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

March 21, 2010
Marco Chiarandini

DM811 - Heuristics for Combinatorial Optimization

Exam Project, Fall 2009

Note 1 The project is carried out individually and it is not permitted 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*. 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 **before 15:00 of Monday, April 19th, 2010**. This is the procedure:

- choose the course DM811 in Blackboard,
- choose "Exam Project Hand in" in the menu on the left,
- fill the form and conclude with submit,
- print the receipt (there will be a receipt also per email).

See Appendix C for details on how to organize the electronic archive. Reports and codes handed in after the deadline will generally not be accepted. System failures, illness, etc. will not automatically give extra time.

---

1 Problem Description

The project consists in solving the Minimum Rainbow Subgraph Problem by means of heuristics. This problem arises in the context of bioinformatics as described in the following.

1.1 The biological problem

A *single nucleotide polymorphism* (SNP, pronounced "snip") is a site of the human genome (i.e., the position of a specific nucleotide) showing a statistically significant variability within a population. Besides very rare exceptions, at each SNP site only two nucleotides (out of A, T, C, and G) are observed, and they are called the SNP alleles.

Humans are diploid organisms; i.e., their DNA is organized in pairs of chromosomes. For each pair of chromosomes, one chromosome copy is inherited from the father and the other copy is inherited from the mother. For a given SNP, an individual can be either

Hapl. 1, paternal: taggtccCtatttCccaggcgcCgtatacttcgacgggTctata
Hapl. 1, maternal: taggtccGtatttAccaggcgcGgtatacttcgacgggTctata

Hapl. 2, paternal: taggtccCtatttAccaggcgcGgtatacttcgacgggTctata
Hapl. 2, maternal: taggtccGtatttCccaggcgcGgtatacttcgacgggCctata

Hapl. 3, paternal: taggtccCtatttAccaggcgcGgtatacttcgacgggTctata
Hapl. 3, maternal: taggtccGtatttAccaggcgcCgtatacttcgacgggCctata

Figure 1: The Haplotypes of Three Individuals, with Four SNPs.

homozygous (i.e., possessing the same allele on both chromosomes) or heterozygous (i.e., possessing two different alleles). The values of a set of SNPs on a particular chromosome copy define a *haplotype*.

In Figure 1, it is illustrated a simplistic example of three individuals and four SNPs. The alleles for SNP 1, in this example, are C and G. Individual 1, in this example, is heterozygous for SNPs 1, 2, and 3, and homozygous for SNP 4. His haplotypes are CCCT and GAGT.

SNPs are the predominant form of differences of human genomes. For this reason they are widely used in therapeutic, diagnostic and forensic applications.

Haplotyping, that is, determining the two haplotypes of a given chromosome for an individual is infeasible by experimental way, because it is not possible to examine the two copies of the chromosome separately. Instead, haplotypes are retrieved computationally by means of the less-informative genotypes that can be easily obtained from experiments.

A *genotype* provides information about the multiplicity of each SNP allele: i.e., for each SNP site, the genotype specifies if the individual is heterozygous or homozygous (in the latter case, it also specifies the allele). Heterozygous sites cause ambiguity in the retrieval of the haplotypes, because one has to decide how to distribute the two allele values on the two chromosome copies. *Resolving* a genotype requires to determine two haplotypes such that, if they are assumed to be the two chromosome copies, then the multiplicity of each SNP allele yields exactly that genotype. Note that for a genotype with k heterozygous sites, there are 2^{k-1} pairs of distinct haplotypes that could resolve the genotype. For example, for the genotype (0212) there are two possible pairs of haplotypes that resolve it: $\{(0110), (0011)\}$ and $\{(0010), (0111)\}$.

Given a set of genotypes, the *Haplotype Inference problem* asks to determine the set of haplotype pairs such that all genotypes are resolved. Since there is an exponential number of possible haplotypes for each genotype, a criterion to discriminate among the possible solutions is needed. One of the possible criteria is the *pure parsimony criterion* that seems consistent with observations in nature. This criterion chooses among the possible resolving sets of haplotypes the one of minimal size.

The Pure Parsimony Haplotyping Problem has been shown by Earl Hubbel to be NP-hard [Gus03].

More formally, given a set of n SNPs, let's arbitrarily fix a binary encoding of the two alleles for each SNP (i.e., call one of the two alleles "0" and the other "1"). Once the encoding has been fixed, each haplotype corresponds to a binary vector of length n. For a haplotype h, we denote by h_i the value of its ith component, with $i = 1, \ldots, n$. Under this encoding of the alleles, the two haplotypes of individual 1 in Figure 1 could be represented as binary vectors $(0, 0, 0, 1)$ and $(1, 1, 1, 1)$.

Given two haplotypes h' and h'', their *sum* is a vector $h' \oplus h''$, where the binary opera-

$$\mathcal{G} = \{(0221), (0011), (2102), (2220)\}$$

Haplotype 1, paternal: 0101 Haplotype 1, maternal: 0011	0221	Genotype 1
Haplotype 2, paternal: 0011 Haplotype 2, maternal: 0011	0011	Genotype 1
Haplotype 3, paternal: 0101 Haplotype 3, maternal: 1100	2102	Genotype 1
Haplotype 4, paternal: 1100 Haplotype 4, maternal: 0010	2220	Genotype 1

$$\mathcal{H} = \{(0101), (0011), (1100), (0010)\}$$

Figure 2: Input $\mathcal{G}$ and output $\mathcal{H}$ for the Pure Parsimony Haplotyping Problem.

tor $\oplus$ is defined, component-wise, as

$$(h' \oplus h'')_i := \begin{cases} 0 & \text{if } h'_i = h''_i = 1 \\ 1 & \text{if } h'_i = h''_i = 1 \\ 2 & \text{if } h'_i \neq h''_i \end{cases} \quad \text{for } i = 1, \ldots, n$$

A vector $g = h' \oplus h''$ is called a genotype. In general, we call a genotype any vector $g \in \{0, 1, 2\}^n$ and each position i such that $g_i = 2$ an ambiguous position.

For g, a genotype, a pair of haplotypes h', h'' such that $g = h' \oplus h''$ is a *resolution* of g. The haplotypes h' and h'' are said to resolve g. Let $\mathcal{G}$ be a set of genotypes and $\mathcal{H}$ be a set of haplotypes such that each g has a resolution in $\mathcal{H}$. Then $\mathcal{H}$ resolves $\mathcal{G}$ and is called a *resolving* set for $\mathcal{G}$.

The parsimony haplotyping problem is formally defined as follows:

Definition 1 (Pure Parsimony Haplotyping Problem)**.**
INPUT: A set $\mathcal{G}$ of p genotypes of length n each.
TASK: Find a resolving set $\hat{\mathcal{H}}$ for $\mathcal{G}$ of minimum cardinality.

We denote this problem by PPHP.

1.2 The graph problem

The Pure Parsimony Haplotyping problem can be transformed in the following graph problem [CST10].

Definition 2 (Minimum Rainbow Subgraph Problem)**.** *Given a graph G, whose edges are colored with p colors, find a subgraph $F \subseteq G$ of G of minimum order and with p edges such that each color occurs exactly once.*

We denote this problem by MRSP.

The transformation from PPHP to MRSP works as follows. For a set $\mathcal{G}$ of p genotypes $g_1, g_2, \ldots, g_p$ let us use p colors $1, 2, \ldots, p$. For each haplotype let us introduce a vertex. If two haplotypes h' and h'' explain a genotype g_l, $l \in \{1, \ldots, p\}$, that is, $g_l = h' \oplus h''$, then the corresponding vertices will be joined by an edge that receives color l. If a genotype is

```
#genotypes
11002
22002
22221
```

```
#haplotypes
01111       01000
10111       10000
00101       10011
01101       01001
11101       11011
10101       01011
00111       11001
11111       00011
10001       00000
00001       11000
```

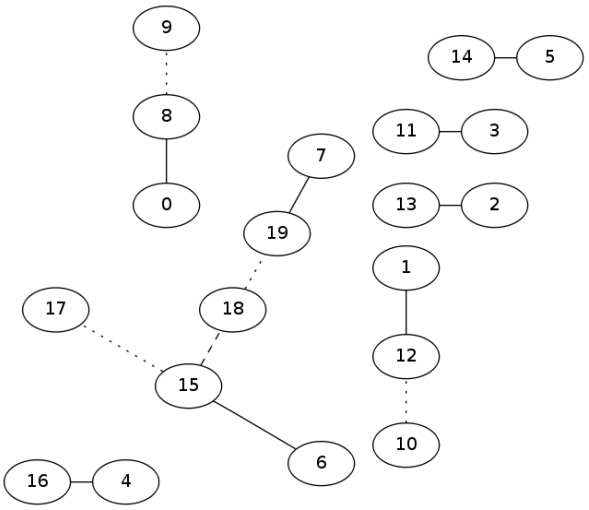

Figure 3: The small demo example. On the left the input data. On the right the corresponding rainbow graph.

explained by two identical haplotypes, then the corresponding vertex is joined by an edge which is called a *loop*. In this way we construct a graph G, whose edges are colored with p colors. Note that this is a proper edge coloring (no vertex is incident with two edges of the same color), since a haplotype h can be used at most once in a pair of haplotypes, which explains a genotype g. In this setting, every set $\mathcal{H}$ of haplotypes that explains $\mathcal{G}$ corresponds to a rainbow subgraph F of G.

An example is illustrated in Figure 3. On the left we have the set G of genotypes and a set $\mathcal{H}$ of 20 haplotypes resolving $\mathcal{G}$. The haplotypes are identified by numbers from 0 to 19 and form the set V of vertices in the rainbow graph on the left. The genotypes form the edges and the colors $1, 2, 3$ that we represent on the edges by using different line style. Any set of edges including exactly 3 colors determines a subset of $\mathcal{H}$ that is a resolving set. The set of haplotypes $\{6, 15, 17, 18\}$ is the resolving set of minimal cardinality.

1.3 Instances

The instances of the PPHP are simulated data generated with the program `ms` by Hudson (2002) [Hud02] following the indications of Lancia and Rizzi [LS09].

Summary statistics relative to the 10 instances generated are reported in Table 1.

2 Project Requirements

The aim of the project is to study efficient heuristic algorithms for solving the 10 test instances of the PPHP defined above after having transformed them in MRSP instances.

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

1. Implement the procedure RandomRainbowSubset of Figure 4 to generate random solutions.
2. Design and implement one or more construction heuristics that perform better than RandomRainbowSubset.
3. Design and implement one or more local search algorithms.

Instance	$\lvert\mathcal{G}\rvert$	$\lvert\mathcal{H}\rvert$	$\min\{\lvert S2\rvert\}$	$av\{\lvert S2\rvert\}$	$\max\{\lvert S2\rvert\}$
01	67	62817	1	7	15
02	68	338294	2	9	16
03	55	172349	1	9	16
04	59	5321333	1	11	21
05	66	2937915	1	10	20
06	62	1665986	2	10	19
07	61	869377	1	9	18
08	64	98735	1	7	15
09	62	271437	1	8	17
10	63	840233	2	11	18

Table 1: The first column reports the number of genotypes, the second the number of compatible haplotypes (a haplotype h is compatible if it has at least one genotype g in $\mathcal{G}$ such that $h_i = g_i$ whenever $g_i \neq h_i$); the third column indicates the number of ambiguous solutions.

```
1 function RandomRainbowSubset(π)
2 input an instance π of Minimum Rainbow Subgraph Problem;
3 output a rainbow subgraph;
4 for c in 1, ..., p do
5     choose uniformly at random an edge from the set of edges available with color c.
6 return (the set of vertices incident to the selected edges);
```

Figure 4: Procedure to generate a random rainbow sets.

4. Design and implement an effective algorithm enhancing the heuristics at the previous two points with the use of stochastic local search methods and metaheuristics.

5. For all the methods above carry out an experimental analysis and draw sound conclusions.

6. For the best algorithm devised in point 4, determine experimentally whether the initial solution should be a good solution provided by a construction heuristic devised in point 2 or whether the use of RandomRainbowSubset would lead to equally good (or possibly better) final results.

In the experimental assessment all algorithms should be given a maximum time per run of **30 seconds** on each single instance.[1]

3 Remarks

Remark 1 For each point above a description of the work undertaken must be provided in the report. In particular for the best algorithms arising from the experimental analysis the amount of details provided must guarantee the reproducibility of the algorithm from the report only (i.e., without having to look at the source code).

Remark 2 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 4, a table must be

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

provided with the best results for each specific instance of Table 1.

Remark 3 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 4 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;
- presence of computational considerations (ie, asymptotic complexity analysis) for the main procedures designed;
- organization of experiments which guarantee reproducibility of conclusions;
- clarity of the report;
- effective use of graphics in the presentation of experimental results.

Appendix A Instance Format

Input data for each instance of PPHP are given in two text file. A file called `haplotypes-`s consists of a binary string per line indicating the haplotypes. Another file called `genotypes-`s reports the strings of elements $\{0, 1, 2\}$ representing the genotypes.

Appendix B Solution Format

In order to check the validity of the results reported, the program submitted must output when finishing the best solution found during its execution in a file called `parsimony`s`.sln`. The file must be in text format and contain the selected haplotypes in the same format as in the file `haplotypes-`s`.txt`. Thus, the number of lines of the solution file gives the cardinality of the resolving set.

Appendix C Handing in Electronically

The electronic archive to hand in must be organized as follows. It expands in a main directory named with the first 6 digits of your CPR number (e.g., 030907). The directory has the following content:

```
CPRN/README
CPRN/report/
CPRN/src/
```

The directory `report` contains a pdf or postscript version of your report. *Do not put your name in the author field of the report, instead put your CPR number*. 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 `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 `mrsp`. It must execute from command line by typing in the directory `CPRN/src/`:

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

where the flags indicate:

- `-i INSTANCE` the input instance;
- `-t TIME` the time limit in seconds;
- `-s SEED` the random seed;
- `-o OUTPUT` the file name where the solution is written.

For example:

[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`.

```
mrsp -i Inst/01.txt -o 01.sln -t 180 -s 1 > mrsp-01.log
```

will run the program on the input files of the instance `01.txt` opportunely retrieved from the given path for 180 seconds with random seed 1 and write the solution in the file `01.sln`.

In its default mode, the program must run the best algorithm developed and must print on the standard output **only one single number** at the end of the run corresponding to the quality of the best solution found during the run.

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 853 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`. Process times include the time to read the instance.

References

[CST10] Stephan Matos Camacho, Ingo Schiermeyer, and Zslot Tuza. Approximation algorithms for the minimum rainbow subgraph problem. *Discrete Mathematics*, 2010. to appear.

[Gus03] D. Gusfield. Haplotype inference by pure parsimony. In R.A. Baeza-Yates, E. Chvez, and M. Crochemore, editors, *Combinatorial Pattern Matching, CPM 2003, Proc. 14th Annual Symposium*, volume 2676 of *Lecture Notes in Computer Science*, pages 144–155. Springer, Berlin, 2003.

[Hud02] R. R. Hudson. Generating samples under a wright-fisher neutral model of genetic variation. *Bioinformatics*, 18:337–338, 2002.

[LS09] Giuseppe Lancia and Paolo Serafini. A set-covering approach with column generation for parsimony haplotyping. *INFORMS Journal on Computing*, 21(1):151–166, 2009.