7.2.6 Inference algorithm

Below, we discuss three local search algorithms for CORRELATION-CLUSTERING. For simplicity, we define $c : S \to \mathbb{R}$ such that

$$\forall \{a, a'\} \in S\colon \quad c_{\{aa'\}} = -\langle \theta, x_{\{a,a'\}} \rangle \tag{7.16}$$

and write the objective function $\varphi : \{0,1\}^S \to \mathbb{R}$ such that

$$\forall y \in \{0,1\}^S\colon \quad \varphi(y) = \sum_{\{a,a'\} \in S} c_{\{a,a'\}}\, y_{\{a,a'\}} \tag{7.17}$$

Greedy joining

The greedy joining algorithm starts from any initial decomposition and searches for decompositions with lower objective value by joining pairs of components recursively. By this procedure, components can only grow, and the number of components decreases by precisely one in every step. Thus, one typically starts from the finest decomposition Π_0 of $G = (A, E)$ into one-elementary components.

Definition 18 For any graph $G = (A, E)$ and any disjoint sets $B, C \subseteq A$, the pair $\{B, C\}$ is called *neighboring* in G iff there exist nodes $b \in B$ and $c \in C$ such that $\{b, c\} \in E$.

For any decomposition Π of a graph $G = (A, E)$, we define

$$\mathcal{E}_\Pi = \left\{ \{B, C\} \in \tbinom{\Pi}{2} \,\middle|\, \exists b \in B\, \exists c \in C\colon \{b, c\} \in E \right\} \ . \tag{7.18}$$

Definition 19 For any decomposition Π of $G = (A, E)$ and any $\{B, C\} \in \mathcal{E}_\Pi$, let $\mathrm{join}_{BC}[\Pi]$ be the decomposition of G obtained by joining the sets B and C in Π, i.e.

$$\mathrm{join}_{BC}[\Pi] = (\Pi \setminus \{B, C\}) \cup \{B \cup C\} \ . \tag{7.19}$$

Algorithm 3 The greedy joining algorithm is defined by the recursion below.

$\Pi' = \text{greedy-joining}(\Pi)$

---

choose $\{B, C\} \in \underset{\{B',C'\} \in \mathcal{E}_\Pi}{\operatorname{argmin}} \ \varphi(y^{\mathrm{join}_{B'C'}[\Pi]}) - \varphi(y^\Pi)$
if $\varphi(y^{\mathrm{join}_{BC}[\Pi]}) - \varphi(y^\Pi) < 0$
    $\Pi' := \text{greedy-joining}(\mathrm{join}_{BC}[\Pi])$
else
    $\Pi' := \Pi$

Exercise 8 *a) Write the difference $\varphi(y^{\mathrm{join}_{B'C'}[\Pi]}) - \varphi(y^\Pi)$ in terms of the c defined in* (7.16).
b) Implement greedy joining efficiently.
c) Establish a bound on the time complexity of your implementation.

Greedy moving

The greedy moving algorithm starts from any initial decomposition, e.g., the fixed point of greedy joining. It seeks to lower the objective value by recursively moving individual nodes from one component to a neighboring component, or to a new component. When a node is moved to a new component, the number of components can increase. When the last node is moved from a component, the number of components decreases.

Definition 20 For any graph G, a component G' of G is called *maximal* iff there is no subgraph of G that is both a strict supergraph of G' and a component of G.

Definition 21 For any graph $G = (A, E)$ and any decomposition Π of G, the decomposition Π is called *coarsest* iff, for every $U \in \Pi$, the component $(U, E \cap \binom{U}{2})$ induced by U is maximal.

Lemma 14 *For any graph G, the coarsest decomposition is unique.*

For any graph G, we denote the coarsest decomposition by Π^*_G.

Definition 22 For any graph $G = (A, E)$, any decomposition Π of A and any $a \in A$, choose U_a to be the unique $U_a \in \Pi$ such that $a \in U_a$, and let

$$\mathcal{N}_a = \{\varnothing\} \cup \{W \in \Pi \,|\, a \notin W \wedge \exists w \in W \colon \{a, w\} \in E\} \tag{7.20}$$

$$G_a = \left(U_a \setminus \{a\},\, E \cap \tbinom{U_a \setminus \{a\}}{2}\right) \tag{7.21}$$

For any $U \in \mathcal{N}_a$, let $\mathrm{move}_{aU}[\Pi]$ the decomposition of A obtained by moving the node a to the set U, i.e.

$$\mathrm{move}_{aU}[\Pi] = \Pi \setminus \{U_a, U\} \cup \{U \cup \{a\}\} \cup \Pi^*_{G_a} \ . \tag{7.22}$$

Algorithm 4 The greedy moving algorithm is defined by the recursion below.

$\Pi' = \text{greedy-moving}(\Pi)$

---

choose $(a, U) \in \underset{(a', U') \in A \times (\Pi \cup \{\varnothing\})}{\operatorname{argmin}} \varphi(y^{\mathrm{move}_{a'U'}[\Pi]}) - \varphi(y^{\Pi})$
if $\varphi(y^{\mathrm{move}_{aU}[\Pi]}) - \varphi(y^{\Pi}) < 0$
    $\Pi' := \text{greedy-moving}(\mathrm{move}_{aU}[\Pi])$
else
    $\Pi' := \Pi$

Exercise 9 *a) Write the difference $\varphi(y^{\mathrm{move}_{aU}[\Pi_t]}) - \varphi(y^{\Pi_t})$ in terms of the c defined in* (7.16).
b) Implement greedy moving.

Greedy moving using the technique of Kernighan and Lin (1970)

Both algorithms discussed above terminate as soon as no transformation (join and move, resp.) leads to a partition with strictly lower objective value. This can be sub-optimal in case transformations that increase the objective value at one point in the recursion can lead to transformations that decrease the objective value at later points in the recursion and the decrease overcompensates the increase. A generalization of greedy local search introduced by Kernighan and Lin (1970) can escape such sub-optimal fixed points. It is applied to greedy moving below.

Algorithm 5 The greedy moving algorithm generalized by the technique of Kernighan and Lin (1970) is defined by the recursion below.

$\Pi' = \text{greedy-moving-kl}(\Pi)$

$\Pi_0 := \Pi$
$\delta_0 := 0$
$A_0 := A$
$t := 0$
repeat (build sequence of moves)
  choose $(a_t, U_t) \in \operatorname{argmin}_{(a,U) \in A_t \times (\Pi \cup \{\varnothing\})} \varphi(y^{\text{move}_{aU}[\Pi_t]}) - \varphi(y^{\Pi_t})$
  $\Pi_{t+1} := \text{move}_{a_t U_t}[\Pi_t]$
  $\delta_{t+1} := \varphi(y^{\Pi_{t+1}}) - \varphi(y^{\Pi_t}) < 0$
  $A_{t+1} := A_t \setminus \{a_t\}$ (move a_t only once)
  $t := t + 1$
until $A_t = \varnothing$
$\hat{t} := \min \operatorname{argmin}_{t' \in \{0,\ldots,|A|\}} \sum_{\tau=0}^{t'} \delta_\tau$ (choose sub-sequence)
if $\sum_{\tau=0}^{\hat{t}} \delta_\tau < 0$
  $\Pi' := \text{greedy-moving-kl}(\Pi_{\hat{t}})$ (recurse)
else
  $\Pi' := \Pi$ (terminate)

Exercise 10 *a) Implement greedy moving using the technique of Kernighan and Lin (1970).*
b) Generalize the greedy joining algorithm using the technique of Kernighan and Lin (1970).
c) Implement greedy joining using the technique of Kernighan and Lin (1970).