1 grober Ablauf

Es gilt in einem bipartiten Graphen (L, R, E) ein maximum Matching zu finden. L ist nochmal aufgeteilt in L_1 und L_2. Ein Knoten in L_2 will be available for matching with given prob p_{l_2}.

In der ersten Stage werden die Knoten in L_1 gematcht nur mit den Informationen, die durch die Wahrscheinlichkeiten gegeben sind. In der zweiten Stage werden entsprechend p die verfügbaren Knoten aus L_2 gewonnen und gematcht.

Und das Ziel ist es, die Größe des finalen Matchings zu maximieren.

2 genauer

Es werden zwei Mengen eingeführt:

$$\boldsymbol{x}_1 = \{x_{l_1 r} \in \{0,1\} | \{l_1, r\} \in E, l_1 \in L_1\} \tag{1}$$

$$\boldsymbol{x}_2 = \{x_{l_2 r} \in \{0,1\} | \{l_2, r\} \in E, l_2 \in L_2\} \tag{2}$$

mit $x_{lr} = 1$ iff (kp wieso nicht einfach nur if) $\{l, r\} \in M \subset E$. Damit wird also angezeigt, dass sich eine Kante im Matching befindet oder nicht. Die x_{lr} sind nur dann 1, wenn diese Kante im Matching enthalten ist. Zusätzlich gibt es jetzt noch:

$$t = \{t_{l_2} \in \{0,1\} | l_2 \in L_2\} \tag{3}$$

mit $t_{l_2} = 1$ iff l_2 is available for matching in the second stage. *Also nachdem entsprechend der Wahrscheinlichkeiten ausgewählt wurde?*

Jetzt mit der cost function $\mathcal{E}(x_1, x_2, t)$, die von den noch verfügbaren Knoten diejenigen zählt, die noch nicht gematcht sind, finde:

$$x_1^* = \underset{x_1}{\operatorname{argmin}} \, \mathbb{E}_t \min_{x_2} \mathcal{E}(x_1, x_2, t). \tag{4}$$

Ich möchte also das x_1 wählen, welches den Teil $\mathbb{E}_t \min_{x_2} \mathcal{E}(x_1, x_2, t)$ minimiert. Dieser Teil ist aber nichttrivial stark von x_1 abhängig und das ist das Hauptproblem.

Wenn ich die Größe des finalen Matchings maximieren möchte muss ich dafür sorgen, dass möglichst wenig Knoten übrig bleiben die nicht mehr dem Matching hinzugefügt werden können. Die cost function verstehe ich so, dass sie am Ende zählt, wie viele Knoten nicht gematcht sind. Wenn x_1 feststeht und die t_{l_2} bestimmt sind, ist es straighforward to find the optimal x_2 (weil das dann einfach nur ein normales Matchingproblem ist?).

Dazu gibt es noch Constraints. Es wird die Hilfsmenge $\partial r = \{l \in L | \{l, r\} \in E\}$, welche alle Knoten in L enthält, die mit dem Knoten $r \in R$ durch eine Kante in E verbunden sind.:

- constraint 1:

$$\sum_{l \in \partial r} x_{lr} \leq 1 \quad \forall r \in R \tag{5}$$

  Für jeden Knoten in R darf es im Matching also nur maximal einen Knoten in L geben, der mit r gematcht ist. Diese Beschränkung setzt also die Definition des Matchings um.

- constraint 2:

$$\sum_{r \in \partial l_1} x_{l_1 r} \leq 1 \quad \forall l_1 \in L_1 \tag{6}$$

  Das selbe wie eben soll jetzt auch für alle Knoten aus L_1 gelten: Es darf maximal eine Kante im Matching von den Knoten l_1 abgehen.

- constraint 3:

$$\sum_{r \in \partial l_2} x_{l_2 r} \leq t_{l_2} \quad \forall l_2 \in L_2 \tag{7}$$

  Das sagt aus, dass falls ein Knoten aus L_2 am Ende nicht durch die Wahrscheinlichkeit realisiert wird, dann darf er auch in keiner Kante im Matching auftauchen. Falls er realisiert wird dann gilt das selbe wie für die anderen Knoten und er darf nur in einer Kante vorkommen.

3 Fragen

3.1 Offen: Werden aus L_1 alle Knoten in der ersten stage gematcht?

Kann es sein, dass die Wahrscheinlichkeiten nahelegen, dass bestimmte Knoten lieber nicht gematcht werden?

Das zweite constraint schreibt auf jeden Fall nicht vor, dass alle Knoten aus L_1 gematcht werden müssen, es kann auch Knoten geben von denen keine Kante im Matching berücksichtigt wird.

Also würde ich eher nein auf die Frage antworten.

3.2 Offen: zur cost function

Gehört mit der ersten Frage zusammen.
Die cost function verstehe ich so, dass sie am Ende zählt, wie viele Knoten nicht gematcht sind. Im Text steht aber: "counting the number of umatched vertice samong the available ones". Zählt die also nur die Knoten für die $t_{l_2} = 1$ war und die am Ende übrig bleiben? Dann würden ja Knoten, die aus L_1 übrig geblieben sind nicht mitgezählt.

3.3 Offen: Ist das jetzt ein lineares Programm?

Und wenn, wieso gibt es das 3. constraint, das sich auf die x_2 bezieht, obwohl wir nach einem x_1 suchen.