DM811

Heuristics for Combinatorial Optimization

Lecture 13

Exercises: GCP

Marco Chiarandini

Department of Mathematics & Computer Science
University of Southern Denmark

Course Overview

- ✔ Combinatorial Optimization, Methods and Models
- ✔ CH and LS: overview
- ✔ Working Environment and Solver Systems
- ˜ Methods for the Analysis of Experimental Results
- ✔ Construction Heuristics
- ✔ Local Search: Components, Basic Algorithms
- Local Search: Neighborhoods and Search Landscape
- Efficient Local Search: Incremental Updates and Neighborhood Pruning
- ˜ Stochastic Local Search & Metaheuristics
- Configuration Tools: F-race
- Very Large Scale Neighborhoods

Examples: GCP, CSP, TSP, SAT, MaxIndSet, SMTWP, Steiner Tree, Unrelated Parallel Machines, p-median, set covering, QAP, ...

Outline

1. Recap.

2. GCP

Outline

1. Recap.

2. GCP

Summary: Local Search Algorithms
(as in [Hoos, Stützle, 2005])

For given problem instance π:

1. search space S_π
2. neighborhood relation $\mathcal{N}_\pi \subseteq S_\pi \times S_\pi$
3. evaluation function $f_\pi : S \to \mathbf{R}$
4. set of memory states M_π
5. initialization function $\texttt{init} : \emptyset \to S_\pi \times M_\pi$
6. step function $\texttt{step} : S_\pi \times M_\pi \to S_\pi \times M_\pi$
7. termination predicate $\texttt{terminate} : S_\pi \times M_\pi \to \{\top, \bot\}$

Efficiency and Effectiveness

After implementation and test of the above components, improvements in efficiency (ie, computation time) can be achieved by:

A. fast incremental evaluation (ie, delta evaluation)

B. neighborhood pruning

C. clever use of data structures

Improvements in effectiveness, ie, quality, can be achieved by:

D. application of a metaheuristic

E. definition of a larger neighborhood

Outline

1. Recap.

2. GCP

Preprocessing rules

Polynomial time reduction of the graph G to G' such that given a feasible k-coloring for G' it is striaghtforward to derive a feasible k-coloring for G.

Searching for a k-coloring (k fixed)

- Remove under-constrained nodes: $v \in V, d(v) < k$
- Remove subsumed nodes: $v \in V,$ if $\exists u \in V, uv \notin E, A(v) \subseteq A(u)$
- Merge nodes that must have the same color: eg, if any nodes are fully connected to a clique of size $k-1$, then these nodes can be merged into a single node with all the constraints of its constituents, because they must have the same color.

Local Search for Graph coloring

[Chiarandini et al., 2007]

Different choices for the candidate solutions:

decision vs optimization	assignment of colors to V	level of feasibility	
k-fixed	complete	proper	
k-fixed	partial	proper	$+++$
k-fixed	complete	improper	$+++$
k-fixed	partial	improper	$-$
k-variable	complete	proper	$++$
k-variable	partial	proper	$-$
k-variable	complete	improper	$++$
k-variable	partial	improper	$-$

imply different neighborhood structures and evaluation functions.

Local Search for GCP

Scheme: k-fixed / complete / improper

Local Search

- Solution representation: `var{int} a[|V|](1..K)`
- Evaluation function: conflicting edges or conflicting vertices
- Neighborhood: one-exchange

Naive approach: $O(n^2k)$
Neighborhood examination

for all $v \in V$ **do**
    for all $k \in 1..k$ **do**
        compute $\Delta(v, k)$

Better approach:

- V^c set of vertices involved in a conflict
- $\boldsymbol{\Delta}(v,k)$ stores number of vertices adjacent to v in each color class k

Procedure Initialise_$\boldsymbol{\Delta}$(G,a)
$\boldsymbol{\Delta} = 0$
for each v in V **do**
    for each u in $A_V(v)$ **do**
        $\boldsymbol{\Delta}(u, \mathsf{a}(v)) = \boldsymbol{\Delta}(u, \mathsf{a}(v)) + 1$

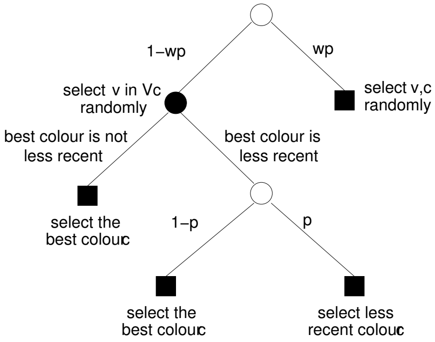
Procedure Examine(G,N(a))
for each v in V^c **do**
    for each $k \in \Gamma$ **do**
        compute $\Delta(v,k) = \boldsymbol{\Delta}(v,k) - \boldsymbol{\Delta}(v, \mathsf{a}(v))$

Procedure Update_$\boldsymbol{\Delta}$(G,a,v,k)
for each u in $A_V(v)$ **do**
    $\boldsymbol{\Delta}(u, \mathsf{a}(v)) = \boldsymbol{\Delta}(u, \mathsf{a}(v)) - 1$
    $\boldsymbol{\Delta}(u, k) = \boldsymbol{\Delta}(u, k) + 1$

Comet examples
Tabu Search

`./coloring.co`

Randomized Iterative Improvement



Guided Local Search

- evaluation function: $f'(s) = f(s) + \lambda \cdot \sum_{i=1}^{|E|} w_i \cdot I_i(\mathcal{C})$
  w_i is the penalty cost associated to edge i;
  $I_i(s)$ is an indicator function that takes the value 1 if edge i causes a colour conflict in s and 0 otherwise;
  parameter λ

- penalty weights are initialised to 0

- updated each time Iterative Improvement reaches a local optimum in f'; increment the penalties of all edges with maximal utility.

$$u_i = I_i(s) \cdot \frac{1}{1 + w_i}.$$

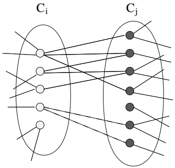
- once a local optimum is reached, the search continues for sw non-worsening exchanges (side walk moves) before the evaluation function f' is updated. Update of w_i and f' is done in the worst case in $O(k|V|^2)$.

Local Search for GCP

Scheme: k-variable / complete / proper

Local Search

- Solution representation: `var{int} a[|V|](1..K)`
- Neighborhood: one-exchange restricted to feasible moves
  Kempe chains



- Evaluation function: $f(s) = -\sum_{i=1}^{k} |C_i|^2$
  favours few large color classes wrt. many small color classes

Local Search for GCP
Iterated Greedy

Scheme: k-variable / complete / proper

Local Search

- Solution representation: `var{int} a[|V|](1..K)`
- Neighborhood: permutation of color classes + greedy algorithm
- Evaluation function: number of colors

Theorem

Let φ be a k-coloring of a graph G and π a permutation such that if $\varphi(v_{\pi(i)}) = \varphi(v_{\pi(m)}) = c$ then $\varphi(v_{\pi(j)}) = c, \forall i \leq j \leq m$.
Applying the greedy algorithm to π will produce a coloring using k or fewer colors.

Local Search for GCP

Scheme: k-variable / complete / improper

Local Search

- Solution representation: `var{int} a[|V|](1..K)`
- Neighborhood: one-exchange
- Evaluation function: $f(s) = -\sum_{i=1}^{k} |C_i|^2 + \sum_{i=1}^{k} 2|C_i||E_i|$

Ev. function chosen in such a way that an improvement in feasibility
(in the worst case by coloring a vertex to a new color class)
offsets any improvement in solution quality
(in the best case by moving a vertex to the first color class).

References

Chiarandini M., Dumitrescu I., and Stützle T. (2007). **Stochastic local search algorithms for the graph colouring problem**. In *Handbook of Approximation Algorithms and Metaheuristics*, edited by T.F. Gonzalez, Computer & Information Science Series, pp. 63.1–63.17. Chapman & Hall/CRC, Boca Raton, FL, USA. Preliminary version available as *Tech. Rep.* AIDA-05-03 at Intellectics Group, Computer Science Department, Darmstadt University of Technology, Darmstadt, Germany.