Several Graph Problems and their LP formulation

Nathann Cohen*

Available since : March 4, 2010
Updated : July 22, 2010

Abstract

This document is meant as an explanation of several graph theoretical functions defined in Sage's Graph Library (http://www.sagemath.org/), which use Linear Programming to solve optimization of existence problems.

Contents

*nathann.cohen@gmail.com

1 Maximum average degree

The average degree of a graph G is defined as $ad(G) = \frac{2|E(G)|}{|V(G)|}$. The maximum average degree of G is meant to represent its densest part, and is formally defined as :

$$mad(G) = \max_{H \subseteq G} ad(H)$$

Even though such a formulation does not show it, this quantity can be computed in polynomial time through Linear Programming. Indeed, we can think of this as a simple flow problem defined on a bipartite graph. Let D be a directed graph whose vertex set we first define as the disjoint union of $E(G)$ and $V(G)$. We add in D an edge between $(e, v) \in E(G) \times V(G)$ if and only if v is one of e's endpoints. Each edge will then have a flow of 2 (through the addition in D of a source and the necessary edges) to distribute among its two endpoints. We then write in our linear program the constraint that each vertex can absorb a flow of at most z (add to D the necessary sink and the edges with capacity z).

Clearly, if $H \subseteq G$ is the densest subgraph in G, its $|E(H)|$ edges will send a flow of $2|E(H)|$ to their $|V(H)|$ vertices, such a flow being feasible only if $z \geq \frac{2|E(H)|}{|V(H)|}$. An elementary application of the max-flow/min-cut theorem, or of Hall's bipartite matching theorem shows that such a value for z is also sufficient. This LP can thus let us compute the Maximum Average Degree of the graph.

LP Formulation :

- Mimimize : z
- Such that :
  - a vertex can absorb at most z

$$\forall v \in V(G), \sum_{\substack{e \in E(G) \\ e \sim v}} x_{e,v} \leq z$$

  - each edge sends a flow of 2

$$\forall e = uv \in E(G), x_{e,u} + x_{e,u} = 2$$

- $x_{e,v}$ real positive variable

REMARK : In many if not all the other LP formulations, this Linear Program is used as a constraint. In those problems, we are always at some point looking for a subgraph H of G such that H does not contain any cycle. The edges of G are in this case variables, whose value can be equal to 0 or 1 depending on whether they belong to such a graph H. Based on the observation that the Maximum Average Degree of a tree on n vertices is exactly its average degree ($= 2 - 2/n < 1$), and that any cycles in a graph ensures its average degree is larger than 2, we can then set the constraint that $z \leq 2 - \frac{2}{|V(G)|}$. This is a handy way to write in LP the constraint that "the set of edges belonging to H is acyclic". For this to work, though, we need to ensure that the variables corresponding to our edges are binary variables.

corresponding method : `Graph.maximum_average_degree`

2 Traveling Salesman Problem

Given a graph G whose edges are weighted by a function $w : E(G) \to \mathbb{R}$, a solution to the TSP is a hamiltonian (spanning) cycle whose weight (the sum of the weight of its edges) is minimal. It is easy to define both the objective and the constraint that each vertex must have exactly two neighbors, but this could produce solutions such that the set of edges define the disjoint union of several cycles. One way to formulate this linear program is hence to add the constraint that, given an arbitrary vertex v, the set S of edges in the solution must contain no cycle in $G - v$, which amounts to checking that the set of edges in S no adjacent to v is of maximal average degree strictly less than 2, using the remark from section 1.

We will then, in this case, define variables representing the edges included in the solution, along with variables representing the weight that each of these edges will send to their endpoints.

LP Formulation :

- Mimimize
$$\sum_{e \in E(G)} w(e) b_e$$
- Such that :
  - Each vertex is of degree 2
$$\forall v \in V(G), \sum_{\substack{e \in E(G) \\ e \sim v}} b_e = 2$$
  - No cycle disjoint from a special vertex v^*
    - Each edge sends a flow of 2 if it is taken
$$\forall e = uv \in E(G - v^*), x_{e,u} + x_{e,v} = 2b_e$$
    - Vertices receive strictly less than 2
$$\forall v \in V(G - v^*), \sum_{\substack{e \in E(G) \\ e \sim v}} x_{e,v} \leq 2 - \frac{2}{|V(G)|}$$
- Variables
  - $x_{e,v}$ real positive variable (flow sent by the edge)
  - b_e binary (is the edge in the solution ?)

corresponding method :
`Graph.traveling_salesman_problem`

3 Edge-disjoint spanning trees

This problem is polynomial by a result from Edmonds. Obviously, nothing ensures the following formulation is a polynomial algorithm as it contains many integer variables, but it is still a short practical way to solve it.

This problem amounts to finding, given a graph G and an integer k, edge-disjoint spanning trees $T_1, \ldots, T_k$ which are subgraphs of G. In this case, we will chose to define a spanning tree as an acyclic set of $|V(G)|-1$ edges.

LP Formulation :

- Maximize : nothing
- Such that :
  - An edge belongs to at most one set

$$\forall e \in E(G), \sum_{i \in [1,\ldots,k]} b_{e,k} \leq 1$$

  - Each set contains $|V(G)| - 1$ edges

$$\forall i \in [1, \ldots, k], \sum_{e \in E(G)} b_{e,k} = |V(G)| - 1$$

  - No cycles
    - In each set, each edge sends a flow of 2 if it is taken

$$\forall i \in [1, \ldots, k], \forall e = uv \in E(G), x_{e,k,u} + x_{e,k,u} = 2b_{e,k}$$

    - Vertices receive strictly less than 2

$$\forall i \in [1, \ldots, k], \forall v \in V(G), \sum_{\substack{e \in E(G) \\ e \sim v}} x_{e,k,v} \leq 2 - \frac{2}{|V(G)|}$$

- Variables
  - $b_{e,k}$ binary (is edge e in set k ?)
  - $x_{e,k,u}$ positive real (flow sent by edge e to vertex u in set k)

corresponding method : `Graph.edge_disjoint_spanning_trees`

4 Steiner tree

Finding a spanning tree in a Graph G can be done in linear time, whereas computing a Steiner Tree is NP-hard. The goal is in this case, given a graph, a weight function $w : E(G) \to \mathbb{R}$ and a set S of vertices, to find the tree of minimum cost connecting them all together. Equivalently, we will be looking for an acyclic subgraph $H$of G containing $|V(H)|$ vertices and $|E(H)| = |V(H)| - 1$ edges, which contains each vertex from S

LP Formulation :

- Minimize :

$$\sum_{e \in E(G)} w(e) b_e$$

- Such that :
  - Each vertex from S is in the tree

$$\forall v \in S, \sum_{\substack{e \in E(G) \\ e \sim v}} b_e \geq 1$$

  - c is equal to 1 when a vertex v is in the tree

$$\forall v \in V(G), \forall e \in E(G), e \sim v, b_e \leq c_v$$

  - The tree contains $|V(H)|$ vertices and $|E(H)| = |V(H)| - 1$ edges

$$\sum_{v \in G} c_v - \sum_{e \in E(G)} b_e = 1$$

  - No Cycles
    - Each edge sends a flow of 2 if it is taken

$$\forall e = uv \in E(G), x_{e,u} + x_{e,u} = 2b_{e,k}$$

    - Vertices receive strictly less than 2

$$\forall v \in V(G), \sum_{\substack{e \in E(G) \\ e \sim v}} x_{e,v} \leq 2 - \frac{2}{|V(G)|}$$

- Variables :
  - b_e binary (is e in the tree ?)
  - c_v binary (does the tree contain v ?)
  - $x_{e,v}$ real positive variable (flow sent by the edge)

corresponding method : `Graph.steiner_tree`

5 Linear arboricity

The linear arboricity of a graph G is the least number k such that the edges of G can be partitionned into k classes, each of them being a forest of paths (the disjoints union of paths – trees of maximal degree 2). The corresponding LP is very similar to the one giving edge-disjoint spanning trees

LP Formulation :

- Maximize : nothing
- Such that :
  - An edge belongs to exactly one set

$$\forall e \in E(G), \sum_{i \in [1,\dots,k]} b_{e,k} = 1$$

  - Each class has maximal degree 2

$$\forall v \in V(G), \forall i \in [1,\dots,k], \sum_{\substack{e \in E(G) \\ e \sim v}} b_{e,k} \leq 2$$

  - No cycles
    - In each set, each edge sends a flow of 2 if it is taken

$$\forall i \in [1,\dots,k], \forall e = uv \in E(G), x_{e,k,u} + x_{e,k,v} = 2b_{e,k}$$

    - Vertices receive strictly less than 2

$$\forall i \in [1,\dots,k], \forall v \in V(G), \sum_{\substack{e \in E(G) \\ e \sim v}} x_{e,k,v} \leq 2 - \frac{2}{|V(G)|}$$

- Variables
  - $b_{e,k}$ binary (is edge e in set k ?)
  - $x_{e,k,u}$ positive real (flow sent by edge e to vertex u in set k)

6 Acyclic edge coloring

An edge coloring with k colors is said to be acyclic if it is proper (each color class is a matching – maximal degree 1), and if the union of the edges of any two color classes is acyclic. The corresponding LP is almost a copy of the previous one, except that we need to ensure that $\binom{k}{2}$ different classes are acyclic.

corresponding methods :
`sage.graphs.graph_coloring.acyclic_edge_coloring`,
`sage.graphs.graph_coloring.linear_arboricity`

7 H-minor

For our purposes, we will just say that finding a minor H in a graph G, consists in :

1. Associating to each vertex $h \in H$ a set S_h of representants in H, different vertices h having disjoints representative sets
2. Ensuring that each of these sets is connected (can be contracted)
3. If there is an edge between h_1 and h_2 in H, there must be an edge between the corresponding representative sets

Here is how we will address these constraints :

1. Easy
2. For any h, we can find a spanning tree in S_h (an acyclic set of $|S_h| - 1$ edges)
3. This one is very costly.
   To each *directed* edge g_1g_2 (I consider g_1g_2 and g_2g_1 as different) and every edge h_1h_2 is associated a binary variable which can be equal to one only if g_1 represents h_1 and g_2 represents g_2. We then sum all these variables to be sure there is at least one edge from one set to the other.

LP Formulation :

- Maximize : nothing
- Such that :
  - A vertex $g \in V(G)$ can represent at most one vertex $h \in V(H)$

$$\forall g \in V(G), \sum_{h \in V(H)} rs_{h,g} \leq 1$$

  - An edge e can only belong to the tree of h if both its endpoints represent h

$$\forall e = g_1g_2 \in E(G), t_{e,h} \leq rs_{h,g_1} \text{ and } t_{e,h} \leq rs_{h,g_2}$$

  - In each representative set, the number of vertices is one more than the number of edges in the corresponding tree

$$\forall h, \sum_{g \in V(G)} rs_{h,g} - \sum_{e \in E(G)} t_{e,h} = 1$$

  - No cycles in the union of the spanning trees
    - Each edge sends a flow of 2 if it is taken

$$\forall e = uv \in E(G), x_{e,u} + x_{e,v} = 2 \sum_{h \in V(H)} t_{e,h}$$

    - Vertices receive strictly less than 2

$$\forall v \in V(G), \sum_{\substack{e \in E(G) \\ e \sim v}} x_{e,k,v} \leq 2 - \frac{2}{|V(G)|}$$

  - $arc_{(g_1,g_2),(h_1,h_2)}$ can only be equal to 1 if g_1g_2 is leaving the representative set of h_1 to enter the one of h_2. (note that this constraints has to be written both for g_1, g_2, and then for g_2, g_1)

$$\forall g_1, g_2 \in V(G), g_1 \neq g_2, \forall h_1h_2 \in E(H)$$

$$arc_{(g_1,g_2),(h_1,h_2)} \leq rs_{h_1,g_1} \text{ and } arc_{(g_1,g_2),(h_1,h_2)} \leq rs_{h_2,g_2}$$

  - We have the necessary edges between the representative sets

$$\forall h_1h_2 \in E(H)$$

$$\sum_{\forall g_1,g_2 \in V(G), g_1 \neq g_2} arc_{(g_1,g_2),(h_1,h_2)} \geq 1$$

- Variables
  - $rs_{h,g}$ binary (does g represent h ? rs = "representative set")
  - $t_{e,h}$ binary (does e belong to the spanning tree of the set representing h ?)
  - $x_{e,v}$ real positive (flow sent from edge e to vertex v)
  - $arc_{(g_1,g_2),(h_1,h_2)}$ binary (is edge g_1g_2 leaving the representative set of h_1 to enter the one of h_2 ?)

corresponding method : `Graph.minor`