Correlation Clustering Revisited

Nir Ailon [1] Edo Liberty [2]

[1] Google Research
[2] Yale University, Google Research.

Local Search

Local search is faced with gluing together many pieces of information regarding a specific business or location into a dependable coherent data source.

There are many questions with only partial answers

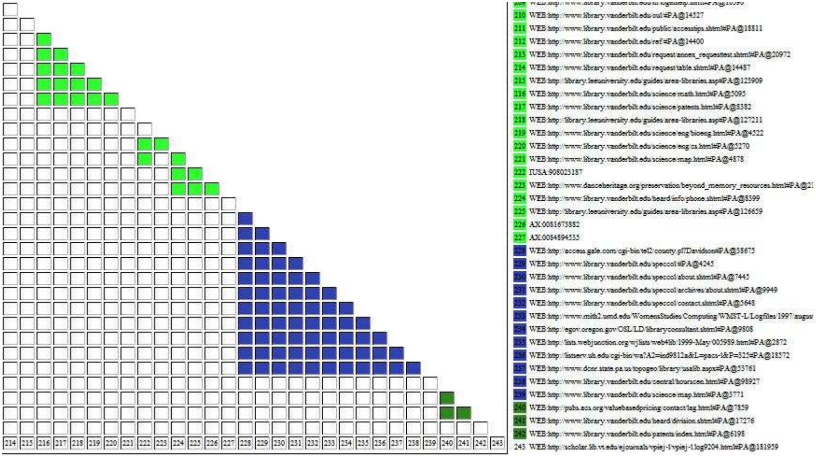
- How do you find duplicate entities?
- What defines a single entity?
- How do you decide what piece of information belongs to what entity?

Local Search

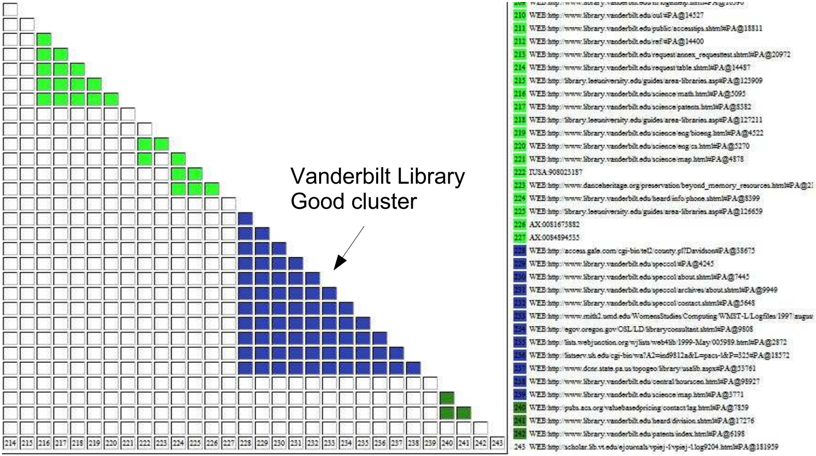
Ideal pipeline:

1. For each piece of information search any others with matching terms. These will be suspects.
2. For each suspect and information element run a classifier to decide whether you think they belong to the same business (most of the work is put into this stage)
3. Divide all information elements into groups such that:
   - Each information element is assigned only to one group.
   - Each group corresponds to one entity.
   - No entity is represented by more than one group.

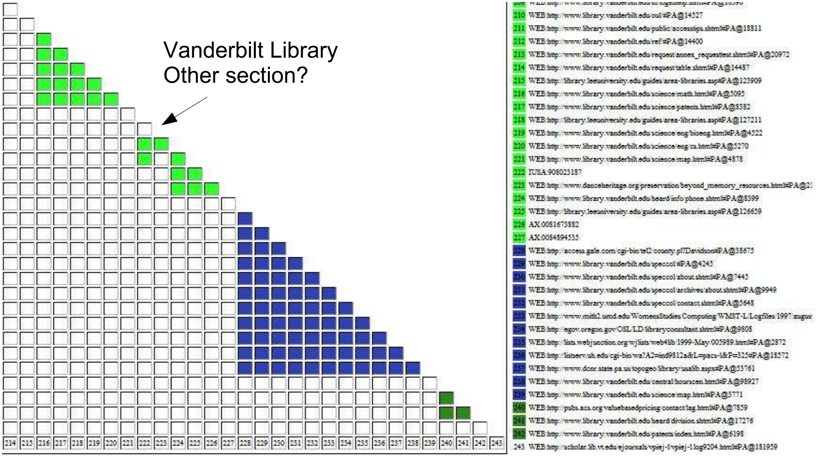
Local Search Graph



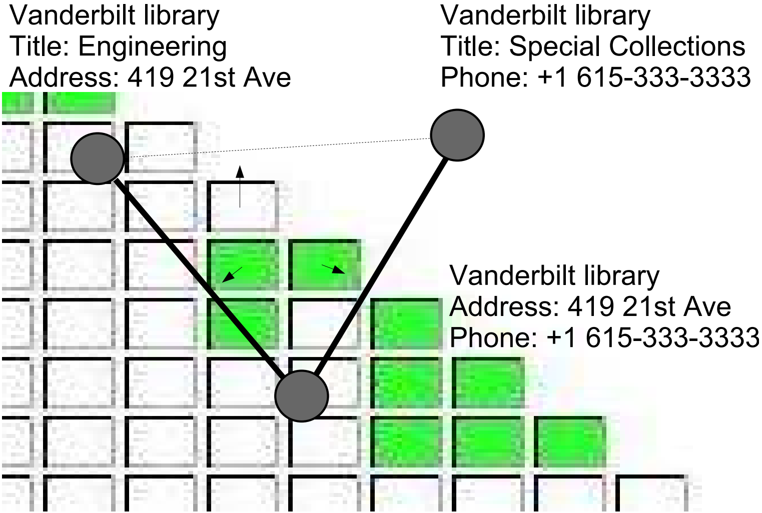
Local Search Graph



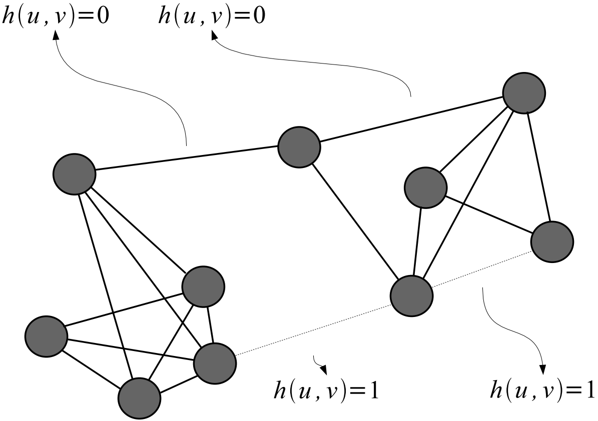
Local Search Graph



Local Search Graph

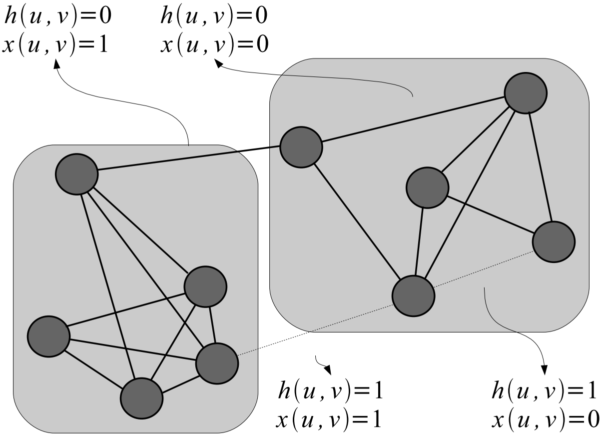


Correlation Clustering Input



This is the input to an algorithm that performs correlation clustering. A set of *n* elements (information pieces) and a binary ”distance” between them *h* (the classifier).

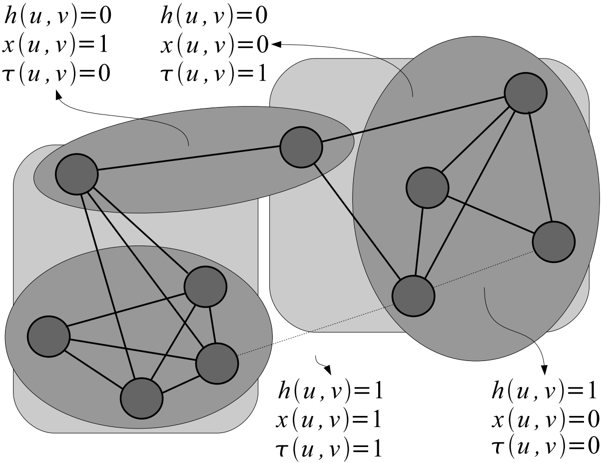
Agnostic Correlation Clustering



Minimize disagreement between output x and input h.

$$f(x,h) = \sum_{u<v} x(u,v)\overline{h(u,v)} + \overline{x(u,v)}h(u,v)$$

Correlation Clustering Revisited



Minimize disagreement between output x and ground truth τ.

$$f(x,\tau^*) = \sum_{u<v} x(u,v)\overline{\tau^*(u,v)} + \overline{x(u,v)}\tau^*(u,v)$$

How can we design such algorithms if we do not know τ^*?

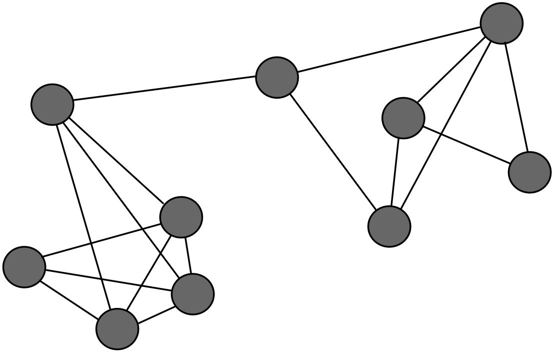
An algorithm is a *C*-approximation if, given *h* it produces *x* in polynomial time such that:

$$\forall\tau \;\; f(x,\tau^*) \leq Cf(h,\tau^*)$$

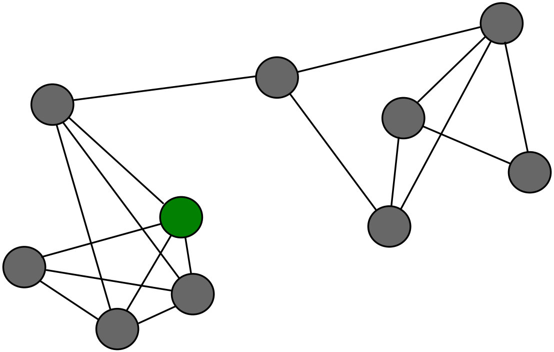
An algorithm is a *randomized* *C*-approximation if, given *h* it draws *x* in polynomial from a distribution $\mathcal{D}$ such that:

$$\forall\tau^* \;\; E_{x\sim\mathcal{D}}[f(x,\tau^*)] \leq Cf(h,\tau^*)$$

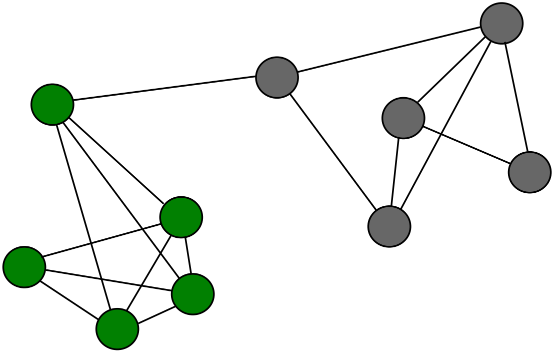
Quick Cluster Algorithm



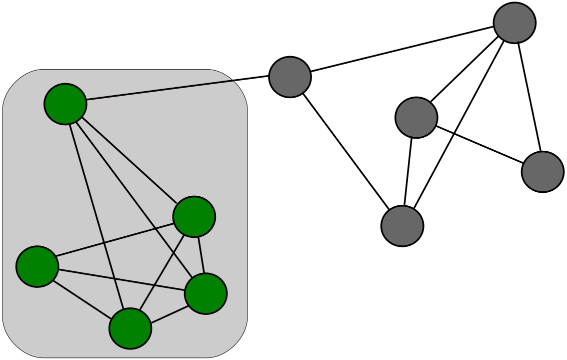
Quick Cluster Algorithm



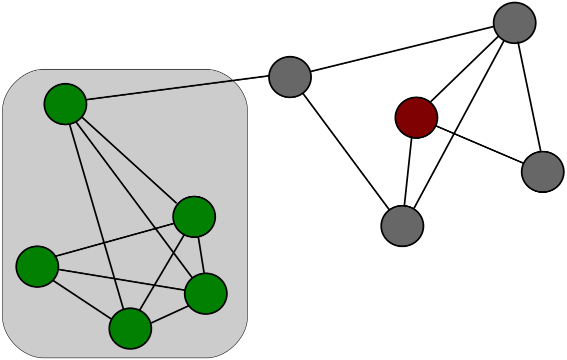
Quick Cluster Algorithm



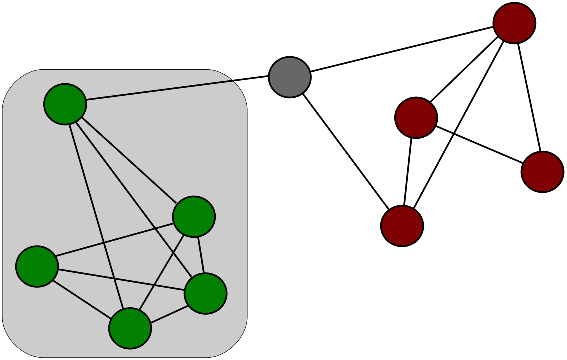
Quick Cluster Algorithm



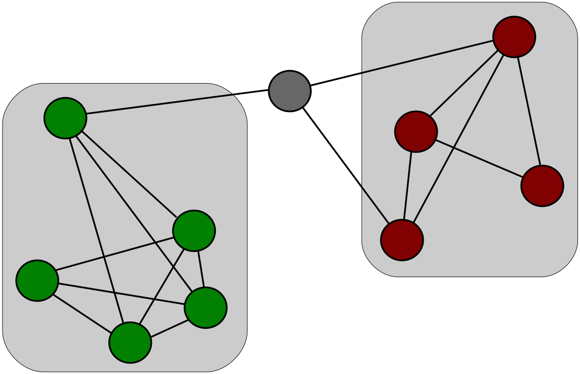
Quick Cluster Algorithm



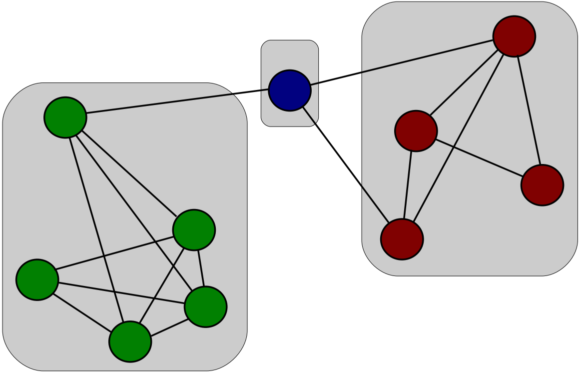
Quick Cluster Algorithm



Quick Cluster Algorithm



Quick Cluster Algorithm



Statement of results

Theorem

Let QC denote the output distribution of QuickCluster .

$$\forall \tau^* \;\; E_{x\sim QC}\left[f(x,\tau^*)\right] \leq 2f(h,\tau^*)$$

We start by calculating

$$E_{x\sim QC}\left[f(x,\tau^*)\right] = E_{x\sim QC}\left[\sum_{u<v} x(u,v)\overline{h(u,v)} + \overline{x(u,v)}h(u,v)\right]$$

Each pair $\{u,v\}$ is either joined or separated once.

$$\begin{aligned}
E_{x\sim QC}\left[\overline{x(u,v)}\right] &= p_{uv}\overline{h(u,v)} + \sum_{w\neq u,v}\frac{1}{3}p_{uvw}[\overline{h(w,u)}\;\overline{h(w,v)}] \\
E_{x\sim QC}\left[x(u,v)\right] &= p_{uv}h(u,v) \\
&\quad + \sum_{w\neq u,v}\frac{1}{3}p_{uvw}[h(w,u)\overline{h(w,v)} + \overline{h(w,u)}h(w,v)]
\end{aligned}$$

$E_{x\sim QC}[f(\tau^*, x)]$ cont'd

We define a few handy notations:

$$\begin{aligned}
L(u,v) &:= h(u,v)\overline{\tau^*(u,v)} + \overline{h(u,v)}\tau^*(u,v) \\
\beta(u,v;w) &:= \overline{h(w,u)}\ \overline{h(w,v)}\tau^*(u,v) \\
&\quad + h(w,u)\overline{h(w,v)}\ \overline{\tau^*(u,v)} + \overline{h(w,u)}h(w,v)\ \overline{\tau^*(u,v)} \\
B(u,v,w) &:= \frac{1}{3}[\beta(u,v;w) + \beta(v,w;u) + \beta(w,u;v)]
\end{aligned}$$

Putting it all together we get that:

$$\boxed{E_{x\in QC}[f(\tau^*, x)] = \sum_{u<v} p_{uv} L(u,v) + \sum_{u<v<w} p_{uvw} B(u,v,w)}$$

Computing $f(h, \tau^*)$

Reminder

$$f(h,\tau^*) = \sum_{u<v} L(u,v) \quad L(u,v) = h(u,v)\overline{\tau^*(u,v)} + \overline{h(u,v)}\tau^*(u,v)$$

We notice that because each pair is decided upon exactly once:

$$1 = p_{uv} + \sum_{w \neq u,v} \frac{1}{3} p_{uvw} \overline{h(w,u)h(w,v)}$$

We have that:

$$\begin{aligned}
\sum_{u<v} L(u,v) &:= \sum_{u<v} p_{uv} L(u,v) + \sum_{u<v<w} p_{uvw} A(u,v,w) \\
A(u,v,w) &:= \frac{1}{3}[\overline{h(w,u)h(w,v)} L(u,v) \\
&\quad + \overline{h(u,v)h(u,w)} L(v,w) + \overline{h(v,w)h(v,u)} L(w,u)]
\end{aligned}$$

$$\boxed{f(h,\tau^*) = \textstyle\sum_{u<v} p_{uv} L(u,v) + \sum_{u<v<w} p_{uvw} A(u,v,w)}$$

Putting it all together

$$\boxed{E_{x \in QC}\left[f(\tau^*, x)\right] = \sum_{u<v} p_{uv} L(u,v) + \sum_{u<v<w} p_{uvw} B(u,v,w)}$$

$$\boxed{f(h, \tau^*) = \sum_{u<v} p_{uv} L(u,v) + \sum_{u<v<w} p_{uvw} A(u,v,w)}$$

It is not hard to verify that:

$$\forall\ u, v, w \quad B(u,v,w) \leq 2A(u,v,w)$$

Which concludes the proof for

$$\boxed{E_{x \in QC}\left[f(\tau^*, x)\right] \leq 2f(h, \tau^*)}$$

Running time and Game Theory?

In order to investigate the best running time of any randomized algorithm we refresh our game theory.

Let us define a zero-sum two player game.

- Players A and h have m_A and m_h possible *pure* strategies, $\{A_1, \dots, A_{m_A}\}$ and $\{h_1, \dots, h_{m_x}\}$.
- For pure strategies A_i and h_j, the payoff T of player h is $T(A_i, h_j)$.
- Each player can choose a *mixed* strategy, which is a probability distribution over *pure* strategies (denoted by **A** and **h**).
- For mixed strategies, the payoff of player h is $E_{h\sim\mathbf{h},\ A\sim\mathbf{A}}[T(A, h)]$
- The payoff of player A is minus that of player h (zero-sum).

Von Neumann's Minimax Theorem

Theorem (Von Neumann)

$$\max_{\mathbf{h}} \min_{A} E_{h\sim\mathbf{h}}[T(A,h)] \leq \min_{\mathbf{A}} \max_{h} E_{A\sim\mathbf{A}}[T(A,h)]$$

If (i) player A chooses an algorithm ,(ii) player h chooses an input, and (iii) $T(A,h)$ gives the running time of A on h, then the above inequality is referred to as Yao's principle.
LHS: The expected running time of the fastest possible *deterministic* algorithm on its worst possible *distribution* over inputs.
RHS: The expected running time of the fastest *randomized* algorithm on its *single* worst input.

Running time with pairwise queries

Consider h such that only one pair $\{u_0, v_0\}$ is clustered together and the rest are singletons. Also think about the case where $\tau^* = h$. The algorithm must return x such that

$$f(x, \tau^*) \leq 2f(h, \tau^*) = 0 \quad \rightarrow x = h$$

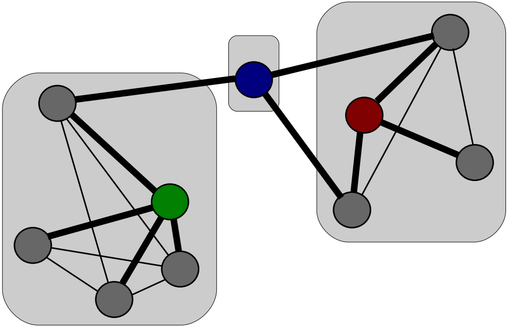
The algorithm must find $\{u_0, v_0\}$.

Any deterministic algorithm performs an expected $O(n^2)$ operations to find $\{u_0, v_0\}$.

Thus: No randomized algorithm executes faster than QuickCluster .

Running time with neighborhood queries

Assume that we are given for each element u the set of neighbors $N(u) = \{u\} \cup \{v \mid h(u, v) = 0\}$.



Each edge in h from a center either captures an element or disagrees with the final clustering x.

Running time with neighborhood queries

We have that

$$T(A, h) \leq n + f(x, h)$$

From the triangle inequality on f we have that:

$$f(x, h) \leq f(x, \tau^*) + f(\tau^*, h)$$

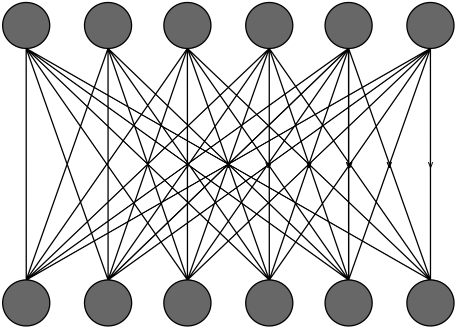
From the theorem we have:

$$E_{x \sim QC}\left[f(x, \tau^*)\right] \leq 2f(\tau^*, h)$$

Finally $E[T(A, h)] = O(n + f(\tau^*, h))$.

Open Questions

The worst single instance for *h* that we know of:



If τ^* clusters everything together then:

$$E_{x \sim QC}\left[f(x, \tau^*)\right] = 1.5 f(\tau^*, h)$$

Can you think of a worse example?
Is QuickCluster a 1.5-approximation algorithm?

Fin