The Orthogonal Array package

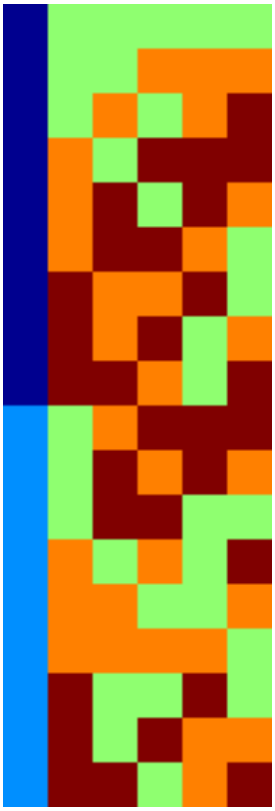
P.T. Eendebak*

July, 2015

Abstract

This document describes the Orthogonal Array package. The package contains tools to work with orthogonal arrays and optimal designs. The package is written in C++, but includes a Python and command-line interface. Functionality is included to calculate reductions to normal form and statistical properties such as generalized wordlength patterns, J-characteristics, D-efficiency, strength and rank. Also complete enumeration of all isomorphism classes of arrays of a specified type is supported.

Package version: 2.3.4



*Corresponding author. E-mail: pieter.eendebak@gmail.com. Address: University of Antwerp, Dept. of Mathematics, Statistics, and Actuarial Sciences, Prinsstraat 13, 2000 Antwerp, Belgium.

Contents

1 Introduction

Orthogonal arrays are an important tool in the design of experiments [Hedayat et al., 1999]. The Orthogonal Array package contains functionality the generate orthogonal arrays and to analyse their properties. The algorithms and methods in this package have been described in [Schoen et al., 2010]. A large collection of arrays can be found on the Orthogonal Array package website [Eendebak, 2012] or on the website of Neil Sloane [Sloane, 2014].

1.1 Example usage

The Orthogonal Array package can be used to work with arrays and calculate statistical properties. For example to calculate the D-efficiency and rank of a design we can use:

Calculate D-efficiency

```
>>> al=oapackage.exampleArray(0)
>>> al.showarray()
array:
  0   0
  0   0
  0   1
  0   1
  1   0
  1   0
  1   1
  1   1
>>> print('D-efficiency %f, rank %d' % (al.Defficiency(), al.rank()) )
D-efficiency 1.000000, rank 2
>>> print('Generalized wordlength pattern: %s' % str(al.GWLP()))
Generalized wordlength pattern: (1.0, 0.0, 0.0)
```

1.2 Interfaces

The Orthogonal Array package has several interfaces. First of all there are command line tools for manipulating arrays and generating. All functions of the package can be used from either C++ or Python. For a restricted set of functionality also Matlab or R can be used.

1.3 Compilation and installation

The program has been tested using Linux and Windows (XP, Windows 7 and Windows 8). The Python interface is available from the Python Package Index https://pypi.python.org/pypi/OApackage/. The package can be installed from the command line using pip:

```
> pip install --user OApackage
```

The R interface to the package is available from CRAN http://cran.r-project.org/web/packages/oapackage/index.html.

The command line tools uses a cmake build system. From the command line type:

```
> mkdir -p build; cd build
> cmake ..
> make
> make install
```

This creates the command line utilities and a C++ library. To compile the Python interface using Linux use

```
> python setup.py build
> python setup.py install --user
```

The Python interface requires Numpy [The Scipy community, 2012], Matplotlib [Hunter, 2007] and Swig. The code has been tested with Python 2.7, 3.4 and 3.5.

Using Windows start Cygwin or the Visual Studio command prompt. From the package source directory run:

```
%> SET VS90COMNTOOLS=%VS110COMNTOOLS%
%> swig -c++ -w503,401,362 -python -Isrc/ oalib.i
> python setup.py bdist_wininst
```

This creates a binary installer package.

1.4 License

The code is available under a BSD style license, see the file LICENSE for details. If you use this code or any of the results, please cite this program as follows:

Complete Enumeration of Pure-Level and Mixed-Level Orthogonal Arrays, P.T. Eendebak, E.D. Schoen, M.V.M. Nguyen, Volume 18, Issue 2, pages 123-140, 2010.

1.5 Acknowledgements

The code and ideas for this package have been contributed by Eric Schoen, Ruben Snepvangers, Vincent Brouerius van Nidek, Alan Vazquez-Alcocer and Pieter Thijs Eendebak.

2 The Orthogonal Array package

An orthogonal array (OA) of strength t, N runs and n factors at s levels is an $N \times n$ array of $0, \ldots, (s-1)$-valued symbols such that for every t columns every t-tuple occurs equally often [Rao, 1947]. The set of all OAs with given strength, runs and levels is denoted by $\mathrm{OA}(N; t; s^n)$. The OAs are represented by arrays (in column-major form).

2.1 Data structures

The package contains several data structures. Here we describe the main structures and their use.

`array_link` The structure containing an orthogonal array is called the `array_link` structure. Lists of arrays are stored in the `arraylist_t` object, which as a `std::deque` container.

`arrayfile_t` This is an objct that allows for reading and writing of arrays to disk.

`arraydata_t` The structure describing a certain class of orthogonal arrays or designs.

`array_transformation_t` This describes a transformation of an array. This includes the row-, column- and level-permutations.

2.1.1 Representing arrays

The structure containing an orthogonal array is called the `array_link` structure. It consists of a specified number of rows and columns, the data (integer values) and an index.

C++ interface

```
struct array_link
{
    //! Number of rows in array
    rowindex_t n_rows;
    //! Number of columns in array
    colindex_t n_columns;
    //! Index number
    int index;
    //! Pointer to an array data
    array_t* array;

    /// Constructor functions
    array_link();
    array_link(rowindex_t nrows, colindex_t ncols, int index);
    ~array_link();
    array_link(const array_link &);

public:
    /// print an array to output stream
    friend std::ostream &operator<<(std::ostream &, const array_link &A);

    /// print array to stdout
    void showarray() const;

    // manipulation of arrays

    /// return array with selected column removed
    array_link deleteColumn(int index) const;

    /// return array with first n columns selected
    array_link selectFirstColumns(int n) const;

    /// return array with last n columns selected
    array_link selectLastColumns(int n) const;

    /// select columns from an array
    array_link selectColumns(const std::vector<int> c) const;

    /// return transposed array
    array_link transposed() const;

    // statistical properties of the array

        ...
}
```

In the Python interface the arraylink object can be indexed just as normal arrays. It is also possible to return a Numpy array. The `array_link` object implements to Python array interface, so most opertations from packages such as Numpy work on the `array_link` object.

Array representation in Python

```
>>> import oapackage
>>> al=oapackage.exampleArray(0)
>>> al.showarray()
array:
  0   0
  0   0
  0   1
  0   1
  1   0
  1   0
  1   1
  1   1
>>> al[2,1]
1L
>>> X=al.getarray()
>>> X
array([[0, 0],
       [0, 0],
       [0, 1],
       [0, 1],
       [1, 0],
       [1, 0],
       [1, 1],
       [1, 1]], dtype=int32)
```

2.1.2 Reading and writing arrays

Reading and writing arrays to disk can be done with the `arrayfile_t` class. For example:

Write an array to disk

```
>>> import oapackage
>>> al=oapackage.exampleArray()
>>> af=oapackage.arrayfile_t('test.oa', al.n_rows, al.n_columns)
>>> af.append_array(al)
>>> print(af)
file test.oa: 8 rows, 2 columns, 1 arrays, mode text, nbits 8
>>> af.closefile()
```

The arrays can be written in text or binary format. For more details on the file format see Section 2.2. The header of the `arrayfile_t` class is listed below.

C++ interface

```
struct arrayfile_t
{

public:
    std::string filename;
    int iscompressed;
    int nrows;
    int ncols;

    /// number of bits used when storing an array
    int nbits;

    /// file mode, can be ATEXT or ABINARY
    arrayfilemode_t mode;
    /// file opened for reading or writing
    afilerw_t rwmode;
```

```
    int narrays;
    int narraycounter;

public:

    /// open existing array file
    arrayfile_t(const std::string fname, int verbose = 1);
    /// open new array file for writing
    arrayfile_t(const std::string fname, int nrows, int ncols,
                int narrays=-1, arrayfilemode_t m = ATEXT, int nb = 8);
    /// destructor function, closes all filehandles
    ~arrayfile_t();

    /// close the array file
    void closefile();
    /// return true if file is open
    int isopen() const;
    /// seek to specified array position
    int seek(int pos);
    /// read array and return index
    int read_array(array_link &a);
    /// return true if the file has binary format
    bool isbinary() const;
    /// append arrays to the file
    int append_arrays(const arraylist_t &arrays, int startidx);
    /// append a single array to the file
    void append_array(const array_link &a, int specialindex=-1);

    ...

}
```

2.1.3 Array transformations

Transformations of (orthogonal) arrays consist of row permutations, level permutations and level transformations. A transformation is represented by the `array_transformation_t` object.

For a given transformation the column permutations are applied first, then the level permutations and finally the row permutations. The level- and column permutations are not commutative.

C++ interface

```
class array_transformation_t
{
public:
    rowperm_t        rperm;                /// row permutation
    colperm_t        colperm;          /// column permutation
    levelperm_t        *lperms;          /// level permutations
    const arraydata_t *ad;          /// type of array

public:
    array_transformation_t ( const arraydata_t *ad );
    array_transformation_t ( );                /// default constructor
    array_transformation_t ( const array_transformation_t  &at );
    array_transformation_t & operator= ( const array_transformation_t &at );
    ~array_transformation_t();  /// destructor

    /// show the array transformation
    void show() const;

    /// return true if the transformation is equal to the identity
```

```
    bool isIdentity() const;

    /// return the inverse transformation
    array_transformation_t inverse() const;

    /// return the transformation to the identity transformation
    void reset();

    /// initialize to a random transformation
    void randomize();

    /// initialize with a random column transformation
    void randomizecolperm();

    /// apply transformation to an array_link object
    array_link apply ( const array_link &al ) const;

    /// composition operator. the transformations are applied from the left
    array_transformation_t operator*(const array_transformation_t b);

    ...
```

2.1.4 Classes of arrays

The `arraydata_t` object represents data about a class of orthogonal arrays, e.g. the class $\mathrm{OA}(N; t; s^k)$.

C++ interface

```
struct arraydata_t
{
    rowindex_t N;        /** number of runs */
    array_t *s; /** pointer to levels of the array */
    colindex_t ncols; /** total number of columns (factors) in the design */
    colindex_t strength;        /** strength of the design */

    ordering_t  order; /** Ordering used for arrays */

public:
    /// create new arraydata_t object
    arraydata_t(std::vector<int> s, rowindex_t N_, colindex_t t, colindex_t nc);
    arraydata_t(carray_t *s_, rowindex_t N_, colindex_t t, colindex_t nc);
    arraydata_t(const arraydata_t &adp);

    ...

    /// return true if the array is of mixed type
    bool ismixed() const;
        /// return true if the array is a 2-level array
        bool is2level() const;
    /// set column group equal to that of a symmetry group
    void set_colgroups(const symmetry_group &sg);
            /// return random array from the class
    array_link randomarray ( int strength = 0, int ncols=-1 ) const;

}
```

2.2 File formats

The Orthogonal Array package stored orthogonal arrays in a custom file format. There is a text format with is easily readable by humans and a binary format with is faster to process and memory efficient.

2.2.1 Plain text array files

Arrays are stored in plain text files with extension .oa. The first line contains the number of columns, the number of rows and the number of arrays (or -1 if the number of arrays is not specified). Then for each array a single line with the index of the array, followed by N lines containing the array.

A typical example of a text file would be:

```
5 8 1
1
0 0 0 0 0
0 0 0 1 1
0 1 1 0 0
0 1 1 1 1
1 0 1 0 1
1 0 1 1 0
1 1 0 0 1
1 1 0 1 0
-1
```

This file contains exactly 1 array with 8 rows and 5 columns.

2.2.2 Binary array files

Every binary file starts with a header, which has the following format:

```
[INT32] 65 (magic identifier)
[INT32] b: Format: number of bits per number. Currently supported are 1 and 8
[INT32] N: number of rows
[INT32] k: kumber of columns
[INT32] Number of arrays (can be -1 if unknown)
[INT32] Binary format number: 1001: normal, 1002: binary diff, 1003: binary diff zero
[INT32] Reserved integer
[INT32] Reserved integer
```

The normal binary format has the following format. For each array (the number is specified in the header):

```
[INT32] Index
[Nxk elements] The elements contain b bits
```

If the number of bits per number is 1 (e.g. a 2-level array) then the data is padded with zeros to a multiple of 64 bits. The data of the array is stored in column-major order. The binary file format allows for random access reading and writing. The `binary diff` and `binary diff zero` formats are special formats.

A binary array file can be compressed using gzip. Most tools in the Orthogonal Array package can read these compressed files transparently. Writing to compressed array files is not supported at the moment.

2.2.3 Data files

The analysis tool (`oaanalyse`) writes data to disk in binary format. The format is consists of a binary header:

```
[FLOAT64] Magic number 30397995;
[FLOAT64] Magic number 12224883;
[FLOAT64] nc: Number of rows
[FLOAT64] nr: Number of columns
```

After the header there follow `nc*nr [FLOAT64]` values.

2.3 Statistical properties of an array

Most properties of an array can be calculated using the `array_link` object. The interface is listed below.

C++ interface

```
struct array_link
{
    ...

public:
    // statistical properties of the array

    /// calculate D-efficiency
    double Defficiency() const;
    /// calculate main effect robustness (or Ds optimality)
    double DsEfficiency(int verbose=0) const;
    /// calculate A-efficiency
    double Aefficiency() const;
    /// calculate E-efficiency
    double Eefficiency() const;
    /// calculate rank of array
    int rank() const;
    /// calculate generalized wordlength pattern
    std::vector<double> GWLP() const;
    /// return true if the array is a foldover array
    bool foldover() const;
    /// calculate centered L2 discrepancy
    double CL2discrepancy() const;
    /// Calculate the projective estimation capacity sequence
    std::vector<double> PECsequence() const;
}
```

The D-efficiency, A-efficiency and E-efficiency are calculated by calculating the SVD of the second order interaction matrix. The efficiencies can then be calculated using the eigenvalues of the SVD. For the definition of the D-, A- and E-efficiency see Definition 1. For the rank of a matrix the LU decomposition of the matrix is calculated using the Eigen package [Guennebaud et al., 2010].

Definition 1 (D-efficiency and average VIF)**.** Let X be an $N \times k$ 2-factor array with second order model $F(X)$. Then we define the D-efficiency and the average variance inflation factor as

$$D(X) = \left(\det F(X)^T F(X)\right)^{1/m} / N, \tag{1}$$

$$\mathrm{VIF}(X) = N \operatorname{tr}\left(\frac{1}{F(X)^T F(X)}\right) / m. \tag{2}$$

The matrix $F(X)^T F(X)$ is called the information matrix. Let $\lambda_1, \ldots, \lambda_m$ be the eigenvalues of the information matrix. Then the E-efficiency of a matrix is defined as

$$E(X) = \min_j \lambda_j. \tag{3}$$

Note that in terms of the eigenvalues we have $D(X) = (\prod_j \lambda_j)^{1/m}/N$ and $\mathrm{VIF}(X) = N(\sum_j \lambda_j^{-1})/m$.

The D_s-effiency is the main effect robustness, see the appendix in [Schoen, 2010] for more details.

2.4 Calculation of D-optimal designs

D-optimal designs can be calculated with the function `Doptimize`. This function uses a coordinate exchange algorithm to generate designs with good properties for the D-efficiency.

An example script with Python to generate optimal designs with 40 runs and 7 factors is shown below.

Doptimize

```
>>> N=40; s=2; k=7;
>>> arrayclass=oapackage.arraydata_t(s, N, 0, k)
>>> print('We generate optimal designs with: %s.\n' % arrayclass)
We generate optimal designs with: arrayclass: N 40, k 7, strength 0, s {2,2,2,2,2,2,2}, order 0.
>>> alpha=[1,2,0]
>>> method=oapackage.DOPTIM_UPDATE
>>> scores, dds, designs, ngenerated = oapackage.Doptimize(arrayclass, nrestarts=40, optimfunc=alpha, selectpar
Doptim: optimization class 40.2-2-2-2-2-2-2
Doptimize: iteration 0/40
Doptimize: iteration 39/40
Doptim: done (8 arrays, 0.6 [s])
>>> print('\nGenerated %d designs, the best D-efficiency is %.4f' % (len(designs), dds[:,0].max() ))
Generated 8 designs, the best D-efficiency is 0.9098
```

The parameters of the function are documented in the code.

To calculate properties of designs we can use the following functions. For D-efficiencies we can use

C++ interface

```
std::vector<double> array_link::Defficiencies ( int verbose ) const;
```

to calculate the D-, D_s- and D_1-efficiency. For details see [Eendebak and Schoen, 2015].

The projective estimation capacity (PEC) sequence from [Loeppky, 2004] can be calculated with:

C++ interface

```
std::vector<double> PECsequence(const array_link &al, int verbose=1);
```

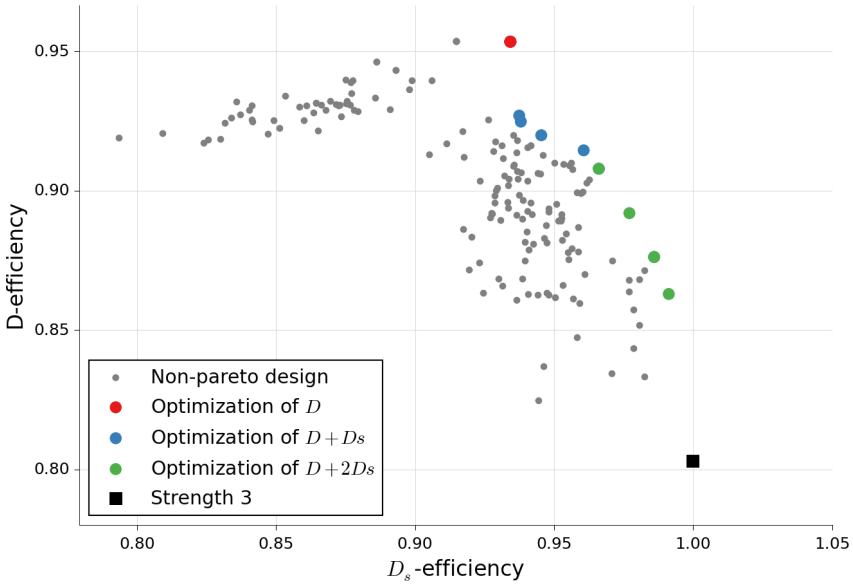


Figure 1: Scatterplot for the D-efficiency and D_s-efficiency for generated designs in OA$(40; 2; 2^7)$. The Pareto optimal designs are colored, while the non-Pareto optimal designs are grey. For reference the strength-3 orthogonal array with highest D-efficiency is also included in the plot.

2.5 GWLP and J-characteristics

From an `array_link` object we can calculate the generalized worldlength patterns [Xu and Wu, 2001], F-values and J-characteristics.

Calculate GWLP and F-values

```
>>> al=oapackage.exampleArray(1)
>>> al.showarray()
array:
  0   0   0   0   0
  0   0   0   0   0
  0   0   0   1   1
  0   0   1   0   1
  0   1   0   1   0
  0   1   1   0   0
  0   1   1   1   1
  0   1   1   1   1
  1   0   0   1   1
  1   0   1   0   1
  1   0   1   1   0
  1   0   1   1   0
  1   1   0   0   1
  1   1   0   0   1
  1   1   0   1   0
  1   1   1   0   0

>>> g=al.GWLP()
>>> print('GWLP: %s'% str(g) )
GWLP: (1.0, 0.0, 0.0, 1.0, 1.0, 0.0)
>>> print('F3-value: %s' % str(al.Fvalues(3)))
F3-value: (4, 6)
>>> print('F4-value: %s' % str(al.Fvalues(4)))
F4-value: (1, 4)
>>> print('J3-characteristis: %s' % str(al.Jcharacteristics(3)))
J3-characteristis: (8, 8, 0, 0, 0, 8, 0, 8, 0, 0)
```

2.6 Reduction to canonical form

If we introduce an ordering on the set of arrays, then for each isomorphism class of arrays the minimal element defines a unique canonical form. The Orthogonal Array package contains functions to reduce any orhogonal array to canonical form with respect to some ordering. The default ordering for arrays is the lexicographic ordering in columns [Schoen et al., 2010]. An alternative ordering is the delete-one-factor projection ordering as described in [Eendebak, 2013].

C++ interface

```
/// reduce an array to canonical form using delete-1-factor ordering
array_link reduceLMCform(const array_link &al);

/// reduce an array to canonical form using delete-1-factor ordering
array_link reduceDOPform(const array_link &al);
```

Another approach to generation of canonical forms for designs is to use graph-isomophism packages such as Nauty [McKay, 1981, McKay and Piperno, 2013] or Bliss [Junttila and Kaski, 2007]. To reduce a general graph to Nauty canonical form one can use `reduceNauty`. For orthogonal arrays we can encode the array structure as a graph. The reduction can then be done with `reduceOAnauty`.

Reduce a design to normal form using Nauty

```
>>> al = oapackage.exampleArray(0).randomperm()
>>> al.showarray()
array:
  0   1
  1   1
  1   0
  0   0
  0   0
  1   1
  1   0
  0   1
>>> t=oapackage.reduceOAnauty(al, verbose=0)
>>> t.show()
array transformation: N 8
column permutation: {0,1}
level perms:
{0,1}
{0,1}
row permutation: {3,4,0,7,2,6,1,5}
>>> alr=t.apply(al)
>>> alr.showarray()
array:
  0   0
  0   0
  0   1
  0   1
  1   0
  1   0
  1   1
  1   1
```

2.7 Generation of arrays

A list of arrays in LMC form can be extended to a list of arrays in LMC form with one additional column. Details for the algorithm are described in [Schoen et al., 2010].

The main function for array extension is the following:

C++ interface

```
/// extend a list of arrays
arraylist_t & extend_arraylist(arraylist_t & alist, arraydata_t &fullad,
                        OAextend const &oaextendoptions);
```

Here `fullad` is the structure describing the type of arrays and `oaxextendoptions` contains various options for the algorithm.

A typical session could be:

Extend an array

```
>>> N=8; ncols=3;
>>> arrayclass=oapackage.arraydata_t(2, N, 2, ncols)
>>> al=arrayclass.create_root()
>>> al.showarray()
array:
  0   0
  0   0
  0   1
  0   1
  1   0
  1   0
  1   1
  1   1
>>>
>>> alist=oapackage.extend_array(al, arrayclass)
>>> for al in alist:
...       al.showarray()

array:
  0   0   0
  0   0   0
  0   1   1
  0   1   1
  1   0   1
  1   0   1
  1   1   0
  1   1   0

array:
  0   0   0
  0   0   1
  0   1   0
  0   1   1
  1   0   0
  1   0   1
  1   1   0
  1   1   1
```

2.8 Conference designs

An $n \times k$ conference design is an $N \times k$ matrix with entries 0, -1, +1 such that i) in each column the symbol 0 occurs exactly one time and ii) all columns are orthogonal to each other. A more detailed description is given in [Wikipedia, 2016].

Generate conference designs with 8 rows

```
>>> import oapackage
>>> ctype=oapackage.conference_t(N=8, k=8)
>>>
>>> al = ctype.create_root_three()
>>> al.showarray()
array:
  0   1   1
  1   0  -1
  1   1   0
  1   1   1
  1   1  -1
  1  -1   1
  1  -1   1
  1  -1  -1
>>> l4=oapackage.extend_conference ( [al], ctype, verbose=0)
>>> l5=oapackage.extend_conference ( l4, ctype, verbose=0)
>>> l6=oapackage.extend_conference ( l5, ctype, verbose=0)
>>>
>>> print('number of non-isomorphic conference designs: %d'  % len(l6) )
number of non-isomorphic conference designs: 11
```

2.9 MD5 sums

To check data structures on disk the packages includes functions to generate MD5 sums. These are:

C++ interface

```
/// calculate md5 sum of a data block in memory
std::string md5(void *data, int numbytes);
/// calculate md5 sum of a file on disk
std::string md5(const std::string filename);
```

3 Command line interface

Included in the packages are several command line tools. For each tool help can be obtained from the command line by using the switch `-h`. These are:

`oainfo` This program reads Orthogonal Array package data files and reports the contents of the files. For example:

```
~eendebakpt$ oainfo result-8.2-2-2-2.oa
Orthogonal Array package 1.8.7
oainfo: reading 1 file(s)
file result-8.2-2-2.oa: 8 rows, 3 columns, 2 arrays, mode text, nbits 0
~eendebakpt$
```

`oacat` Show the contents of a file with orthogonal arrays for a data file.

`oacheck` Check or reduce an array to canonical form.

`oaextendsingle` Extend a set of arrays in LMC form with one or more columns.

`oacat` Show the contents of an array file or data file.

```
Usage: oacat [OPTIONS] [FILES]
```

`oajoin` Read one or more files from disk and join all the array files into a single list.

```
Orthogonal Arrays 1.8.7
For more details see the files README.txt and LICENSE.txt

Orthonal Array Join: join several array files into a single file
Usage: oajoin [OPTIONS] [FILES]

 -h --help                      Prints this help
 -s --sort                      Sort the arrays
 -l --latex                     Output with LaTeX format
 -o [FILE] --output [FILE]      Output prefix (default: standard output)
 -f [FORMAT]                    Output format (TEXT, BINARY (default),
                                D (binary difference) )
```

`oasplit` Takes a single array file as input and splits the arrays to a specified number of output files.

`oapareto` Calculates the set of Pareto optimal arrays in a file with arrays.

`oaanalyse` Calculates various statistics of arrays in a file. The statistics are described in section 2.3.

References

[Eendebak, 2012] Eendebak, P. T. (2012). Orthogonal array page.

[Eendebak, 2013] Eendebak, P. T. (2013). A canonical form for non-regular arrays based on generalized worldlength pattern values of delete-one-factor projections. working paper of University of Antwerp.

[Eendebak and Schoen, 2015] Eendebak, P. T. and Schoen, E. D. (2015). Two-level orthogonal arrays to estimate all main effects and two-factor interactions. to be submitted.

[Guennebaud et al., 2010] Guennebaud, G., Jacob, B., et al. (2010). Eigen v3. http://eigen.tuxfamily.org.

[Hedayat et al., 1999] Hedayat, A., Sloane, N., and Stufken, J. (1999). *Orthogonal arrays : theory and applications*. Springer.

[Hunter, 2007] Hunter, J. D. (2007). Matplotlib: A 2D graphics environment. *Computing In Science & Engineering*, 9(3):90–95.

[Junttila and Kaski, 2007] Junttila, T. and Kaski, P. (2007). Engineering an efficient canonical labeling tool for large and sparse graphs. In Applegate, D., Brodal, G. S., Panario, D., and Sedgewick, R., editors, *Proceedings of the Ninth Workshop on Algorithm Engineering and Experiments and the Fourth Workshop on Analytic Algorithms and Combinatorics*, pages 135–149. SIAM.

[Loeppky, 2004] Loeppky, J. (2004). *Ranking Non-regular Designs*. Canadian theses on microfiche. Simon Fraser University.

[McKay, 1981] McKay, B. (1981). Practical graph isomorphism. *Congressus Numerantium*, 30:45–87.

[McKay and Piperno, 2013] McKay, B. D. and Piperno, A. (2013). Practical graph isomorphism, ii. *CoRR*, abs/1301.1493.

[Rao, 1947] Rao, C. (1947). Factorial experiments derivable from combinatorial arrangements of arrays. *Journal of the Royal Statistical Society Supplement*, 9:128–139.

[Schoen, 2010] Schoen, E. (2010). Optimum designs versus orthogonal arrays for main effects and two-factor interactions. *Journal of Quality Technology*, 42:197–208.

[Schoen et al., 2010] Schoen, E. D., Eendebak, P. T., and Nguyen, M. V. M. (2010). Complete enumeration of pure-level and mixed-level orthogonal arrays. *Journal of Combinatorial Designs*, 18(2):123–140.

[Sloane, 2014] Sloane, N. (2014). A library of orthogonal arrays. http://neilsloane.com/oadir/index.html.

[The Scipy community, 2012] The Scipy community (2012). *NumPy Reference Guide*. SciPy.org.

[Wikipedia, 2016] Wikipedia (2016). Conference matrix — wikipedia, the free encyclopedia. [Online; accessed 4-April-2016].

[Xu and Wu, 2001] Xu, H. and Wu, C. F. J. (2001). Generalized minimum aberration for asymmetrical fractional factorial designs. *Annals of Statistics*, 29:1066–1077.