Minimum maximal and maximum minimal problems

In many of the optimization problems that we have come across, greedy algorithms do not work because we can get stuck on a local optimum. For example, with the maximum matching problem, greedily adding independent edges to a set may not achieve a maximum set. However, with this problem, we saw that there is in fact a greedy algorithm that works: we just have to generalize the incremental step to be an augmenting path augmentation instead of only edge addition. Indeed, the maximum matching problem is in P on general graphs. Since this problem can get stuck in a local maximum in the first case, we can ask the question of how bad this can be. Let a *maximal matching* be a matching that is maximal with respect to edge addition. Then a *minimum maximal matching* (MMM) is such a set of minimum size. This has its own applications: such a set of edges has the property that it intersects with all other edges of the graph, and can equivalently be defined as an independent *edge dominating set*. (Analoguous to vertex domination which we have seen several variations of.) Unlike the maximum matching problem, the MMM problem is NP-hard, just like vertex domination.

As an example of a maximum minimal problem, consider the following greedy algorithm for proper vertex coloring. Start with all vertices a unique color, and repeatedly merge color classes that have no edges between them, until no such pairs exist. This greedy algorithm attempts to find a proper coloring with few colors. It always finds a proper coloring (which we will call a *total coloring*, but not necessarily minimum. We can now ask the question, what is the largest number of colors in a total coloring of a graph G? This is called the *achromatic number* of G. Just like proper vertex coloring, total coloring is NP-hard in general. Recall that the k-Vertex-Coloring problem is in P when $k = 2$ and NP-complete for $k \geq 3$. We claim see that k-Total-Toloring is in P when k is a constant, by noting that G has a total coloring using k colors if and only if it contains one of a constant set of subgraphs on at most $k(k-1)$ vertices. Checking for the existnce of those subgraphs can then be done in $O(n^{k(k-1)})$ time.

Open problem: ordered vertex coloring

Consider the following problem. You are given a graph G with an ordering on the vertices, labeled as $v_1, \ldots, v_n$. We say that a proper coloring respects the vertex order if every color class is an interval in the ordering: $v_i, v_{i+1}, \ldots, v_j$, for some $1 \leq i \leq j \leq n$. Now, we wish to find the smallest number of colors in a proper coloring that respects the order of the graph, and we will denote this number $\chi_{ord}(G^*)$, where G^* is the graph with the ordering.

Exercise 1. *Give an example of a graph G and two orderings which give different values for $\chi_{ord}(G^*)$.*

Exercise 2. *Show that the problem of determining $\chi_{ord}(G^*)$ is in P.*

Approximation algorithms and NP optimization problems

The class NPO of NP optimization problems is the class of problems whose solutions can be verified in polynomial time, and try to optimize (min or max) some function that maps solutions positive real numbers via some polynomial-time computable function. In many cases, despite the exact problem being NP-hard, there exist efficient approximation algorithms. An α-approximation algorithm for a problem is a polynomial-time algorithm which

finds a solution with a value that is within a constant factor α of the value of an optimal solution. Depending on the problem, one might also consider additive instead of multiplicative approximation.

Minimum maximal matching

Although, as we have seen, MMM is NP-hard in general, it has a straightforward 2-approximation. We know that finding a maximum (which is necessarily also maximal) matching M can be done in polynomial time. Any maximal matching must cover at least one vertex of each edge in M, and therefore any maximal matching (including the minimum one) has at least $|M|/2$ edges. Therefor every algorithm that finds *some* maximal matching is a 2-approximation algorithm.

Exercise 3. *Adapt the above to give a 2-approximation algorithm for the vertex cover problem.*

Exercise 4. *We can compare maximum matching/MMM in the edge setting to maximum independent set/minimum independent dominating set in the vertex setting. Show that the vertex-analogue of above approach does not work as an approximation algorithm for minimum independent dominating set.*

Set-covers

This section is a sketch based on the proof in Chandra Chekuri's notes (2021). We consider a more general setting. We have a set of vertices $V = \{v_1, \dots, v_n\}$, and a set of subsets $\mathcal{S} = S_1, \dots, S_m$ of V. Two related problems: find a subset $\mathcal{T} \subseteq \mathcal{S}$ of cardinality k which covers the most vertices (has the largest union), k-Cover, and, find the smallest number of sets that cover all of U, Set-Cover. Both problems are known to be NP-hard in general. For both problems, consider a greedy algorithm that starts with an empty set $\mathcal{T}_0$ and obtains $\mathcal{T}_{i+1}$ for $0 \leq i \leq k-1$ by adding a set S_{a_i} to $\mathcal{T}_i$ that maximizes the increase in coverage.
We focus on k-Cover one first. To analyze this greedy algorithm, imagine an optimal k-cover $\mathcal{T}'$ and let U be the set of vertices covered. Note that our algorithm does not attempt to cover U specifically. (In fact, U is unknown to our algorithm.) With slgiht abuse of notation, let $U_i = U - \mathcal{T}_i$. We know that U_i is covered by at most k sets from $\mathcal{T}'$, and therefore one of them covers at least $|U_i|/k$ of its vertices. Such a set is not in $\mathcal{T}_i$ since it contains vertices outside of its union. Therefore, $|S_{a_i} - \mathcal{T}_i| \geq |U_i|/k$. Each time we add a set, we decrease the difference between our coverage and the optimal number set by a factor of at least $(1 - 1/k)$. Again, note that these are just numbers; we may not cover U itself at all. If we miss U_i, this only improves our lower bound in the next step. Therefore, if W is the set of vertices covered by the greedy algorithm, we have

$$|W| = (1 - (1 - 1/k)^k)|U| \geq (1 - 1/e)|U| \approx .63|U|.$$

Therefore, the greedy algorithm (which is easily checked to be polynomial) is a $(1 - 1/e)$-approximation algorithm.

Exercise 5. *Deduce that the greedy algorithm for* Set-Cover *gives a* $(\log n + O(1))$*-approximation algorithm.*