IT Licentiate theses
2007-005

Parallel Algorithms and Implementations for Genetic Analysis of Quantitative Traits

Mahen Jayawardena

UPPSALA UNIVERSITY
Department of Information Technology

UPPSALA
UNIVERSITET

Parallel Algorithms and Implementations for Genetic Analysis of Quantitative Traits

BY

MAHEN JAYAWARDENA

September 2007

DIVISION OF SCIENTIFIC COMPUTING
DEPARTMENT OF INFORMATION TECHNOLOGY
UPPSALA UNIVERSITY
UPPSALA
SWEDEN

Dissertation for the degree of Licentiate of Philosophy in Scientific Computing
at Uppsala University 2007

Parallel Algorithms and Implementations for Genetic Analysis of Quantitative Traits

Mahen Jayawardena

Mahen.Jayawardena@it.uu.se

Division of Scientific Computing
Department of Information Technology
Uppsala University
Box 337
SE-751 05 Uppsala
Sweden

http://www.it.uu.se/

ISSN 1404-5117
Printed by the Department of Information Technology, Uppsala University, Sweden

Abstract

Many important traits in plants, animals and humans are quantitative, and most such traits are generally believed to be regulated by multiple genetic loci. Standard computational tools for analysis of quantitative traits use linear regression models for relating the observed phenotypes to the genetic composition of individuals in a population. However, using these tools to simultaneously search for multiple genetic loci is very computationally demanding. The main reason for this is the complex nature of the optimization landscape for the multidimensional global optimization problems that must be solved. This thesis describes parallel algorithms and implementation techniques for such optimization problems. The new computational tools will eventually enable genetic analysis exploiting new classes of multidimensional statistical models, potentially resulting in interesting results in genetics.

We first describe how the algorithm used for global optimization in the standard, serial software is parallelized and implemented on a grid system. Then, we also describe a parallelized version of the more elaborate global optimization algorithm DIRECT and show how this can be deployed on grid systems and other loosely-coupled architectures. The parallel DIRECT scheme is further developed to exploit both coarse-grained parallelism in grid systems or clusters as well as fine-grained, tightly-coupled parallelism in multi-core nodes. The results show that excellent speedup and performance can be archived on grid systems and clusters, even when using a tightly-coupled algorithms such as DIRECT. Finally, a pilot implementation of a grid portal providing a graphical front-end for our code is implemented. After some further development, this portal can be utilized by geneticists for performing multidimensional genetic analysis of quantitative traits on a regular basis.

List of Papers

This thesis is a summary of the following papers. They will be referred to as Paper A, B, C and D.

A M. Jayawardena, S. Holmgren and K. Ljungberg. Using Parallel Computing and Grid Systems for Genetic Mapping of Quantitative Traits. In *Proceedings of PARA 2006, Umeå, Sweden, June 18-21, 2006*, Springer Lecture Notes in Computer Science (In press).

B M. Jayawardena and S. Holmgren. Grid-Enabling an Efficient Algorithm for Demanding Global Optimization Problems in Genetic Analysis. Submitted to *3rd IEEE International Conference on e-Science and Grid Computing, Bangalore, India, December 10-13, 2007*.

C M. Jayawardena, H. Löf and S. Holmgren. Grids and Clusters with Multi-Core Nodes: A Genetics Application Perspective . To be submitted to *22:nd IEEE International Parallel and Distributed Processing Symposium, Miami, USA, April 14-18, 2008.*

D S. Toor, M. Jayawardena, J. Lindemann and S. Holmgren. A Grid Portal Implementation for Genetic Mapping of Multiple QTL. Manuscript, Department of Information Technology, Uppsala University, 2007.

Acknowledgements

I am grateful to my supervisor Docent Sverker Holmgren for believing in me always. He has shown me what research is and helped me overcome many of the obstacles that I encountered. I also thank Dr. Ruvan Weerasinghe my Sri Lankan supervisor for all his valuable advice and help in making this work possible. Docent Örjan Carlborg from LCB who was very helpful in all the genetics in this project and the effort he has made to make this thesis a success. I also thank Salman Toor, Dr. Kajsa Ljungberg and Dr. Henrik Löf for all the fruitful discussions and contributions they have made in the thesis.

The Swedish International Development and Cooperation Agency (SIDA) has been very generous in funding my studies here and my thanks go to all the people involved with the Split PhD program both in Sweden and Sri Lanka. The late Vidya Jyothi Professor V.K.Samaranayake who envisioned this program needs to be mentioned specially . Also I wish to mention the late Mr. Samanthilaka who helped at the initial stages of the project. I offer my sincere gratitude to Dr. Kasun De Zoysa at UCSC and Docent Richard Wait at Uppsala University who have coordinated this program with a lot of effort.

Also the staff at UCSC including Dr. Nalin Ranasinghe who has collaborated in several projects and also Ajitha and Renuka for all their help in the administration of the PhD program. Marianne Ahrne, Ulrika Andersson and Carina Lindgren at the department who have helped us immensely need mentioning. UPPMAX headed by Docent Ingela Nyström and the support staff including Dr. Jukka Komminaho, Dr. Tore Sundqvist and Dr. Mattias Ellert who have provided all the computational infrastructure for the work done in this thesis and all the technical support they have provided is greatly appreciated.

I also thank all my fellow PhD students both at the department and also the other Split PhD students in Sweden for all their company and friendship. A special thanks to the entire Sri Lankan community in Uppsala for their kindness and the warm meals to make us feel at home.

Most of all my gratitude goes to my family. My loving wife Sadhani for keeping me company during the highs and specially the lows and to my parents and sister for all their loving support, without which this would never be possible.

Contents

1 Introduction

Some disorders, e.g. cystic fibrosis and Huntington's disease in humans, are caused by a single gene. Historically, genetics has been quite successful in locating the genes responsible for such monogenic traits. However, most economically and medically important traits in plants, animals and humans are determined by a combination of an unknown number of genes (normally several) and environmental factors. Examples are susceptibility to breast cancer, diabetes, heart disease and malaria in humans, lean meat production in farm animals, and crop yield in corn and rice. Normally, these multifactorial traits are quantitative, i.e. they can be measured on a continues scale. Locating the regions in the genome affecting such quantitative traits is a great challenge, and even though much progress has been made during the last decade, the development of more powerful analysis tools are needed.

Which regions in the genome that affect a multifactorial trait can be examined by setting up a statistical model of the relation between the observed trait and the genetic composition of individuals in experimental populations. The regions are called QTL (Quantitative Trait Loci) and the procedure of finding them is called QTL mapping. Here, the genetic composition of all individuals in the experimental population is determined at a set of marker locations in the genome. Together with the observed values of the trait, this data is input to a QTL mapping computer code where the computation of the model fit and the search for the most probable positions of the QTL in the genome are implemented using numerical algorithms. Once the most probable set of QTL positions are determined, further computations are needed to establish the statistical significance of the result. Reviews of QTL mapping methods are given in e.g. [53] and [26].

Because of the complexity of the computations in the mapping procedure, standard software can normally only perform mapping of a single QTL at a time. This thesis describes new computational algorithms and their implementations on high-performance computers for QTL mapping using models containing several, possibly interacting loci. Using such a model makes the mapping procedure very computationally demanding. Finding the most likely position of d QTL influencing a trait results in that a d-dimensional global optimization problem must be solved. To determine the statistical significance of the result, 1000-10000 such problems must possibly be solved. The evaluation of the objective function is performed by computing the statistical model fit for a given set of d QTL positions in the genome, and if an elaborate model is used each model fit computation can be quite time-consuming.

So far, standard QTL mapping software [11, 15, 47, 60] have used a robust but computationally expensive algorithm for solving the global optimization problem. Furthermore, the computational requirement of this algorithm grows exponentially with d. Models with multiple QTL are still fitted on a regular basis using a technique where a sequence of one-dimensional mapping problems are solved. However, it is not clear how accurate this technique is for general QTL models and the interest in simultaneous mapping of multiple QTL has increased lately. Partly, the interest is motivated by analysis of real data sets [19, 61, 64] where certain interactions [18] between pairs of QTL have been found to only be detectable by solving the full two-dimensional optimization problem.

1. In paper A, we show that it is possible to perform at least a few high-dimensional QTL mapping computations using the standard (robust but demanding) global optimization algorithm. The computations are performed using different types of parallel computers, including grid systems. The parallel code described in paper A also provides a basis for the implementation of the more efficient optimization schemes presented in papers B and C in a variety of high performance computing environments.

2. In papers B and C, we describe parallel versions and implementations of the DIRECT scheme for global optimization, and we use the parallel codes to perform representative QTL analysis computations using models with up to 5 QTL. Several types of parallelism are exploited, and flexible implementations on shared memory systems, clusters and grid systems are described.

   Paper B describes how the DIRECT algorithm is parallelized by partitioning the search space, arriving at an implementation for grid and other loosely-coupled systems. In paper C, multithreading is added at an inner level for improved performance on systems with multicore nodes.

3. In paper D, a preliminary version of a grid portal is implemented. This portal utilizes the parallelized code described above and provides a pilot implementation of some of the functionality needed by biologists to be able to use the software on a regular basis.

The remainder of this summary is organized in the following way: Section 2 discusses genetics and the representative models that are necessary to understand the problems that we deal with in this thesis. Sections 3 and 4 deal with the computational aspects of the problem and how it relates to the QTL problem. High performance computing and its use in QTL analysis is the topic of Sections 5. The data sets that we have used in our experiments are briefly discussed in Section 6, and Section 7 gives a brief summary of the papers A,B,C and D which complete this thesis. Some future work envisaged is highlighted in Section 8.

2 Genetics, Multifactorial Traits and QTL

In this section, we start by giving a brief introduction to basic genetics. Then we introduce the concepts of multifactorial and quantitative traits and quantitative trait loci (QTL), which are central to the work in this thesis. Finally, a brief review of the QTL models and the QTL mapping procedures used in papers A-D is given.

2.1 Basic Genetics

The genetic information in organisms is encoded in long strands of DNA (deoxyribonucleic acid) called *chromosomes*. For example, humans have 2×23 chromosomes, pigs have 2×19 and maize have 2×10. The data is built up of four different molecular building blocks, commonly called *genetic bases*, denoted by A,C,G and T. Using this notation, the genetic structure of a chromosome or a part of a chromosome can be described. The DNA has a double-helix structure, where A only bonds with T and G only with C. This base-pairing effect is exploited in the replication of DNA.

The function of a major part of the DNA is still unknown. Some portions of the DNA contain code for RNA (ribonucleic acid) production, and these DNA segments are referred to as *genes*. A specific genetic position in the chromosome is called a *locus (pl. loci)*. Every gene has its fixed locus, but it may appear in different versions, *alleles*. The polymorphism caused by the different alleles is the basis for the study of genetics.

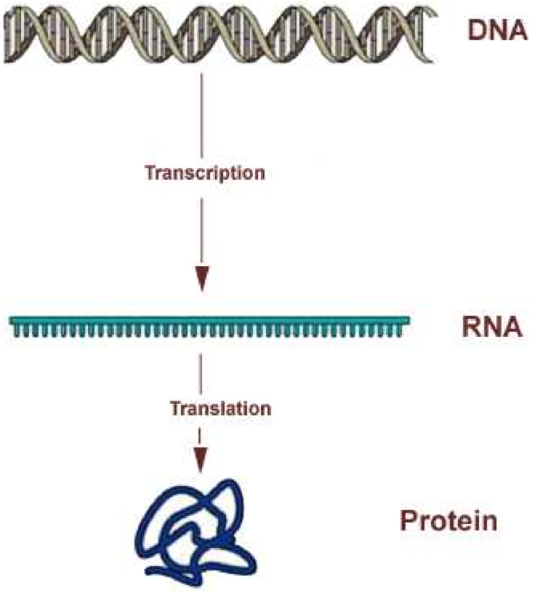


Figure 1: The central dogma of genetics

When a gene is activated, this often leads to the production of a protein. The most important steps in this process are described in Figure 1. Proteins are the basic chemical compounds that make up the structure of cells and direct their activities. Normally, organisms are capable of producing thousands of different proteins which have different functions (there is some overlapping of the tasks that they perform).

Phenotypes and Genotypes

A *phenotype* is a measurable trait of an individual (e.g. blood group, hair color and body weight for humans). A phenotype depends on the individuals *genotype*, i.e. the genetic information in the DNA, and often also the environment. The fraction of the phenotypic variation that is determined by the genetic information is called the *heritability* of the trait. For example, blood group is only genetically determined, while the heritability of lung cancer is very small.

Genetic Effects

The chromosomes are organized in pairs of *homologous* chromosomes. In each pair, one is inherited from the father and the other from the mother. This means that for a simple trait, we have two alleles of the DNA segments affecting it. Individuals with the same allele in both chromosomes are called *homozygotes* while those with different alleles are called *hetrozygotes*. For a completely *dominant* allele, its genetic instructions are followed if it is present in at least one of the chromosome pairs, while a completely *recessive* alleles instructions come into play only if it is present in both chromosomes. Dominance and recessitivity are often considered as deviations from *additivity*, where the effect of two alleles equals the sum of the individual effects, i.e. the phenotype of the heterozygote is equal to the mean of the two possible homozygotes.

The concepts of recessitivity and dominance are used for homologous alleles at the same locus. When the total effect of genes at multiple loci is different from the sum of the individual effects, this is called *epistasis*.

Crossover, Genetic Linkage, Genetic Markers and Genetic Distance

In the process of *meiosis*, the homologous chromosomes exchange large segments when germ cells, i.e. egg and sperm cells, are formed. As a result of this, each chromosome that a child inherits from a parent will normally contain segments from both grandparents. For example, there are on average 2-3 such *crossovers* per chromosome during human meiosis.

Two loci are said to be *linked* if there is a tendency for them to be inherited together. Loci on different chromosomes are unlinked because they segregate independently. If the loci are located close together on the same chromosome they will normally be linked. However, because of recombination this is not always the case. The further apart the loci are the more likely it is that a recombination will occur, and if the loci are very far apart there is an equal chance that they will be on either chromosome in the pair and they are thus unlinked.

Using *genetic markers*, i.e. known DNA sequences that can be identified in experiments, the pattern of inheritance can be tracked through families. Also, finding a marker linked to a genetic disorder can lead to location of the gene causing the disorder. Genetic markers provide an important tool for *genetic mapping*, i.e. the process of connecting a trait to a locus in the genome.

The distance between loci can be expressed as physical or genetic distance. The physical distance is the number of base pairs between them. The genetic distance is defined by the average crossover frequency between the loci, and is expressed in Morgan, M. or centi-Morgan, cM. Genetic distance can be measured with a marker analysis over generations in a pedigree, counting how often two markers located on the same chromosome in the paternal generation are separated in the next generation.

Genetic Variation

Together with environmental effects, genetic variation is the origin of trait differences between individuals in a population. Genetic variation can occur due to *mutations*, i.e. a change of the base pairs in the DNA molecule. When reproductive cells containing a mutation combine to produce offspring, the mutation will be present in all cells of the offspring. Mutations can occur due to exposure to mutagens such as radiation and chemicals that destroy DNA. A mutation can result in that the instructions that direct the production of proteins is changed, which in turn can affect traits of the offspring. Most mutations are harmless in the sense that they do not result in any deleterious changes to traits. Also, mutations are an important driving force in evolution since the corresponding change of a trait may be helpful to the survival of the species. However, sometimes mutations cause problems in the functions of an organism. These problems are then called *genetic disorders*.

2.2 Traits

Traits are often classified according to its inheritance pattern. Basically, this can be single-gene or multifactorial.

Single-Gene Traits

Single-gene (also called Mendelian or monogenic) traits are caused by variations in the DNA sequence of a single gene. For example, cystic fibrosis, sickle cell anemia, Marfan syndrome, Huntington's disease, and hereditary hemochromatosis are disorders in humans which are single-gene disorders. Single-gene traits are inherited in a few, recognizable patterns, for example autosomal dominant or autosomal recessive. For a disorder with autosomal dominant inheritance, an affected individual has at least one affected parent, the disorder affects either sex and can be transmitted by either sex, and an affected heterozygous individual has a 50% chance of passing the defect on to their offspring. Examples of autosomal dominant conditions in humans are achondroplasia and Huntington disease. In autosomal recessive inheritance, affected individuals are usually born by unaffected parents, i.e. a parent can be a carrier of the mutation without being affected. Cystic fibrosis has a autosomal recessive inheritance pattern.

The first step involved in a genetic study of a trait is to do a pedigree study where the family tree is studied for certain phenotypic patterns. For mutations caused by a single gene, the inheritance pattern can then often be identified. The next step is to map the trait to a specific chromosome and locus. Sometimes, visual examination of the chromosomes is enough, for example if a large part of a chromosome is missing. However, in most cases the isolation of the locus is done by using genetic markers and linkage analysis. The markers are used in paternity tests to check which chromosomes were inherited from each parent. These genetic markers can also be used to see if a particular marker has a similar inheritance pattern with the trait under study. If so, it can be assumed that the marker is located close to the mutated gene. After the initial mapping analysis, the identified region is narrowed down, for example by introducing more genetic markers. Once the region is sufficiently small, several techniques can be used to locate the specific gene and sequence it.

Multifactorial and Quantitative Traits

Multifactorial (also called complex or polygenic) traits are affected by the variation in multiple genes and often also the environment. For example, genes that influence breast cancer susceptibility in humans have been found on several chromosomes [30]. For multifactorial traits, it is normally quite hard to identify the genetic loci affecting the trait. Simple pedigree methods can not be used because the effect of segregation of alleles of one gene is at least to some degree concealed by the effect of other genes and environmental effects. Thus, individuals with identical genotypes usually exhibit different phenotypes and simple patterns such as dominance or recessiveness are not present. As noted earlier, most economically and medically important traits in humans, animals and plants are multifactorial, and identifying the genetic variation affecting these traits is an important problem in genetics.

Multifactorial traits that can be measured on a continuous scale are called *quantitative*. A gene can be seen as a variable with K possible states, where each state corresponds to a possible allele-pair combination. If a trait is affected by n independent genes, there are K^n different genotypes available. According to the central limit theorem of statistics, the overall phenotype will be normally distributed if n is large enough. The phenotype difference between similar genotypes can be small, and further blurred by environmental variation, giving a continuous distribution overall.

If the heritability is large, many genes are needed to give a continuous distribution. If the heritability is small, the trait can be continuous even with a small number of genes because of environmental effects. If a complex trait has a normal distribution, it will have a mean (μ) and a variance (σ). The variance can be assumed to be the composed of a genetic and an environmental component,

$$\sigma_P^2 = \sigma_G^2 + \sigma_E^2.$$

The broad sense heritability is given by

$$H^2 = \frac{\sigma_G^2}{\sigma_P^2},$$

and measures the importance of genetic variation relative to the environmental contribution for the studied trait.

Table 1: Some heritability values based on twin studies

Trait	Heritability
Longevity	0.29
Height	0.85
Maximum heart rate	0.84
Memory	0.47
Verbal ability	0.63
Maximum blood lactate	0.34
Weight	0.63

As an illustration, Table 1 shows the heritability for some quantitative traits in humans determined using twin studies. As identical twins share the same genes, the variance between members of an identical twin pair is equal to σ_E^2.

Experimental populations

By providing a form of ”controlled genetic variation”, *experimental populations* are very important sources of information for analyzing the genetics behind a multifactorial trait. An individual that is homozygous at all locus is said to be *inbred*. Starting with two inbred lines of a species, different kinds of controlled crossed can be bred, for example:

- **F$_1$:** Produced by crossing individuals from two different inbred lines. The individuals in the first F$_1$ generation will all be genetically identical and heterozygous for all loci. The F$_1$ generation is the basis for other crosses, like F$_2$ or a backcross, which are used for the genetic analysis.

- **F$_2$ (or Intercross):** Produced by crossing individuals from the same F$_1$ generation.

- **Backcross:** Produced by crossing individuals from the F$_1$ generation with individuals from either of the two parental lines.

Both F$_2$ and backcross individuals will exhibit genetic variation because of crossover, and this variation is modeled in the statistical framework used for the analysis.

For some organisms, like humans and many other mammals, it is not possible to create inbred populations. In some cases, for example for farm animals, outbred populations created by breeding programs can still be employed for creating experimental populations that can be used for genetical analysis of the traits that were subject to breeding. Also, organisms where inbred populations are available, like mice, are often used as model organisms for e.g. humans.

The non-genetic effects on a trait can be controlled when searching for genes in plants and animals by placing them in controlled environments.

2.3 Quantitative Trait Loci, QTL

A QTL (quantitative trait locus) is a location/region in the genome that affects a quantitative trait. Finding an important QTL for a trait is the first step in a genetic analysis of the trait.

Historically, the field of quantitative genetics was mainly focused on the study of the aggregate effects of all the genes causing variation. This approach has given estimates of the genetic contribution to the observed phenotypic variation (heritability) and also the genetic correlation between various traits. For example, this type of knowledge has been used in animal breeding programs for a long time. However, during the last decades, significant progress has been made in the development of methods for mapping the effects of a trait to locations in the genome, *QTL mapping*. This has resulted in that several major genes responsible for quantitative traits have been located [24, 31].

One of the key developments in this area has been the establishment of large collection of genetic markers, which can be used to construct genetic maps of experimental and domestic species. These maps help to identify regions of the genome that may have a statistical association with the phenotype under study.

Genetic mapping of QTL

The first step in a standard QTL mapping experiment is to construct an experimental population as described above. In experimental populations the heritability can be increased by reducing the environmental influence by maintaining a constant environment for all individuals.

Genetic analysis of QTL can be done using a candidate gene approach or by using a genome scan based on linkage analysis. The papers A-D describe algorithms and implementations for the genome scan/linkage analysis approach. Here, multiple genetic markers are used to identify the regions in the genome that affect the trait. This method is general in the sense that it may provide that new and previously unlocated regions/genes are identified. However, it is important to select markers having sufficient information to maximize the probability to detect the co-segregation between marker and QTL [14, 36]. Since the markers will be located some distance apart, genome scanning can only resolve a QTL location with 10-30 cM resolution. It is necessary to invoke other methods to locate the QTL more precisely [23].

In practice, the genome scan approach is often combined with candidate gene analysis. If a gene has been previously identified to influence the trait under study, or if a gene in a another organism with sequence similarity and exhibiting similar variations in phenotype has been found, these genes can be used as candidate genes. Using statistical methods, the effect of the candidate gene on the variation of the quantitative trait can be analyzed. This approach has the disadvantage that only a limited region of the genome is analyzed, also this method will not provide information about new QTL.

QTL models

Models for the statistical association between genetic markers and QTL are reviewed in e.g. [26]. Some standard models can be found in [48]. Various statistical methods can be used when evaluating the models, ranging from simple ANOVA (Analysis of variation) tests to more complex approaches.

A QTL can be modeled by an allele substitution effect, which is commonly called an additive effect. In many cases the heterozygote phenotype deviates from the mean of the two homozygote, i.e. the trait is partly dominant or recessive. A dominance effect can be modeled by including a parameter for the deviation of the heterozygous phenotype from the mean of the two homozygotes. Using data from a backcross population, the combined effect of the additive and dominance effects of a locus can be modeled, while an F_2 population allows separation of the additive and a dominance effects in the model. In general, several traits are considered in each study and the level and direction of dominance depends on each trait.

The additive and dominance effects are also refereed to as *marginal genetic effects* since they only depend on a single locus. Quantitative traits are normally affected by the products of multiple genes. *Epistasis* effects describe such interaction between genes, i.e. when the action of one gene is modified by one or several other genes.

The F_2 also makes a more thorough investigation of epistasis possible, but a large population size is needed to obtain the same power to detect epistasis.

The first QTL mapping techniques were based on linkage analysis between single markers and phenotype [62, 65]. A major problem with this kind of analysis is that it is not possible to distinguish a closely linked QTL of small effect and a loosely linked QTL of large effect (linked to a single marker). Later, the development of *interval mapping* [33, 43, 45] solved this problem, providing schemes where it is possible to estimate both the effect and the position of a QTL.

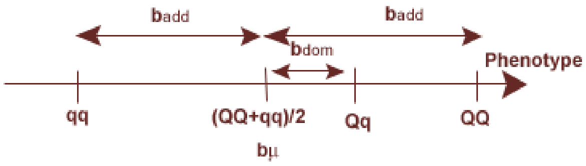


Figure 2: Relationship between additive and dominant effects

To illustrate the type of mathematical models used in QTL mapping, we consider an F_2 population with a particular potential QTL having alleles Q and q. Then the possible genotypes are then qq, QQ, Qq and qQ. Let y^i denote the phenotype of individual i and b_μ the average phonotype of the population. Also, let b_{add} denote the additive effect, b_{dom} the dominance effect, and ϵ^i the environmental noise. The model is then described by the equations

$$\begin{array}{rcl} qq : y^i & = & b_\mu - b_{add} + \epsilon^i \\ qQ \quad \text{or} \quad Qq : y^i & = & b_\mu + b_{dom} + \epsilon^i \\ QQ : y^i & = & b_\mu + b_{add} + \epsilon^i \end{array}$$

This is illustrated in figure 2 and can be summarized as

$$y^i = b_\mu + a^i_{add} b_{add} + a^i_{dom} b_{dom} + \epsilon^i,$$

where the indicator variables are defined by $a^i_{add} = -1$ and $a^i_{dom} = 0$ for genotype qq; $a^i_{add} = 0$ and $a^i_{dom} = 1$ for genotypes qQ or Qq; and $a^i_{add} = 1$ and $a^i_{dom} = 0$ for genotype QQ. Using this type of model, the three parameters b_μ, b_{add} and b_{dom} can be estimated. Using the same type of approach, models with different parameters can be formed. As a general rule, there should be many observations (individuals) per parameter in order to obtain reliable estimates. This means that large populations are required to make analysis with models with many parameters applicable.

Single QTL models of the type described above can be generalized for multiple QTL [39, 57, 66, 67] by adding more terms to the model. Also, it is possible to include epistatic interactions [61].

In general, assume that a model including d QTL is used and that a sequential map of the chromosomes and the genetic information within them is given. A position in the genome is identified by a number $x \in [0, G]$, where G is the total genome length. Let the vector $\mathbf{x} = [x_1 \quad x_2 \ldots x_d]$ denote potential positions of the d QTL and let m be the number of individuals in the experimental population. Also, let k be the total number of parameters in the model and let the vector y contain the m phenotype observations, the vector b contain the k regression parameters and the vector ϵ contain the m noise components. A general QTL model can then be written as

$$y = A(\mathbf{x})b + \epsilon, \tag{1}$$

where the design matrix $A(\mathbf{x})$ is an $m \times k$ matrix of coefficients for fixed effects and QTL effects. Here, only the QTL effects depend on $\mathbf{x}$. The matrix A has one row per phenotype observations while the number of columns is given by $k = k_{fix} + k_{QTL}$.

Maximum Likelihood and Linear Regression Statistics

Any set of d hypothetical QTL positions $\mathbf{x}$ can be used as input when building $A(\mathbf{x})$, but the genotypes of the individuals are (at best) only known at the marker loci. Several approaches can be used for estimating the regression parameters at loci between markers. One standard approach is maximum likelihood interval mapping [42, 45]. Here the parameter estimation is a nonlinear problem which must be solved iteratively, often using the EM [25] or ECM algorithms [55]. Another standard approach is the linear regression method [33, 54]. Since only a single, linear least-squares problem needs to be solved, this method is normally much less computationally expensive than if the maximum likelihood approach is used. When multiple QTL are included in the model, this becomes an important criterion when selecting the statistical approach. The main objective of the QTL analysis is to detect important regions for further experimental study, and using linear regression methods may make models including multiple QTL computationally feasible. This is also the approach used in this thesis. However, it should be noted that by implementing an alternative routine for maximum likelihood estimation, the codes presented in papers A-D can be easily modified to include also this approach.

In [41], and analytic and numerical investigation of the difference between QTL mapping based on maximum likelihood and linear regression is made. This study indicates that the maximum likelihood-based methods can in some cases be more accurate, precise and powerful. However, the methods were not compared for real data, where violations of model assumptions such as unequal variance within QTL genotype classes, segregation distortion and unusual inheritance patterns are likely to be present. It is difficult to asses the relative properties of the two approaches for analysis of experimental data sets.

Genetic Background Effects and Forward Selection

Methods for mapping of single QTL where the effects of other QTL believed to be present is taken into account have been derived [38, 39, 66]. When a QTL has been identified it can then be included in the model as a fixed effect, and then additional searches for other QTL can be made. This procedure is called *forward selection*, and drastically reduces the computational demand compared to a simultaneous search for multiple QTL. However, as remarked earlier, it has been shown [18] that such methods can be ineffective in detecting interacting QTL.

Significance and Randomization Testing

Since QTL mapping involves multiple statistical tests throughout the genome, the selection of a significance threshold is a key issue of the procedure. Since the use of a nominal significance threshold will lead to an elevated type I error (large number of false positives) various methods have been suggested to deal with the multiple comparisons [12, 44, 45, 63]. Empirical estimation of the overall significance thresholds can be done in a wide range of population designs by re-sampling techniques such as randomization testing [22]. Here the observed trait values are randomly shuffled over individuals (genotype) generating a sample with the original marker information, but with trait values randomly assigned over genotypes. The major drawback is that this method requires that on the order of 1000 independent QTL mapping problems are solved, implying the corresponding increase in computational complexity.

3 The Computational Problems in QTL analysis

The goal of a QTL search is to optimize the model fit over all possible positions $\mathbf{x}$. Using linear regression interval mapping, this corresponds to computing the optimal residual sum of squares, RSS_{opt} according to

$$RSS_{opt} = \min_{b,\mathbf{x}} (A(\mathbf{x})b - y)^T (A(\mathbf{x})b - y), \tag{2}$$

The location $\mathbf{x}$ that minimizes (2) is denoted by $\mathbf{x}_{opt}$, and is the most probable set of QTL positions for the given model. The expression (2) is a separable non-linear least-squares problem where the model is a linear combination of non-linear functions. The solution of (2) can be separated into two parts: For linear regression interval mapping, the inner problem will be a linear least-squares problem given by

$$RSS(\mathbf{x}) = \min_{b} (A(\mathbf{x})b - y)^T (A(\mathbf{x})b - y). \tag{3}$$

We refer to the process of solving (3) as *evaluating the objective function*. If another statistical approach is used, e.g. maximum-likelihood interval mapping, the evaluation of the objective function will instead require the solution of a non-linear problem.

The outer problem,

$$\min_{\mathbf{x}} RSS(\mathbf{x}), \tag{4}$$

is referred to as the *global search problem*. This is a d-dimensional, non-linear global optimization problem.

3.1 Evaluating the Objective Function

Given a particular set of QTL positions $\mathbf{x}$, the solution of the inner problem (3) determined the model fit for the linear regression statistical model. The least-squares problem can be solved using any standard algorithm. However, more efficient schemes that take the specific structure of the QTL mapping problems in to account [49] have also been developed.

3.2 Solving the Global Search Problem

The global optimization problem (4) normally has a large number of local minima. In Figure 3, a surface plot of a part of a typical objective function for a problem where $d = 2$ is shown.

In principle, the global search should be performed over all $\mathbf{x}$ in a d-dimensional hypercube, where the side is given by the size of the genome. However, the optimal value of $RSS(\mathbf{x})$ is independent of the ordering of QTL in the set. Therefore the problem exhibits a $d!$-fold symmetry and the search space can be reduced accordingly.

There are two levels of structure in the search space; The outer level is given by that the hyper-cube consists of a set of C^d d-dimensional, unequally sized *chromosome*

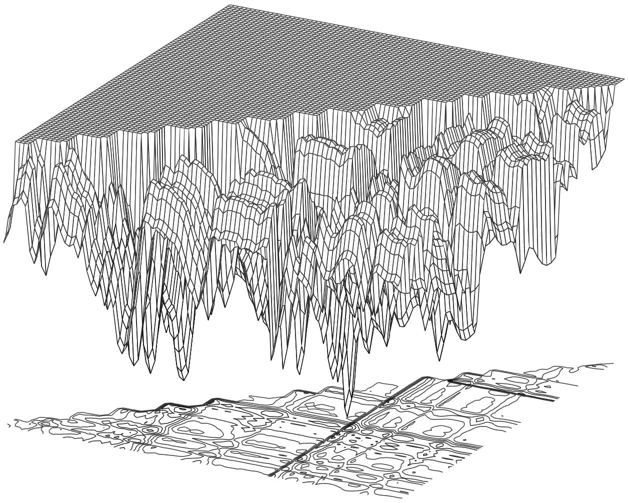


Figure 3: Part of a typical optimization landscape for a model where $d = 2$

combination boxes, cc-boxes, which can be identified by a d-vector of chromosome numbers. The function (3) is continuous within a cc-box, but discontinuous at the cc-box boundaries. The second level of structure is defined by the genetic markers. Each cc-box consists of a set of d-dimensional, unequally sized *marker boxes, m-boxes*, defined by the marker positions and the endpoints of the chromosomes. The function (3) is smooth within an m-box, but in general has a discontinuous derivative at the m-box boundaries.

The global search problem is solved using some global optimization scheme. In the next section, we first give a brief overview of global optimization problems and some classes or algorithms that can be used for solving them. Then we describe the DIRECT algorithm in some more detail. This is the basis of the parallel algorithms and implementations for QTL analysis in papers B and C. Finally, we give an account for earlier work on global optimization schemes for serial and parallel QTL computations.

4 Global Optimization

The objective of an algorithm for global optimization is to locate the globally best objective function value in the presence of multiple local optima. Global optimization problems arise in many areas where non-linear models are used. Examples are found in a range of applications in engineering design and scientific modeling as well as in biotechnology and financial planning.

A general global optimization problem can be formulated as

$$\begin{aligned} &\min f(\mathbf{x}) \\ &\text{subject to } \mathbf{x} \in D \end{aligned} \tag{5}$$

where the objective function $f : \mathbb{R}^d \rightarrow \mathbb{R}$ fulfills some regularity criterion (e.g. Lipschitz continuity), $\mathbf{x}$ is a d-vector of decision variables, and the feasible set D is non-empty. There is no straight-forward general algebraic characterization of the solution to a global optimization problem, corresponding to the Karush-Kuhn-Tucker conditions for local optimization problems. Also, if no global information about the function, e.g. a global Lipschitz constant, is available there is in general no way of *guaranteeing* that the solution obtained by a finite algorithm is a global optimum. This property has given global optimization a bad reputation as an "unsolvable problem". However, in practice there are many algorithms available, and some of them are often surprisingly successful. Some of these schemes are general in nature and applicable to large classes of problems, while others are tailored for specific problem types.

4.1 Algorithms for Global Optimization

If a local optimization method is applied to a global optimization problem then, depending on the starting point, the scheme can be trapped in a local minimum which is not the global optimum. Hence, special global search algorithms have to be used. These can be divided into different classes, e.g.:

- *Exact methods*, that exploit some known global information about the objective function to guarantee that the solution is found to some pre-set accuracy, versus *heuristic methods* that use some heuristic scheme to find a global minimum with supposedly high probability.
- *Deterministic methods*, that generate the same sequence of approximations if they are applied to the same problem repeatedly, versus stochastic methods where the approximations have an element of randomness included.

Some examples of methods are, further described in e.g. [58], are:

- **Exhaustive Search:** This is a brute-force method where the objective function is evaluated for all points in a fine mesh covering the search space. For an objective function with a known Lipschitz constant, the accuracy of this type of procedure is given a-priori. Also, this type of algorithm is normally very easy to implement. However, if the search space is large and/or the objective function is expensive to evaluate, the computational cost of an exhaustive search algorithm will often be prohibitive. The basic exhaustive search scheme provides a basis for other, more efficient deterministic algorithms, e.g., of branch and bound-type.
- **Branch and Bound:** This is a class of methods where, after evaluation of the objective function at certain points, search space regions where the minimum

can not be located are recursively cut away. To be able to perform such pruning of the search tree, some global information about the objective function must be available.

- **Random Search:** This is a random version of the exhaustive search algorithm where the objective function is evaluated at randomly chosen points instead of on a predefined mesh. Again, for problems with large search spaces and/or expensive objective function evaluations, the cost can be prohibitive. The basic random search scheme provides a basis for evolutionary algorithms.

- **Evolutionary Algorithms:** This is a large class of methods containing e.g. genetic algorithms. These schemes mimic evolution, starting with a population of search space points (individuals) represented by a set of data (a genome). Using operators such as mutations and crossover, the population reproduces to form a new generation. The fitness (objective function values) of the offspring is evaluated, and individuals with better function values are kept in the population with higher probability than the others. Genetic algorithm are rather easy to implement and often exhibit robust behavior, but they also often need many evaluations of the objective function to produce a good result.

- **Simulated Annealing:** This type of method is based on mimicking the cooling process of metals, where the atoms tend to crystalize in a configuration which corresponds to a global minimum in energy. Thus, the objective function is the energy state and the change in the design variables is similar to the change of the positions of the atoms during the cooling process. The optimization starts at a random point in the design space and makes a random step. If this step reduces the energy of the total system then the change is accepted and the algorithm proceeds. If the energy is not reduced, the move is either accepted or rejected based on some random probability function.

A problem with algorithms with pre-determined accuracy, like exhaustive search, is that the computational work normally grows exponentially with the number of variables d. For high-dimensional problems such algorithms can not be used, and we have to accept that some form of heuristics is included in the scheme. Algorithms for global optimization can be compared based on the following characteristics:

- Generality - The algorithm is applicable to wide ranges of problems.

- Robustness - The algorithm reliably computes the correct solution (up to some accuracy) in a reasonable amount of time and independently of how it is initialized.

- Efficiency - The algorithm computes the correct solution quickly

- Ease of use - The algorithm is easy to implement and understand, and it does not use a large number of user-determined parameters.

4.2 DIRECT

The DIRECT algorithm was first described in [40]. It falls into a category of global optimization algorithms which can be called *Branch without Bound*. Normally, the algorithm is implemented such that, if it is run for a long time, it performs an exhaustive search. However, DIRECT initially tries to focus the search to the “most interesting” regions of the search space. This could be compared to algorithms of Branch and Bound-type, where search space regions where the optimum can not reside are fully excluded from further search based e.g. on a Lipschitz criterion. However, DIRECT is related to the Lipschitz approach in the sense that in the algorithm, DIRECT views the Lipschitz constant as an unknown parameter that is used to decide whether local or global search should be given priority. Some properties of DIRECT are that it:

- can be easily applied to problems with simple constraints.
- does not use any derivative information.
- neither uses a Lipschitz constant, nor any estimate of it.
- if the objective function is Lipscitz continues, the algorithm will eventually solve the optimization problem to a predetermined accuracy (since the scheme then performs an exhaustive search). However, as for exhaustive search, DIRECT can be directly applied to problems with a discontinues objective function.

A number of implementations of DIRECT are available, including various minor [1,7,13] and sometimes major modifications [10,29,35] of the original algorithm. The name DIRECT is short-hand for the phrase *DIviding RECTangles*, which indicates how the algorithm works. The following description of a DIRECT implementation is found in [13]:

> The first step in the DIRECT algorithm is to transform the search space to be the unit hypercube. The function is then sampled at the center-point of this cube. Computing the function value at the center-point instead of doing it at the vertices is an advantage when dealing with problems in higher dimensions. The hypercube is then divided into smaller hyperrectangles whose center- points are also sampled. Instead of using a Lipschitz constant when determining the rectangles to sample next, DIRECT identifies a set of potentially optimal rectangles in each iteration. All potentially optimal rectangles are further divided into smaller rectangles whose center-points are sampled. When no Lipschitz constant is used, there is no natural way of defining convergence (except when the optimal function value is known as in the test problems). Instead, the procedure described above is performed for a predefined number of iterations.

The DIRECT algorithm

We now give a more detailed description of the original DIRECT algorithm. As indicated above, the first step is to transform the search space into the unit hyper-cube, i.e.,

$$\overline{\Omega} = \{\mathbf{x} \in \mathbb{R}^d : 0 \leq x_i \leq 1, \quad i = 1...d\}$$

The algorithm works in this normalized space, referring to the problem-specific space only when making function calls. The algorithm is initiated by computing $f(c_1)$, where c_1 is the center of the normalized search space. The next step is to evaluate the objective function at the points $c_1 \pm \delta e_i, i = 1, ..., d$, where δ is one-third of the side-length of the hyper-cube and e_i is the ith unit vector. Next, the pair-wise minimums for each direction are computed,

$$w_i = min(f(c_1 + \delta e_i), f(c_1 - \delta e_i)), 1 \leq i \leq d,$$

and the boxes with the smallest values of w_i are divided into thirds so that $c_1 \pm \delta e_i$ are the centers of new hyper-rectangles. This pattern is repeated for all directions, choosing the next direction by determining the next smallest w_i.

The algorithm now enters a loop where potentially optimal hyper-rectangles are identified, divided appropriately, and the objective function values at the centers of the new hyper-rectangles are sampled. Let $\epsilon > 0$ be chosen, and let f_{min} be the currently best function value. A hyper-rectangle j is said to be potentially optimal if there exists some $\widehat{L} > 0$ such that

$$\begin{aligned} f(c_j) - \widehat{L}d_j &\leq f(c_i) - \widehat{L}d_i, \forall i \ , \quad \text{and} \\ f(c_j) - \widehat{L}d_j &\leq f_{min} - \epsilon|f_{min}| \end{aligned}$$

In this definition, c_j is the center of hyper-rectangle j, and d_j is the distance from c_j to a vertex. The definition of the potentially optimal hyper-rectangles is based on an estimate of the smallest function value possible in each hyper-rectangle. The unknown parameter $\widehat{L}$ is assumed to provide the maximum rate of change of the objective function, i.e. it is the Lipschitz constant. The first equation is used to select the hyper-rectangle which has the smallest objective function value within a group of similarly sized hyper-rectangles. The second equation is used to make that DIRECT does not always select the hyper-rectangle with the best current objective function value, hereby making sure that the search does not get trapped in a local optimum.

Figure 4 gives a geometric representation of the how the definition above is used. Each point in the graph represents a hyper-rectangle in the search space. The potentially optimal hyper-rectangles that form the lower convex hull of the set of points. These points are the ones selected for subdivision in the next phase of the algorithm. The parameter ϵ is used so that $f(c_j)$ exceeds our current best solution by a non-trivial amount. Experiments have shown that, provided $1 \times 10^{-2} \leq \epsilon \leq 1 \times 10^{-7}$ the value for ϵ has a negligible effect on calculations. In papers B-D, we use $\epsilon = 1 \times 10^{-4}$

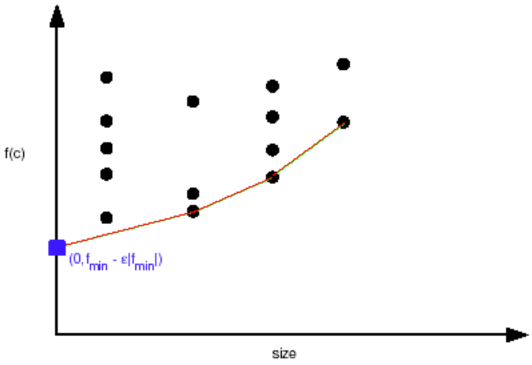


Figure 4: Convex Hull, source [29]

In general, there are more than one potentially optimal rectangle found in each iteration. Once these potentially optimal hyper-rectangles have been identified, the iteration is completed by dividing them. The divisions are restricted to be done along the longest direction(s) of the hyper-rectangle. This restriction ensures that the rectangles will shrink in every direction. If the hyper-rectangle is a hyper-cube, divisions will be done in all directions as was the case for the initial step.

In Figure 5, the DIRECT division pattern for three DIRECT iterations applied to a test problem is illustrated. For each iteration, the potentially optimal rectangles are marked as grey.

Termination

For most, if not all, widely used methods for global optimization, no well-founded, practical stopping criterion is available. This is also the situation for DIRECT. For a problem with a known Lipschitz constant the algorithm can of course be run until a sufficiently accurate exhaustive search has been performed, but this approach can not be used in practice. Some form of heuristics must be used in the stopping criterion. One popular solution is to run the algorithm until a predetermined number of function evaluations have been performed. Another approach, which is the one chosen in papers B-D, is to combine a finest resolution of the mesh with a criterion where a predefined number of objective function evaluations have been performed without any further improvement of the global minimum.

Parallelization

In [10], a parallel DIRECT algorithm is implemented. This code uses a version of the DIRECT algorithm, called aggressive DIRECT. Here, the idea of using the Lipschitz

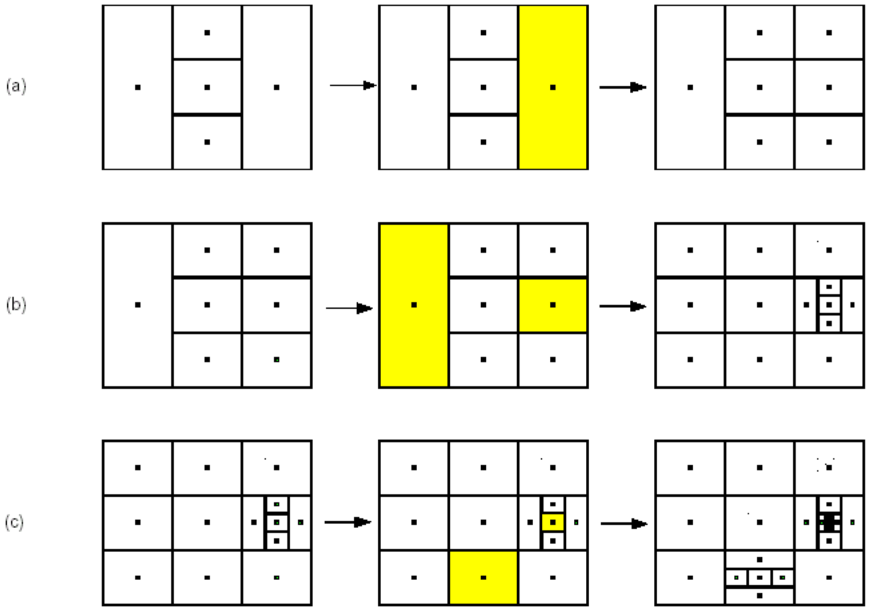


Figure 5: DIRECT iterations, source [28]

constant is discarded and all hyper-rectangles with the smallest objective function in each class are divided. This provides additional parallelism. The implementation uses a master-slave model where a single copy of the central hyper-rectangle data structure is updated and the objective function evaluations are distributed.

In [34], a parallelization strategy for the original DIRECT algorithm is described. This work is further developed in [35], where a parallelized algorithm suitable for very large scale problems is described. This code uses both search-space level and complex-hull-level parallelization, and the approach is somewhat similar to what is described in paper C. However, one important difference is that in paper C, the original algorithm is modified to be applicable also on very loosely-connected systems like grids.

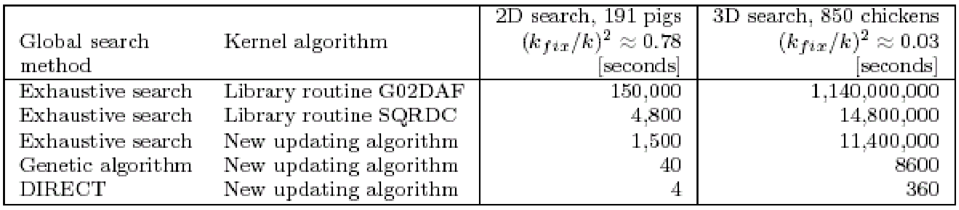
4.3 Global Optimization for QTL mapping problems

The global search problem in QTL analysis described in Section 3 is a global optimization problem in d dimensions. The first models for mapping of QTL we presented in [33, 45]. These models were able to model the effect of single QTL ($d = 1$) and the implementations used exhaustive search for solving the global optimization problem. Later, models were developed which were able model the effect of multiple QTL and their interactions. Some exhaustive search problems where $d = 2$ have been solved to detect epistatic QTL [21, 27, 37, 46]. The reason for the prohibitive cost for $d > 2$ is that an exhaustive search on a d-dimensional 1 cM mesh in an G cM genome requires G^d solutions of the kernel problem. In a typical example, $G = 2,500$ cM and an exhaustive search for $d = 3$ would require 16×10^9 function evaluations. If exhaustive search is considered to be the only viable optimization method, performing QTL mapping using simultaneous search for two QTL combined with randomization testing is even today a very demanding computation.

One of the early approached to tackle this problem was to reduce the computational complexity by reducing the search regions [42], increasing the step size in the exhaustive search, or ignoring effects such as epistasis in the model. However, this reduces the accuracy and statistical power of the search. Also, models that use forward selection [38,39,66] were developed. However, as remarked earlier, such models have been shown to be less efficient for detecting epistatic QTL [18].

In [19], a genetic optimization algorithm implemented in a library has been used for QTL anaysis problems where $d = 2$. The genetic algorithm had some difficulty in finding the global optimum when epistasis was included in the model. It was observed that the genetic algorithm sometimes failed when a QTL pair lacked significant marginal effects. This can be explained by that the genetic algorithm has an inherent forward selection property.

Lately, the DIRECT scheme described above has also been applied for solving the global search problem [50], leading to a dramatic improvement in efficiency compared to when exhaustive search is used. Some of the results are summarized in Figure 6.



Global search method	Kernel algorithm	2D search, 191 pigs $(k_{fix}/k)^2 \approx 0.78$ [seconds]	3D search, 850 chickens $(k_{fix}/k)^2 \approx 0.03$ [seconds]
Exhaustive search	Library routine G02DAF	150,000	1,140,000,000
Exhaustive search	Library routine SQRDC	4,800	14,800,000
Exhaustive search	New updating algorithm	1,500	11,400,000
Genetic algorithm	New updating algorithm	40	8600
DIRECT	New updating algorithm	4	360

Figure 6: Results for QTL analysis, source [52]

In [50], the original DIRECT algorithm is adopted to take the special structure of the search space into account. The function (3) is discontinuous between cc-boxes, and therefore the search space is divided into cc-boxes all ready at initialization, and the center of each box is sampled. This is sufficient to fulfill the Lipschitz continuity condition of DIRECT, since the Lipschitz method is used for bounding (4) within (and not across) hyper-boxes. Also, in contrast to the original method, the box sizes are not normalized in order to retain the relation between the distance measure cM and change in the genotype. Finally, the algorithm does not divide boxes smaller than $1cM$ further.

In [51], further improvements of the DIRECT scheme for the global search problem are presented. Here, a hybrid global-local scheme is developed where DIRECT is run for a fixed number of function evaluations and then a local search is performed

Table 2: Stopping rule parameters. The minimum number of function evaluations allowed without improvement of minimum

d	N_f
2	841
3	11487
4	117740
5	965468
6	6597367

as a refinement step. When a hyper-rectangle smaller than a certain limit is chosen for subdivision it is sent to the local phase. There it is first investigated if the box lies completely within a m-box. If not the box is divided along the marker interval boundaries, ensuring that the local algorithm is only applied within a region where the objective function is smooth. Three different methods were used for the local search:

- DIRECT
- Steepest Descent
- Quasi-Newton

The DIRECT scheme in [51] is terminated when N_f function evaluations have been performed without further improvement of the optimal value. For a model with d QTL, the size of the search space is $G^d/d!$. This motivates that N_f is chosen as $N_f = (P_{alg}.G)^d/d!$, where the parameters P_{alg} was determined by performing a large number of numerical experiments, adjusting P_{alg} so that the global optimum is located in each data set. The values for N_f is Table 2 have been calculated based on these values. $P_{alg}.G$ for DIRECT without any local search is 41. The values for other local search algorithms can be found in [51].

5 High Performance Computing: Systems and Programming Models

The term High Performance Computing (HPC) refers to the use of parallel computing systems such as clusters and supercomputers for solving computational problems. The systems contain multiple processing nodes connected with some type of interconnect. Today, HPC is a standard tool in many research fields. Computational Science and Engineering, i.e. mathematical modeling and large scale simulations, is becoming a third pillar of research in science and technology, complementing theory and experiments.

5.1 HPC Computer Architectures

Symmetric Multiprocessors (SMPs) and Chip Multicore Processors (CMPs)

A symmetric multiprocessor, or SMP, is a parallel computer architecture where several identical processors are connected to a single, shared main memory. This allows each of the processors to access all data in the global memory, i.e. the system has shared address space. Communication between the processors is performed via the shared memory, and the time required for communicating a single data entry is comparable to a reference to the main memory. The tightly-coupled shared memory system is rather costly, and it also imposes a scalability limit for how many processors that can be attached to it. Today, SMPs are often used for running commercial applications like databases and other transaction systems and e.g. web servers.

During the last couple of years, chip multicore processors, or CMPs, have emerged. These are microprocessors which can be viewed as having the processors and part of the memory system of an SMP on a single chip. When programming a CMP, the same type of programming models as for an SMP can be used. However, new targets for optimization should be considered since, e.g., communication between the cores of a CMP is much faster than a main memory reference.

Clusters

A computer cluster is a set of computer nodes connected via a high-speed interconnect. Each node has a local memory, and a processor can not directly access the memory in another node. Instead, transfer of data is handled by some form of message passing. The cluster nodes can have a single processor or they can be SMP systems with several processors. Today, most new clusters have nodes with one or more CMP processors. For clusters, it is easier to provide the scalability needed to build very large parallel computers, and the largest systems in the world are clusters. The cluster network is cheaper than the SMP memory system, and since the cluster nodes are normally commodity components, clusters often provide a cost-effective way of achieving the computing power needed.

Grids

A grid is a possibly heterogeneous collection of computers connected to each other via the internet. The administration of the grid, including scheduling of work and set-up of communication, is handled by a software layer called the middleware. A grid is a highly loosely-coupled parallel computer system, and the behavior is often non-deterministic since there is interaction with many other users and applications.

A primary advantage of grid systems is that of cost. Commodity hardware computing nodes are combined, and together they can provide the same processing power of a supercomputer at a fraction of the cost. Also, distributing the hardware makes issues such as cooling and power requirements easier to deal with. For parallel computing, the primary disadvantages are the slow communication and the stochastic nature of scheduling. Also, since the grid approach is rather new, the complex middleware systems still need to be improved.

To be suitable for implementation on grid systems, an algorithm must be pleasantly parallel in the sense that the main part of the work can be subdivided into sufficiently large chunks that can be executed in a highly independent way, but where possibly some local/serial preprocessing, synchronization at a few workflow stages, and/or some local/serial postprocessing can be included.

Many challenging problems like protein folding, financial modeling, earthquake and climate modeling have already been approached using grid computing. Grid computing is also recently introduced as a way of providing computing power in a commercial setting, selling cpu-capacity to clients requiring it on an on-demand basis.

5.2 Programming models

MPI

The message passing interface (MPI) [4,32] is a language independent communication protocol, mainly aimed at cluster architectures with local memory nodes. For such systems, it has become the de-facto standard for parallel programming. There are many implementations of message passing sharing the MPI API. MPI bindings are available for C and Fortran, and also for languages like Java and Python. It should be noted that MPI libraries are normally also available on shared memory systems, implying that message passing codes can be executed without modification on these systems also.

OpenMP

On a parallel computer with a shared memory, a programming model where several threads of control are run simultaneously, communicating via memory references, is often used. For computational applications, OpenMP has become the most widely used tool of this type. Using a shared memory model, the data does not have to explicitly partitioned over the different processors and it is often easier to get a parallel code going than if a message passing model is used. However, shared memory programming is also error-prone, since it is easy to introduce e.g. race conditions when modifying shared variables.

OpenMP consists of a set of compiler directives and a few routines to interact with the run-time environment. Most vendors provide implementations of OpenMP, and recently OpenMP is also added in the the Gnu compilers. Using a shared memory model combined with compiler directives provides a possibility of introducing the parallelism incrementally in a serial program, where work on one portion of the program at a time can be done without dramatic changes in the code. OpenMP also provides for both coarse-grained and fine-grained parallelism.

Traditionally, OpenMP has been used for large-scale shared memory systems of SMP type. However, with the deployment of personal computers with multicore chips, the OpenMP programming model will possibly be used much more widely in the future.

Distributed Shared Memory

The most cost-effective and scalable way to build an HPC system is possibly via a cluster with local memory nodes. However, for the programmer there are many advantages offered by the shared memory model. Thus there has been much effort in trying to combine these two approaches, providing a programming environment for using a virtual shared memory. Unified Parallel C (UPC) [8], Linda [20] and JavaSpaces [3] are some of tools along these lines.

Hybrid Local Memory - Shared Memory programming

Both MPI and OpenMP are portable across a wide variety of platforms. MPI codes are usually written to dynamically decompose the problem among MPI processes. These processes run their piece of the problem on different processors usually having their own local memory. OpenMP codes, on the other hand, have directives that describe how parallel sections or loop iterations are to be split up among threads. On a cluster with SMP nodes, a hybrid programming models where MPI is used at an outer (node) level while OpenMP is used inside a node is natural. Developing hybrid codes present additional challenges compared to using only a single level of parallelism. The code should perform correctly with what ever combination of multithreading and message passing. Hybrid code will possibly be used widely when clusters and grids with multi-core chips will become standard.

5.3 QTL mapping using HPC systems

Most of the popular QTL softwares [11, 15, 60] are built from serial codes in C or R, and some of them are plug-ins to other software such as Matlab. These codes are used on single processors only, and we do not know of any attempts to parallelize them.

However, parallelizing the exhaustive search algorithm for the global search in QTL mapping problems is a rather straight-forward task. A first implementation was also presented in [17], where a two dimensional search was parallelized by distributing the calculations for cc-boxes over the different processors. The code was written in Fortran 90, and MPI was used for message passing. Computations where genetically interesting results were derived were performed on a cluster and a Cray T3E system. The code used a master-slave model where all the slaves performed their local calculations independently and the results were output to files. These output files were then merged afterwards to form the final result. Also, the genetic algorithm described in [16] has been parallelized using the same code base by including calls to the parallelized PGA-Pack library.

For grid systems, the gridQTL project [2] aims at grid-enabling the application QTLexpress [60] and making it accessible via a grid portal.

6 Data Sets

For the experiments in Papers A-D, we have used two representative data sets. One data set consists of experimental data from two F_2 crosses between outbred lines of wild boars and domestic pigs [9] with 191 individuals and a genome size of 2300cM. All the studied traits for this data set were growth related. The other set of data is derived by simulating a mouse F_2 intercross with 500 individuals, more details can be found in [51].

7 Summary of attached papers

In papers A-D, we describe several flexible parallel algorithms and implementations for simultaneous mapping of multiple QTL using the linear regression statistical method and different global optimization methods for solving the global search problem.

In the papers, we use the models described in Section 2 and the data sets described in Section 6 as representative examples. We do not consider the relevance of the models used, nor do we consider the important problem of which statistical model to use. Also, we do not attempt to establish the statistical significance of the results, and we do not draw any form of genetic implications from the computations. However, the algorithms and implementations described in the papers provides a basis for future studies of all these issues, and for performing complete QTL mapping analysis using models including multiple QTL.

7.1 Paper A

The codes regularly used for QTL analysis by geneticists are serial and use the brute-force exhaustive search algorithm for solving the global search problem.

In paper A, we develop a parallelized code for simultaneous search of multiple, possible interacting, QTL using the standard exhaustive search algorithm. Even though this scheme is very computationally expensive, a parallel implementation is valuable. Using results for representative data sets and models computed using exhaustive search, it is possible to accurately evaluate the accuracy and efficiency of more elaborate optimization methods. Also, the code can be used as a basis for implementing other schemes for performing the global search,

When searching for d QTL, the search space for the problem is a d-dimensional hypercube, where each side is given by the size of the genome. The genome is divided into C chromosomes. This means that we can view the search space as a collection of d-dimensional *chromosome combination boxes*, cc-boxes. Since genes on different chromosomes are unlinked, the objective function is normally discontinuous at the cc-box boundaries. This means that the QTL search could be viewed as essentially consisting of $n \approx C^d/d!$ independent global optimization problems, one for each cc-box included in the search space.

This partitioning of the problem is a natural basis for a straight-forward parallelization of multi-dimensional QTL searches:

```
do (in parallel) i=1:n
  l_sol(i) = global_optimization(cc-box(i));
end
Find the global solution among l_sol(:);
```

Since the objective function evaluations (model fit computations) are rather expensive, the work for performing global minimization in a cc-box is almost exclusively determined by the number of calls to the objective function evaluation routine. For an exhaustive search algorithm, this number is known a priori. However, since the size of the different cc-boxes varies a lot (the chromosomes have very different lengths), the work will be very different for different boxes. Hence, the two main issues when implementing a parallel version of exhaustive search is load-balancing and granularity for the parallel loop.

The parallel code is based on existing serial QTL codes in Fortran 90 and C. The implementation uses a hybrid, two-level scheme scheme for partitioning the tasks corresponding to global optimization in cc-boxes over a set of parallel processors. On the outer level a static partitioning of tasks is used, and on the inner level dynamic partitioning is exploited. For the outer, static level a separate code partitions the cc-boxes over p_s different jobs. The preparation code stores the description of the partitioning in p_s input files, which are then used by p_s instances of the computational code. A job control script submits the p_s computational jobs and wait for them to finish. Finally, when all jobs have finished, a small separate code compares the local results and outputs the global optimum. The inner, dynamic level of parallelization is implemented using a MPI master-slave model. The master process maintains a queue of tasks, each consisting of global optimization in a single cc-box. These tasks are dynamically handed out to the worker processes as soon as they have completed their previous task.

The outer level parallelism, using a job control script, is easily implemented on various parallel architectures, including grids. The inner level parallelism requires the availability of MPI, and can normally be used only within clusters. We perform experiments using the Swedish national grid system Swegrid [6], and the results show that the scheduling policies used in this system normally favors using only the outer level parallelism. For comparison, we also perform experiments using a shared memory server (SunFire 15k).

7.2 Paper B

In [50, 51], the global optimization algorithm DIRECT is used instead of exhaustive search for solving the global search problem in QTL mapping. In paper B, we discuss how to parallelize this tightly-coupled algorithm efficiently on loosely-coupled systems such as grids. We describe a new version of the algorithm, based on partitioning the search space and removing some of the global communication at an outer level of the algorithm. We show that, for the QTL mapping problems, the serial performance of the new version of DIRECT is in fact sometimes better than that of the original algorithm.

The implementation of the parallel DIRECT scheme is based on an existing C code [51]. The code uses the same type of outer level parallelism as in paper A; First a separate code is used to partition the global search space into local search spaces, cc-box-sets. The partitioning code stores the cc-box-set data on separate files which are used as input by the DIRECT code which run on each node. A job manager scrip submits each job with a different cc-box-set and waits for them to finish and gets the final output. For a grid system each job is described in a job description file, and when using a cluster or a shared memory system a similar batch script is used. For each job, the executable is the same, the only difference is in the cc-box-set file which describes the local search space. Once the code is executed it outputs the local minimum. The job-manager script downloads each of these outputs and computes the global minimum using a minor serial computation.

For the DIRECT algorithm, it is not possible to determine the work for each cc-box set beforehand. It is not known where the algorithm will perform most of the objective function evaluations. However, a block-cyclic distribution of the cc-boxes is used to get reasonable load balance.

Again, we perform experiments on the Swegrid system and an Sunfire 15k shared memory server. We also performed some experiments using a cluster with a standard interconnect.

7.3 Paper C

With the introduction of multicore processors, a new level of parallelism is becoming available. In the future, clusters and grids will have nodes with one or more multicore processors. In paper C we enhance the parallel DIRECT code from paper B by introducing thread-level parallelism at an inner level.

There are several levels of parallelism available within the DIRECT algorithm. In paper B, a partitioning of the search space was used. Several cc-boxes were blocked together and DIRECT was executed for the corresponding search region. Here, each search can be performed independently. In paper C, we add thread-level parallelism, implemented using OpenMP, to parallelize the initialization of the different cc-boxes in each cc-box set, and also when evaluating the objective function in the potentially optimal hyper-rectangles inside the DIRECT scheme. The resulting hybrid scheme, where threading and search space partitioning are both utilized, mainly targets computing resources like grids and clusters with shared memory or multicore nodes.

We perform experiments mainly on a cluster with dual-processor nodes where each processor has two cores. The results show that normally a combination of search space partitioning and multithreading results in the most efficient computations.

7.4 Paper D

A problem with grid systems is that an important set of tools is still missing. When using the grid, the researcher has to use a scripting language for job submission. For example, using the ARC middleware in the Swegrid [6] infrastructure, we have to write a xRSL (extended Resource Specification Language) file that describes the hardware and software requirements of the job. If we want to submit multiple jobs (assuming each job is independent of the others) then we must build scripts for this purpose. Also, submitting multiple jobs usually requires developing a 'babysitting' applications which check if the jobs have completed and then downloads the results. These tasks can preferably be undertaken by a grid portal. Such a portal can be specific to an application (application portal) or it can be generic.

In paper D, we present a pilot implementation of the QTL mapping software presented in Papers A-C within the Lunarc Application Portal framework [5]. A GUI-based interface for submitting grid jobs for the parallel DIRECT code is developed.

We implemented this system using the Lunarc application portal framework on one of the clusters in the Swegrid system.

8 Future Work

The experiments performed in this thesis have been aimed at evaluating the performance of the new algorithms and implementations. No experiments attempting to derive results of interest in genetics have been attempted. It would be a natural step to use the software framework developed for performing genetically interesting investigations. This require further development of the grid portal to arrive at a tool which is easy to use for biologists. Exhaustive testing of the grid portal needs to be done, and the development should be done in close collaboration with users. The proper GUI design should be created. Also, functionality for users to upload their own data sets and examine the results needs to be added and integrated into the system.

In [51], a hybrid global-local algorithm based on DIRECT is developed. This scheme has been shown to perform better for the data sets examined. We have not used the hybrid algorithm in our code, and it would be interesting to examine if the efficiency of the parallel DIRECT implementations can be improved too.

We can use different objective functions and models for QTL interactions. These different models would possibly give different genetic outcome. It is rather easy to modify the code so that other models are used. Variance component based models for QTL analysis is an interesting area. The genetic models and the computational methods used for the objective function evaluation in this type of work are different to the ones used in this thesis. These objective function evaluations are much more expensive, indicating that the need for parallel computing is even larger. Recently, new formulations of the the model and new algorithms have been studied [56, 59]. It would be highly interesting to combine these schemes with our parallelized global search routines.

Thread-level parallelism will be available in all future clusters and grid nodes. How to harness that power is also interesting; we have evaluated some of this in paper C. However, further development of algorithms and codes for multicore chips is needed to fully take advantage of the architectural properties of these systems.

References

[1] Basic direct code. http://www4.ncsu.edu/ definkel/research/index.html.

[2] The GridQTL project. www.gridqtl.org.uk/.

[3] Jini and JavaSpaces. www.jini.org.

[4] LAM/MPI. www.lam-mpi.org.

[5] Lunarc application portal toolkit. www.lunarc.lu.se/Software/lap/.

[6] Swegrid. www.swegrid.se.

[7] Tomlab-mathlab optimization environment. http://www.ima.mdh.se/tom.

[8] Unified Parallel C. http://upc.gwu.edu/.

[9] L. Andersson, C. Haley, H. Ellegren, S. Knott, M. Johansson, K. Andersson, L. Andersson-Eklund, I. Edfors-Lilja, M. Fredholm, and I. Hansson. Genetic mapping of quantitative trait loci for growth and fatness in pigs. *Science*, 263:1771–1774, 1994.

[10] C.A. Baker, L.T. Watson, B. Grossman, R.T. Haftka, and W.H. Mason. Parallel global aircraft configuration design space exploration. In *High performance symposium 2000*, pages 101–106, 2000.

[11] C. Basten, B. Weir, and Z.-B. Zeng. *QTL Cartographer, version 1.15*. Dept. of Statistics, North Carolina State University, Raleigh, NC, 2001.

[12] Y. Benjamini and Y. Hochbery. Controlling the false discovery rate: A practical and powerful approach to multiple testing. *Journal of the Royal Statistical Society, Series B*, 57:289–300, 1995.

[13] Mattias Bjorkman and Kenneth Holmstrom. Global optimization using DIRECT algorithm in matlab. *Advanced Modeling and Optimization*, 1(2):17–37, 1999.

[14] D. Botstein, R.L. White, M. Skolnick, and R.W. Davis. Construction of a genetic linkage map in man using restriction fragment length polymorphisms. *American Journal of Human Genetics.*, 32(3):314–331, 1980.

[15] K.W. Broman, H. Wu, S. Sen, and G.A. Churchill. R/qtl mapping in experimental crosses. *Bioinformatics*, 19(7):889–890, 2003.

[16] Ö. Carlborg, L. Andersson, and B. Kinghorn. The use of a genetic algorithm for simultaneous mapping of multiple interacting quantitative trait loci. *Genetics*, 155:2003–2010, 2000.

[17] Ö. Carlborg, L. Andersson-Eklund, and L. Andersson. Parallel computing in regression interval mapping. *Journal of Heredity*, 92(5):449–451, 2001.

[18] Ö. Carlborg and C.S. Haley. Epistasis: too often neglected in complex trait studies? *Nature Reviews Genetics*, 5:618–625, 2004.

[19] Ö. Carlborg, S. Kerje, K. Schtz, L. Jacobsson, P. Jensen, and L. Andersson. A global search reveals epistatic interactions between QTL for early growth in chicken. *Genome Res.*, 13:413–421, 2003.

[20] Nicholas Carriero and David Gelernter. Linda in context. *Commun. ACM*, 32(4):444–458, 1989.

[21] K. Chase, F.R. Adler, and K.G. Lark. Epistat:a computer program for identifying and testing interactions between pairs of quantitative trait loci. *Theoretical and Applied Genetics*, 94:724–730, 1997.

[22] G. Churchill and R. Doege. Empherical threshold values for quantitative trait mapping. *Genetics*, 138:963–971, 1994.

[23] A. Darvasi. Experimental strategies for the genetic dissection of complex traits in animal models. *Nature Genetics*, 18:19–24, 1998.

[24] G.H. Davis, G.W. Montgomery, A.J. Allison, R.W. Kelly, and A.R. Bray. Segregation of a major gene influencing fecundity in progeny of booroola sheep. *New Zealand Journal of Agricultural Research*, 25:525–529, 1982.

[25] A.P. dempster, N.M. Laird, and D.B. Rubin. Maximum likelihood from incomplete data via EM algorithm (with discussion). *J. R. Statis. Soc.*, 39:1–38, 1977.

[26] R.W. Doerge. Mapping and analysis of quantitative trait loci in experimental populations. *Nature reviews-genetics*, 3:43–52, 2002.

[27] M.D. Edwards, C.W. Stuber, and J.F. Wendel. Molecular-marker-facilitated investigations of quantitative trait loci in maize. i. numbers, genomic distribution and types of gene action. *Genetics*, 116:113–125, 1987.

[28] Daniel E Finkel. *DIRECT Optimization Algorithm User Guide*. North Carolina State University, March 2003.

[29] D.E. Finkel and C.T. Kelly. An adaptive restart implementation of DIRECT. In *International Conference on Continuous Optimization*, 2004.

[30] M.H. Green. Genetics of breast cancer. *Mayo Clinic Proceedings*, 72(1), 1997.

[31] L. Grobet, L.J. Martin, D. Poncelet, D. Pirottin, B. Brouwers, J. Riquet, A. Schoeberlein, S. Dunner, F. Menissier, J. Massabanda, R. Fries, R. Hanset, and M. Georges. A deletion in the bovine myostatin gene causes the double-muscled phenotype in cattle. *Nature Genetics*, 17:71–74, 1997.

[32] W. Gropp, E. Lusk, N. Doss, and A. Skjellum. A high-performance, portable implementation of the MPI message passing interface standard. *Parallel Computing*, 22(6):789–828, 1996.

[33] C.S. Haley and S.A. Knott. A simple regression method for mapping quantitative trait loci in line crosses using flanking markers. *Heredity*, 69:315–324, 1992.

[34] J. He, M. Sosonkina, C.A. Shaffer, J.J. Tyson, L.T. Watson, and J.W. Zwolak. A hierarchical parallel scheme for a global search algorithm. In *High performance computing symposium*, 2004.

[35] J. He, A. Verstak, L.T. Watson, and M. Sosonkina. Design and implementation of a massively parallel version of DIRECT. Technical report, Technical Report TR-06-02, Computer Science, Virginia Tech., 2006.

[36] C.M. Hearne, S. Ghosh, and J.A. Todd. Microsatellites for linkage analysis of genetic traits. *Trends Genet.*, 8(8):288–294, 1992.

[37] J.B. Holland, L.S. Moser, L.S. O'Donoughe, and M. Lee. QTLs and epistasis associated with vernalization response in oat. *Crop Science*, 37:1306–1316, 1997.

[38] R. Jansen. A general mixture model for mapping quantitative trait loci by using molecular markers. *Theoretical And Applied Genetics*, 85:252–260, 1992.

[39] R. Jansen. Interval mapping of multiple quantitative trait loci. *Genetics*, 135:205–211, 1993.

[40] D. Jones, C. Perttunen, and B. Stuckman. Lipschitzian optimization without the lipschitz constant. *J. Optimization Theory App*, 79:157–181, 1993.

[41] C.-H. Kao. On the difference between maximum likelihood and regression interval mapping in the analysis of quantitative trait loci. *Genetics*, 156:855–865, 2000.

[42] C.-H. Kao, Z.-B. Zeng, and R. Teasdale. Multiple interval mapping for quantitative trait loci. *Genetics*, 152:1203–1216, 1999.

[43] S. Knapp, W. Bridges, and D. Birkes. Mapping quantitative trait loci using molecular marker linkage maps. *Theoretical and Applied Genetics*, 79:583–592, 1990.

[44] L. Kruglyak and E.S. Lander. High-resolution genetic mapping of complex traits,. *American Journal of Human Genetics*, 56:1212–1223, 1995.

[45] E. Lander and D. Botstein. Mapping mendelian factors underlying quantitative traits using RFLP linkage maps. *Genetics*, 121:185–199, 1989.

[46] Z. Li, S.R. Pinson, W.D. Park, A.H. Paterson, and J.W. Stansel. Epistasis for three grain yield components in rice(oryza sativa l.). *genetics*, 145:453–465, 1997.

[47] S. Lincoln, M. Daly, and E. Lander. Mapping genes controlling quantitative traits with mapmaker/qtl1.1. Technical report 2nd edition, Whitehead Institute, 1992.

[48] B.-H. Liu. *Statistical Genomics*, volume first edition. CRC Press, 1998.

[49] K. Ljungberg, S. Holmgren, and Ö. Carlborg. Efficient algorithms for quantitative trait loci mapping problems. *Journal of Computational Biology*, 9(6):793–804, 2002.

[50] K. Ljungberg, S. Holmgren, and Ö. Carlborg. Simultaneous search for multiple QTL using the global optimization algorithm DIRECT. *Bioinformatics*, 20:1887–1895, 2004.

[51] K. Ljungberg, M. Kateryna, and S. Holmgren. Efficient algorithms for multi-dimensional global optimization in genetic mapping of complex traits. Technical report, Division of Scientific Computing,Dept of IT, Uppsala University, 2005.

[52] Kajsa Ljungberg, Sverker Holmgren, and Orjan Carlborg. Global optimization in QTL analysis. Poster - International Conference on Research in Computational Molecular Biology, 2004.

[53] M. Lynch and B. Walsh. *Genetics and Analysis of Quantitative Traits*. Sinauer Associates, Inc., 1998.

[54] O. Martinez and R. Curnow. Estimating the location and the sizes of effects of quantitative trait loci flanking markers. *Theor Appli Genet*, 85:480–488, 1992.

[55] X.-L. Meng and D. Rubin. Maximum likelihood estimation via the ECM algorithm. a general framework. *Biometrika*, 80:267–278, 1993.

[56] K. Mishchenko, S. Holmgren, and L. Rönnegård. Efficient implementation of the AI-REML iteration for variance component QTL analysis. Preliminary Technical Report, June 2007.

[57] J. Moreno-Gonzalez. Genetic models to estimate additive and non-additive effects of marker-associated QTL using multiple regression techniques. *Theor Appl Genet*, 85:435–444, 1992.

[58] Panos M Pardalos and H Edwin Romeijn. *Handbook of Global Optimization*, chapter 15. Kluwer Academic Publishers, 2002.

[59] L. Rönnegård, K. Mishchenko, S. Holmgren, and Ö. Carlborg. Increasing the efficiency of variance component QTL analysis by using reduced rank IBD matrices. *Genetics*, Accepted for publication.

[60] G. Seaton, C. Haley, S. Knott, M. Kearsey, and P. Visscher. QTL express: mapping quantitative trait loci in simple and complex pedigrees. *Bioinformatics*, 18:339–340, 2002.

[61] K. Shimomura, S.S. Low-Zeddies, D.P. King, T.D. Steeves, A. Whiteley, J. Kushla, P.D. Zemenides, A. Liu, M.H. Vitaterna, G.A. Churchill, and J.S. Takahashi. Genome-wide epistatic interaction analysis reveals complex genetic determinants of circadian behavior in mice. *Genome Res*, 11:959–980, 2001.

[62] M. Soller, T. Brody, and A. Genizi. On the power of experimental design for the detection of linkage between marker loci and quantitative trait loci in crosses between inbred lines. *Theor Appl Genet*, 47:35–39, 1976.

[63] B.R. Southey and R.L. Fernando. Controlling the proportion of false positives among significant results in QTL detection. In *Proceedings of the 6th world congress of genetics applied to livestock production*, volume 26, pages 341–244, Armidale, NSW, Australia., 1998.

[64] F. Sugiyama, G.A. Churchill, D.C. Higgins, C. Johns, K.P. Makaritsis, H. Gavras, and B. Paigen. Concordance of murine quantitative trait loci for salt-induced hypertension with rat and human loci. *Genomics*, 71:70–77, 2001.

[65] J.I. Weller. Maximum likelihood techniques for the mapping and analysis of quantitative trait loci with the aid of genetic markers. *Biometrics*, 42:627–640, 1986.

[66] Z.-B. Zeng. Theoretical basis for separation of multiple linked gene effects in mapping quantitative trait loci. *Proc Nat Acad Sci USA*, 90:10972–10976, 1993.

[67] Z.-B. Zeng. Precision mapping of quantitative trait loci. *Genetics*, 136:1457–1468, 1994.

Recent licentiate theses from the Department of Information Technology

2007-004	Olof Rensfelt: *Tools and Methods for Evaluation of Overlay Networks*
2007-003	Thabotharan Kathiravelu: *Towards Content Distribution in Opportunistic Networks*
2007-002	Jonas Boustedt: *Students Working with a Large Software System: Experiences and Understandings*
2007-001	Manivasakan Sabesan: *Querying Mediated Web Services*
2006-012	Stefan Blomkvist: *User-Centred Design and Agile Development of IT Systems*
2006-011	Åsa Cajander: *Values and Perspectives Affecting IT Systems Development and Usability Work*
2006-010	Henrik Johansson: *Performance Characterization and Evaluation of Parallel PDE Solvers*
2006-009	Eddie Wadbro: *Topology Optimization for Acoustic Wave Propagation Problems*
2006-008	Agnes Rensfelt: *Nonparametric Identification of Viscoelastic Materials*
2006-007	Stefan Engblom: *Numerical Methods for the Chemical Master Equation*
2006-006	Anna Eckerdal: *Novice Students' Learning of Object-Oriented Programming*
2006-005	Arvid Kauppi: *A Human-Computer Interaction Approach to Train Traffic Control*
2006-004	Mikael Erlandsson: *Usability in Transportation – Improving the Analysis of Cognitive Work Tasks*
2006-003	Therese Berg: *Regular Inference for Reactive Systems*
2006-002	Anders Hessel: *Model-Based Test Case Selection and Generation for Real-Time Systems*
2006-001	Linda Brus: *Recursive Black-box Identification of Nonlinear State-space ODE Models*

UPPSALA
UNIVERSITET

Department of Information Technology, Uppsala University, Sweden