John Hershberger, Matthew Maxel, and Subhash Suri. Finding the k shortest simple paths: a new algorithm and its implementation. *ACM Transactions on Algorithms*, 3(4):45, November 2007.

[54] Petteri Hintsanen. The most reliable subgraph problem. In Joost N. Kok, Jacek Koronacki, Ramon Lopez de Mantaras, Stan Matwin, Dunja Mladenič, and Andrzej Skowron, editors, *Knowledge Discovery in Databases: PKDD 2007*, volume 4702 of *Lecture Notes in Artificial Intelligence*, pages 471–478, Warsaw, Poland, September 2007. Springer.

[55] Petteri Hintsanen, Petteri Sevon, Päivi Onkamo, Lauri Eronen, and Hannu Toivonen. An empirical comparison of case–control and trio-based study designs in high-throughput association mapping. *Journal of Medical Genetics*, 43(7):617–624, 2006.

[56] Petteri Hintsanen and Hannu Toivonen. Finding reliable subgraphs from large probabilistic graphs. *Data Mining and Knowledge Discovery*, 17(1):3–23, 2008.

[57] Petteri Hintsanen, Hannu Toivonen, and Petteri Sevon. Fast discovery of reliable subnetworks. In Nasrullah Memon and Reda Alhajj, editors, *International Conference on Advances in Social Networks Analysis and Mining, ASONAM 2010*, pages 104–111, Odense, Denmark, August 2010. IEEE Computer Society.

[58] Dimitar Hristovski, Borut Peterlin, Joyce A. Mitchell, and Susanne M. Humphrey. Using literature-based discovery to identify disease candidate genes. *Medical informatics*, 74(2–4):289–298, March 2005.

[59] Richard R. Hudson. Properties of a neutral allele model with intragenic recombination. *Theoretical Population Biology*, 23(2):183–201, April 1983.

[60] Richard R. Hudson. Generating samples under a Wright–Fisher neutral model of genetic variation. *Bioinformatics*, 18(2):337–338, 2002.

[61] Kin-Ping Hui, N. Bean, M. Kraetzl, and Dirk P. Kroese. The cross-entropy method for network reliability estimation. *Annals of Operations Research*, 134(1):101–118, 2005.

[62] Sarah Hunter, Rolf Apweiler, Teresa K. Attwood, Amos Bairoch, Alex Bateman, David Binns, Peer Bork, Ujjwal Das, Louise Daugherty, Lauranne Duquenne, Robert D. Finn, Julian Gough, Daniel Haft, Nicolas Hulo, Daniel Kahn, Elizabeth Kelly, Aurélie Laugraud, Ivica Letunic, David Lonsdale, Rodrigo Lopez, Martin Madera, John Maslen, Craig McAnulla, Jennifer McDowall, Jaina Mistry, Alex Mitchell, Nicola Mulder, Darren Natale, Christine Orengo, Antony F. Quinn, Jeremy D. Selengut, Christian J. A. Sigrist, Manjula Thimma, Paul D. Thomas, Franck Valentin, Derek

Wilson, Cathy H. Wu, and Corin Yeats. InterPro: the integrative protein signature database. *Nucleic Acids Research*, 37:D211–D215, January 2009.

[63] John P.A. Ioannidis, Thomas A. Trikalinos, Evangelia E. Ntzani, and Despina G. Contopoulos-Ioannidis. Genetic associations in large versus small studies: an empirical assessment. *Lancet*, 361(9357):567–571, February 2003.

[64] Glen Jeh and Jennifer Widom. SimRank: a measure of structural-context similarity. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 538–543, Edmonton, Canada, July 2002. ACM.

[65] Lars J. Jensen, Michael Kuhn, Manuel Stark, Samuel Chaffron, Chris Creevey, Jean Muller, Tobias Doerks, Philippe Julien, Alexander Roth, Milan Simonovic, Peer Bork, and Christian von Mering. STRING 8—a global view on proteins and their functional interactions in 630 organisms. *Nucleic Acids Research*, 37:D412–D416, January 2009.

[66] Minoru Kanehisa, Susumu Goto, Miho Furumichi, Mao Tanabe, and Mika Hirakawa. KEGG for representation and analysis of molecular networks involving diseases and drugs. *Nucleic Acids Research*, 38:D355–D360, January 2010.

[67] David R. Karger. A randomized fully polynomial time approximation scheme for the all-terminal network reliability problem. *SIAM Review*, 43(3):499–522, 2001.

[68] Richard M. Karp, Michael Luby, and Neal Madras. Monte-Carlo approximation algorithms for enumeration problems. *Journal of Algorithms*, 10(3):429–449, September 1989.

[69] Melissa Kasari, Hannu Toivonen, and Petteri Hintsanen. Fast discovery of reliable k-terminal subgraphs. In Mohammed J. Zaki, Jeffrey Xu Yu, B. Ravindran, and Vikram Pudi, editors, *Advances in Knowledge Discovery and Data Mining*, volume 6119 of *Lecture Notes in Artificial Intelligence*, pages 168–177. Springer, June 2010.

[70] Leo Katz. A new status index derived from sociometric analysis. *Psychometrika*, 18(1):39–43, March 1953.

[71] Juha Kere. Human population genetics: lessons from Finland. *Annual Review of Genomics and Human Genetics*, 2:103–128, September 2001.

[72] Motoo Kimura. The number of heterozygous nucleotide sites maintained in a finite population due to a steady flux of mutations. *Genetics*, 61(4):893–903, April 1969.

[73] J. F. C. Kingman. The coalescent. *Stochastic Processes and their Applications*, 13(3):235–248, September 1982.

[74] Sebastian Köhler, Sebastian Bauer, Denise Horn, and Peter N. Robinson. Walking the interactome for priorization of candidate disease genes. *American Journal of Human Genetics*, 82(4):949–958, April 2008.

[75] Yehuda Koren, Stephen C. North, and Chris Volinsky. Measuring and extracting proximity graphs in networks. In Eliassi-Rad et al. [31], pages 245–255.

[76] Dirk P. Kroese, Kin-Ping Hui, and Sho Nariai. Network reliability optimization via the cross-entropy method. *IEEE Transactions on Reliability*, 56(2):275–287, June 2007.

[77] Stephen M. Krone and Claudia Neuhauser. Ancestral processes with selection. *Theoretical Population Biology*, 51(3):210–237, June 1997.

[78] Zoé Lacroix, Louiqa Raschid, and Maria-Esther Vidal. Efficient techniques to explore and rank paths in life science data sources. In Erhard Rahm, editor, *Data Integration in the Life Sciences, First International Workshop, DILS 2004*, pages 187–202, Leipzig, Germany, 2004. Springer.

[79] Tarja Laitinen, Mark J. Daly, John D. Rioux, Paula Kauppi, Catherine Laprise, Tuula Petäys, Todd Green, Michele Cargill, Tari Haahtela, Eric S. Lander, Lauri A. Laitinen, Thomas J. Hudson, and Juha Kere. A susceptibility locus for asthma-related traits on chromosome 7 revealed by genome-wide scan in a founder population. *Nature Genetics*, 28:87–91, 2001.

[80] Tarja Laitinen, Anne Polvi, Pia Rydman, Johanna Vendelin, Ville Pulkkinen, Paula Salmikangas, Siru Mäkelä, Marko Rehn, Asta Pirskanen, Anna Rautanen, Marco Zucchelli, Harriet Gullstén, Marina Leino, Harri Alenius, Tuula Petäys, Tari Haahtela, Annika Laitinen, Catherine Laprise, Thomas J. Hudson, Lauri A. Laitinen, and Juha Kere. Characterization of a common susceptibility locus for asthma-related traits. *Science*, 304(5668):300–304, April 2004.

[81] Laura Langohr. Link analysis in heterogeneous biological networks. Diplomarbeit, Friedrich-Schiller-Universität Jena, Jena, Germany, May 2008.

[82] Guillaume Laval and Laurent Excoffier. SIMCOAL 2.0: a program to simulate genomic diversity over large recombining regions in a subdivided population with a complex history. *Bioinformatics*, 20(15):2485–2487, April 2004.

[83] Eugene L. Lawler. A procedure for computing the k best solutions to discrete optimization problems and its application to the shortest path problem. *Management Science*, 18(7):401–405, March 1972.

[84] Samuel Levy, Granger Sutton, Pauline C. Ng, Lars Feuk, Aaron L. Halpern, Brian P. Walenz, Nelson Axelrod, Jiaqi Huang, Ewen F. Kirkness, Gennady Denisov, Yuan Lin, Jeffrey R. MacDonald, Andy Wing Chun Pang, Mary Shago, Timothy B. Stockwell, Alexia Tsiamouri, Vineet Bafna, Vikas Bansal, Saul A. Kravitz, Dana A. Busam, Karen Y. Beeson, Tina C. McIntosh, Karin A. Remington, Josep F. Abril, John Gill, Jon Borman, Yu-Hui Rogers, Marvin E. Frazier, Stephen W. Scherer, Robert L. Strausberg, and J. Craig Venter. The diploid genome sequence of an individual human. *PLoS Biology*, 5(10):e254, October 2007.

[85] David Liben-Nowell and Jon Kleinberg. The link prediction problem for social networks. *Journal of the American Society for Information Science and Technology*, 58(7):1019–1031, May 2007.

[86] Shin Lin, Aravinda Chakravarti, and David J. Cutler. Exhaustive allelic transmission disequilibrium tests as a new approach to genome-wide association studies. *Nature Genetics*, 36:1181–1188, 2004.

[87] Donna Maglott, Jim Ostell, Kim D. Pruitt, and Tatiana Tatusova. Entrez Gene: gene-centered information at NCBI. *Nucleic Acids Research*, 35:D26–D31, January 2007.

[88] Johanna Malinen. A method for mapping multiple interacting genes. Master's thesis, University of Helsinki, Helsinki, Finland, 2008.

[89] Paul Marjoram and Jeff D. Wall. Fast "coalescent" simulation. *BMC Genetics*, 7(1):16, 2006.

[90] Mary Sara McPeek and Andrew Strahs. Assessment of linkage disequilibrium by the decay of haplotype sharing, with application to fine-scale genetic mapping. *American Journal of Human Genetics*, 65(3):858–875, September 1999.

[91] Gilean A. T. McVean and Niall J. Cardin. Approximating the coalescent with recombination. *Philosophical Transactions of the Royal Society B*, 360(1459):1387–1393, July 2005.

[92] M. Möhle. Weak convergence to the coalescent in neutral population models. *Journal of Applied Probability*, 36(2):446–460, June 1999.

[93] P. Mork, R. Shaker, A. Halevy, and P. Tarczy-Hornoch. PQL: a declarative query language over dynamic biological schemata. In *Proceedings of the American Medical Informatics Association Annual Symposium 2002*, pages 533–537, 2002.

[94] A. Morris, J. Whittaker, and D. Balding. Little loss of information due to unknown phase for fine-scale linkage-disequilibrium mapping with single-nucleotide-polymorphism genotype data. *American Journal of Human Genetics*, 74(5):945–953, May 2004.

[95] Richard W. Morris and Norman L. Kaplan. On the advantage of haplotype analysis in the presence of multiple disease susceptibility alleles. *Genetic Epidemiology*, 23(3):221–233, October 2002.

[96] C. Neuhauser. *Mathematical models in population genetics*, chapter 19, pages 577–601. Volume 2 of Balding et al. [5], second edition, 2003.

[97] Claudia Neuhauser and Stephen M. Krone. The genealogy of samples in models with selection. *Genetics*, 145(2):519–534, February 1997.

[98] Mark E. J. Newman. Clustering and preferential attachment in growing networks. *Physical Review E*, 64(2):025102, July 2001.

[99] M. Nordborg. *Coalescent theory*, chapter 20, pages 602–635. Volume 2 of Balding et al. [5], second edition, 2003.

[100] McKusick-Nathans Institute of Genetic Medicine, Johns Hopkins University and National Center for Biotechnology Information, National Library of Medicine. Online Mendelian Inheritance in Man, OMIM™. World Wide Web. `http://www.ncbi.nlm.nih.gov/omim/`.

[101] Vesa Ollikainen. *Simulation techniques for disease gene localization in isolated populations*. PhD thesis, University of Helsinki, Department of Computer Science, Helsinki, Finland, August 2002.

[102] Bo Peng, Christopher I. Amos, and Marek Kimmel. Forward-time simulations of human populations with complex diseases. *PLoS Genetics*, 3:e47, 2007.

[103] Bo Peng and Marek Kimmel. simuPOP: a forward-time population genetics simulator. *Bioinformatics*, 21(18):3686–3687, 2005.

[104] Carolina Perez-Iratxeta, Matthia Wjst, Peer Bork, and Miguel A. Andrade. G2D: a tool for mining genes associated with disease. *BMC Genetics*, 6:45, 2005.

[105] Shaun Purcell, Pak Sham, and Mark J. Daly. Parental phenotypes in family-based association analysis. *American Journal of Human Genetics*, 76(2):249–259, February 2005.

[106] L. De Raedt, K. Kersting, A. Kimmig, K. Revoredo, and H. Toivonen. Compressing probabilistic Prolog programs. *Machine Learning*, 70(2–3):151–168, 2008.

[107] Proton Rahman, Albert Jones, Joseph Curtis, Sylvia Bartlett, Lynette Peddle, Bridget A. Fernandez, and Nelson B. Freimer. The Newfoundland population: a unique resource for genetic investigation of complex diseases. *Human Molecular Genetics*, 12:R167–R172, 2003.

[108] Pasi Rastas, Jussi Kollin, and Mikko Koivisto. Fast Bayesian haplotype inference via context tree weighting. In Keith A. Crandall and Jens Lagergren, editors, *Algorithms in Bioinformatics, 8th International Workshop, WABI 2008*, number 5251 in Lecture Notes in Computer Science, pages 259–270, Karlsruhe, Germany, September 2008. Springer.

[109] Neil J. Risch and Kathleen Merikangas. The future of genetic studies of complex human diseases. *Science*, 273(5281):1516–1517, September 1996.

[110] Gabriel Robins and Alexander Zelikovsky. Improved Steiner tree approximation in graphs. In *Proceedings of the Eleventh Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 770–779, San Francisco, USA, January 2000. ACM/SIAM.

[111] Liam Roditty. On the k shortest simple paths problem in weighted directed graphs. *SIAM Journal on Computing*, 39(6):2363–2376, 2010.

[112] Eric W. Sayers, Tanya Barrett, Dennis A. Benson, Evan Bolton, Stephen H. Bryant, Kathi Canese, Vyacheslav Chetvernin, Deanna M. Church, Michael DiCuccio, Scott Federhen, Michael Feolo, Lewis Y. Geer, Wolfgang Helmberg, Yuri Kapustin, David Landsman, David J. Lipman, Zhiyong Lu, Thomas L. Madden, Tom Madej, Donna R. Maglott, Aron Marchler-Bauer, Vadim Miller, Ilene Mizrachi, James Ostell, Anna Panchenko, Kim D. Pruitt, Gregory D. Schuler, Edwin Sequeira, Stephen T. Sherry, Martin Shumway, Karl Sirotkin, Douglas Slotta, Alexandre Souvorov, Grigory Starchenko, Tatiana A. Tatusova, Lukas Wagner, Yanli

Wang, W. John Wilbur, Eugene Yaschenko1, and Jian Ye. Database resources of the national center for biotechnology information. *Nucleic Acids Research*, 38:D5–D16, January 2010.

[113] Paul Scheet and Matthew Stephens. A fast and flexible statistical model for large-scale population genotype data: Applications to inferring missing genotypes and haplotypic phase. *American Journal of Human Genetics*, 78(4):629–644, April 2006.

[114] Petteri Sevon. *Algorithms for association-based gene mapping*. PhD thesis, University of Helsinki, Department of Computer Science, Helsinki, Finland, June 2004.

[115] Petteri Sevon and Lauri Eronen. Subgraph queries by context-free grammars. *Journal of Integrative Bioinformatics*, 5(2):100, 2008.

[116] Petteri Sevon, Lauri Eronen, Petteri Hintsanen, Kimmo Kulovesi, and Hannu Toivonen. Link discovery in graphs derived from biological databases. In Ulf Leser, Felix Naumann, and Barbara A. Eckman, editors, *Data Integration in the Life Sciences, Third International Workshop, DILS 2006*, number 4075 in Lecture Notes in Computer Science, pages 35–49, Hinxton, UK, July 2006. Springer.

[117] Petteri Sevon, Hannu Toivonen, and Vesa Ollikainen. TreeDT: Tree pattern mining for gene mapping. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 3(2):174–185, 2006.

[118] Sagiv Shifman and Ariel Darvasi. The value of isolated populations. *Nature Genetics*, 28:309–310, 2001.

[119] Robert Sladek, Ghislain Rocheleau, Johan Rung, Christian Dina, Lishuang Shen, David Serre, Philippe Boutin, Daniel Vincent, Alexandre Belisle, Samy Hadjadj, Beverley Balkau, Barbara Heude, Guillaume Charpentier, Thomas J. Hudson, Alexandre Montpetit, Alexey V. Pshezhetsky, Marc Prentki, Barry I. Posner, David J. Balding, David Meyre, Constantin Polychronakos, and Philippe Froguel. A genome-wide association study identifies novel risk loci for type 2 diabetes. *Nature*, 445:881–885, 2007.

[120] M. Stephens. *Inference under the coalescent*, chapter 21, pages 636–661. Volume 2 of Balding et al. [5], second edition, 2003.

[121] Fumio Tajima. Evolutionary relationship of DNA sequences in finite populations. *Genetics*, 105(2):437–460, October 1983.

[122] Simon Tavaré. Line-of-descent and genealogical processes, and their applications in population genetic models. *Theoretical Population Biology*, 26(2):119–164, October 1984.

[123] A. Tenesa, A. F. Wright, S. A. Knott, A. D. Carothers, C. Hayward, A. Angius, I. Persico, G. Maestrale, N. D. Hastie, M. Pirastu, and P. M. Visscher. Extent of linkage disequilibrium in a Sardinian sub-isolate: sampling and methodological considerations. *Human Molecular Genetics*, 13(1):25–33, 2004.

[124] Hannu Toivonen, Päivi Onkamo, Kari Vasko, Vesa Ollikainen, Petteri Sevon, Heikki Mannila, Mathias Herr, and Juha Kere. Data mining applied to linkage disequilibrium mapping. *American Journal of Human Genetics*, 67(1):133–145, July 2000.

[125] Hannu Toivonen, Päivi Onkamo, Petteri Hintsanen, Evimaria Terzi, and Petteri Sevon. Data mining for gene mapping. In Mehmed N. Katardzic and Jozef Zurada, editors, *Next Generation of Data-Mining Applications*, chapter 12, pages 263–293. John Wiley & Sons, Hoboken, 2005.

[126] Hanghang Tong and Christos Faloutsos. Center-piece subgraphs: problem definition and fast solutions. In Eliassi-Rad et al. [31].

[127] Frances S. Turner, Daniel R. Clutterbuck, and Colin A. M. Semple. POCUS: Mining genomic sequence annotation to predict disease genes. *Genome Biology*, 4(11):R75, 2003.

[128] Esko Ukkonen. Finding founder sequences from a set of recombinants. In Roderic Guigó and Dan Gusfield, editors, *Algorithms in Bioinformatics, Second International Workshop, WABI 2002*, number 2452 in Lecture Notes in Computer Science, pages 277–286, Rome, Italy, September 2002. Springer.

[129] Jacobo Valdes, Robert Endre Tarjan, and Eugene L. Lawler. The recognition of series-parallel digraphs. *SIAM Journal on Computing*, 11(2):298–313, May 1982.

[130] Leslie G. Valiant. The complexity of computing the permanent. *Theoretical Computer Science*, 8(2):189–201, 1979.

[131] Leslie G. Valiant. The complexity of enumeration and reliability problems. *SIAM Journal on Computing*, 8(3):410–421, August 1979.

[132] John Wakeley. The limits of theoretical population genetics. *Genetics*, 169(1):1–7, October 2005.

[133] Jeffrey D. Wall and Jonathan K. Pritchard. Haplotype blocks and linkage disequilibrium in the human genome. *Nature Reviews Genetics*, 4:587–597, 2003.

[134] Ying Wang and Bruce Rannala. Simulating a coalescent process with recombination and ascertainment. In *Lecture Notes in Bioinfomatics*, volume 2983, pages 84–95. Springer, Berlin, 2004.

[135] Ying Wang and Bruce Rannala. In silico analysis of disease-association mapping strategies using the coalescent process and incorporating ascertainment and selection. *American Journal of Human Genetics*, 76(6):1066–1073, June 2005.

[136] Zuoheng Wang and Mary Sara McPeek. An incomplete-data quasi-likelihood approach to haplotype-based genetic association studies on related individuals. *Journal of the American Statistical Association*, 104:1251–1260, 487 2009.

[137] D. E. Weeks, J. Ott, and G. M. Lathrop. SLINK: a general simulation program for linkage analysis. *American Journal of Human Genetics*, 47:A204, 1990.

[138] Kenneth M. Weiss and Joseph D. Terwilliger. How many diseases does it take to map a gene with SNPs? *Nature Genetics*, 26:151–157, 2000.

[139] Carsten Wiuf and Jotun Hein. Recombination as a point process along sequences. *Theoretical Population Biology*, 55(3):248–259, June 1999.

[140] Alan F. Wright, Andrew D. Carothers, and Mario Pirastu. Population choice in mapping genes for complex diseases. *Nature Genetics*, 23:397–404, 1999.

[141] Fred A. Wright, Hanwen Huang, Xiaojun Guan, Kevin Gamiel, Clark Jeffries, William T. Barry, Fernando Pardo-Manuel de Villena, Patrick F. Sullivan, Kirk C. Wilhelmsen, and Fei Zou. Simulating association studies: a data-based resampling method for candidate regions or whole genome scans. *Bioinformatics*, 23(19):2581–2588, 2007.

[142] Hongyu Zhao, Ruth Pfeiffer, and Mitchell H. Gail. Haplotype analysis in population genetics and association studies. *Pharmacogenomics*, 4(2):171–178, March 2003.

TIETOJENKÄSITTELYTIETEEN LAITOS
PL 68 (Gustaf Hällströmin katu 2 b)
00014 Helsingin yliopisto

DEPARTMENT OF COMPUTER SCIENCE
P.O. Box 68 (Gustaf Hällströmin katu 2 b)
FIN-00014 University of Helsinki, FINLAND

JULKAISUSARJA **A**

SERIES OF PUBLICATIONS **A**

Reports may be ordered from: Kumpula Science Library, P.O. Box 64, FIN-00014 University of Helsinki, FINLAND.

A-2004-1 M. Koivisto: Sum-product algorithms for the analysis of genetic risks. 155 pp. (Ph.D. Thesis)

A-2004-2 A. Gurtov: Efficient data transport in wireless overlay networks. 141 pp. (Ph.D. Thesis)

A-2004-3 K. Vasko: Computational methods and models for paleoecology. 176 pp. (Ph.D. Thesis)

A-2004-4 P. Sevon: Algorithms for Association-Based Gene Mapping. 101 pp. (Ph.D. Thesis)

A-2004-5 J. Viljamaa: Applying Formal Concept Analysis to Extract Framework Reuse Interface Specifications from Source Code. 206 pp. (Ph.D. Thesis)

A-2004-6 J. Ravantti: Computational Methods for Reconstructing Macromolecular Complexes from Cryo-Electron Microscopy Images. 100 pp. (Ph.D. Thesis)

A-2004-7 M. Kääriäinen: Learning Small Trees and Graphs that Generalize. 45+49 pp. (Ph.D. Thesis)

A-2004-8 T. Kivioja: Computational Tools for a Novel Transcriptional Profiling Method. 98 pp. (Ph.D. Thesis)

A-2004-9 H. Tamm: On Minimality and Size Reduction of One-Tape and Multitape Finite Automata. 80 pp. (Ph.D. Thesis)

A-2005-1 T. Mielikäinen: Summarization Techniques for Pattern Collections in Data Mining. 201 pp. (Ph.D. Thesis)

A-2005-2 A. Doucet: Advanced Document Description, a Sequential Approach. 161 pp. (Ph.D. Thesis)

A-2006-1 A. Viljamaa: Specifying Reuse Interfaces for Task-Oriented Framework Specialization. 285 pp. (Ph.D. Thesis)

A-2006-2 S. Tarkoma: Efficient Content-based Routing, Mobility-aware Topologies, and Temporal Subspace Matching. 198 pp. (Ph.D. Thesis)

A-2006-3 M. Lehtonen: Indexing Heterogeneous XML for Full-Text Search. 185+3 pp. (Ph.D. Thesis)

A-2006-4 A. Rantanen: Algorithms for ^{13}C Metabolic Flux Analysis. 92+73 pp. (Ph.D. Thesis)

A-2006-5 E. Terzi: Problems and Algorithms for Sequence Segmentations. 141 pp. (Ph.D. Thesis)

A-2007-1 P. Sarolahti: TCP Performance in Heterogeneous Wireless Networks. (Ph.D. Thesis)

A-2007-2 M. Raento: Exploring privacy for ubiquitous computing: Tools, methods and experiments. (Ph.D. Thesis)

A-2007-3 L. Aunimo: Methods for Answer Extraction in Textual Question Answering. 127+18 pp. (Ph.D. Thesis)

A-2007-4 T. Roos: Statistical and Information-Theoretic Methods for Data Analysis. 82+75 pp. (Ph.D. Thesis)

A-2007-5 S. Leggio: A Decentralized Session Management Framework for Heterogeneous Ad-Hoc and Fixed Networks. 230 pp. (Ph.D. Thesis)

A-2007-6 O. Riva: Middleware for Mobile Sensing Applications in Urban Environments. 195 pp. (Ph.D. Thesis)

A-2007-7 K. Palin: Computational Methods for Locating and Analyzing Conserved Gene Regulatory DNA Elements. 130 pp. (Ph.D. Thesis)

A-2008-1 I. Autio: Modeling Efficient Classification as a Process of Confidence Assessment and Delegation. 212 pp. (Ph.D. Thesis)

A-2008-2 J. Kangasharju: XML Messaging for Mobile Devices. 24+255 pp. (Ph.D. Thesis).

A-2008-3 N. Haiminen: Mining Sequential Data – in Search of Segmental Structures. 60+78 pp. (Ph.D. Thesis)

A-2008-4 J. Korhonen: IP Mobility in Wireless Operator Networks. (Ph.D. Thesis)

A-2008-5 J.T. Lindgren: Learning nonlinear visual processing from natural images. 100+64 pp. (Ph.D. Thesis)

A-2009-1 K. Hätönen: Data mining for telecommunications network log analysis. 153 pp. (Ph.D. Thesis)

A-2009-2 T. Silander: The Most Probable Bayesian Network and Beyond. (Ph.D. Thesis)

A-2009-3 K. Laasonen: Mining Cell Transition Data. 148 pp. (Ph.D. Thesis)

A-2009-4 P. Miettinen: Matrix Decomposition Methods for Data Mining: Computational Complexity and Algorithms. 164+6 pp. (Ph.D. Thesis)

A-2009-5 J. Suomela: Optimisation Problems in Wireless Sensor Networks: Local Algorithms and Local Graphs. 106+96 pp. (Ph.D. Thesis)

A-2009-6 U. Köster: A Probabilistic Approach to the Primary Visual Cortex. 168 pp. (Ph.D. Thesis)

A-2009-7 P. Nurmi: Identifying Meaningful Places. 83 pp. (Ph.D. Thesis)

A-2009-8 J. Makkonen: Semantic Classes in Topic Detection and Tracking. 155 pp. (Ph.D. Thesis)

A-2009-9 P. Rastas: Computational Techniques for Haplotype Inference and for Local Alignment Significance. 64+50 pp. (Ph.D. Thesis)

A-2009-10 T. Mononen: Computing the Stochastic Complexity of Simple Probabilistic Graphical Models. 60+46 pp. (Ph.D. Thesis)

A-2009-11 P. Kontkanen: Computationally Effcient Methods for MDL-Optimal Density Estimation and Data Clustering. 75+64 pp. (Ph.D. Thesis)

A-2010-1 M. Lukk: Construction of a global map of human gene expression - the process, tools and analysis. 120 pp. (Ph.D. Thesis)

A-2010-2 W. Hämäläinen: Efficient search for statistically significant dependency rules in binary data. 163 pp. (Ph.D. Thesis)

A-2010-3 J. Kollin: Computational Methods for Detecting Large-Scale Chromosome Rearrangements in SNP Data. 197 pp. (Ph.D. Thesis)

A-2010-4 E. Pitkänen: Computational Methods for Reconstruction and Analysis of Genome-Scale Metabolic Networks. 115+88 pp. (Ph.D. Thesis)

A-2010-5 A. Lukyanenko: Multi-User Resource-Sharing Problem for the Internet. 168 pp. (Ph.D. Thesis)

A-2010-6 L. Daniel: Cross-layer Assisted TCP Algorithms for Vertical Handoff. 84+72 pp. (Ph.D. Thesis)

A-2011-1 A. Tripathi: Data Fusion and Matching by Maximizing Statistical Dependencies. 89+109 pp. (Ph.D. Thesis)

A-2011-2 E. Junttila: Patterns in permuted binary matrices. 155 pp. (Ph.D. Thesis)