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

December 28, 2007
Marco Chiarandini

DM63 - Heuristics for Combinatorial Optimization Problems

Exam Project, Fall 2007

Note 1 The project is carried out individually and it is not permitted to collaborate.

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 serve to verify the correctness of the results presented. The report may be written either in English or Danish.

Note 2 Additional material to the project description is available on the web at: `http://www.imada.sdu.dk/Courses/DM63/project.php`. Please take vision of this link before starting the project.

Note 3 Corrections or updates to the project description will be announced on the same web site above and communicated to the students who registered for the exam. The registration is done by sending an email to the lecturer `marco@imada.sdu.dk`. In any case, it remain students' responsibility to check for updates on the web page.

Note 4 *Submission.* Two printed copies of the written report must be handed in at the secretary office **before 12:00 of Monday, 17 December 2007**. Ask the secretary for a receipt showing that you have handed in the report in time. Contextually, an electronic version of the report and the program source code must be sent by email to the lecturer. A reply will be sent as receipt.

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

Hypergraphs generalize the concept of graphs by letting edges be any subset of vertices [Ber73]. Let V denote a finite non-empty set and let $\mathcal{E}$ denote a collection of non-empty subsets of V, that is, $E \subseteq V$ for each $E \in \mathcal{E}$. The pair $H = (V, \mathcal{E})$ is a *hypergraph* with the set V of *vertices* and the set $\mathcal{E}$ of *edges* or *hyperedges*. A hypergraph $H = (V, \mathcal{E})$ is called *r-uniform* if each edge $E \in \mathcal{E}$ has exactly r elements. A 2-uniform hypergraph is a *graph*.

A subset $I \subseteq V$ of a hypergraph $H = (V, \mathcal{E})$ is called *independent* if I contains no edges from H, i.e., $E \not\subset I$ for each edge $E \in \mathcal{E}$. A *k-coloring* of a hypergraph $H = (V, \mathcal{E})$ is a mapping $\varphi : V \to \{1, \ldots, k\}$ and it is called *proper* if every edge $E \in \mathcal{E}$ with $|E| > 1$ contains two vertices with two distinct colors (or equivalently, if no edge has *all* vertices of the same color). The *chromatic number* of H, denoted by $\chi(H)$, is the minimal number

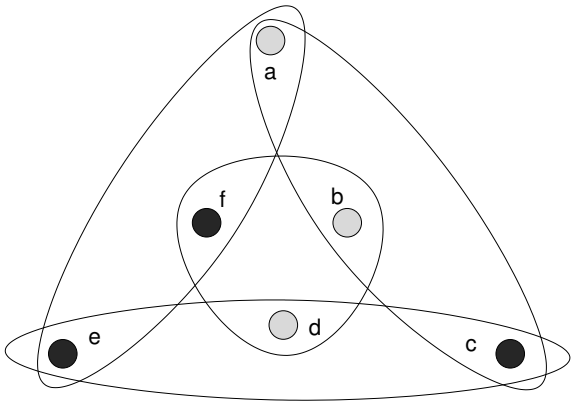

Figure 1: Example of 3-uniform hypergraph with vertex set $V = \{a, b, c, d, e, f\}$ and edges $E_1 = \{abc\}, E_2 = \{cde\}, E_3 = \{bdf\}, E_4 = \{aef\}$. Set $\{acdf\}$ is an independent set and a proper 2-coloring is shown.

of colors for which H admits a proper coloring. An example illustrating the concepts introduced is given in Figure 1.

The subject of the project is on the 3-Uniform Hypergraph Coloring Problem that can be defined as follows:

Definition: 3-Uniform Hypergraph Coloring Problem
Input: *A 3-uniform hypergraph $H = (V, \mathcal{E})$ and a set of colors Γ with $|\Gamma| = k$.* **Task:** *Find a proper k-coloring with minimal k.*

Coloring 3-uniform hypergraphs is NP-hard [Lov73, Bro96, PR84]. Assuming $NP \neq ZPP$ there are no polynomial time algorithms for the chromatic number with approximation ratio of $n^{1-\epsilon}$ for any $\epsilon > 0$ [KS03, HL98]. The best results for polynomial time approximation algorithms for r-uniform hypergraphs are the performance ratios of $O(n/(\log^{(r-1)} n)^2))$ [HL98] and $O(n(\log\log n)^2)/(\log n)^2)$ [KS03].

The 3-Uniform Hypergraph Coloring Problem might be useful to model real life applications. The following is an example arising in the shunting of trains in railway logistics.

During the low traffic hours or overnight passenger trains and trams need to be parked in stations and depots. Similarly, incoming freight trains need to be split up and rearranged according to their destinations in railroad shunting yards. In both these two cases, there is an ordering of arriving *units*, and one has to decide for each unit on which track it will be stored [SK04]. The choice is limited by the fixed number of available tracks and by the accessibility of the tracks. The objective is to use the minimum number of tracks to host all trains without additional shunting movements during input or output operations.

Let focus on the case in which each track can receive trains only from a single side while the output can be done from both sides (single input, double output, SIDO). The opposite case (double input, single output, DISO) is more relevant in practice, because it reflects the need to park trains in the depot in the night in the best possible conditions so that in the morning they can exit from the single output in the direction of the station. However, it can be shown that the two cases are equivalent from a solution point of view [SK04]. Let also assume that the first departure takes place after the last arrival. Under these

conditions, the *SIDO shunting problem* can be modeled as a 3-Uniform Hypergraph Coloring Problem.

Let train units be numbered according to the departure time from the depot area. In particular, the first train leaving the depot is numbered 1, the second 2, and so on. Then the arriving sequence of n train units at the depot area is a permutation $\pi = [\pi_1, \pi_2, \ldots, \pi_n]$ of the outgoing sequence $[1, 2, \ldots, n]$. For example the sequence $[2, 3, 1]$ means that three trains are involved, the first incoming train will be the second to go out, the second incoming train will be the third to leave the depot and the last one will be the first to go out.

Let also introduce a few definitions on sequences of integer numbers. Given a sequence $S = [s_1, s_2, \ldots, s_n]$ of n distinct elements, a sub-sequence of S is a sequence $S' = [s_{i_1}, s_{i_2}, \ldots, s_{i_m}]$ such that $1 \leq i_j < i_k \leq n$ for each $j < k$. If S and T are two sub-sequences such that $|S| = n$ and $|T| = m$, then the sequence given by the concatenation of S and T, denoted by $S \cdot T$, is the sequence having S as the sub-sequence of the first n elements and T as the sub-sequence of the last m elements, and $|T + S| = m + n$. The concatenation of two sequences, one increasing and one decreasing, is called *unimodal sequence*.

In order to minimize shunting movements, in the SIDO situation, the sequence $S = [s_1, s_2, \ldots, s_m]$ of the trains stored in a track must form a *unimodal sequence*. Indeed, let R be the side of the track used for incoming trains and let L be the opposite side. The sequence S is such that $S = S_L \cdot S_R$, where $S_L = [s_1, s_2, \ldots s_t]$ is the sub-sequence of trains going out from the L side and $S_R = [s_{t+1}, s_{t+2}, \ldots s_m]$ is the sequence of trains going out from the R side. The trains in S_L must form an increasing sequence $s_1 < s_2 < \ldots < s_t$ whereas the trains in S_R must form a decreasing sequence $s_{t+1} > s_{t+2} > s_m$, being s_m the first train leaving the track from this side.

The *SIDO shunting problem* can then be reformulated as follows. Given a sequence S of n incoming trains, find a partition of S with the minimum number of unimodal sub-sequences.

It is interesting to note that already sequences of only three trains may require two tracks. Let $[t_1, t_2, t_3]$ be a sequence of three incoming trains, and let us suppose they are stored in a single track. If $t_1 > t_2$ and $t_2 < t_3$ the train t_2 must leave the depot before the other two which is impossible without shunting operations. This last example shows that sub-sequences of S with a "valley" shape must not be put into a single track. This kind of "no valley in a single track" constraint can be modeled in terms of a special kind of hypergraph.

Given a sequence $S = [s_1, s_2, \ldots, s_n]$ of n distinct integers, the *valley hypergraph induced by* S is the hypergraph $H_S = (V, \mathcal{E})$ where $V = \{s \,|\, s \in S\}$ and $\{s_i, s_j, s_k\} \in \mathcal{E}$ is and only if $s_i > s_j$, $s_j < s_k$, and $i < j < k$. Then, it is proved in [SK04] that given a sequence S of n incoming trains, the SIDO problem is solvable using k tracks if and only if the valley hypergraph $H_S = (V, \mathcal{E})$ is colorable using k colors. See Figure 2 for an illustration of this correspondence.

The problem of coloring "valley" hypergraphs with the minimal number of colors remains NP-hard [SKLZ06].

2 Project Content

The aim of the project is to study heuristic algorithms to solve the 3-Uniform Hypergraph Coloring Problem in two sets of instances.

A first set of instances, called random, is made of general 3-uniform hypergraphs. These

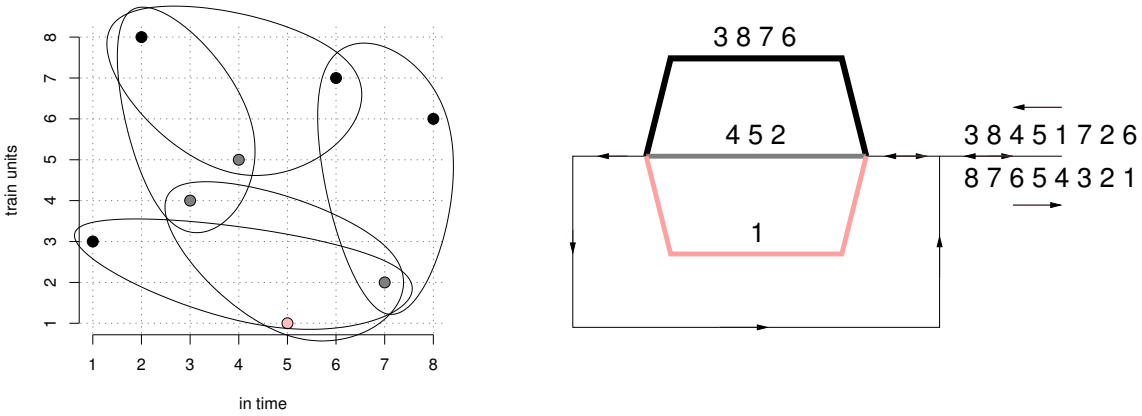

Figure 2:
The hypergraph generated by some of the "valley" constraints associated with the shunting problem represented on the right. A proper 3-coloring of the hypergraph corresponds to an allocation of trains in the track infrastructure without additional shunting movements.

hypergraphs are generated selecting randomly a fraction p of the $\binom{|V|}{3}$ possible distinct edges. A second set, called shunting, is made of valley hypergraphs described above and used to model the SIDO shunting problem.

For each of these two sets of instances, 6 benchmark instances and a random instance generator are made available at `http://www.imada.sdu.dk/Courses/DM63/project.php`. See also the Appendix of this document for details on the format of the instances.

All the four tasks below must be addressed in order to pass the exam.

Task 1 – Construction Heuristics

- Implement the two following construction heuristics:

  1. A *greedy heuristic* similar to the one for graph coloring studied during the lectures. The greedy algorithm takes in input a randomly ordered list of vertices and assigns the smallest feasible color to the vertices visited in the order. The heuristic works on both sets of instances random and shunting. Hence, it can be used as reference heuristic.
  2. The *longest unimodal sub-sequence heuristic* described in [SK04, pp. 26-27]. This heuristic works only on the shunting instances and colors the hypergraph by iteratively removing the longest unimodal sub-sequence.

- Undertake a computational study to determine the most convenient order of vertices to pass to the greedy heuristic.

- Design and implement *at least one more* construction heuristic which can solve both instances of type random and shunting. Inspiration can be taken from the graph coloring problem treated in detail during the lectures.

- For each construction heuristic in the previous points of these task provide a complexity analysis.

- Undertake a computational study to compare the heuristics implemented and report in a table the numerical results (time and quality) for the 6 benchmark instances per class.

Task 2 – Construction Heuristics

In the report address the following issues:

- How could the longest unimodal sub-sequence heuristic be modified in order to work also with general 3-uniform graphs? In other terms, on which principle this heuristic for coloring hypergraphs is based?
- Is there for any hypergraph a permutation of vertices for which the greedy algorithm will produce an optimal coloring? Motivate the answer.
- Is it still valid the following assertion:
  "If we take a proper coloring and any permutation of vertices in which the vertices of each color class are maintained adjacent, then applying the greedy coloring will produce a proper coloring at least as good"?
- Provide an example where the longest unimodal heuristic does not find the optimal coloring.

Task 3 – Local Search

- Design and implement *at least two* local search *schemes* with natural termination criterion (e.g., in a local optimum). Describe the solution representation, the initial solutions, the evaluation function, the neighborhood structures and the search strategy.
- Provide details on the data structures used in the implementation and on the computational complexity of the evaluation function computation, in the initialization and in the update consequent to a move.
- Compare experimentally the local search algorithms, also in combination with different construction heuristics and within a variable neighborhood descent procedure.

Task 4 – Metaheuristics

- Devise, implement and describe *at least two* metaheuristic (or hybridization thereof) algorithms (check the course content on the web-page for the list of methods described in the course). One of the two metaheuristics must be a population based method.
- Address the problem of configuring and tuning the components and the parameters inherent the metaheuristics chosen.
- Undertake an experimental comparison to decide the best algorithm. In order to address this question you may take into consideration the following issues:
  - different time limits;
  - scaling;
  - change in performance with respect to instance features.

- Report in a table the numerical results of your best algorithms on the 6 benchmark instances per class. Use 360 seconds as time limit on these instances.

3 Remarks

Remark 1 Beside the precise request for tables there is absolute freedom to add other graphics or table that might facilitate the presentation in the report.

Remark 2 The good performance of the algorithms presented, the study of a number of algorithms above the minimal number requested and the use of very large scale neighborhood techniques will contribute to achieve a higher mark.

Remark 3 The total length of the report should not be less than 10 pages and not be more than 16 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 algorithms, it is allowed to use short algorithmic sketches but not to include program codes.

Remark 4 This is a list of further factors that will be taken into account in the evaluation:

- 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;
- quality of the final results (however, showing that a promising approach does not work well in practice will also be considered equally well if there is the attempt to explain why);
- effective use of graphics in the presentation of experimental results;
- clarity of the report.

Appendix A Instance Format

The instance format is an extension of the DIMACS format used for the instances of graph coloring. Each line of the file begins with a letter that defines the rest of the line. The legal lines are:

- `c` Comment: remainder of line ignored.
- `p edge n m` where `n` is the number of nodes (to be numbered $1, \ldots n$) and `m` the number of edges.
- `s` Incoming sequence: this is present only in the `shunting` instances and serves to apply the longest unimodal heuristic.
- `e` Edge: is of the form `e v1 v2 v3` where `v1`, `v2`, `v3` are the vertices that compose the edge.

Appendix B Instance Generator

Two scripts to generate instances random and instances shunting are made available. The generators are two R functions and should be launched within R as follows:

```
> source("random.R")
> generate(30,0.5,1,1)
> source("shunting.R")
> generate(30,1,1)
```

Read the comments on the two scripts for an explanation of the arguments of the functions.

Appendix C Solution Format

In order to check the validity of the results claimed the program submitted must output the solution in a file when finishing. The syntax remains the same as for the graph coloring: a column of numbers corresponding to the colors assigned to vertices. After each entry the character '`\n`' (new line) has to be printed. Colors start from 1 and the first color in the column represents the color assigned to vertex 1.

References

[Ber73] C. Berge. *Graphs and hypergraphs*, volume 6 of *North-Holland mathematical library*. North-Holland Publishging Company, Amsterdam, Holland., 1973.

[Bro96] J.I. Brown. The complexity of generalized graph colorings. *Discrete Applied Mathematics*, 69(9):257–270, 1996.

[HL98] T. Hofmeister and H. Lefmann. Approximating maximum independent sets in uniform hypergraphs. In Lubos Brim, Jozef Gruska, and Jirí Zlatuska, editors, *MFCS*, volume 1450 of *Lecture Notes in Computer Science*, pages 562–570. Springer, 1998.

[KS03] M. Krivelevich and B. Sudakov. Approximate coloring of uniform hypergraphs. *Journal of Algorithms*, 49(1):2–12, 2003.

[Lov73] L. Lovász. Coverings and colorings of hypergraphs. In *Proceedings of the 45th Southeastern Conf. Combinatorics, Graph Theory and Computing*, pages 3–12, 1973.

[PR84] K.T. Phelps and V. Rödl. On the algorithmic complexity of coloring simple hypergraphs and steiner triple systems. *Combinatorics*, 1984.

[SK04] G. Di Stefano and M.L. Koci. A graph theoretical approach to the shunting problem. *Electr. Notes Theor. Comput. Sci.*, 92:16–33, 2004.

[SKLZ06] G. Di Stefano, S. Krause, M.E. Lübbecke, and U.T. Zimmermann. On minimum -modal partitions of permutations. In José R. Correa, Alejandro Hevia, and Marcos A. Kiwi, editors, *LATIN*, volume 3887 of *Lecture Notes in Computer Science*, pages 374–385. Springer, 2006.