Department of Mathematics and Computer Science
University of Southern Denmark, Odense

December 14, 2008
Marco Chiarandini

DM811 - Heuristics for Combinatorial Optimization

Re-exam, Fall 2008

Note 1 The project is carried out individually and it is not allowed to collaborate. It consists of algorithm design, implementation, experimentation and written report.

The evaluation of the project is based on the report. However, a program that implements the best algorithm described in the report must also be submitted. The program will serve to verify the correctness of the results presented. The report may be written in either English or Danish.

Note 2 Corrections or updates to the project description will be published on the course web page and will be announced by email to the addresses available in the Blackboard system. In any case, it remains students' responsibility to check for updates on the web page.

Note 3 *Submission*. **Two** printed copies of the written report must be handed in at the secretary office **before 12:00 of Thursday, January 22, 2009**. Ask the secretary a receipt of the submission.

In addition, with the same deadline, an electronic archive containing the source code of the program must be sent by email to the lecturer (marco@imada.sdu.dk). A return email will be sent as receipt within one day. See Appendix B for details on how to organize the electronic archive.

In order to fulfill the requirements of the exam both report and code must be handed in within the deadline. Late submissions will generally not be accepted. System failures, illness, etc. will not automatically give extra time.

---

1 Problem Description

Let $x = \{x_0, x_1, \dots x_{N-1}\}$ be a sequence of N elements of value +1 or -1. The aperiodic correlations are defined by

$$c_k = \sum_{j=0}^{N-k-1} x_j x_{j+k} \qquad k = 1, ..., N-1.$$

The *energy* of the sequence is defined by

$$E(x) = \sum_{k=1}^{N-1} c_k^2$$

and the *merit factor*, introduced by M. Golay [M.J77], by

$$M = \frac{N^2}{2E}.$$

N	$E(x)$									
1-10	0	0	1	2	2	7	3	8	12	13
11-20	5	10	6	19	15	24	32	25	29	26
21-30	26	39	47	36	36	45	37	50	62	59
31-40	67	64	64	65	73	82	86	87	99	108
41-50	108	101	109	122	118	131	135	140	136	153
51-60	153	166	170	175	171	192	188	197	205	218
61-70	226	235	207	208	240	265	241	250	274	295
71-77	275	300	308	349	341	338	366			

Table 1: Best known results for various N. Up to $N = 60$ these results are optimal.

The Merit Factor Problem (MFP) asks for a given N to find the sequence that maximizes the merit factor, or equivalently, one that minimizes the energy.

This is an example of energy and merit computation for two sequences of length $N = 3$.

$x = \{1, 1, -1\}$	$x = \{1, 1, 1\}$
$C_1 = (1 * 1) + (1 * -1) = 1 + (-1) = 0$	$C_1 = (1 * 1) + (1 * 1) = 2$
$C_2 = (1 * -1) = -1$	$C_2 = (1 * 1) = 1$
$E(x) = C_1^2 + C_2^2 = 0^2 + (-1)^2 = 0 + 1 = 1$	$E(x) = 2^2 + 1^2 = 5$
$F(x) = 3^2/(2 * 1) = 9/2 = 4.5$	$F(x) = 3^2/(2 * 5) = 0.9$

Binary sequences with small aperiodic correlations play an important role in many applications in communication and electrical engineering, ranging from radar to modulation and testing of systems. In particular, the merit factor relates with spectral properties of the signal corresponding to the sequence. It measures how much the amplitude spectrum of the signal deviates from the constant value N, and a sequence with maximal merit factor M gives a signal with maximally flat spectrum for a fixed N. For a list of known algebraic results on this problem see [Høh06].

In Table 1 we report the best results in terms of energy. Up to $N = 60$ the results are proven optimal, for $N \geq 61$ they have been obtained by a state-of-the-art heuristic solver. The details of the solutions provided by this solver are reported in Table 2.

2 Project Requirements

All the following points must be addressed to pass the exam:

1. Design and implement one or more construction heuristics.
2. Design and implement one or more local search algorithms.
3. Design and implement a high performing algorithm enhancing the heuristics at the previous two points with the use of stochastic local search methods.
4. For all the methods above carry out an experimental analysis and draw sound conclusions. For the algorithms at point 3, compare the results with those of Table 2, limiting your computation times to a maximum of 5 minutes per run.[1]

[1]Times refer to machines in IMADA terminal room, Intel Core 2 CPU 6300 at 1.86GHz, with 2048 KB cache and 2 GB RAM, running an Ubuntu 8.04 distribution of Linux with kernel 2.6.24-19-generic.

N	$E(x)$	$F(x)$	Runtime	Limit
61	226	8.23	3 m	1.1 h
62	235	8.18	8 m	1.5 h
63	207	9.59	4 m	2.0 h
64	208	9.85	47 m	2.7 h
65	240	8.80	2.2 h	3.7 h
66	265	8.22	3.1 h	4.9 h
67	241	9.31	4.1 h	6.6 h
68	250	9.25	6.6 h	8.8 h
69	274	8.69	8.2 h	11.8 h
70	295	8.31	12.4 h	15.8 h
71	275	9.17	7.8 h	10.0 h
72	300	8.64	2.4 h	10.0 h
73	308	8.65	1.2 h	10.0 h
74	349	7.85	0.2 h	10.0 h
75	341	8.25	8.0 h	10.0 h
76	338	8.54	4.6 h	10.0 h
77	366	8.10	3.9 h	10.0 h

Table 2: Best results obtained by a state-of-the-art heuristic solver. Times refer to a 2.33 GHz Core2 Duo PC.

3 Remarks

Remark 1 Only results on instances with $N \geq 55$ should be discussed in the report.

Remark 2 For each point above a description must be provided in the report of the work undertaken. In particular for the best algorithms arising from the experimental analysis enough details must be provided in order to guarantee the reproducibility of the algorithm from the report only (i.e., without having to look at the source code).

Remark 3 Besides the description of the algorithm, it will make a case for higher grade an analysis on the computational cost of the procedures implemented and details on the data structure used.

Remark 4 The results of the experiments must be reported either in graphical form or in form of tables. Moreover, for the best solver resulting from the point 3, a table must be provided with the best results for each specific N of Table 2.

Remark 5 The total length of the report should not be less than 8 pages and not be more than 14 pages, appendix included (lengths apply to font size of 11pt and 3cm margins). Although these bounds are not strict, their violation is highly discouraged. In the description of the algorithms, it is allowed (and encouraged) to use short algorithmic sketches in form of pseudo-code but not to include program codes.

Remark 6 This is a list of factors that will be taken into account in the evaluation:

- quality of the final results;
- level of detail of the study;

- complexity and originality of the approaches chosen;
- originality of the experimental questions;
- organization of experiments which guarantee reproducibility and correctness of the conclusions;
- clarity of the report;
- effective use of graphics in the presentation of experimental results.

Appendix A Solution Format

In order to check the validity of the results reported the program submitted must output the best solution in a file when finishing. The format of the file is a row containing a sequence of +1,-1 separated by commas.
Example:

```
-1,1,-1,1,-1,-1,1,1,1,1,-1,-1,1,-1,-1,-1,1,-1,-1,1,1,-1,1,1,1,1
```

Appendix B Handing in Electronically

The electronic archive to hand in must be organized as follows. It expands in a main directory:

```
CPRN/
```

where CPRN is the student's first 6 digits of the CPR number (e.g., 030907) and its content:

```
CPRN/README
CPRN/Report/
CPRN/src/
```

The file README contains the manual for the compilation of the program. The directory src contains the sources which may be in C, C++, Java or other languages. If needed a Makefile can be included either in the root directory or in src. After compilation the executable must be placed in `src`. For java programs, a jar package can also be submitted.

Programs must work on IMADA's computers under Linux environment and with the compilers and other applications present on IMADA's computers. Students are free to develop their program at home, but it is their own responsibility to transfer the program to IMADA's system and make the necessary adjustments such that it works at IMADA.[2]

The executable must be called `mfp`. It will be run by typing in the directory CPRN/src/:

```
mfp -i INSTANCE -t TIME -s SEED -o OUTPUT
```

- `-n N` to define the N solved.
- `-t TIME` to stop the program execution after TIME seconds.
- `-s SEED` to initialize the random generator.

[2]Past issue: the java compiler path is `/usr/local/bin/javac`; in C, any routine that uses subroutines from the math.c library should be compiled with the `-lm` flag – eg, `cc floor.c -lm`.

- `-o OUTPUT` the file name where the solution is written

For example:

```
mfp -n 60 -o mfp-60.sol -t 300 -s 1
```

will run the program for a sequence of length N for 300 seconds with random seed 1 and write the solution in the file `mfp-60.sol`.

It is advisable to have a log of algorithm activities during the run. This can be achieved by printing further information on the standard error or in a file. A suggested format is to output a line whenever a new best solution is found containing at least the following pieces of information:

```
best 301 time 10.000000 iter 1000
```

All process times are the sum of user and system CPU time spent during the execution of a program as returned by the linux library routine `getrusage`. Process times include the time to read the instance.

References

[Høh06] Tom Høholdt. The merit factor problem for binary sequences. In *Applied Algebra, Algebraic Algorithms and Error-Correcting Codes*, Lecture Notes in Computer Science, pages 51–59. Springer, 2006.

[M.J77] M.J.E.Golay. Sieves for low autocorrelation binary sequences. *IEEE Trans. Inform. Theory*, IT-23(1):43–51, 1977.