CaWoF

Reference Manual

10th January 2017

Contents

Chapter 1

Introduction

1.1 State-of-Art

The (binary) generic decoding problem is a interesting subject in Coding Theory because, for example, it is very usefull for decoding random codes. But, in Code-Based Cryptography is a vital issue, because the security of cryptosystem repose in the fact the generic decoding is a hard problem, so better resolving algorithm implies that these cryptosystem become less secure (independently of the code the cryptosystem was based). Therefore, for a designer code-based cryptosystem is very important to know which is the exact mesure of the security of his cryptosystem against this kind of direct attack, with this reason in mind, we provide an simple program thats calculates the complexity of some very important generic decoding algorithms.

One of the first solving algorithms was made by Prange in 1962, who introduced the notion of *information set* and gave a probabilistic solution when the chosen information set is the right one. A remarkable improvement of this algorithm was by Stern in 1989, this algorithm subdivides the decoding problem in to two parts: in betting the information set and in solving a smaller decoding subproblem (another similar version of Dumer's algorithm in 1991).

Recently, we had improvements done by May, Meurer and Thomae in 2011, which uses the previous scheme and use the technique of representations (by Joux and Becker) to solve the decoding subproblem. The next improvement was done Becker, Joux, May and Meurer in 2014; this algorithm improves against the resolution of the decoding subproblem. The last improvement was due by May et Ozerov in 2015, their algorithm has a better technique to match the produced lists of the previous algorithm.

All these algorithms compete for the smallest **Work Factor**, this is the asymptotic expression of their optimal complexity. The common form of Work Factor is $2^{c \cdot n}$, where n is the length of code and c is a constant that depends only of the code rate, error rate and the algorithm. Therefore, the evolution of this algorithm consist in reducing the constant c for a set of values of code rate and error rate, normally we examine the case where error rate is smaller of Gilbert-Varshamov's bound. The program calculates the constant c for the mentioned algorithms.

1.2 How to use

CaWoF uses **GSL** (GNU Scientific library) to minimize functions and solving equations in its calculations. Some Linux distributions containts precompilated binary packages of GSL or it could be from the GNU website.

After the installation of GSL, we only must unzip the file CaWoF.zip and use the command `MAKE` in the directory `\CaWoF`. You can find the executable `cawof` in the directory `\src`.

This program calculates the constant c in the asymptotic expresion of the Work Factor for an algorithm and a code rate. For example, a simple execution (in command line) is

```
..\CaWoF\src$ ./cawof -a BJMM -k 0.5
The work factor of BJMM's algorithm is assymptotically 2^(0.0999852060n),
when the code rate is 0.5000000000 and error rate is 0.1100278644
```

The default error rate is the Gilbert-Varshamov's bound (respect the given code rate), but the option `-w` let us choice any value between 0 and 1. For example,

```
..\CaWoF\src$ ./cawof -w 0.5 -a MMT -k 0.5
The work factor of MMT's algorithm is assymptotically 2^(0.6225562489n),
when the code rate is 0.5000000000 and error rate is 0.5000000000
```

However, error rate values greater than $1-k$ will not be analyzed. But, we can find polynomial behaviour in error rate equals to this bound.

Finally, there are more options and we invite you to discover them by the option `-h`

```
..\CaWoF\src$ ./cawof -h
```

1.3 Acknowledgement

This program could be done (at this point) without the help, orientation and emphasis of my PhD. advisor Nicolas Sendrier. And, I cannot forget to thank the ethernal good mood in my project-team SECRET at INRIA-Paris.

Chapter 2

File Index

2.1 File List

Here is a list of all files with brief descriptions:

Chapter 3

Theorical Aspects

In this chapter, we will describe briefly about the formulas to obtain the work factor of the presented algorithms: Prange, Stern, Dumer, MMT, BJMM and Nearest Neighbors (NN). All of these algorithm are ISD algorithms, so they will have a *sucess probability* $\mathscr{P}$[1], so we will repeat the principal instructions of these algorithms in a loop a number of times $\mathscr{P}^{-1}$. With the exception of Prange, this principal instructions are just the instruction to solve a decoding subproblem. Therefore we calculate the workfactor with a formula like that:

$$\mathrm{WF}_{\mathscr{A}} = \mathscr{P}^{-1}\mathscr{I}$$

where $\mathscr{I}$ is the number of operations made inside the principal loop. For almost all cases, $\mathscr{I}$ will be the number of operations made by the subroutine which solves the decoding subproblem.

3.1 Prange's algorithm

We avoid the polynomial terms in the Prange's number of operations, so we will have $\mathscr{I} = 1$ and the succes probability will be $\mathscr{P} = \binom{n-k}{w} / \binom{n}{w}$. Therefore

$$\mathrm{WF}_{Prange} = \frac{\binom{n}{w}}{\binom{n-k}{w}}.$$

3.2 Stern's algorithm

In this algorithm and the next ones, p will be the parameter to describes the weight of error in the decoding subproblem and l means the length of syndrome. So the succes probability will be $\mathscr{P} = \binom{n-k-l}{w-p}\binom{k}{p} / \binom{n}{w}$. We include the number of operations and we obtain

[1] This is the probability that the chosen Information Set was the right one

$$\mathrm{WF}_{Stern} = \frac{\binom{n}{w}}{\binom{n-k-l}{w-p}\binom{k}{p}}\left(\sqrt{\binom{k}{p}} + \frac{\binom{k}{p}}{2^l}\right).$$

3.3 Dumer's algorithm

From this ISD algorithm, the sucess probability will not change and it will be $\mathscr{P} = \binom{n-k-l}{w-p}\binom{k+l}{p}/\binom{n}{w}$. Therefore the workfactor will be

$$\mathrm{WF}_{Dumer} = \frac{\binom{n}{w}}{\binom{n-k-l}{w-p}\binom{k+l}{p}}\left(\sqrt{\binom{k+l}{p}} + \frac{\binom{k+l}{p}}{2^l}\right)$$

3.4 MMT Algorithm

The expression of Work Factor in its published article was

$$\mathrm{WF}_{MMT} = \frac{\binom{n}{w}}{\binom{n-k-l}{w-p}\binom{k+l}{p}}\left(\sqrt{\binom{k+l}{p/2}} + \frac{\binom{k+l}{p/2}}{2^{l_2}} + \frac{\binom{k+l}{p/2}^2}{2^{l+l_2}}\right).$$

But, we use simplify this expression by reducing an optimal case when $l_2 = p$ (the number of representations). This choice comes from the fact the number of total must divided by the number of representations, in this way be obtain no repeated solutions to be tested. Therefore, we use this reduced and eficiant version

$$\mathrm{WF}_{MMT} = \frac{\binom{n}{w}}{\binom{n-k-l}{w-p}\binom{k+l}{p}}\left(\sqrt{\binom{k+l}{p/2}} + \frac{\binom{k+l}{p/2}}{2^{p}} + \frac{\binom{k+l}{p/2}^2}{2^{l+p}}\right).$$

3.5 BJMM Algorithm

In this algorithm, we have $p_1, p2$ as the parameter of the lowers step of BJMM's algorithm. We denote the probability μ_2 that two words of length $k+l$ with weight p_2 match into a word of weight p_1, respectly $\mu_1$and μ_2. These probalities satisfy the relation

$$\binom{k+l}{p_1}R_2 = \mu_2\binom{k+l}{p_2} \qquad \text{and} \qquad \binom{k+l}{p}R_1 = \mu_1\binom{k+l}{p_1},$$

where R_2 is the number of representations of a word of lenght $k+l$ and weight p_1 comes from a match of two words of weight p_2, respectly R_1. Therefore, we have the final expresion

$$\mathrm{WF}_{BJMM} = \frac{\binom{n}{w}}{\binom{n-k-l}{w-p}}\left(\frac{\sqrt{\binom{k+l}{p_2}}}{\binom{k+l}{p}} + \frac{\binom{k+l}{p_1}}{\mu_2\binom{k+l}{p_2}\binom{k+l}{p}} + \frac{1}{\mu_2\mu_1\binom{k+l}{p_1}} + \frac{1}{\mu_1 2^l}\right).$$

3.6 NN Algorithm

We use exactly the same expression than in the published article

$$\mathrm{WF}_{BJMM} = \frac{\binom{n}{w}}{\binom{n-k-l}{w-p}\binom{k+l}{p}}\left(\sqrt{\binom{k+l}{p_2}} + \frac{\binom{k+l}{p_2}}{2^{l_2}} + \frac{\binom{k+l}{p_2}^2}{2^{l+l_2}} + 2^{\mu} + 2^{y(1-k-l)}\right).$$

where

$$\mu = \binom{k+l}{p_1}/2^l \qquad \text{and} \qquad y = (1-\gamma)\Big(1 - h\big(h^{-1}(1-\frac{\mu}{1-k-l}) - \frac{\gamma}{2}\big)/(1-\gamma)\Big)$$

with $\gamma = \frac{1-k-l}{w-p}$ and h as binary entropy function.

3.7 When there are more than one solution

This special case is reduced just to change the probability $\mathscr{P}$ by the general one

$$\mathscr{P}^* = \max\{1, \mathscr{N} * \mathscr{P}\},$$

here $\mathscr{N}$ is the number of solutions and $\mathscr{P}$ is the respective probability of the algorithm in the standart case. If the weight in the decoding problem is bigger than the Gilbert-Vershamov bound, then the number of solution could be calculated by $\binom{w}{n}/2^{n-k}$.

3.8 Implementation of formulas

To describe these work factors, we need some notations and conventions. We denote as n the lenght of code and we will describe all the others values as quotient by n, for example k will be denotes the *code rate* and w its *error rate*. In the same way, we represent the values for the parameters of an algorithm by a quotient of its value (in the algorithm) by n. For example, p will be the quotient between the weight of the error to find in the subroutine of the algorithm and n. We do the same for the parameters l, p_1 and p_2.

Chapter 4

Data Structure Documentation

4.1 wf_params Struct Reference

Parameters of the decoding algorithms.

`#include <entropy_tools.h>`

Data Fields

- double k
- double w
- double N
- double l
- double p
- double p1
- double p2
- double a
- double wf

4.1.1 Detailed Description

Parameters of the decoding algorithms.

One variable of this type will be use in any function about work factor to transfer the parameters of decoding algorithms. After the optimization of a function this struct saves the optimal parameters to achieve the Work Factor.

Definition at line 50 of file entropy_tools.h.

4.1.2 Field Documentation

double **k**

This is the code rate with values between 0 and 1. Its value is 0.5 by default.

double **w**

This is the error rate with values between 0 and 1. Its value is the Gilbert-Varshamov bound $h^{-1}(1-k)$ by default.

double **N**

This is the binary logarithm of number of solutions of the decoding problem. Its value is 0 by default.

double **l**

This is quotient of the parameter l with length code in the ISD algorithms with exception to Prange's algorithm.

double **p**

This is quotient of the parameter p with length code in the ISD algorithms with exception to Prange's algorithm.

double **p1**

This is quotient of the parameter p1 with length code in the BJMM and NN algorithms.

double **p2**

This is quotient of the parameter p1 with length code in the BJMM and NN algorithms.

double **a**

Theorical parameter between 0 and 1 for the reduced bound function B_a.

double **wf**

Exponential coefficient c in the expression 2^{cn} of Work Factor of an algorithm (here n is the length code).

The documentation for this struct was generated from the following file:

- include/entropy_tools.h

Chapter 5

File Documentation

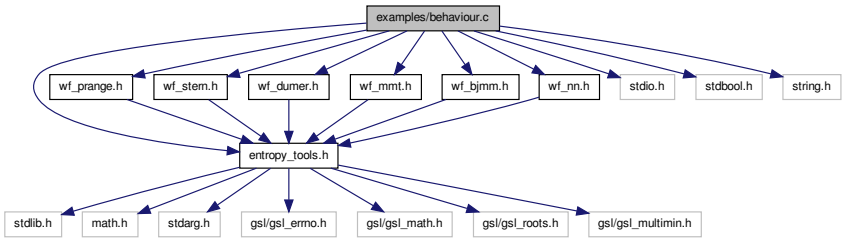
5.1 examples/behaviour.c

Example program, it shows wf/w vs w.

Dependencies

```
#include <entropy_tools.h>
#include <wf_prange.h>
#include <wf_stern.h>
#include <wf_dumer.h>
#include <wf_mmt.h>
#include <wf_bjmm.h>
#include <wf_nn.h>
#include <stdio.h>
#include <stdbool.h>
#include <string.h>
```

Include dependency graph for behaviour.c:



Functions

- int main (int argc, char ∗argv[])
- void usage (void)

Variables

- FILE ∗ streamout
- wf_params wfp
- double(∗ wf_algo)(wf_params ∗)
- char algo [20]

5.1.1 Detailed Description

Example program, it shows wf/w vs w. For a code rate k, this program calculates wf/w respects the error rate w (with values between 0 and Gilbert-Varshamov bound). We can see the wf/w becomes near to expect value -log_2(1-k) when w approachs 0.

Definition in file behaviour.c.

5.1.2 Function Documentation

5.1.2.1 int main (int *argc,* char ∗ *argv[]*)

It shows wf/w vs w. Definition at line 63 of file behaviour.c.

5.1.2.2 void usage (void)

It prints help.

Definition at line 54 of file behaviour.c.

5.1.3 Variable Documentation

5.1.3.1 char algo[20]

This variable contains the name of algorithm to use.

Definition at line 46 of file behaviour.c.

5.1.3.2 FILE∗ streamout

This variable will saves in a file the obtained values after optimization of wf.

Definition at line 43 of file behaviour.c.

5.1.3.3 double(∗ wf_algo)(wf_params ∗)

This variable saves the adress of the optimization function of work factor.

Definition at line 45 of file behaviour.c.

5.1.3.4 wf_params wfp

This variable contains the value of k, w, wf/w, l, p, p1, p2 after optimization of wf.

Definition at line 44 of file behaviour.c.

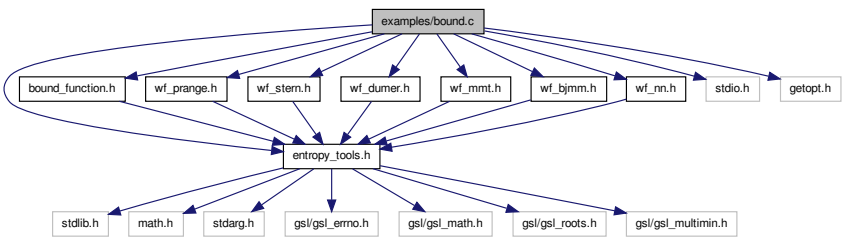
5.2 examples/bound.c

Auxilairy program which calculates the index a of the bound fonction.

Dependencies

```
#include <entropy_tools.h>
#include <bound_function.h>
#include <wf_prange.h>
#include <wf_stern.h>
#include <wf_dumer.h>
#include <wf_mmt.h>
#include <wf_bjmm.h>
#include <wf_nn.h>
#include <stdio.h>
#include <getopt.h>
```

Include dependency graph for bound.c:



Functions

- int main (int argc, char ∗argv[])

- void usage (void)

Variables

- FILE * streamout
- wf_params wfp
- double(* wf_algo)(wf_params *)

5.2.1 Detailed Description

Auxilairy program which calculates the index a of the bound fonction. This program takes a generic decoding algorithm algo as input and returns a in]0,1[such that min B_a = WF(algo) for many code rate k in [0,1] (and error rate w as Gilbert Varshamov's bound).

Definition in file bound.c.

5.2.2 Function Documentation

5.2.2.1 int main (int *argc*, char * *argv[]*)

It calculates the index a of the bound fonction.

Definition at line 82 of file bound.c.

5.2.2.2 void usage (void)

It prints help.

Definition at line 54 of file bound.c.

5.2.3 Variable Documentation

5.2.3.1 FILE* streamout

This variable will saves in a file the obtained values after optimization of wf.

Definition at line 44 of file bound.c.

5.2.3.2 double(* wf_algo)(wf_params *)

This variable saves the adress of the optimization function of work factor.

Definition at line 45 of file bound.c.

5.2.3.3 wf_params wfp

This variable contains the value of k, w, wf/w, l, p, p1, p2 after optimization of wf.

Definition at line 46 of file bound.c.

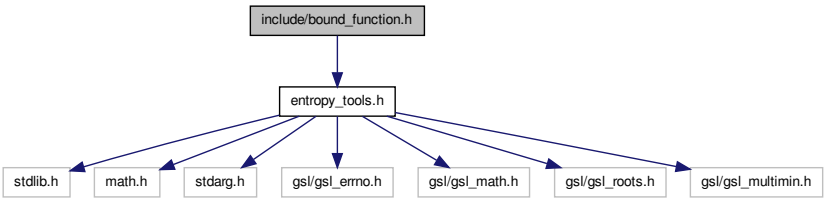
5.3 include/bound_function.h File Reference

Library for calculation of Bound Function.

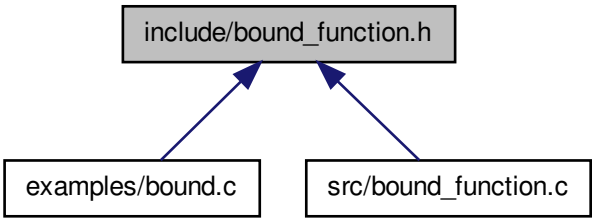
Dependencies

```
#include "entropy_tools.h"
#include <gsl/gsl_roots.h>
```

Include dependency graph for bound_function.h:



This graph shows which files directly or indirectly include this file:



Defines

- #define BOUND_FUNTION_H

Functions

- double pp (double l, wf_params *params)
- double dif_pp (double l, void *params)
- double pp_df (double l, wf_params *params)
- double reduced_Ba (double l, wf_params *params)
- double reduced_Ba_df (double l, void *params)
- double l_max (wf_params *params)
- double Optimal_reduced_Ba (wf_params *params)
- double dif_Optimal_reduced_Ba (double a, void *params)
- double find_coefficient (wf_params *params)

5.3.1 Detailed Description

Library for calculation of Bound Function. This Library calculate theorical Bound - Function, its optimal value for a coefficient a of min B_a and find, for a work factor wf, the coefficient a for the equation min B_a = wf for k, w fixed.

Definition in file bound_function.h.

5.3.2 Define Documentation

5.3.2.1 #define BOUND_FUNTION_H

Definition at line 31 of file bound_function.h.

5.3.3 Function Documentation

5.3.3.1 double pp (double *l*, wf_params * *params*)

Calculates the p such that 2^l=binomial(k+l,ap).

Declaration at line 42 of file bound_function.h.

5.3.3.2 dif_pp (double *l*, void * *params*)

the diference pp(l,a)-w.

Declaration at line 49 of file bound_function.h.

5.3.3.3 pp_df (double *l,* wf_params * *params*)

Derivate of function pp.

Declaration at line 56 of file bound_function.h.

5.3.3.4 double reduced_Ba (double *l,* wf_params * *params*)

Reduced version of Ba(l,p). This function will achieve the same minimum value of function Ba(l,p).

Declaration at line 65 of file bound_function.h.

5.3.3.5 double reduced_Ba_df (double *l,* void * *params*)

Derivate of function reduced_Ba.

Declaration at line 72 of file bound_function.h.

5.3.3.6 double l_max (wf_params * *params*)

Obtains the maximum value of l such Ba is calculable.

Declaration at line 79 of file bound_function.h.

5.3.3.7 double Optimal_reduced_Ba (wf_params * *params*)

Obtains the minimum value of Ba.

Declaration at line 86 of file bound_function.h.

5.3.3.8 double dif_Optimal_reduced_Ba (double *a,* void * *params*)

The difference Optimal_reduced_Ba - workfactor.

Declaration at line 93 of file bound_function.h.

5.3.3.9 double find_coefficient (wf_params * *params*)

Find a such that reduced_Ba_min(a)=wf.

Declaration at line 100 of file bound_function.h.

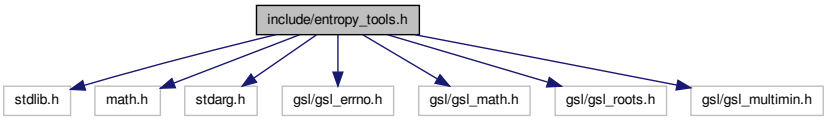
5.4 include/entropy_tools.h File Reference

Auxiliary library for calculation of Work Factor.

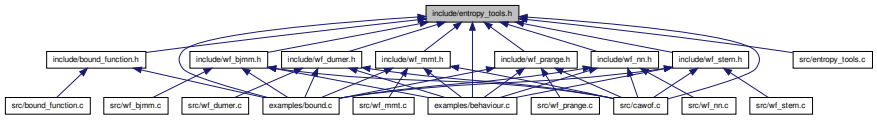
Dependencies

```
#include <stdlib.h>
#include <math.h>
#include <stdarg.h>
#include <gsl/gsl_errno.h>
#include <gsl/gsl_math.h>
#include <gsl/gsl_roots.h>
#include <gsl/gsl_multimin.h>
```

Include dependency graph for entropy_tools.h:



This graph shows which files directly or indirectly include this file:



Data Structures

- struct wf_params

Functions

- double min (double x, double y)
- double max (int num,...)
- double entropy (double x)
- double dif_entropy (double x, void *params)
- double binomial (double x, double y)
- double entropy_df (double x, void *params)
- void entropy_fdf (double x, void *params, double *y, double *dy)
- double entropy_inverse (double y)

5.4.1 Detailed Description

Auxiliary library for calculation of Work Factor. This Library contains the structure wf-_params which comunicate several parameters in the optimization of several work factors.

Definition in file entropy_tools.h.

5.4.2 Function Documentation

5.4.2.1 min (double *x*, double *y*)

Minimum of two double values.

Declaration at line 58 of file entropy_tools.h.

5.4.2.2 max (int *num*, ...)

Maximum of num double values.

Declaration at line 65 of file entropy_tools.h.

5.4.2.3 entropy (double *x*)

This function calculates the (binary) entropy.

Declaration at line 72 of file entropy_tools.h.

5.4.2.4 dif_entropy (double *x*, void * *params*)

The diference entropy(x)-params.

Declaration at line 79 of file entropy_tools.h.

5.4.2.5 binomial (double *x*, double *y*)

This function approximates Binomial coefficient by entropy function.

Declaration at line 86 of file entropy_tools.h.

5.4.2.6 entropy_df (double *x*, void * *params*)

Derivate of entropy function.

Declaration at line 93 of file entropy_tools.h.

5.4.2.7 entropy_fdf (double *x*, void * *params*, double * *y*, double * *dy*)

Obtains the dif_entropy function and its derivate.

Declaration at line 100 of file entropy_tools.h.

5.4.2.8 entropy_inverse (double *y*)

Obtains the inverse of entropy function and return a real in [0,0.5].

Declaration at line 107 of file entropy_tools.h.

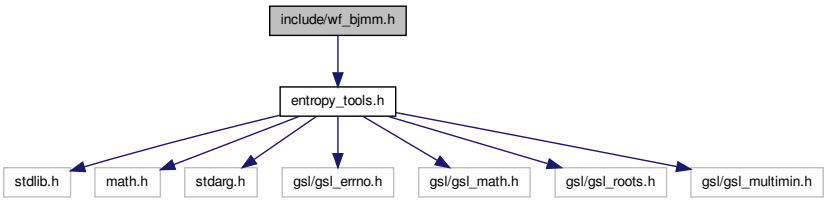
5.5 include/wf_bjmm.h File Reference

Library for calculation of BJMM's Work Factor.

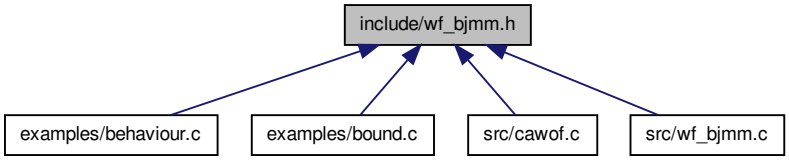
Dependencies

```
#include "entropy_tools.h"
```

Include dependency graph for wf_bjmm.h:



This graph shows which files directly or indirectly include this file:



Functions

- double wf_BJMM (const gsl_vector *v, void *paramslp)
- double Optimal_wf_BJMM_p12 (const gsl_vector *vv, void *params)
- double Optimal_wf_BJMM (wf_params *params)

5.5.1 Detailed Description

Library for calculation of BJMM's Work Factor.

Definition in file wf_bjmm.h.

5.5.2 Function Documentation

5.5.2.1 double wf_BJMM (const gsl_vector * v, void * paramslp)

Work Factor of BJMM's algorithm. Parameters p1 and p2 in v and k, w, N, l ,p in paramslp.

Declaration at line 37 of file wf_bjmm.h.

5.5.2.2 double Optimal_wf_BJMM_p12 (const gsl_vector * vv, void * params)

Optimal Work Factor of BJMM's algorithm for l and p fixed. Parameters l and p in vv and k, w, N in params. It saves Optimal Parameters p1 and p2 in paramslp.

Declaration at line 47 of file wf_bjmm.h.

5.5.2.3 double Optimal_wf_BJMM (wf_params * params)

Optimal Work Factor of BJMM's algorithm. Parameters k, w, N in params. It saves Optimal Parameters l, p, p1 and p2 in params.

Declaration at line 57 of file wf_bjmm.h.

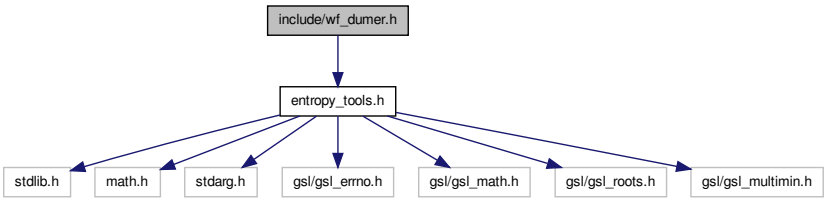
5.6 include/wf_dumer.h File Reference

Library for calculation of Dumer's Work Factor.

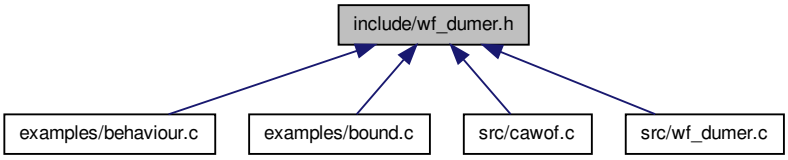
Dependencies

```
#include "entropy_tools.h"
```

Include dependency graph for wf_dumer.h:



This graph shows which files directly or indirectly include this file:



Functions

- double wf_Dumer (const gsl_vector ∗v, void ∗params)
- double Optimal_wf_Dumer (wf_params ∗params)

5.6.1 Detailed Description

Library for calculation of Dumer's Work Factor.

Definition in file wf_dumer.h.

5.6.2 Function Documentation

5.6.2.1 double wf_Dumer (const gsl_vector ∗ *v*, void ∗ *params*)

Work Factor of Dumer's algorithm. Parameters l and p in v and k, w, N in params.

Definition at line 37 of file wf_dumer.h.

5.6.2.2 double Optimal_wf_Dumer (wf_params * *params*)

Optimal Work Factor of Dumer's algorithm. Parameters k, w, N in params. It saves Optimal Parameters l and p in params.

Definition at line 46 of file wf_dumer.h.

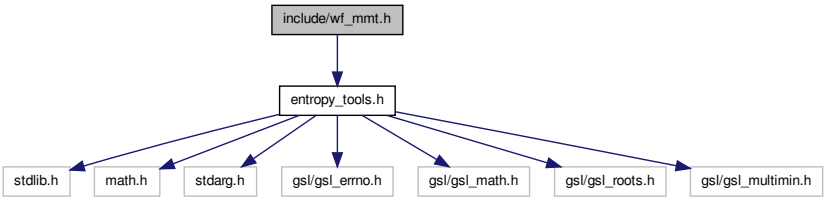
5.7 include/wf_mmt.h File Reference

Library for calculation of MMT's Work Factor.

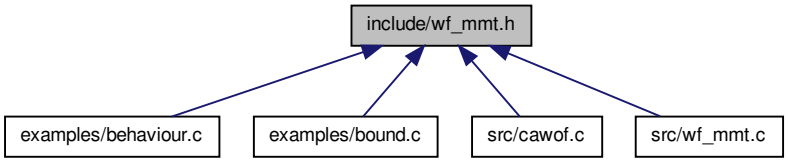
Dependencies

```
#include "entropy_tools.h"
```

Include dependency graph for wf_mmt.h:



This graph shows which files directly or indirectly include this file:



Functions

- double wf_MMT (const gsl_vector *v, void *params)
- double Optimal_wf_MMT (wf_params *params)

5.7.1 Detailed Description

Library for calculation of MMT's Work Factor. Definition in file wf_mmt.h.

5.7.2 Function Documentation

5.7.2.1 double wf_MMT (const gsl_vector * *v*, void * *params*)

Work Factor of MMT's algorithm. Parameters l and p in v and k, w, N in params.

Declaration at line 37 of file wf_mmt.c.

5.7.2.2 double Optimal_wf_MMT (wf_params * *params*)

Optimal Work Factor of MMT's algorithm. Parameters k, w, N in params. It saves - Optimal Parameters l and p in params.

Declaration at line 46 of file wf_mmt.c.

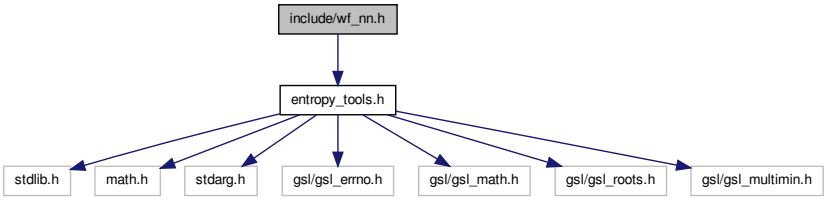
5.8 include/wf_nn.h File Reference

Library for calculation of Nearest Neighboord's Work Factor.

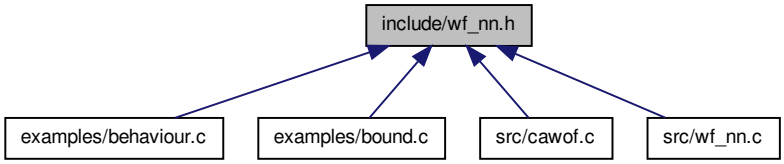
5.8.1 Dependencies

```
#include "entropy_tools.h"
```

Include dependency graph for wf_nn.h:



This graph shows which files directly or indirectly include this file:



Functions

- double wf_NN (double p2, void ∗paramslpp1)
- double Optimal_wf_NN_p (const gsl_vector ∗v, void ∗params)
- double Optimal_wf_NN (wf_params ∗params)

5.8.2 Detailed Description

Library for calculation of Nearest Neighboord's Work Factor.

Definition in file wf_nn.h.

5.8.3 Function Documentation

5.8.3.1 double wf_NN (double *p2,* void ∗ *paramslpp1*)

Work Factor of Nearest Neighbord's algorithm. Parameters k, w, N, l ,p, p1 in paramslpp1.

Declaration at line 37 of file wf_nn.h.

5.8.3.2 double Optimal_wf_NN_p (const gsl_vector ∗ *v,* void ∗ *params*)

Optimal Work Factor of Nearest Neighbord's algorithm for l, p fixed. Parameters l and p in v and k, w, N in params. It saves Optimal Parameters p2 in paramslp (p1 is chosen from l and p).

Declaration at line 48 of file wf_nn.h.

5.8.3.3 double Optimal_wf_NN (wf_params ∗ *params*)

Optimal Work Factor of Nearest Neighbord's algorithm. Parameters k, w, N in params. It saves Optimal Parameters l, p, p1 and p2 in params.

Declaration at line 58 of file wf_nn.h.

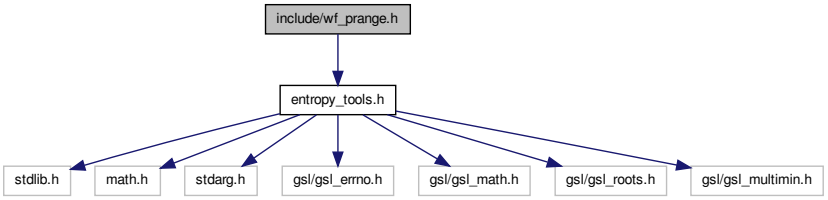
5.9 include/wf_prange.h File Reference

Library for calculation of Prange's Work Factor.

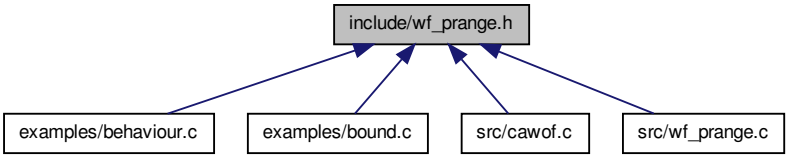
Dependencies

```
#include "entropy_tools.h"
```

Include dependency graph for wf_prange.h:



This graph shows which files directly or indirectly include this file:



Functions

- double wf_Prange (wf_params *params)

5.9.1 Detailed Description

Library for calculation of Prange's Work Factor.

Definition in file wf_prange.h.

5.9.2 Function Documentation

5.9.2.1 double wf_Prange (wf_params * *params*)

Work Factor of Prange's algorithm.

Declaration at line 34 of file wf_prange.h.

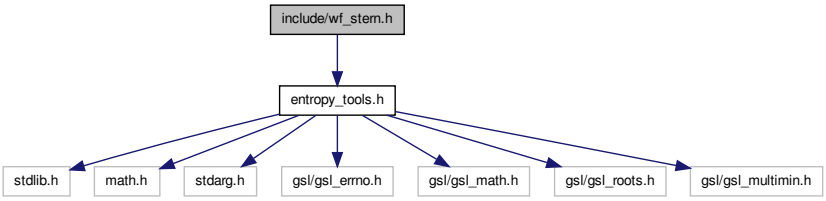
5.10 include/wf_stern.h File Reference

Library for calculation of Stern's Work Factor.

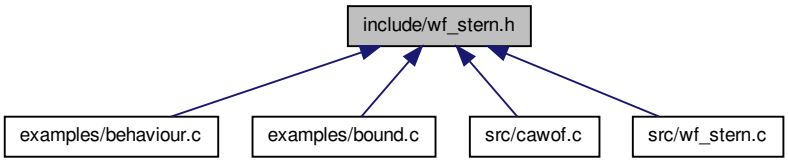
Dependencies

```
#include "entropy_tools.h"
```

Include dependency graph for wf_stern.h:



This graph shows which files directly or indirectly include this file:



Functions

- double wf_Stern (const gsl_vector *v, void *params)
- double Optimal_wf_Stern (wf_params *params)

5.10.1 Detailed Description

Library for calculation of Nearest Neighboord's Work Factor.

Definition in file wf_stern.h.

5.10.2 Function Documentation

5.10.2.1 double wf_Stern (const gsl_vector * v, void * params)

Work Factor of Stern's algorithm. Parameters l and p in v and k, w, N in params.

Declaration at line 37 of file wf_stern.h.

5.10.2.2 double Optimal_wf_Stern (wf_params * params)

Optimal Work Factor of Stern's algorithm. Parameters k, w, N in params. It saves Optimal Parameters l and p in params.

Declaration at line 46 of file wf_stern.h.

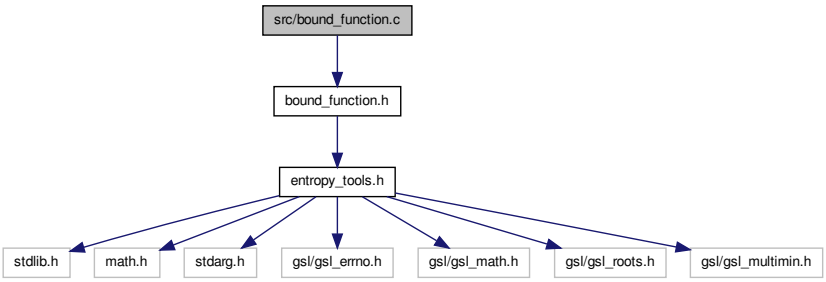
5.11 src/bound_function.c File Reference

Implementation of include/bound_function.h.

Dependencies

```
#include <bound_function.h>
```

Include dependency graph for bound_function.c:



Functions

- double pp (double l, wf_params ∗params)
- double dif_pp (double l, void ∗params)
- double pp_df (double l, wf_params ∗params)
- double reduced_Ba (double l, wf_params ∗params)
- double reduced_Ba_df (double l, void ∗params)
- double l_max (wf_params ∗params)
- double Optimal_reduced_Ba (wf_params ∗params)
- double dif_Optimal_reduced_Ba (double a, void ∗params)
- double find_coefficient (wf_params ∗params)

5.11.1 Detailed Description

Implementation of include/bound_function.h.

Definition in file bound_function.c.

5.11.2 Function Documentation

5.11.2.1 double pp (double *l*, wf_params ∗ *params*)

Calculates the p such that 2^l=binomial(k+l,ap).

Implementation at line 28 of file bound_function.c.

5.11.2.2 dif_pp (double *l*, void ∗ *params*)

the diference pp(l,a)-w. Implementation at line 38 of file bound_function.c.

5.11.2.3 pp_df (double *l,* wf_params * *params*)

Derivate of function pp.

Implementation at line 44 of file bound_function.c.

5.11.2.4 double reduced_Ba (double *l,* wf_params * *params*)

Reduced version of Ba(l,p). This function will achieve the same minimum value of function Ba(l,p).

Implementation at line 56 of file bound_function.c.

5.11.2.5 double reduced_Ba_df (double *l,* void * *params*)

Derivate of function reduced_Ba.

Declaration at line 67 of file bound_function.h.

5.11.2.6 double l_max (wf_params * *params*)

Obtains the maximum value of l such Ba is calculable.

Implementation at line 82 of file bound_function.c.

5.11.2.7 double Optimal_reduced_Ba (wf_params * *params*)

Obtains the minimum value of Ba.

Implementation at line 131 of file bound_function.c.

5.11.2.8 double dif_Optimal_reduced_Ba (double *a,* void * *params*)

The difference Optimal_reduced_Ba - workfactor.

Implementation at line 184 of file bound_function.c.

5.11.2.9 double find_coefficient (wf_params * *params*)

Find a such that reduced_Ba_min(a)=wf.

Implementation at line 195 of file bound_function.c.

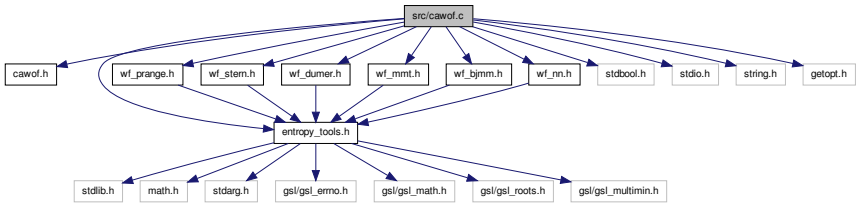
5.12 src/cawof.c File Reference

Main program of calculation of Work Factor.

Dependencies

```
#include "cawof.h"
#include <entropy_tools.h>
#include <wf_prange.h>
#include <wf_stern.h>
#include <wf_dumer.h>
#include <wf_mmt.h>
#include <wf_bjmm.h>
#include <wf_nn.h>
#include <stdbool.h>
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <getopt.h>
```

Include dependency graph for cawof.c



Functions

- void version (void)
- void usage (int status)
- int main (int argc, char ∗argv[])

Variables

- bool verbose
- bool quiet
- bool format
- wf_params wfp
- char algo [20]
- double(∗ wf_algo)(wf_params ∗)

5.12.1 Detailed Description

Calculates of Work Factor of an ISD algorithm in the code rate and error rate. Default algorithm is BJMM, default code rate is 0.5 and defaut error rate is s Gilbert-Varshamov's bound. Available algorithms are PRANGE, STERN, DUMER, MMT, BJMM and NN.

Definition in file cawof.c.

5.12.2 Function Documentation

5.12.2.1 void version (void)

Print the version of program.

Definition at line 56 of file cawof.c.

5.12.2.2 void usage (int *status*)

Print the help of program.

Definition at line 68 of file cawof.c.

5.12.2.3 int main (int *argc*, char * *argv[]*)

Calculates of Work Factor of an ISD algorithm in the code rate and error rate.

Definition at line 96 of file cawof.c.

5.12.3 Variable Documentation

5.12.3.1 bool verbose

Decides if the program prints the values of optimal parameters for the algorithm.

Definition at line 46 of file cawof.c.

5.12.3.2 bool quiet

Decides if the program prints explanation of results.

Definition at line 46 of file cawof.c.

5.12.3.3 bool format

Decides if the work factor is expressed in terms of w or n.

Definition at line 46 of file cawof.c.

5.12.3.4 wf_params wfp

It contains the parameters of coding algorithm.

Definition at line 47 of file cawof.c.

5.12.3.5 char algo[20]

It saves the name of algorithm to analyze.

Definition at line 48 of file cawof.c.

5.12.3.6 double(* wf_algo)(wf_params *)

It contains the optimization function to use.

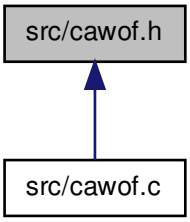
Definition at line 49 of file cawof.c.

5.13 src/cawof.h File Reference

Definiton of constants for the principal program.

Dependencies

This graph shows which files directly or indirectly include this file:



Defines

- #define PROG_NAME "CaWoF"
- #define PROG_VERSION 1
- #define PROG_SUBVERSION 0

- #define PROG_REVISION 0

5.13.1 Detailed Description

Definiton of constants for the principal program.

Definition in file cawof.h.

5.13.2 Define Documentation

5.13.2.1 #define PROG_NAME "CaWoF"

Definition at line 30 of file cawof.h.

5.13.2.2 #define PROG_VERSION 1

Definition at line 32 of file cawof.h.

5.13.2.3 #define PROG_SUBVERSION 0

Definition at line 33 of file cawof.h.

5.13.2.4 #define PROG_REVISION 0

Definition at line 34 of file cawof.h.

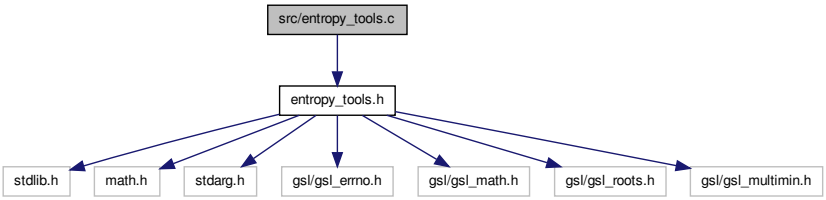
5.14 src/entropy_tools.c File Reference

Implementation of include/entropy_tools.h.

Dependencies

```
#include <entropy_tools.h>
```

Include dependency graph for entropy_tools.c:



Functions

- double max (int num,...)
- double min (double x, double y)
- double entropy (double x)
- double binomial (double x, double y)
- double dif_entropy (double x, void *params)
- double entropy_df (double x, void *params)
- void entropy_fdf (double x, void *params, double *y, double *dy)
- double entropy_inverse (double y)

5.14.1 Detailed Description

Implementation of include/entropy_tools.h.

Definition in file entropy_tools.c.

5.14.2 Function Documentation

5.14.2.1 double max (int *num*, ...)

Maximum of num double values.

Implementation at line 28 of file entropy_tools.c.

5.14.2.2 double min (double *x*, double *y*)

Minimum of two double values.

Implementation at line 49 of file entropy_tools.c.

5.14.2.3 double entropy (double *x*)

This function calculates the (binary) entropy.

Implementation at line 57 of file entropy_tools.c.

5.14.2.4 double binomial (double *x*, double *y*)

This function approximates Binomial coefficient by entropy function.

Implementation at line 65 of file entropy_tools.c.

5.14.2.5 double dif_entropy (double *x*, void * *params*)

The diference entropy(x)-params.

Implementation at line 73 of file entropy_tools.c.

5.14.2.6 double entropy_df (double *x*, void * *params*)

Derivate of entropy function.

Implementation at line 82 of file entropy_tools.c.

5.14.2.7 void entropy_fdf (double *x*, void * *params*, double * *y*, double * *dy*)

Obtains the dif_entropy function and its derivate.

Implementation at line 88 of file entropy_tools.c.

5.14.2.8 double entropy_inverse (double *y*)

Obtains the inverse of entropy function and return a real in [0,0.5]

Implementation at line 96 of file entropy_tools.c.

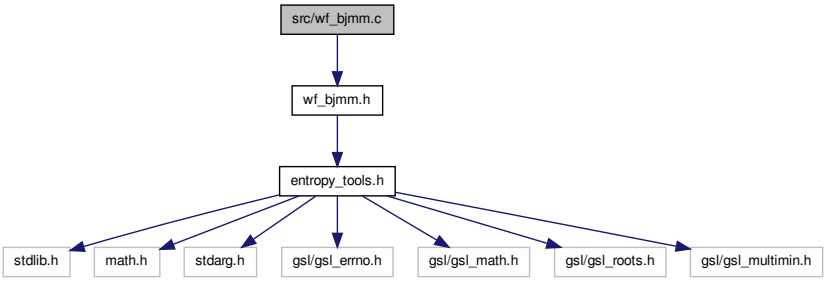
5.15 src/wf_bjmm.c File Reference

Implementation of include/wf_bjmm.h.

Dependencies

```
#include <wf_bjmm.h>
```

Include dependency graph for wf_bjmm.c:



Functions

- double wf_BJMM (const gsl_vector ∗v, void ∗paramslp)
- double Optimal_wf_BJMM_p12 (const gsl_vector ∗vv, void ∗params)
- double Optimal_wf_BJMM (wf_params ∗params)

5.15.1 Detailed Description

Implementation of include/wf_bjmm.h.

Definition in file wf_bjmm.c.

5.15.2 Function Documentation

5.15.2.1 double wf_BJMM (const gsl_vector ∗ *v,* void ∗ *paramslp*)

Work Factor of BJMM's algorithm. Parameters p1 and p2 in v and k, w, N, l ,p in paramslp.

Implementation at line 29 of file wf_bjmm.c.

5.15.2.2 double Optimal_wf_BJMM_p12 (const gsl_vector ∗ *vv,* void ∗ *params*)

Optimal Work Factor of BJMM's algorithm for l and p fixed. Parameters l and p in vv and k, w, N in params. It saves Optimal Parameters p1 and p2 in paramslp.

Definition at line 60 of file wf_bjmm.c.

5.15.2.3 double Optimal_wf_BJMM (wf_params * *params*)

Optimal Work Factor of BJMM's algorithm. Parameters k, w, N in params. It saves Optimal Parameters l, p, p1 and p2 in params.

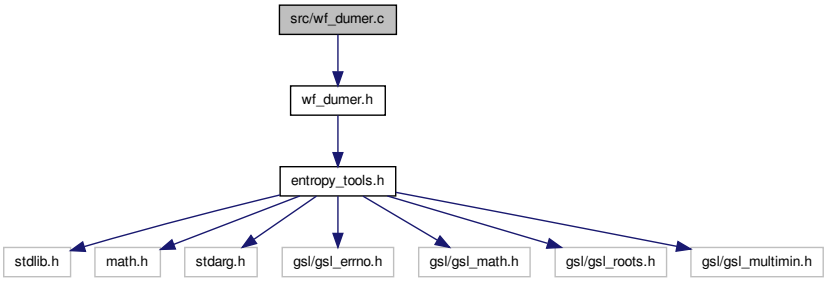
Definition at line 133 of file wf_bjmm.c.

5.16 src/wf_dumer.c File Reference

Dependencies

```
#include <wf_dumer.h>
```

Include dependency graph for wf_dumer.c:



Functions

- double wf_Dumer (const gsl_vector *v, void *params)
- double Optimal_wf_Dumer (wf_params *params)

5.16.1 Detailed Description

Implementation of include/wf_dumer.h.

Definition in file wf_dumer.c.

5.16.2 Function Documentation

5.16.2.1 double wf_Dumer (const gsl_vector * *v*, void * *params*)

Work Factor of Dumer's algorithm. Parameters l and p in v and k, w, N in params.

Implementation at line 29 of file wf_dumer.c.

5.16.2.2 double Optimal_wf_Dumer (wf_params * *params*)

Optimal Work Factor of Dumer's algorithm. Parameters k, w, N in params. It saves Optimal Parameters l and p in params.

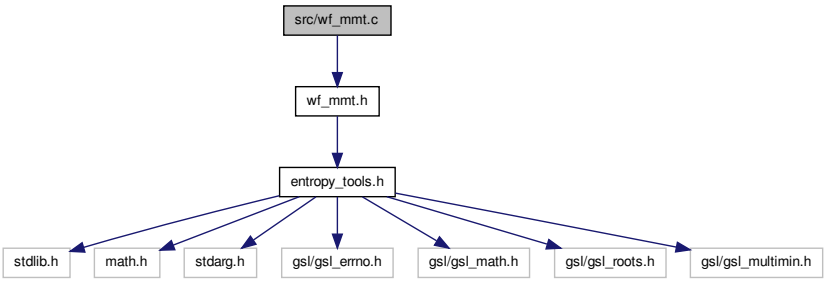
Implementation at line 53 of file wf_dumer.c.

5.17 src/wf_mmt.c File Reference

Dependencies

```
#include <wf_mmt.h>
```

Include dependency graph for wf_mmt.c:



Functions

- double wf_MMT (const gsl_vector *v, void *params)
- double Optimal_wf_MMT (wf_params *params)

5.17.1 Detailed Description

Implementation of include/wf_mmt.h. Definition in file wf_mmt.c.

5.17.2 Function Documentation

5.17.2.1 double wf_MMT (const gsl_vector * *v*, void * *params*)

Work Factor of MMT's algorithm. Parameters l and p in gsl_vector v and k, w, N in params.

Implementation at line 29 of file wf_mmt.c.

5.17.2.2 double Optimal_wf_MMT (wf_params * *params*)

Optimal Work Factor of MMT's algorithm. Parameters k, w, N in params. It saves - Optimal Parameters l and p in params.

Implementation at line 52 of file wf_mmt.c.

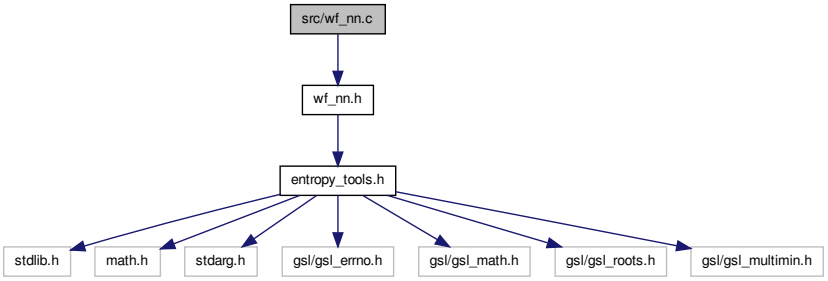
5.18 src/wf_nn.c File Reference

Implementation of include/wf_nn.h.

Dependencies

```
#include <wf_nn.h>
```

Include dependency graph for wf_nn.c:



Functions

- double wf_NN (double p2, void *paramslpp1)
- double Optimal_wf_NN_p (const gsl_vector *v, void *params)
- double Optimal_wf_NN (wf_params *params)

5.18.1 Detailed Description

Implementation of include/wf_nn.h.

Definition in file wf_nn.c.

5.18.2 Function Documentation

5.18.2.1 double wf_NN (double *p2,* void * *paramslpp1*)

Work Factor of Nearest Neighbord's algorithm. Parameters k, w, N, l ,p, p1 in paramslpp1.

Implementation at line 29 of file wf_nn.c.

5.18.2.2 double Optimal_wf_NN_p (const gsl_vector * *v,* void * *params*)

Optimal Work Factor of Nearest Neighbord's algorithm for l, p fixed. Parameters l and p in v and k, w, N in params. It saves Optimal Parameters p2 in paramslp (p1 is chosen from l and p).

Implementation at line 62 of file wf_nn.c.

5.18.2.3 double Optimal_wf_NN (wf_params * *params*)

Optimal Work Factor of Nearest Neighbord's algorithm. Parameters k, w, N in params. It saves Optimal Parameters l, p, p1 and p2 in params.

Implementation at line 122 of file wf_nn.c.

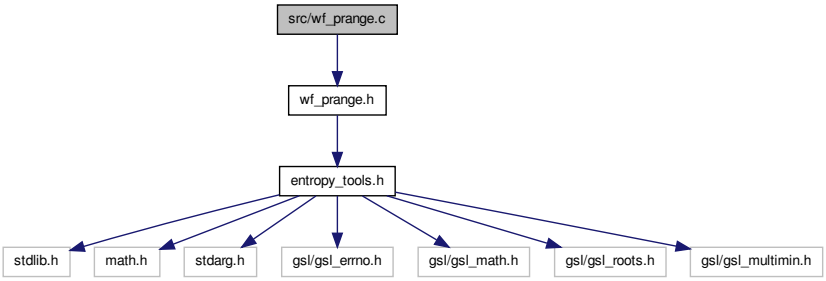
5.19 src/wf_prange.c File Reference

Implementation of include/wf_prange.h.

Dependencies

```
#include <wf_prange.h>
```

Include dependency graph for wf_prange.c:



Functions

- double wf_Prange (wf_params *params)

5.19.1 Detailed Description

Implementation of include/wf_prange.h.

Definition in file wf_prange.c.

5.19.2 Function Documentation

5.19.2.1 double wf_Prange (wf_params * *params*)

Work Factor of Prange's algorithm.

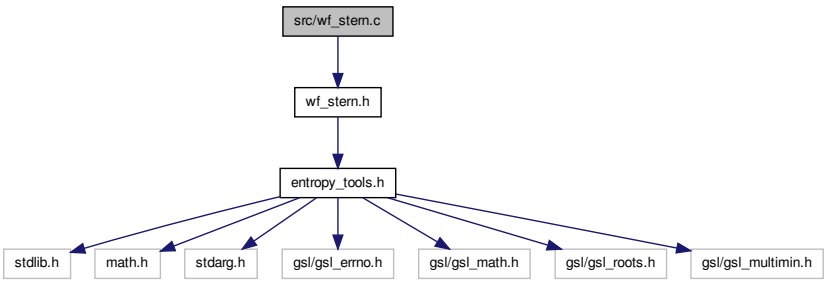
Implementation at line 29 of file wf_prange.c.

5.20 src/wf_stern.c File Reference

Implementation of include/wf_stern.h.

Dependencies

```
#include <wf_stern.h>
```



Functions

- double wf_Stern (const gsl_vector *v, void *params)
- double Optimal_wf_Stern (wf_params *params)

5.20.1 Detailed Description

Implementation of include/wf_stern.h.

Definition in file wf_stern.c.

5.20.2 Function Documentation

5.20.2.1 double wf_Stern (const gsl_vector * v, void * params)

Work Factor of Stern's algorithm. Parameters l and p in v and k, w, N in params.

Implementation at line 29 of file wf_stern.c.

5.20.2.2 double Optimal_wf_Stern (wf_params * params)

Optimal Work Factor of Stern's algorithm. Parameters k, w, N in params. It saves Optimal Parameters l and p in params.

Implementation at line 53 of file wf_stern.c.

Index