DM841

Discrete Optimization

Lecture 7

Stochastic Local Search and Metaheuristics

Marco Chiarandini

Department of Mathematics & Computer Science
University of Southern Denmark

Review

1. Combinatorial Optimization and Terminology, TSP
2. Solution Methods
3. SAT Example: enumeration, MIP, local search, backtracking
4. Local Search: Modelling and components
5. N-Queens example
6. C++: object passing, Encapsulation, Constructors, Inheritance, Templates, STL, virtual functions, headers, namespaces, copy constructors, destructors
7. EasyLocal framework. Examples
8. Local Search: Iterative Improvement, SAT

The Vertex Coloring Problem

Given: A graph G and a set of colors Γ.

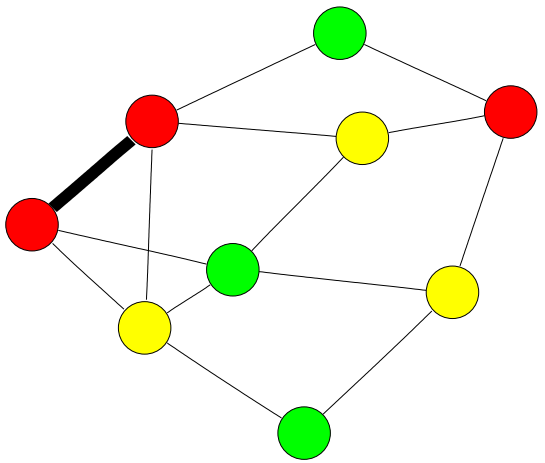
A proper coloring is an assignment of one color to each vertex of the graph such that adjacent vertices receive different colors.

Decision version (k-coloring)

Task: Find a proper coloring of G that uses at most k colors.

Optimization version (chromatic number)

Task: Find a proper coloring of G that uses the minimal number of colors.



Exercise

Map coloring:

Timetabling as a graph coloring problem

Definition

Find an assignment of lectures to time slots and rooms which is

Feasible

- rooms are only used by one lecture at a time,
- each lecture is assigned to a suitable room,
- no student has to attend more than one lecture at once,
- lectures are assigned only time slots where they are available;

} Hard Constraints

and Good

- no more than two lectures in a row for a student,
- unpopular time slots avoided (last in a day),
- students do not have one single lecture in a day.

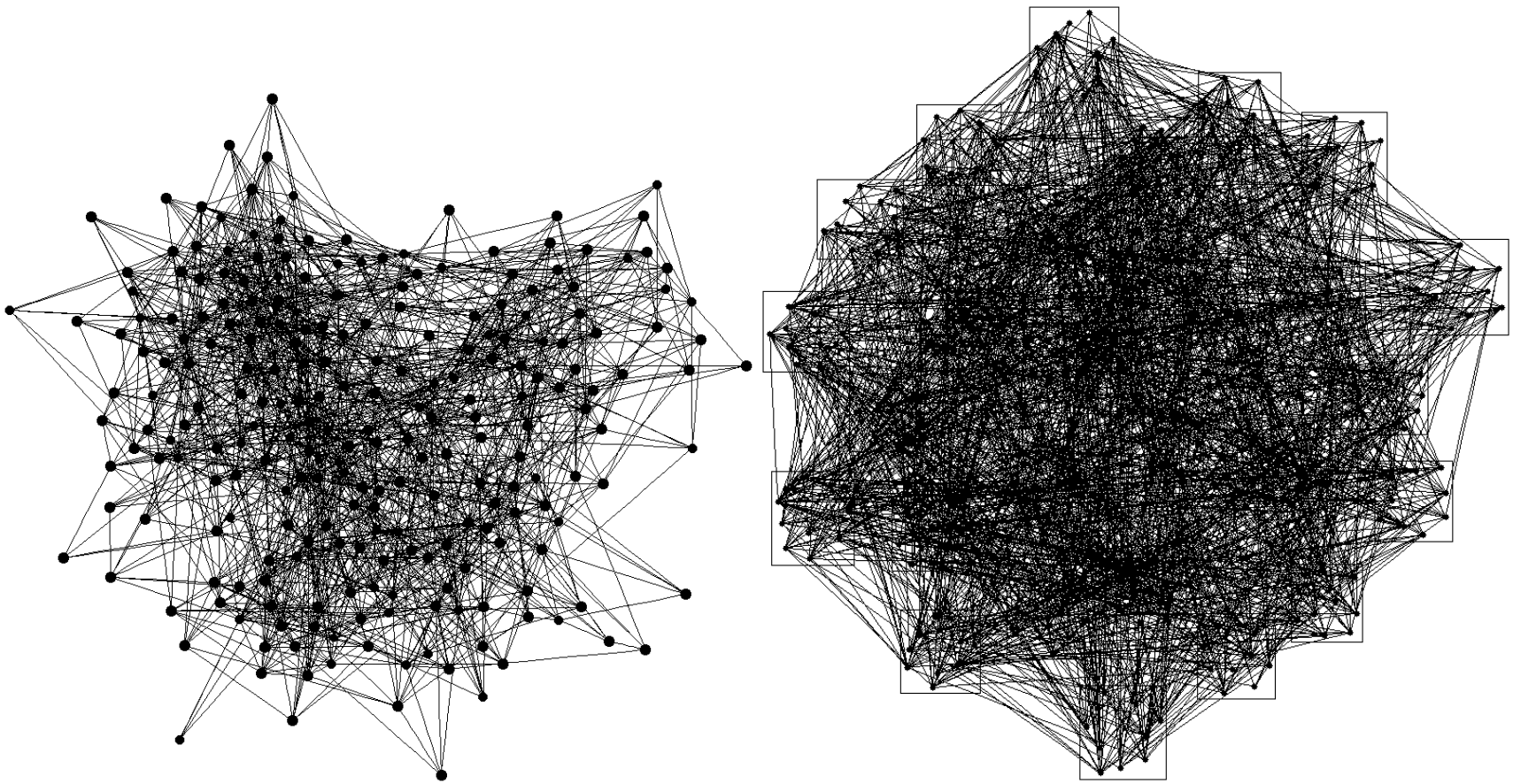
} Soft Constraints

A look at the instances

ID	year	lecs	studs	rooms	lecs/stud	studs/lec	rooms/le	degree	slots/lec	slots/lec	slots/lec	Prec.	Rel. Prec.
1	2007	400	500	10	21.02	26.27	4.08	0.34	16	25.34	34	40	14
2	2007	400	500	10	21.03	26.29	3.95	0.37	17	25.69	33	36	14
3	2007	200	1000	20	13.38	66.92	5.04	0.47	19	25.54	33	20	11
4	2007	200	1000	20	13.40	66.98	6.40	0.52	15	25.66	33	20	9
5	2007	400	300	20	20.92	15.69	6.80	0.31	16	25.43	34	120	43
6	2007	400	300	20	20.73	15.54	5.07	0.30	13	25.39	36	119	32
7	2007	200	500	20	13.47	33.66	1.57	0.53	9	17.86	26	20	10
8	2007	200	500	20	13.83	34.58	1.92	0.52	11	17.17	26	21	13
9	2007	400	500	10	21.43	26.79	2.91	0.34	17	25.42	34	41	18
10	2007	400	500	10	20.98	26.23	3.20	0.38	14	25.47	34	40	13
11	2007	200	1000	10	13.61	68.04	3.38	0.50	17	25.32	35	21	17
12	2007	200	1000	10	13.61	68.03	3.35	0.58	15	25.67	35	20	13
13	2007	400	300	20	21.19	15.89	8.68	0.32	17	25.75	34	116	34
14	2007	400	300	20	20.86	15.64	7.56	0.32	17	25.44	36	118	46
15	2007	200	500	10	13.05	32.63	2.23	0.54	11	17.38	24	21	13
16	2007	200	500	10	13.64	34.09	1.74	0.46	10	17.57	25	19	10

These are large scale instances.

A look at the basic Graph Model (vertices correspond to lectures)



Constraint Satisfaction Problem

The **domain** of a variable x, denoted $D(x)$, is a finite set of elements that can be assigned to x.

A **constraint** C on X is a subset of the Cartesian product of the domains of the variables in X, i.e., $C \subseteq D(x_1) \times \cdots \times D(x_k)$ (extensional form). A tuple $(d_1, \ldots, d_k) \in C$ is called a solution to C.
Equivalently, we say that a solution $(d_1, ..., d_k) \in C$ is an assignment of the value d_i to the variable $x_i, \forall 1 \leq i \leq k$, and that this assignment satisfies C (intentional form). If $C = \emptyset$, we say that it is inconsistent.

Constraint Satisfaction Problem

Constraint Satisfaction Problem (CSP)

A CSP is a finite set of variables X, together with a finite set of constraints C, each on a subset of X. A **solution** to a CSP is an assignment of a value $d \in D(x)$ to each $x \in X$, such that all constraints are satisfied simultaneously.

Constraint Optimization Problem (COP)

A COP is a CSP P defined on the variables $x_1, \ldots, x_n$, together with an objective function $f : D(x_1) \times \cdots \times D(x_n) \rightarrow Q$ that assigns a value to each assignment of values to the variables. An **optimal solution** to a minimization (maximization) COP is a solution d to P that minimizes (maximizes) the value of $f(d)$.

CP-model

CP formulation:

$$
\begin{aligned}
\textit{variables}: \quad & \texttt{domain}(\texttt{y}_\texttt{i}) = \{1, \ldots, K\} && \forall i \in V \\
\textit{constraints}: \quad & y_i \neq y_j && \forall ij \in E(G) \\
& \texttt{alldifferent}(\{\texttt{y}_\texttt{i} \mid \texttt{i} \in \texttt{C}\}) && \forall C \in \mathcal{C}
\end{aligned}
$$

Outline

Escaping Local Optima

Possibilities:

- Restart: re-initialize search whenever a local optimum is encountered.
  (Often rather ineffective due to cost of initialization.)
- Non-improving steps: in local optima, allow selection of candidate solutions with equal or worse evaluation function value, *e.g.*, using minimally worsening steps.
  (Can lead to long walks in *plateaus*, *i.e.*, regions of search positions with identical evaluation function.)
- Diversify the neighborhood: multiple, variable-size, rich (while still preserving incremental algorithmics insights)

Note: None of these mechanisms is guaranteed to always escape effectively from local optima.

Diversification *vs* Intensification

- Intensification: aims at greedily increasing solution quality, *e.g.*, by exploiting the evaluation function.
- Diversification: aims at preventing search stagnation, that is, the search process getting trapped in confined regions.
- Goal-directed and randomized components of LS strategy need to be balanced carefully.

Examples:

- Iterative Improvement (II): *intensification* strategy.
- Uninformed Random Walk/Picking (URW/P): *diversification* strategy.

Balanced combination of intensification and diversification mechanisms forms the basis for advanced LS methods.

Outline

Randomized Iterative Impr.

Key idea: In each search step, with a fixed probability perform an uninformed random walk step instead of an iterative improvement step.

Randomized Iterative Improvement (RII):
determine initial candidate solution s
while termination condition is not satisfied **do**
  With probability `wp`:
    choose a neighbor s' of s uniformly at random
  Otherwise:
    choose a neighbor s' of s such that $f(s') < f(s)$ or,
      if no such s' exists, choose s' such that $f(s')$ is minimal
  $s := s'$

Example: Randomized Iterative Improvement for SAT

procedure *RIISAT*(F, `wp`, `maxSteps`)
  input: *a formula* F, *probability* `wp`, *integer* `maxSteps`
  output: *a model* φ *for* F **or** $\emptyset$

  choose assignment φ for F uniformly at random;
  steps := 0;
  while not(φ is not proper) **and** (*steps* $<$ *maxSteps*) **do**
    with probability `wp` **do**
      select x in X uniformly at random and flip;
    otherwise
      select x in X^c uniformly at random from those that
        maximally decrease number of clauses violated;
    change φ;
    steps := *steps*+1;
  end
  if φ is a model for F **then return** φ
  else return $\emptyset$
  end
end *RIISAT*

Example: Randomized Iterative Improvement for GCP

procedure *RIIGCP*(F, `wp`, `maxSteps`)
  input: *a graph* G *and* k, *probability* `wp`, *integer* `maxSteps`
  output: *a proper coloring* φ *for* G **or** $\emptyset$

  choose coloring φ of G uniformly at random;
  steps := 0;
  while not(φ is not proper) **and** (*steps* $<$ *maxSteps*) **do**
    with probability `wp` **do**
      select v in V and c in Γ uniformly at random;
    otherwise
      select v in V^c and c in Γ uniformly at random from those that
        maximally decrease number of edge violations;
    change color of v in φ;
    steps := *steps*+1;
  end
  if φ is proper for G **then return** φ
  else return $\emptyset$
  end
end *RIIGCP*

Note:

- No need to terminate search when local minimum is encountered

  Instead: Impose limit on number of search steps or CPU time, from beginning of search or after last improvement.

- Probabilistic mechanism permits arbitrary long sequences of random walk steps

  Therefore: When run sufficiently long, RII is guaranteed to find (optimal) solution to any problem instance with arbitrarily high probability.

- GWSAT [Selman et al., 1994], was at some point state-of-the-art for SAT.