Examensarbete

An iterative solution method for p-harmonic functions on finite graphs with an implementation

Karl Tomas Andersson

LiTH - MAT - EX - - 2009 / 03 - - SE
An iterative solution method for p-harmonic functions on finite graphs with an implementation

Applied Mathematics, Linköpings universitet

Karl Tomas Andersson

LiTH - MAT - EX - - 2009 / 03 - - SE

Examensarbete: 30 hp

Level: D

Supervisor: Anders Björn,
Applied Mathematics, Linköpings universitet

Examiner: Anders Björn,
Applied Mathematics, Linköpings universitet

Linköping: April 2009
An iterative solution method for p-harmonic functions on finite graphs with an implementation

In this paper I give a description and derivation of Dirichlet’s problem, a boundary value problem, for p-harmonic functions on graphs and study an iterative method for solving it. The method’s convergence is proved and some preliminary results about its speed of convergence are presented. There is an implementation accompanying this thesis and a short description of the implementation is included. The implementation will be made available on the internet at http://www.mai.liu.se/~anbjo/pharmgraph/ for as long as possible.

Nyckelord
Dirichlet’s problem, graph, iteration, numerical solution, p-harmonic function.
Abstract

In this paper I give a description and derivation of Dirichlet’s problem, a boundary value problem, for $p$-harmonic functions on graphs and study an iterative method for solving it.

The method’s convergence is proved and some preliminary results about its speed of convergence are presented.

There is an implementation accompanying this thesis and a short description of the implementation is included. The implementation will be made available on the internet at http://www.mai.liu.se/~anbjo/pharmgraph/ for as long as possible.

Keywords: Dirichlet’s problem, graph, iteration, numerical solution, $p$-harmonic function.
Acknowlegdements

I would like to thank my supervisor, Anders Björn, for his great help and amazing patience with me. My opponent Hannes Uppman also deserves my thanks.

I would also like to thank my friends and family for supporting me during my work on this paper.
Notation

Most of the reoccurring abbreviations and symbols are described here.

Symbols

\[ G = (N, \partial N, E) \]  The graph consisting of nodes \( N \), edges \( E \) and boundary nodes \( \partial N \).

\( n_i \)  The interior node with index \( i \).

\( u \)  A function assigning a value to each node of the graph.

\( N_i \)  The value at the node with index \( i \), \( N_i = u(n_i) \).

\( F^p_i \)  The update function for a given \( p \) centered at node \( n_i \).

\( F^p \)  The composition of all local update funtions,

\[ F^p = F^p_n \circ F^p_{n-1} \circ \cdots \circ F^p_1. \]

\( e(X) \)  The \( p \)-energy of the graph when the nodes have values \( u(N) = X \).

\( x \sim y \)  Denotes that \( x \) and \( y \) are adjacent nodes in the graph.

\( \sum_{x \sim y} \)  Summation over all edges in the graph.

\( \sum_{y: x \sim y} \)  Summation over all nodes \( y \) adjacent to \( x \).
Contents

1 Introduction 1
   1.1 Definition of the problem .............................. 1
   1.2 Equivalent formulations .............................. 4
   1.3 The case $p = \infty$ ................................. 6
   1.4 Another derivation of the problem ................. 8

2 Method 13
   2.1 Convergence of the method ....................... 15

3 Implementation details 19

4 Speed of convergence 21

5 Conclusions 25
   5.1 Further studies ................................. 25

6 Appendix A 27
Chapter 1

Introduction

This text is written as a master of science final thesis at Linköpings universitet by Karl Tomas Andersson with Anders Björn as supervisor and examiner, during 2006 to 2009.

This first chapter will first define the problem this paper attempts to solve and give some well known results about it. It will also give definitions of \textit{p-harmonic functions} in the more general sense and describe how they are related to the problem at hand.

1.1 Definition of the problem

The purpose of this thesis is to describe a method of solving the \textit{Dirichlet problem}\footnote{Also known as the boundary value problem.} for \textit{p-harmonic} functions on graphs, the problem consists of finding a function \(u\) that for a given graph minimizes the so called \(p\)-energy function.

\textbf{Definition 1.1.} Given a finite, connected graph \(G\) where some of the nodes are defined to be \textit{boundary nodes}, \(\partial N\), and the rest are \textit{interior nodes} we are asked to find a function \(u\) such that \(u(n)\) takes a specified value when \(n\) is a boundary node and the function

\[ e(u) = \sum_{n_i \sim n_j} |u(n_i) - u(n_j)|^p, \]

called the \textit{p-energy function}, is minimized.\footnote{The notation \(x \sim y\) means that \(x\) and \(y\) are neighbours in the graph.} The function \(u\) is called \textit{p-harmonic} over \(N \setminus \partial N\).

The number \(p\) is assumed to be \(1 < p \leq \infty\) throughout this entire text, the case \(p = \infty\) is covered separately in Section 1.3. We will look at a simple example.

\textbf{Example 1.2.} The graph in this example (Figure 1.1) has four boundary nodes, marked with a ring around them, with given boundary values \(a, b, c, d\) and two interior nodes with values \(n_1\) and \(n_2\). If we set \(p = 2\) the function we want to minimize is

\[ f(n_1, n_2) = |n_1 - a|^2 + |n_1 - b|^2 + |n_1 - n_2|^2 + |n_2 - c|^2 + |n_2 - d|^2 \]
and since $|x|^2 = x^2$ this is easy to expand and simplify to

$$f(n_1, n_2) = 3n_1^2 - 2n_1n_2 + 3n_2^2 - 2(a + b)n_1 - 2(c + d)n_2 + (a^2 + b^2 + c^2 + d^2).$$

Taking the gradient of $f$ gives us

$$\nabla f = \left(\begin{array}{c} 6n_1 - 2n_2 - 2(a + b) \\ -2n_1 + 6n_2 - 2(c + d) \end{array}\right).$$

Setting $\nabla f = 0$ and rearranging the resulting system of equations gives us

$$\begin{pmatrix} n_1 \\ n_2 \end{pmatrix} = \frac{1}{3} \begin{pmatrix} n_2 + a + b \\ n_1 + c + d \end{pmatrix}.$$  

So we see that each of our interior nodes’ value is the average value of its neighbours’ values. Solving the equations gives

$$\begin{pmatrix} n_1 \\ n_2 \end{pmatrix} = \frac{1}{8} \begin{pmatrix} 3(a + b) + (c + d) \\ 3(c + d) + (a + b) \end{pmatrix}.$$  

As we could see in the example, the case $p = 2$ is easy to handle, since the equations for the gradient are linear. When $p \neq 2$ on the other hand, the equations become non-linear and much harder to handle.

The Dirichlet problem on graphs always has a unique solution as long as the set of boundary nodes is not empty, in that case any constant function is of course a solution with $p$-energy = 0. To show this we will first show that there exists a solution and then that it must be unique.

We will need the following definition a few times in the thesis.

**Definition 1.3.** The set $S$ is defined as

$$S = \{ X \mid X_i \in [N_{\min}, N_{\max}] \text{ for } i = 1, 2, \ldots, k \}.$$  

where $N_{\min}$ and $N_{\max}$ are the smallest and largest values of $u$ at the boundary nodes, respectively, and $k$ is the number of interior nodes.
1.1. Definition of the problem

Since this a closed, bounded subset of $\mathbb{R}^n$ it is compact. [3, p. 77]

**Theorem 1.4.** The Dirichlet problem has a solution.

**Proof.** First we note that if there is a solution $u$, then the vector

$$(u(n_1), u(n_1), \ldots, u(n_k)) \in S.$$  

This follows because if we find all nodes where $u(n_i) < N_{\min}$ and all nodes where $u(n_i) > N_{\max}$ and create a new function with values $N_{\min}$ or $N_{\max}$ at those points instead, the $p$-energy will be decreased. To see this note that when we change all nodes with too large value to $N_{\max}$, each changed term $|u(x) - u(y)|^p$ in the $p$-energy will become 0 if both values were too large and it will decrease if only one of them was changed. The same argument is true when we increase the nodes with too small values to $N_{\min}$.

We can now just look at the values at the nodes, $N_i = u(n_i)$, and consider this as an optimization problem of a continuous function from a compact subset of $\mathbb{R}^n$ to $\mathbb{R}$ and it is well known that every real valued, continuous function on a compact set has both a minimum and maximum. [4, p. 34] 

**Theorem 1.5.** The solution to the Dirichlet problem is unique when $\partial N \neq \emptyset$.

To prove this we will need the following lemma.

**Lemma 1.6.** For all real numbers $a$ and $b$ the inequality

$$|a + b|^p \leq 2^{p-1}(|a|^p + |b|^p)$$

holds, with equality only when $a = b$.

**Proof.** We first divide both sides with $2^p$ and get

$$\left| \frac{a + b}{2} \right|^p \leq \frac{|a|^p + |b|^p}{2}.$$  

If we now define $f(x) = |x|^p$ we rewrite the inequality as

$$f \left( \frac{a + b}{2} \right) \leq \frac{f(a) + f(b)}{2}.$$  

Note that $f$ is strictly convex since $f'$ is a strictly increasing function. For all strictly convex functions we have that if $a \neq b$ then $f(at + (1-t)b) < tf(a) + (1-t)f(b)$ for all $t \in (0, 1)$. If we let $t = \frac{1}{2}$ we get $f(\frac{a+b}{2}) < \frac{f(a) + f(b)}{2}$.

The case $a = b$ however gives

$$f \left( \frac{a + a}{2} \right) = f(a) = \frac{f(a) + f(a)}{2}.$$  

Combining these two cases completes the proof. □

**Proof of Theorem 1.5.** Assume that there are two solutions with minimal $p$-energy, $u$ and $v$, such that $e(u) = e(v) = \hat{e}$. We then form the function

$$w = \frac{1}{2}(u + v).$$
If we now calculate the $p$-energy of $w$ we get
\[
e(w) = \sum_{n_i \sim n_j} \frac{|u(n_i) + v(n_i) - u(n_j) - v(n_j)|^p}{2}
= \frac{1}{2^p} \sum_{n_i \sim n_j} |(u(n_i) - u(n_j)) + (v(n_i) - v(n_j))|^p
\]
and by using the lemma
\[
e(w) \leq \frac{1}{2^p} \sum_{n_i \sim n_j} 2^{p-1}(|u(n_i) - u(n_j)|^p + |v(n_i) - v(n_j)|^p)
= \frac{1}{2} \sum_{n_i \sim n_j} (|u(n_i) - u(n_j)|^p + |v(n_i) - v(n_j)|^p) = \frac{1}{2}(e(u) + e(v)) = \hat{\varepsilon}.
\]

But since $\hat{\varepsilon}$ is minimal, we must have $e(w) = \hat{\varepsilon}$. By the lemma, the inequality is an equality only when $u(n_i) - u(n_j) = v(n_i) - v(n_j)$ for all neighbouring nodes $n_i, n_j$. Since the graph is assumed to be connected, we can for each interior node $n_0$ find a path to a boundary node, $n_0 \sim n_1 \sim \cdots \sim n_k \in \partial N$. As $u$ and $v$ agree on the boundary node $n_k$ they agree on $n_{k-1}$ because $u(n_i) - u(n_j) = v(n_i) - v(n_j)$. By induction $u$ and $v$ agree on every node and therefore $u = v$ and the solution is unique.

\[\square\]

### 1.2 Equivalent formulations

Using Definition 1.1 directly and using optimization techniques is not usually the best way of solving the Dirichlet problem. We will show two equivalent formulations of the problem.

**Theorem 1.7.** A function is $p$-harmonic if and only if
\[
\sum_{n_i \sim n_j} |u(n_i) - u(n_j)|^{p-2}(u(n_i) - u(n_j))(w(n_i) - w(n_j)) = 0 \quad (1.1)
\]
for every, so called, test function $w$ with $w(n_i) = 0$ for $n_i \in \partial N$.

In the above definition we also define $|a - b|^{p-2}(a - b) = 0$ when $p < 2$ and $a = b$ in order to maintain continuity everywhere.

The idea of the following proof is taken from Holopainen–Soardi [2].

**Proof.** Suppose that $u$ is $p$-harmonic and let $u_t = u + tw$, where $t \in \mathbb{R}$ and $w$ has boundary values equal to 0 as above. Because $u$ minimizes the $p$-energy we get
\[
0 = \frac{d}{dt}(e(u_t))_{t=0}
= \frac{d}{dt} \left( \sum_{n_i \sim n_j} |u(n_i) - u(n_j) + t(w(n_i) - w(n_j))|^p \right)_{t=0}
= p \sum_{n_i \sim n_j} |u(n_i) - u(n_j)|^{p-2}(u(n_i) - u(n_j))(w(n_i) - w(n_j)).
\]
This shows one direction of the equivalence. Now suppose that $u$ satisfies (1.1) for all test functions. If we let $v$ be a function with the same boundary values as $u$ and use $u - v$ as our test function in (1.1) we get
\[
0 = \sum_{n_i \sim n_j} |u(n_i) - u(n_j)|^{p-2}(u(n_i) - u(n_j))(v(n_i) - u(n_i)) - (v(n_i) - v(n_j)) \cdot \]
Rewriting this gives us
\[
e(u) = \sum_{n_i \sim n_j} |u(n_i) - u(n_j)|^{p-2}(u(n_i) - u(n_j))(v(n_i) - v(n_j))
\leq \sum_{n_i \sim n_j} |u(n_i) - u(n_j)|^{p-1}|v(n_i) - v(n_j)|.
\]
Using Hölder’s inequality we get
\[
e(u) \leq \left( \sum_{n_i \sim n_j} |u(n_i) - u(n_j)|^p \right)^{\frac{p-1}{p}} \left( \sum_{n_i \sim n_j} |v(n_i) - v(n_j)|^p \right)^{\frac{1}{p}}
= e(u)^{\frac{p-1}{p}} e(v)^{\frac{1}{p}}.
\]
So we have
\[
e(u) \leq e(u)^{\frac{p-1}{p}} e(v)^{\frac{1}{p}},
\]
dividing\(^3\) by $e(u)^{\frac{p-1}{p}}$ gives
\[
e(u)^{\frac{1}{p}} \leq e(v)^{\frac{1}{p}},
\]
and therefore $e(u) \leq e(v)$ for all functions $v$ so $u$ is $p$-harmonic.

Another formulation, which is the one we will be using most of the time, is as follows.

**Theorem 1.8.** A function $u$ is $p$-harmonic if and only if
\[
\Delta_p u(n_i) = \sum_{j:n_i \sim n_j} |u(n_i) - u(n_j)|^{p-2}(u(n_i) - u(n_j)) = 0
\]
for each interior node $n_i$.

In the above definition, just like above, we also define $|a - b|^{p-2}(a - b) = 0$ when $p < 2$ and $a = b$.
This proof is also partly based on one by Holopainen–Soardi [2].

**Proof.** If $w$ is an arbitrary test function, we can write $w = \sum_i w_i$ where each

\(^3\)If $e(u) = 0$ then it is obviously minimal since the $p$-energy is always non-negative.
$w_i(n_j) = 0$ when $i \neq j$ and $w_i(n_i) = w(n_i)$. We can then rewrite (1.1) as

$$0 = \sum_{n_i \sim n_j} |u(n_i) - u(n_j)|^{p-2}(u(n_i) - u(n_j))(w(n_i) - w(n_j))$$

$$= \sum_{n_i \sim n_j} |u(n_i) - u(n_j)|^{p-2}(u(n_i) - u(n_j))(w(n_i) - 0)$$

$$+ \sum_{n_i \sim n_j} |u(n_i) - u(n_j)|^{p-2}(u(n_i) - u(n_j))(0 - w(n_j))$$

$$= \sum_{i} \sum_{j : n_i \sim n_j} |u(n_i) - u(n_j)|^{p-2}(u(n_i) - u(n_j))(w_i(n_i) - 0)$$

$$+ \sum_{j} \sum_{i : n_i \sim n_j} |u(n_i) - u(n_j)|^{p-2}(u(n_i) - u(n_j))(0 - w_j(n_j))$$

$$= 2 \sum_{i} \sum_{j : n_i \sim n_j} |u(n_i) - u(n_j)|^{p-2}(u(n_i) - u(n_j))w_i(n_i)$$

$$= 2 \sum_{i} w_i(n_i) \Delta_p u(n_i).$$

So we have

$$0 = 2 \sum_{i} w_i(n_i) \Delta_p u(n_i) \quad (1.2)$$

as an equivalent formulation of (1.1).

It follows that if $\Delta_p u(n_i) = 0$ for all interior nodes then (1.2) is obviously true.

Now assume that $u$ is $p$-harmonic and fix an interior node $x$ and pick a test function $w$ such that $w(n_i) = 1$ and $w(n_j) = 0$ for all other nodes $n_j$. By Theorem 1.7 we see that

$$0 = \sum_{n_i \sim n_j} |u(n_i) - u(n_j)|^{p-2}(u(n_i) - u(n_j))(w(n_i) - w(n_j))$$

$$= \sum_{n_i \sim n_j} |u(n_i) - u(n_j)|^{p-2}(u(n_i) - u(n_j))(w(n_i) - 0)$$

$$+ \sum_{n_i \sim n_j} |u(n_i) - u(n_j)|^{p-2}(u(n_i) - u(n_j))(0 - w(n_j))$$

$$= 2 \sum_{n_i \sim n_j} |u(n_i) - u(n_j)|^{p-2}(u(n_i) - u(n_j))w(n_i)$$

$$= 2 \Delta_p u(n_i).$$

Therefore $\Delta_p u(n_i) = 0$ at all interior nodes $n_i$, for all $p$-harmonic functions. \square

### 1.3 The case $p = \infty$

We have specified that $1 < p \leq \infty$, but the case when $p = \infty$ is of course special and requires its own derivation.

If we just take Definition 1.1 and let $p$ tend to $\infty$ we will not get a sensible definition, since all the terms in the sum will tend to 0, 1 or $\infty$ and it is easy to
construct example graphs that have minimal \( p \)-energy = 0 for infinitely many solutions.

So instead we will use the formulation from Theorem 1.8 as the basis for our definition.

**Definition 1.9.** A function, \( u \), is \( \infty \)-harmonic if

\[
u(n_i) = \lim_{p \to \infty} u_p(n_i)
\]

for all interior nodes \( n_i \), where \( u_p \) is the function that satisfies

\[
\Delta_p u_p(n_i) = 0.
\]

This can however be simplified to a much more manageable form as the following theorem shows.

**Theorem 1.10.** A function \( u \) is \( \infty \)-harmonic if

\[
u(n_i) = \max_N u(n_i) + \min_N u(n_i)
\]

where \( N \) are the neighbours of \( n_i \), for all interior nodes \( n_i \).

To show this we will use the following lemma.

**Lemma 1.11.** If \( u \) is \( p \)-harmonic, with boundary values given by the function \( v \) then for any two numbers, \( a \) and \( b \), the function \( \tilde{u}(n) = au(n) + b \) is also \( p \)-harmonic with boundary values given by \( \tilde{v}(n) = av(n) + b \).

**Proof.** If we put the new function \( \tilde{u} \) into the \( p \)-energy function we get

\[
e(\tilde{u}) = \sum_{n_i \sim n_j} |au(n_i) + b - au(n_j) - b|^p
\]

\[
= |a|^p \sum_{n_i \sim n_j} |u(n_i) - u(n_j)|^p
\]

\[
= |a|^p e(u),
\]

which is obviously minimized if \( e(u) \) is minimized. \( \square \)

The main consequence of Lemma 1.11 is that we can rescale the values of the nodes to any interval we want before we calculate the \( p \)-energy or \( \Delta_p u \) and then switch back to our old scale afterwards.

**Proof of Theorem 1.10.** Due to Lemma 1.11 we can always change scale so that the smallest value at \( n_i \)’s neighbours is 0 and the largest is 2. We now let \( u_+(n_i) = 1 + \epsilon \) for some \( \epsilon > 0 \) in our new scale and calculate

\[
\Delta_p u_+(n_i) = \sum_{j: n_j \sim n_i} |u_+(n_i) - u_+(n_j)|^{p-2}(u_+(n_i) - u_+(n_j))
\]

\[
= A(1 + \epsilon)^{p-1} + \sum_i a_i^{p-1} - \sum_j b_j^{p-1} - B(1 - \epsilon)^{p-1},
\]

where \( A \) and \( B \) is the number of neighbours with value 0 and 2, respectively, \( a_i < 1 + \epsilon \) and \( b_j < 1 \) for all \( i \) and \( j \). Note that the first term, \( (1 + \epsilon)^p \),
grows exponentially with regards to $p$ and all the other terms have a base with smaller absolute value than $1+\epsilon$, so the first term will dominate the other terms eventually. That means that we can pick $P_+$ such that

$$\Delta_p u_+(n_i) > 0, \quad \text{when } p > P_+.$$ 

If we now let $u_-(n_i) = 1 - \epsilon$ we can show in the same way that there is a number $P_-$ such that

$$\Delta_p u_-(n_i) < 0, \quad \text{when } p > P_-.$$ 

Since we know that $\Delta_p u_p(n_i) = 0$ we get the inequality

$$\Delta_p u_-(n_i) < \Delta_p u_p(n_i) < \Delta_p u_+(n_i), \quad \text{when } p > \max(P_-, P_+).$$

Because $\Delta_p$ is strictly increasing with respect to $u(n_i)$ this means that

$$1 - \epsilon < u_p(n_i) < 1 + \epsilon, \quad \text{when } p > \max(P_-, P_+),$$

$$|u_p(n_i) - 1| < \epsilon, \quad \text{when } p > \max(P_-, P_+).$$

This shows that $\lim_{p \to \infty} u_p(n_i) = 1$. If we change scale back to whatever we started with we get

$$\lim_{p \to \infty} u_p(n_i) = \frac{\max_N u(n_i) + \min_N u(n_i)}{2}$$

because the rescaling operation maps the midpoint of the interval $(0, 2)$ to the midpoint of $(\min_N u(n_i), \max_N u(n_i))$. \hfill \Box

1.4 Another derivation of the problem

The Dirichlet problem for harmonic functions usually refers to a partial differential equation in $\mathbb{R}^n$, that is defined in the following way.

**Definition 1.12.** $\Delta f(x) = \text{div}(\nabla f(x)) = 0$ for $x \in D$ and $f(x) = g(x)$ for $x \in \partial D$, where $g$ is a specified function on $D$’s boundary, $\partial D$.

Harmonic functions on $\mathbb{R}^n$ are of course well understood and studied in great detail due to their great importance in electromagnetism, complex analysis and many other diverse fields.

The Dirichlet problem can also be formulated as the completely equivalent formulation that a continuous function $f$ is harmonic in $D$ if

$$E(f) = \int_D |\nabla f(x)|^2 \, dx$$

is minimal for $f$, with $f(x) = g(x)$ for $x \in \partial D$ for a given function $g$.\footnote{This is not quite true if $D$ is irregular, but let us not go into that here.}

This definition can be generalized by defining a $p$-harmonic function to be a solution in the weak sense\footnote{In other words, both $f$ and $g$ can be distributions, not just functions. The derivate of a distribution $f$ is defined by $\int_{\mathbb{R}^n} f' u = -\int_{\mathbb{R}^n} f u'$ for all infinitely differentiable functions $u$ with compact support.} to the following equation.
1.4. Another derivation of the problem

Definition 1.13. A function $f$ is $p$-harmonic in a set $D$ if
\[
\Delta_p f(x) = \text{div}(|\nabla f(x)|^{p-2} \nabla f(x)) = 0 \quad \text{for } x \in D
\]
and $f(x) = g(x)$ for $x \in \partial D$, where $g$ is a specified function on $D$'s boundary.

\[
E(f) = \int_D |\nabla f(x)|^p \, dx
\]
is minimal for $f$, with $f(x) = g(x)$ for $x \in \partial D$ for a given function $g$.

It is clear that the classical version is a special case of the $p$-harmonic functions, when $p = 2$.

To see the connection between Definition 1.13 and Definition 1.1 above, we first make a (seemingly) different definition of $p$-harmonic functions on graphs. This is due to Shanmugalingam [6].

In the case of graphs, we consider functions on graphs as being composed of one function $u$, from nodes to $\mathbb{R}$, and for each edge $(n_i, n_j) \in E$, an absolutely continuous function $f_{n_i, n_j} \in C^1[0, 1]$, whose values agree at the endpoints
\[
\begin{align*}
  f_{n_i, n_j}(0) &= u(n_i), \\
  f_{n_i, n_j}(1) &= u(n_j), \\
  f_{n_i, n_j}(x) &= f_{n_j, n_i}(1 - x).
\end{align*}
\]

Finding the $p$-harmonic function in this case means that we should minimize the sum of the integrals $\int_0^1 |\nabla f_{n_i, n_j}(t)|^p \, dt$ and since the functions are one-dimensional this is just $\int_0^1 |f'_{n_i, n_j}(t)|^p \, dt$.

Theorem 1.14. The integral $\int_0^1 |f'(t)|^p \, dt$, with boundary values
\[
f(0) = a \quad \text{and} \quad f(1) = b
\]
is uniquely minimized by the absolutely continuous function
\[
f(x) = a + (b - a)x.
\]

A formal definition of absolutely continuous functions can be found in [5, p. 145], for this paper it is sufficient to know that a function $f$ is absolutely continuous if $f' \in L^1$ and the fundamental theorem of calculus
\[
f(x) - f(a) = \int_a^x f'(t) \, dt
\]
holds.

Proof. By Hölder’s inequality we have
\[
\left| \int_0^1 f'(t) \, dt \right| \leq \int_0^1 |f'(t)| \, dt \leq \left( \int_0^1 |f'(t)|^p \, dt \right)^{\frac{1}{p}} \left( \int_0^1 |1|^q \, dt \right)^{\frac{1}{q}},
\]
We can also consider a more general case where each edge may have different lengths, but in this thesis we will only consider the case where all edges have length 1.
where \( \frac{1}{p} + \frac{1}{q} = 1 \).

By the fundamental theorem of calculus, as \( f \) is absolutely continuous, we get
\[
\left| \int_0^1 f'(t) \, dt \right| = |f(1) - f(0)| = |b - a|
\]
and it is also obvious that \((\int_0^1 |t|^q \, dt)^{1/q} = 1\). Therefore we can simplify the inequality to
\[
|b - a| \leq \left( \int_0^1 |f'(t)|^p \, dt \right)^{\frac{1}{p}}.
\]
This holds for all \( f \) with the given boundary values. However in the case \( f(x) = a + (b - a)x \) we get
\[
\left( \int_0^1 |f'(t)|^p \, dt \right)^{\frac{1}{p}} = \left( \int_0^1 |b-a|^p \, dt \right)^{\frac{1}{p}} = |b - a|.
\]
So there is an equality instead of an inequality, therefore it is a minimizer since the inequality holds for arbitrary \( f \).

To see that the linear function is the only minimizer, we assume that there are two functions, \( f \) and \( g \) with the given boundary values, that are both minimizers,
\[
\left( \int_0^1 |f'(t)|^p \, dt \right)^{\frac{1}{p}} = \left( \int_0^1 |g'(t)|^p \, dt \right)^{\frac{1}{p}} = |b - a|
\]
and create a new function from these two as
\[
h = \frac{f + g}{2}.
\]
We then use Lemma 1.6 and get
\[
\left| \frac{f'(t) + g'(t)}{2} \right|^p \leq \frac{1}{2} \left( |f'(t)|^p + |g'(t)|^p \right)
\]
with equality only at the points where \( f'(t) = g'(t) \). Therefore if we integrate we get
\[
\int_0^1 \left| \frac{f'(t) + g'(t)}{2} \right|^p \, dt \leq \frac{1}{2} \int_0^1 \left( |f'(t)|^p + |g'(t)|^p \right) = |b - a|^p.
\]
And since the integrand on the left-hand side is smaller than the integrand on the right except when \( f'(t) = g'(t) \) almost everywhere. We see that we can only have equality when \( f' = g' \) almost everywhere. As \( f \) and \( g \) are absolutely continuous and have the same boundary values we conlude that \( f = g \) and thus the solution is unique. \( \square \)

This also shows that the \( p \)-energy is only dependant on the values at the nodes since we have
\[
\sum_{n_i \sim n_j} \int_0^1 |f'_{n_i,n_j}(t)|^p \, dt \geq \sum_{n_i \sim n_j} |u(n_j) - u(n_i)|^p.
\]
1.4. Another derivation of the problem

So since we know that the functions will all be linear between the nodes, the function $u$ uniquely defines the $p$-harmonic function, thus giving us Definition 1.1.
Chapter 2

Method

The method studied in this thesis uses an iterative approach to solving the Dirichlet problem on graphs, as defined in Definition 1.1. The goal is to find a function $u$ over the graph’s nodes that minimizes the sum

$$\sum_{n_i \sim n_j} |u(n_i) - u(n_j)|^p = 0$$

or equivalently, by Theorem 1.8, satisfies

$$\Delta_p u(n_i) = \sum_{j: n_j \sim n_i} |u(n_i) - u(n_j)|^{p-2}(u(n_i) - u(n_j)) = 0 \quad \text{for all } n_i \in (N \setminus \partial N)$$

with $u(n_i) = v(n_i)$ for $n_i \in \partial N$ for some specified function $v$.

To simplify things we will instead look at the problem as finding a vector $\hat{N}$ where each element corresponds to the value of $u$ at a specific node $n_i$ in the graph $N_i = u(n_i)$. This is just to make the notation more convenient and easier to read.

With this notation we can rewrite the problem as finding a vector $N = (N_1, N_2, \ldots, N_k)$ such that

$$\sum_{j: n_j \sim n_i} |N_i - N_j|^{p-2}(N_i - N_j) = 0 \quad \text{for all interior nodes } n_i, \quad (2.1)$$

given a vector $\hat{N} = (N_{k+1}, N_{k+2}, \ldots, N_n)$ of boundary values.

Notice that the equation determining the value for a single node requires knowledge of the value at every neighbouring node, so all the equations will have to be solved simultaneously. Now we could proceed by viewing this as a non-linear equation in $\mathbb{R}^n$ and use some general method for solving non-linear equations numerically, such as Newton’s method or similar. We will however try a different, more specialized approach.

Informally our method is based on the idea of solving each of the equations in the system, one at a time, while (naively) assuming that all the other nodes already have their correct values. Then we iterate this process, looping through all the nodes, until the changes performed are within some given tolerance.
Definition 2.1. We define the local update function for each interior node \( n_i \), called \( F^p_i \), as

\[
F^p_i : \mathbb{R}^n \rightarrow \mathbb{R}^n, \quad (F^p_i(N))_j = \begin{cases} N_i & \text{if } j \neq i, \\ N_{\text{solve}} & \text{if } j = i, \end{cases}
\]

where

\[
\sum_{j: n_j \sim n_i} |N_{\text{solve}} - N_j|^p - 2(N_{\text{solve}} - N_j) = 0.
\]

In other words, \( F^p_i \) is a function that changes the value at node \( n_i \) to the solution of its local function, which we for now assume we can calculate exactly, and leaves all other nodes unchanged. We will refer to these as the local update functions.

Note that when \( p = 2 \) then \( N_{\text{solve}} \) is just the average value of the nodes adjacent to \( N_i \).

In the case \( p = \infty \) we define \( N_{\text{solve}} = (N_{\text{max}} + N_{\text{min}})/2 \), where \( N_{\text{max}} \) and \( N_{\text{min}} \) are the largest and smallest values at \( n_i \)’s neighbours, respectively.

If we perform this once for every interior node \( n_i \) we get what we will call the global update function.

Definition 2.2. The global update function \( F^p \) is the composition of all the local update functions,

\[
F^p = F^p_k \circ F^p_{k-1} \circ \cdots \circ F^p_2 \circ F^p_1.
\]

When we prove the convergence of the method we will need the following.

Theorem 2.3. \( F^p_i \) is continuous and so is \( F^p \).

Proof. We first define

\[
P : \mathbb{R} \times \mathbb{R}^n \rightarrow \mathbb{R}, \quad P(x, N) = \sum_{i=1}^{n} |x - N_i|^p - 2(x - N_i)
\]

and note that if we fix \( N \), then \( P \) is a strictly increasing, continuous function from \( \mathbb{R} \) to \( \mathbb{R} \) so it has an inverse and we can define a new function \( Q \) from \( \mathbb{R}^n \) to \( \mathbb{R} \) by

\[
Q(N) = x, \quad \text{when } P(x, N) = 0.
\]

Lemma 2.4. The function \( Q \) is continuous.

Proof. We pick an \( N \in \mathbb{R}^n \) and choose an \( x \) such that \( P(x, N) = 0 \) and thus \( Q(N) = x \). We now pick an \( \epsilon > 0 \) and an \( M \) such that \( |M_i - N_i| < \epsilon \) or equivalently \( M_i - \epsilon < N_i < M_i + \epsilon \) and \( N_i - \epsilon < M_i < N_i + \epsilon \) for all \( i \).

If we put this \( M \) and \( x + \epsilon \) into \( P \) we get

\[
P(x + \epsilon, M) = \sum_{i=1}^{n} |x + \epsilon - M_i|^p - 2(x + \epsilon - M_i) = \sum_{i=1}^{n} |x - (M_i - \epsilon)|^p - 2(x - (M_i - \epsilon)).
\]

\(^{1}\text{In practice it will of course be solved approximately by some one-dimensional numerical method.}\)
Since \( f(x) = |x|^{p-2}x \) is a strictly increasing function we have
\[
\sum_{i=1}^{n} |x - (M_i - \epsilon)|^{p-2}(x - (M_i - \epsilon)) > \sum_{i=1}^{n} |x - N_i|^{p-2}(x - N_i) = P(x, N) = 0.
\]
So \( P(x + \epsilon, M) \geq 0 \) and by putting in \( x - \epsilon \) instead of \( x + \epsilon \) we can show in the same way that \( P(x - \epsilon, M) \leq 0 \).

Since \( P \) is continuous there exists an \( x_0 \) such that \( P(x_0, M) = 0 \) and
\[
x < x_0 < x + \epsilon, \text{ or in other words}
\]
\[
Q(N) - \epsilon < Q(M) < Q(N) + \epsilon
\]
which of course implies that \( |Q(M) - Q(N)| < \epsilon \) and therefore \( Q \) is uniformly continuous. \(^2\)

We now continue the proof of Theorem 2.3. If we look at the definition of \( F_i^p \) we see that \( (F_i^p(N))_i = Q(\bar{N}) \), where \( \bar{N} \) are the neighbours of \( N_i \).

If we now, similar to the proof of the lemma, form the vector \( M \) such that
\[
|M_i - N_i| < \epsilon \quad \text{for all} \quad i,
\]
we get
\[
|F_i^p(M) - F_i^p(N)| \leq \sum_{k=1}^{n} |(F_i^p(M))_k - (F_i^p(N))_k|
\]
by the triangle inequality. The terms \( |(F_i^p(M))_k - (F_i^p(N))_k| = |M_k - N_k| < \epsilon \) when \( k \neq i \) and when \( i = k \), \( |(F_i^p(M))_k - (F_i^p(N))_k| = |Q(M) - Q(\bar{N})| < \epsilon \) by the proof of Lemma 2.4. So by adding these we get
\[
|F_i^p(M) - F_i^p(N)| \leq n\epsilon
\]
and therefore \( F_i^p \) is uniformly continuous and since \( F^p \) is a composition of continuous function, it too is uniformly continuous. \( \square \)

Now it is straightforward to define a sequence of values at internal nodes \( \{\bar{N}^k\}_{k=0}^{\infty} \) by
\[
\bar{N}^0 = \text{any element in} \; S,
\]
\[
\bar{N}^k = F^p(\bar{N}^{k-1}).
\]

It is our hope that this sequence will approach the solution.

### 2.1 Convergence of the method

In this chapter we will prove that the sequence \( \{\bar{N}^k\}_{k=0}^{\infty} \) actually approaches the solution. To do this we will use the minimizing definition of the problem instead of the local equations. The function we wish to minimize is the \( p \)-energy function, defined as
\[
e(N) = \sum_{n_i \sim n_j} |N_i - N_j|^p. \tag{2.2}
\]

We want to prove that the sequence \( \{\bar{N}^k\}_{k=0}^{\infty} \) approaches a value \( \bar{N} \) that minimizes the \( p \)-energy. We will do this by proving that

\(^2\)It is even Lipschitz continuous, which is a stronger condition than continuity.
• The $p$-energy of the sequence values converges.
• The sequence has one or more convergent subsequences.
• Any limit to one of those subsequences must be $p$-harmonic and thus it must be the unique solution.

To show that the $p$-energy converges we will first show that the $p$-energy decreases each iteration.

It is enough to show that $F_p$ decreases the $p$-energy. For then $F^p$ is a composition of $p$-energy-decreasing functions, it must then also decrease the $p$-energy.

**Theorem 2.5.** For each local update function we have

$$e(F^p_i(X)) \leq e(X) \text{ for all } X.$$ 

**Proof.** We have

$$\frac{\partial e(N)}{\partial N_i} = p \sum_{j:n_j \sim n_i} |N_i - N_j|^{p-2}(N_i - N_j), \quad (2.3)$$

$$\frac{\partial^2 e(N)}{\partial N_i^2} = p(p-1) \sum_{j:n_j \sim n_i} |N_i - N_j|^{p-2}. \quad (2.4)$$

From Definition 2.1 it is clear that $\frac{\partial e(F^p_i(N))}{\partial N_i} = 0$, since it is defined as changing the value of $N_i$ so the partial derivative is 0, and since $\frac{\partial^2 e(N)}{\partial N_i^2}$ is a sum of absolute values it is positive. Since we have only changed the value at node $N_i$ we can conclude that the $p$-energy is decreased by $F^p_i$ and therefore also by $F^p$. \qed

It is obvious that the $p$-energy can never be negative, since the $p$-energy is calculated as a sum of absolute values, so 0 is a lower bound for the $p$-energy $\{e(\tilde{N}_k)\}_{k=0}^\infty$.

So the $p$-energy sequence is decreasing and has a lower bound. It follows that the $p$-energy of the sequence has a limit which we will call $\tilde{e}$.

This proves that the energy of the sequence converges, but not that the sequence itself converges.

**Lemma 2.6.** If $X \in S$ then $F^p_i(X) \in S$.

**Proof.** For each local update function $F^p_i$ we note that

$$\min \tilde{X}_i \leq (F^p_i(N))_i \leq \max \tilde{N}_i,$$

where $\tilde{N}_i$ are the neighbours of the node $N_i$. To see this, note that in the local equation

$$\sum_{j:n_j \sim n_i} |N_j - N_i|^{p-2}(N_j - N_i) = 0$$

each term will become negative if

$$\min(\tilde{N}_i) > (F^p_i(N))_i$$
2.1. Convergence of the method

and positive if

\[(F^p_i(N_i))^i > \max(N_i)\]

so therefore the sum cannot be zero. It follows from the definition of \(S\) that \(F^p(N) \in S\) if \(N \in S\) since no node has neighbours smaller than \(\min(\partial N)\) or larger than \(\max(\partial N)\).

By Lemma 2.6 and the fact that \(\bar{N}^0 \in S\) we see that \(\{\bar{N}^i \mid i \geq 0\} \subset S\). And since \(S\) is compact we know, by the Bolzano–Weierstrass theorem, that there exists a convergent subsequence \((\bar{N}^i)_j^{\infty} \rightarrow \bar{N}\). And because we know that the \(p\)-energy converges we must have \(e(\bar{N}) = \bar{e}\).

**Theorem 2.7.** The limit, \(\bar{N}\) of any convergent subsequence to \(\{\bar{N}^i\}_{i=0}^{\infty}\) is \(p\)-harmonic.

**Proof.** Assume that \(\bar{N}\) is not \(p\)-harmonic, then \(\bar{N}\) is not optimal in at least one node and therefore the value will be changed and the \(p\)-energy decreased by \(F^p_i\), due to Theorem 2.5,

\[
F^p(\bar{N}) \neq \bar{N} \tag{2.5}
\]

\[
e(\bar{N}) < \bar{e}. \tag{2.6}
\]

Since \(F^p\) is continuous by Theorem 2.3 we see that

\[
\bar{N}^i \rightarrow \bar{N}
\]

implies

\[
F^p(\bar{N}^i) \rightarrow F^p(\bar{N}).
\]

But if we calculate the \(p\)-energy of this, using the fact that \(e\) is continuous, we get

\[
e(\bar{N}^i) \rightarrow e(\bar{N}).
\]

so

\[
e(\bar{N}^i + 1) \rightarrow e(\bar{N}) < e(\bar{N}).
\]

This is a contradiction since this gives us a new subsequence whose \(p\)-energy converges to a lower value than \(\bar{e}\). The assumption must therefore be false, and \(\bar{N}\) is \(p\)-harmonic.

This shows that all convergent subsequences converges to the solution of the Dirichlet problem. Since we know that the energy converges, this shows that the limit is also the optimal \(p\)-energy.

We now assume that the sequence \(\{\bar{N}^k\}_{k=1}^{\infty}\) does not converge to any element. Then, for some \(\epsilon > 0\) there is no \(n\) such that

\[
|\bar{N}^i - \bar{N}| < \epsilon, \text{ when } i > n.
\]

This means that there is an infinite number of elements with \(|\bar{N}^i - \bar{N}| > \epsilon\) since if they were finite, we could pick an \(n\) after the last of them. But then, since we know that \(N^i \in S\), which is compact, there must exist a point \(\bar{N}\) with \(|\bar{N} - \bar{N}| > \epsilon\) such that there is an infinite number of elements with \(|N - \bar{N}| < \epsilon\). Therefore there exists a convergent subsequence that tends to \(\bar{N}\), but this is a contradiction since we know that all convergent subsequences converge to the unique solution of the Dirichlet problem, the assumption is therefore false and we know that the sequence converges.
Chapter 3

Implementation details

As a part of my thesis I wrote a computer program that implements the described algorithm, as well as presents the user with a simple GUI (Graphical User Interface) allowing simple graph display and construction.

The programming language used is Sun Java version 1.5.0 [1]. The choice of programming language was based on three criteria.

- Familiarity, I did not have to spend time learning a new language.

- Portability, not having to recompile the program on different platforms is a big plus.

- It has a good, ubiquitous, standard GUI-library.

If performance had been the top priority of the implementation, the choice of language had probably been different, but for this project, ease of implementation was more important and learning a new programming language was not part of the goals for this thesis.

Much of the program is just dealing with the GUI\(^1\) and not very interesting to describe in detail here. The program internally represents both nodes and edges as Objects and a graph is simply a list of Nodes and a list of Edges. Each node stores two values, its actual value and its value rescaled to a value in the interval \([0, 1]\), such that the largest value of the boundary nodes maps to 1 and the smallest maps to 0. This is to reduce numerical problems due to too large numbers for the computer when we calculate \(p\)th powers while solving the local equations for large \(p\).

The main logic of the program is in the two Solver classes SimpleSolver and InfSolver, where InfSolver is used for the special case \(p = \infty\) and SimpleSolver is used in all other cases. Both classes implements a solve function that iterates through the nodes until the maximal (rescaled) change in any node is less than \(10^{-13}\) at which point it halts.

When the program calculates the roots of the local equations it simply uses the interval halving method. This is because Newton–Raphson’s method, which I tried to use first, did not always converge. It is possible that some other method, such as the secant method might be faster.

A listing of the most relevant parts of the program is included as Appendix A.

\(^1\)Graphical user interface. Written using Sun’s Swing toolkit.
Chapter 4

Speed of convergence

The number of iterations needed to get a good approximation is mostly dependent on the value of $p$ and the number of interior nodes. In general, the closer $p$ gets to 1 the more iterations are needed. We will look at two examples and the number of iterations needed for different values of $p$ with the same starting values.

Example 4.1. The first graph is a very small example with two boundary nodes and three interior nodes and looks like in Figure 4.1.\(^1\)

![Figure 4.1: Graph for Example 4.1.](image)

<table>
<thead>
<tr>
<th>Number of iterations</th>
</tr>
</thead>
<tbody>
<tr>
<td>$p$</td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>

As we can see, the number of iterations are more or less the same for $p > 2$ but starts to grow rapidly as $p$ decreases.

Example 4.2. The second example, Figure 4.2, is slightly larger, with nine interior nodes, constructed mostly randomly.

<table>
<thead>
<tr>
<th>Number of iterations</th>
</tr>
</thead>
<tbody>
<tr>
<td>$p$</td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>

\(^1\)The graphics is taken from the program, the nodes with circles around them are boundary nodes, the numbers next to the boundary nodes are boundary values and the numbers next to the interior nodes are the starting values.
Once again it is apparent that small values of $p$ require very large number of iterations, when I tried to solve with $p = 1.2$ it did not finish in several hours.

The reason that the convergence is so much slower for smaller values of $p$ is that the solution of the local equation is much more volatile and changes more when the neighbouring nodes change. For example, we can look at a node with three neighbours, two having values $N_1 = 0$ and $N_2 = 10$ and we let the third one’s value vary between $N_3 = 3, 4, 5$, and look at how the graph of the local equation changes when the value of $N_3$ changes. We will look at three cases, with $p = 4$, $p = 2$ and $p = 1.2$, see Figures 4.3, 4.4 and 4.5.

We can see that the root of the function stays almost the same when $p = 4$ but changes quite a bit when $p = 1.2$. This suggests that the reason why the convergence is so slow is that whenever we do a local update, the $p$-energy in the adjacent nodes changes too much and we get a ‘ripple effect’ since we then have to update all the neighbours, which then change the $p$-energy of their neighbours and so on.
Figure 4.3: Local function for $p = 4$.

Figure 4.4: Local function for $p = 2$. 
Figure 4.5: Local function for $p = 1.2$. 


Chapter 5
Conclusions

The studied iteration method for finding $p$-harmonic functions on graphs does always converge, however the speed of convergence is very slow when $p$ is close to 1 and in those cases some other method should probably be considered. For larger $p$’s the rate of convergence is pretty good and the method appears to be a viable algorithm.

The main advantage over traditional methods such as *Newton–Raphson’s method* is the method’s simplicity, instead of doing one pretty complex operation each iteration we do several simple, almost trivial, operations.

5.1 Further studies

There are several aspects of the problem that I did not have time to explore as much as they might deserve. First of all it would be interesting to get some better way of predicting the number of iterations needed, it is clear that it is heavily dependant on the value of $p$, but I was unable to determine some more exact bounds.

In the program I used interval halving to find the root of the local equations, this was because Newton–Raphson’s method did not always converge. It is known that a fixed point iteration $x = \phi(x)$ converges if $|\phi'(x)| \leq m < 1$, which in our case simplifies to

$$\left| \frac{f(x)f''(x)}{f'(x)f'(x)} \right| \leq m < 1,$$

where $f$ is the local equation for the current node. This inequality is however very complex and I couldn’t find a simple, useful criterion for when it was possible to use Newton–Raphson’s method. There are of course other methods that could work but were not investigated.

The prospect of parallelizing the algorithm to run on multicore processors is promising, but has not been tried in practice, mostly because I did not have easy access to a multicore computer.
Chapter 6

Appendix A

The code presented here is not the complete program, only the classes related to actually finding the solution are included. Methods and fields in these classes related to the display and manipulation of the model have also been excluded from the listings.

The abstract class Solver represents different ways of finding a solution. the implementing classes, InfSolver and SimpleSolver implement the missing function findValue.

Listing 6.1: The Solver classes.

```java
public abstract class Solver {
    private Collection<Node> nodes;
    private Collection<Edge> edges;

    // Tolerance for solving the local equations.
    public static final double LOCAL_TOLERANCE = 1.0e-13;
    // Minimal change in any node before we are done.
    public static final double GLOBAL_TOLERANCE = 1.0e-13;

    public void solve() throws IsolatedNodeException {
        String output = " ";
        double diff = 100;
        int i = 0;
        while (diff > GLOBAL_TOLERANCE) {
            diff = iterateOnce();
            i++;
            if (i % 10 == 0) {
                output = "Iterations: " + i + " diff: " + diff;
                System.out.println(output);
            }
        }
        System.out.print("Total number of Iterations: " + i);
    }

    private double iterateOnce() {
        // Implement the algorithm here.
    }
}
```

Andersson, 2009. 27
```java
throws IsolatedNodeException {
    LinkedList<Double> ns = new LinkedList<Double>();
    double diff = 0;
    for (Node n : this.nodes) {
        // Abort if it's a boundary node
        if (!n.isBoundary()) {
            // Find all adjacent nodes
            for (Edge e : this.edges) {
                if (e.getNode1() == n) {
                    ns.add(e.getNode2().getScaledValue());
                } else if (e.getNode2() == n) {
                    ns.add(e.getNode1().getScaledValue());
                }
            }
            try {
                double newValue = findValue(ns);
                diff = Math.max(diff, Math.abs(newValue - n.getScaledValue()));
                n.setScaledValue(newValue);
                ns.clear();
            } catch (NoSuchElementException e) {
                throw (new IsolatedNodeException(n));
            }
        }
    }
    return diff;
}

abstract double findValue(LinkedList<Double> values)
    throws NoSuchElementException;
}

public class SimpleSolver extends Solver {

    private double p;

    /**
     * Finds the value for a node given a list of
     * the values at its neighbours
     */
    @Override
    double findValue(LinkedList<Double> values)
        throws NoSuchElementException {
        double min = values.getFirst();
        double max = values.getFirst();
        int count = 0;
        // Finds the max and min value of the neighbours
        for (double d : values) {
            if (d > max) {
                max = d;
            } else if (d < min) {
                min = d;
            }
            count++;
        }
        return (max + min) / count;
    }
```
double middle = (max + min) / 2;
// Interval halving method
while ((max - min) > LOCAL_TOLERANCE) {
    double valueAtMiddle = calcEnergy(values, middle);
    if (valueAtMiddle <= 0) {
        min = middle;
    } else {
        max = middle;
    }
    middle = (max + min) / 2;
    count++;
} return middle;
}

double calcEnergy(LinkedList<Double> vs, double c) {
    double sum = 0;
    for (double v : vs) {
        sum += (c - v) * Math.pow(Math.abs(c - v), p - 2);
    }
    return sum;
}

public class InfSolver extends Solver {
    double findValue(LinkedList<Double> values) {
        double min = values.getFirst();
        double max = values.getFirst();
        // Finds the max and min value of the neighbours
        for (double d : values) {
            if (d > max) {
                max = d;
            }
            if (d < min) {
                min = d;
            }
        }
        return ((max + min) / 2);
    }
}

The Node and Edge classes are very simple, the only thing to note is the scaling factors used to avoid numerical errors.
Listing 6.2: The Node and Edge classes.

```java
public class Node {
    /** The actual value currently at the node */
    private double value;
    /** The value scaled to a value between 0 and 1 */
    private double scaledValue;
    /**
     * The relationship between value and scaledValue.
     * It should always hold that
     * scaledValue = value * a - b
     **/
    private double a;
    private double b;

    private boolean boundary;

    public double getValue() {
        return this.value;
    }

    public double getScaledValue() {
        return this.scaledValue;
    }

    public void setValue(double newValue) {
        this.value = newValue;
    }
    // Calculates the scaled value.
    public void scaleValue(double newA, double newB) {
        this.scaledValue = this.value * newA - newB;
        this.a = newA;
        this.b = newB;
    }

    public void setScaledValue(double newValue) {
        this.scaledValue = newValue;
        this.value = (newValue + b) / a;
    }

    public boolean isBoundary() {
        return this.boundary;
    }

    public void setBoundary(boolean b) {
        this.boundry = b;
    }
}
```

```java
public class Edge {
```

Chapter 6. Appendix A
private Node node1, node2;

public Edge(Node n1, Node n2) {
    this.node1 = n1;
    this.node2 = n2;
}

public Node getNode1() {
    return this.node1;
}

g

public Node getNode2() {
    return this.node2;
}
Bibliography


Copyright

The publishers will keep this document online on the Internet - or its possible replacement - for a period of 25 years from the date of publication barring exceptional circumstances. The online availability of the document implies a permanent permission for anyone to read, to download, to print out single copies for your own use and to use it unchanged for any non-commercial research and educational purpose. Subsequent transfers of copyright cannot revoke this permission. All other uses of the document are conditional on the consent of the copyright owner. The publisher has taken technical and administrative measures to assure authenticity, security and accessibility. According to intellectual property law the author has the right to be mentioned when his/her work is accessed as described above and to be protected against infringement. For additional information about the Linköping University Electronic Press and its procedures for publication and for assurance of document integrity, please refer to its WWW home page: http://www.ep.liu.se/

Upphovsrätt

Detta dokument hålls tillgängligt på Internet - eller dess framtida ersättare - under 25 år från publiceringsdatum under förutsättning att inga extraordinära omständigheter uppstår. Tillgång till dokumentet innebär tillstånd för var och en att läsa, ladda ner, skriva ut enstaka kopior för enskilt bruk och att använda det oförändrat för ickekommersiell forskning och för undervisning. överföring av upphovsrätten vid en senare tidpunkt kan inte upphäva detta tillstånd. All annan användning av dokumentet kräver upphovsmannens medgivande. För att garantera äktheten, säkerheten och tillgängligheten finns det lösningar av teknisk och administrativ art. Upphovsmannens ideella rätt innefattar rätt att bli nämnad som upphovsman i den omfattning som god sed kräver vid användning av dokumentet på ovan beskrivna sätt samt skydd mot att dokumentet ändras eller presenteras i sådan form eller i sådant sammanhang som är krävande för upphovsmannens litterära eller konstnärliga anseende eller egennart. För ytterligare information om Linköping University Electronic Press se förlagets hemsida http://www.ep.liu.se/

© 2009, Karl Tomas Andersson