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

October 22, 2012
Marco Chiarandini

DM811 - Heuristics for Combinatorial Optimization

Exam Project, Fall 2012

Remark 1 The project is carried out individually and it is not allowed to collaborate.

Remark 2 The project 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 be used to verify the correctness of the results presented. The report may be written in Danish or in English.

Remark 3 Corrections or updates to the project description, if any, 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 new announcements.

Remark 4 *Submission.* An archive containing the electronic version of the written report and the source code of the program must be handed in through the BlackBoard system with deadline

Friday, November 2, 2012 at 14:00.

The submission procedure is the following:

- choose the course DM811 in BlackBoard,
- choose "SDU Assignment" in the menu on the left,
- fill the form and conclude with submit,
- print and conserve the receipt (there will be a receipt also per email).

See Appendix B for details on how to organize the electronic archive.

In addition to the electronic submission you must deposit two printed copies of your report at the teacher's mailbox in the secretary office.

Reports and codes handed in after the deadline will generally not be accepted. System failures, illness, etc. will not automatically give extra time.

Remark 5 Write your name, your CPR number and your user ID as it appeared during the graph coloring assignments in the front page of the report.

Remark 6 Make sure you have read the whole document before you start to work.

Remark 7 The evalautions of the exams will be sent to the study office not later than three weeks after the deadline has expired.

1 The Problem

A board consists of $N \times N$ cells arranged in an $N \times N$ grid. Each cell is connected to at most eight neighbors along the four axes: vertical, horizontal and two diagonal. A wrap-around board has connections also between the left-most and right-most columns and between the top and bottom rows.

For two strings s and r of K bits $b_1 b_2 \dots b_K$ with $b_i \in \{0, 1\}$ the Hamming distance $d(s, r)$ is the number of bits at which the two strings differ.

Given a wrap-around board $N \times N$ and an integer $D \in \mathbb{Z}^+$, we wish to find an assignment of distinct strings all made of $K \geq \lceil 2 \log_2 N \rceil$ binary bits to each cell of the board in such a way that the maximal Hamming distance between any pair of neighboring cells is smaller than or equal to D. In the optimization sense, given a wrap-around board $N \times N$ we wish to determine the smallest D for which such assignment exists.

For example, the board 4×4 and bit strings of length $K = \lceil 2 \log_2(4) \rceil = 4$ admits a solution for $D = 2$ (see Figure 1) but it does not admit one for $D = 1$. Hence, $D = 2$ is the best possible maximal Hamming distance for this case. The binary strings can be represented by integer numbers in base-10 in which case we wish to assign to each of the N^2 cells a distinct integer from $\mathbb{Z}_{2^K}$.

In the following we will identify an instance of the problem by the triple (N, K, D).

2 Motivation

A current focus in management science is the investigation of human decision-making when searching for new alternatives. Experimental work in this area uses the NK-model of rugged performance landscapes [Kau93].

The NK model defines a combinatorial search space, consisting of every (binary) string of length N. For each string in this search space, a scalar value, called the fitness, is defined. If a distance metric is also defined between strings, the resulting structure is a landscape. Fitness values are defined according to a specific model, but the key feature of the NK model is that the fitness of a given string s is the sum of contributions from each bit (locus) b_i in the string:

$$F(s) = \sum_{i=1}^{N} f(b_i)$$

and the contribution from each bit in general depends on the bit itself and the value of K other bits:

$$f(b_i) = f(b_i, b_1^i, \dots, b_K^i)$$

where b_j^i are the other bits upon which the fitness of b_i depends. Hence, the fitness function is a mapping between strings of length $K + 1$ and scalar values. (Source: `http://en.wikipedia.org/wiki/NK_model`.) An example of NK landscapes at varying K is given in Figure 2.

In management experiments, human subjects are asked to search for high-performing product configurations. They combine several attributes — the N parameter in the NK model — to specify a product configuration. The value or payoff of a particular configuration is initially unknown and is only discovered after trying out the configuration. The complexity of alternatives — the K parameter — is captured by allowing for interactions among the attributes in the payoff function. As complexity

1111	0101	1100	0110
0111	1101	0100	1110
1011	0001	1000	0010
0011	1001	0000	1010

15	5	12	6
7	13	4	14
11	1	8	2
3	9	0	10

Figure 1: An assignment with $D = 2$ for the 4×4 board. On the left the binary strings disposed on the board; on the right the representation of the strings in base-10.

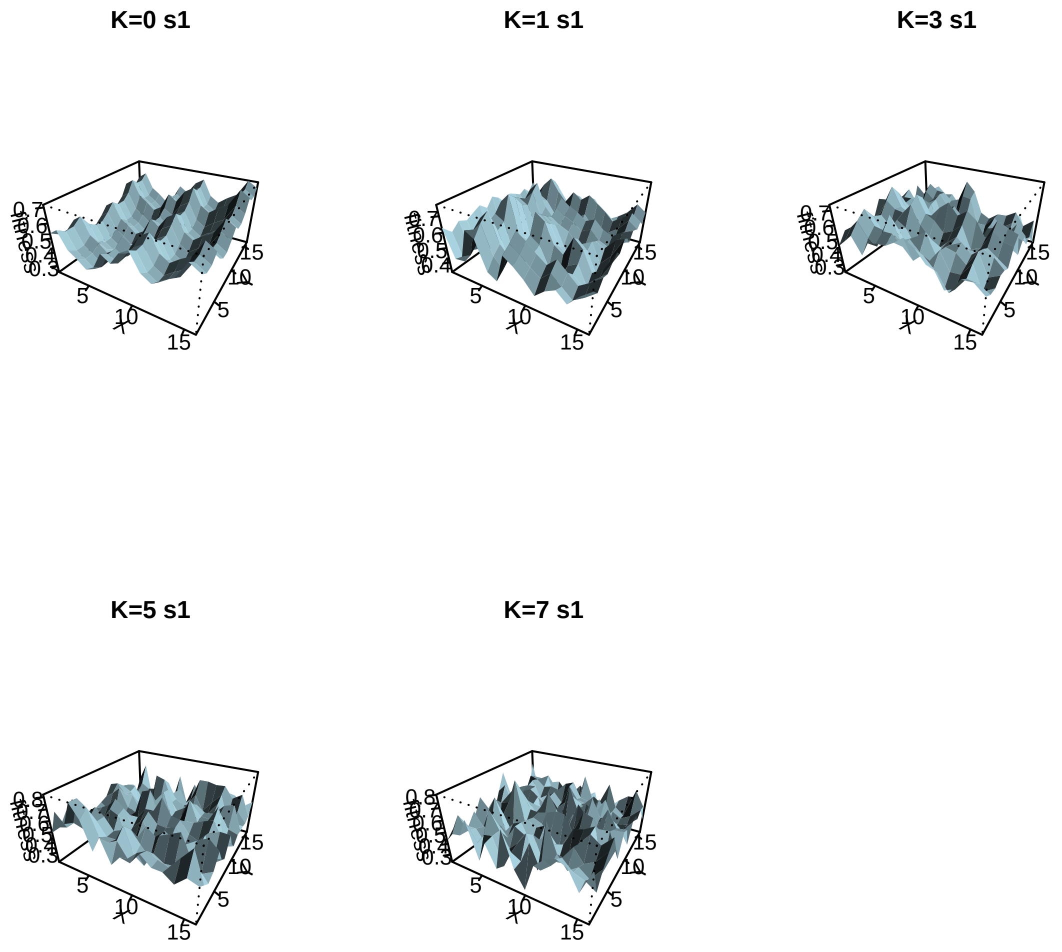

Figure 2: Different NK landscapes.

increases and interactions among attributes proliferate, the problem of finding a high-performing configuration becomes more challenging and difficult. The subjects in the experiments have only a limited number of search trials, far fewer than the number of all possible configurations. The aim of the experiments is then studying the opportunity cost of searching a new alternative varying the experimental treatment of performance landscape complexity K.

A dominant search strategy detected in the experiments is *neighborhood search*: the move to another position occurs by changing from 1 to 3 attributes. More specifically, for $N = 10$, experiments showed that out of 7539 active searches, 75.9% were within distance 3 from the current position with an average distance of 2.53, 84.5% within distance 4 and 90.3% within distance 5. In other terms, humans do not change radically the attributes available but perform local changes. Other observations show the importance of performance feedback [BSS10].

Given these results there is interest to design an experiment in which the distance factor is removed. In other terms, researchers in management would like to have an NK model in which the user is given the possibility to only move to new search states that are in the neighborhood of the current one. A way to achieve this is to provide to the humans an easy graphical representation of the possible moves available. A chess board in which moves are only allowed between neighboring cells seems well suited for this task. The problem is then to dispose the 2^K different search alternatives in the board in such a way that the maximum distance between any pair of adjacent cells is minimized.

3 Known facts

Søren Haagerup, a former student of this course, has shown that for $K = \lceil 2\log_2 N \rceil$:

1. the problem can be reformulated in graphs terms as the problem of finding an embedding f of the quadri-axial torus G in the Hamming graph $H(K)$ with restricted dilation of D. Since H has no clique of size 3 but there are several cliques of size 3 in G then there exist no solution with $D = 1$ for any N.
2. there is an easy method to construct solutions with $D = 2$ for all N that are powers of 2, ie, $N = 2^i$, for $i = 1, 2, 3, \ldots$
3. for any $N > 1$ with $2\lceil \log_2(N) \rceil = \lceil 2\log_2(N) \rceil$, one can construct a solution with $D = 2$ if N is even and with $D = 4$ if N is odd. For example:

N	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
$2\lceil \log_2(N)$	4	4	6	6	6	6	8	8	8	8	8	8	8	8	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10
$\lceil 2\log_2(N) \rceil$	4	4	5	6	6	6	7	7	7	8	8	8	8	8	9	9	9	9	9	9	10	10	10	10	10	10	10	10	10
UB	2	4	–	4	2	4	–	–	–	4	2	4	2	4	–	–	–	–	–	–	2	4	2	4	2	4	2	4	2

Table 1: Upper bounds by obtained by construction for N=2,...,30.

4 Open problems

- We do not know if the theoretical lower and upper bounds could get tighter for odd N and $2\lceil \log_2(N) \rceil = \lceil 2\log_2(N) \rceil$. For example: Does there exist an odd N, with solutions for $D = 2$? The first case without answer up to now is $(N, K, D) = (7, 6, 2)$.
- For all N where $2\lceil \log_2(N) \rceil \neq \lceil 2\log_2(N) \rceil$ can we get some upper bounds?
- Table 2 reports the best impossibility and possibility results for D when $K = \lceil 2\log(N) \rceil$. An assignment with the best known D has been found either by constructive method or by computational methods such as integer programming and constraint programming. Instances with an asterisks are closed, that is the best possible D known is also the optimal.

5 Project Requirements

The aim of the project is to study heuristic algorithms for solving the *decision version* of the problem described above. An instance of the problem is fully determined simply by the triple (N, K, D). Since very little is known about the hardness of theses instances we will initially focus on instances (N, K, D) with $K = \lceil 2\log(N) \rceil$, $N = \{2, \ldots, 20\}$, and $\{D = 2, \ldots, 10\}$. (If these instances turn out to be too easy you may increase N).

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

1. Design and implement one or more stochastic local search or metaheuristics algorithms.

N	K	Impossible	Possible	
2	2	1	2	*
3	4	3	4	*
4	4	1	2	*
5	5	2	3	*
6	6	1	2	*
7	6	1	3	
8	6	1	2	*
9	7	1	3	
10	7	1	3	
11	7	1	3	
12	8	1	2	*
13	8	1	3	
14	8	1	2	*
15	8	1	3	
16	8	1	4	
17	9	1	3	
18	9	1	3	
19	9	1	3	
20	9	1	3	

Table 2: Best known results in terms of D for instances with N from 2 to 20 and $K = \lceil 2\log(N) \rceil$

2. Carry out an experimental comparison or configuration of the algorithms at the previous point.
3. Report the results in a table like Table 3. In the cell write the time in seconds that your algorithm needed to find a solution to the instance. Limit the run time of your algorithm to a maximum of 2 minutes for each instance (N, K, D).[1]
4. Describe the work in the previous points in a report.

6 Remarks

Remark 1 For each point above a description must be provided in the report of the work undertaken. In particular for the best algorithm arising from the experimental analysis enough details must be provided in order to guarantee the reproducibility of the algorithm from the report alone (i.e., without need for looking at the source code). It is important to give account of the computational cost of the operations in the local search.

Remark 2 The total length of the report should not be less than 5 pages and not be more than 12 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 3 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;

[1]Times refer to machines in IMADA terminal room.

	1	2	3	4	5	6	7	8	9	10
1	–	–	–	–	–	–	–	–	–	–
2	–	–	–	–	–	–	–	–	–	–
3	–	–	–	–	–	–	–	–	–	–
4	–	–	–	–	–	–	–	–	–	–
5	–	–	–	–	–	–	–	–	–	–
6	–	–	–	–	–	–	–	–	–	–
7	–	–	–	–	–	–	–	–	–	–
8	–	–	–	–	–	–	–	–	–	–
9	–	–	–	–	–	–	–	–	–	–
10	–	–	–	–	–	–	–	–	–	–
11	–	–	–	–	–	–	–	–	–	–
12	–	–	–	–	–	–	–	–	–	–
13	–	–	–	–	–	–	–	–	–	–
14	–	–	–	–	–	–	–	–	–	–
15	–	–	–	–	–	–	–	–	–	–
16	–	–	–	–	–	–	–	–	–	–
17	–	–	–	–	–	–	–	–	–	–
18	–	–	–	–	–	–	–	–	–	–
19	–	–	–	–	–	–	–	–	–	–
20	–	–	–	–	–	–	–	–	–	–

Table 3: Example of table for presenting the final results: on the rows N and on the columns D.

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

Remark 4 In the project requirements of Sec. 5 the words "one or more algorithms" do NOT imply "the more algorithms, the better the final grade". A few, well thought algorithms are better in this sense than many naive ones!

Appendix A Solution format and solution checker

The number in decimal representation assigned to each cell of the board must be written in a text file with one number per line and the board visited in row-wise order.

A C++ program to check solutions in this format is available at course webpage. Compile the program with

```
g++ checker.cpp -o checker
```

and run the program with

```
checker -n 16 -k 8 -d 2 -c 16-8-2.sol
```

The file 16-8-2.sol is also available at the course web-page.

Appendix B Handing in Electronically

Your work must be handed in electronically via BlackBoard. In addition two printed copies of the report must be deposited at the teacher's mailbox in the secretary office. The official receipt will be obtained at the BlackBoard submission.

This section describes how you must organize your electronic submission.

Main directory:

```
userID/
```

where `userID` is your login at SDU email

```
userID/README
userID/Makefile
userID/src/
userID/bin/
userID/res/
userID/doc/
```

The directory `doc` contains a pdf or postscript version of your report. The file README provides instructions for 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 `bin`. For java programs, a jar package can also be created via Makefile.

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 `nk`. It must execute from command line by typing in the directory `userID/bin/nk`:

```
nk -n N -k K -d D -s SEED -c OUTPUT -t TIME
```

where beside the obvious flags to indicate the instance it is:

[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`.

- `-t TIME` the time limit in seconds;
- `-s SEED` the random seed;
- `-c OUTPUT` the file name where the solution is written.

For example:

```
nk -n 4 -k 4 -d 3 -c 4-4-3.sol -t 120 -s 1 > 4-4-3.log
```

will run the program on the instance $(4, 4, 3)$ for 120 seconds with random seed 1 and write the solution in the file `4-4-3.sol`.

In its default mode, the program must run the best algorithm developed and must print on the standard output 0 if the program never found a solution when it terminates and 1 if the program found a solution written in the solution file.

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 53 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 C library routine `getrusage`. If you are using the frameworks made available in the Graph Coloring Assignments continue to use the same functions for time checking.

References

[BSS10] Stephan Billinger, Nils Stieglitz, and Terry R. Schumacher. Search on rugged landscapes: An experimental study. *SSRN eLibrary*, 2010.

[Kau93] Stuart A. Kauffman. *The Origins of Order: Self-Organization and Selection in Evolution.* Oxford University Press, 1993.