The Satisfactory Partition Problem

Ziyou Hu, Nancy Xing

May 2024

Abstract

A satisfactory partition of a graph is a partition of the nodes into (V_1, V_2) such that $\forall v_1 \in V_1$, $d_{V_1}(v_1) \geq d_{V_2}(v_1)$ and $\forall v_2 \in V_2$, $d_{V_1}(v_2) \leq d_{V_2}(v_2)$. Note that $d_{V_i}(v_j)$ denotes the number of neighbors of v_j in V_i. i is either 1 or 2. This project is an extension of the problem of finding non-trivial Pure Strategy Nash Equilibrium (PSNE) of the masking game, and implements several solution concepts outlined by Bazgan et. al (2006).

1 Introduction

The satisfactory partition problem consists of partitioning a graph into 2 subsets such that for every node in each subset, it has at least as many neighbors in the other subset. Within game theory, this problem is equivalent to the masking game. In the masking game, a player will choose to either mask or not mask based on the majority choice of its neighbors. If the majority of a player's neighbors do not mask, the player will not mask; likewise, if the majority of a player's neighbors mask, the player will mask. If the number of neighbors that mask equals the number of neighbors who do not mask, then the player is indifferent. A PSNE in the masking game can be every player playing the same choice (all masking or unmasking), but such results are trivial. However, finding a PSNE that consists of mixed-choices is NP-hard with some exceptions. This project describes and implements several solution concepts to the satisfactory partition problem described by Bazgan et al. (2006).

Solving the satisfactory problem consists of 1) determining whether a satisfactory partition (i.e., a satisfactory pair) exists in a graph and 2) determining which nodes to include in each part of the partition's satisfactory pair. Bazgan et al. summarized graphs where there is a guaranteed to have a satisfactory partition, and provided a linear time algorithm for regular graphs of degree 3 or 4 that have strictly more than 10 vertices. **Please refer to the appendix (separately attached) for a detailed summary.**

2 Problem definition

Satisfactory Partition Problem: Given an unweighted graph, decide if a given graph has a partition of its vertex set into 2 nonempty subsets such that each

vertex has at least many neighbors in its subset as in the other subset.

Let the two subsets consist of the satisfactory pair (V_1, V_2) where each subset contains the nodes on one side of the division. I.e., if $d_y(v)$ represents the number of v's vertices which are adjacent to y, then in the graph's satisfactory pair (V_1, V_2), for every $v \in V$, $d_{v_i}(v) \geq d(v)/2$. A graph G = (V, E) is partitionable if and only if it contains a satisfactory pair (V_1, V_2). Moreover, if a satisfactory pair (V_1, V_2) is given, then a satisfactory partition of G can be determined in polynomial time.

3 Program

Our program is implemented as a python program, sat_part, which adapts several functions from the networkx library. sat_part accepts representations of a graph, determines if a satisfactory partition exists, and outputs the nodes in a satisfactory pair.

Our program assumes that the provided graphs are simple, unweighted, and undirected, non-empty. This fits in with modeling assumptions about the masking game. The graph is simple as neighbors do not influence each other more than once, and multiple edges and self loops are meaningless to gameplay. The graph is unweighted and undirected, implying that players consider their neighbors with equal weight, and that if two neighbors are connected by an edge, both of them will be affected by the other's opinion.

Further, we assume all node will be numbers from 0 to n-1. A trivial partition where 1 set is the vertices set and the other set is the empty set exist for any graph. We only consider non-trivial satisfactory partition, that is, satisfactory partition such that both sets are non-empty. If there is only trivial results, our program will produce no satisfactory partition found.

In the following sections, we describe our module's approach to the four listed sub-cases.

4 3 and 4-regular graphs, $|V| > 10$

Bazgan et. al proves that any 3 and 4-regular graph is partitionable, excluding the special cases of graphs K3,3, K4, and K5, which are not partitionable.

The given algorithm works consistently for 3 and 4-regular graphs with more than 10 nodes. For 3 and 4-regular graphs with less than 10 nodes, the problem will be solvable in constant time via a number of corner cases which are outside the scope of this project.

The algorithm searches for a cycle within the graph of length less than n/2. If such a cycle exists, the nodes within the cycle can be initially added to the first set in the satisfactory pair, V_1. Due to the nature of the cycle, each node in V_1 is guaranteed to have at least as many neighbors in V_1 as in the rest of the graph. The algorithm probes the remaining nodes for possible addition to V_1, i.e., nodes with at least as many neighbors in V_1 as in the rest of the graph.

When the algorithm finishes, V_1 is complete, and the other partition V_2 consists of the remaining nodes in the graph.

5 Graphs bound by a maximum degree of 4

For non-regular graphs, it is possible to solve the satisfactory partition problem in polynomial time. This is possible via identifying disjoint cycles within the graph. Two cycles in a graph are disjoint if they do not share common nodes, and they indicate the basis for a satisfactory pair.

5.1 Graphs bound by a minimum degree of 3 and a maximum degree of 4

The requirements for a satisfactory partition are stricter if the graph has both an upper bound of 4 degrees and a lower bound of 3 degrees. In this instance, the graph is only partitionable if two disjoint cycles exist in the graph. The algorithm first identifies all simple cycles within the graph and evaluates whether they are disjoint. If there are disjoint cycles, the algorithm creates a partition using the disjoint cycles as a basis.

5.2 Graphs only bounded by a maximum degree of 4

The requirements for a satisfactory partition are looser if the graph only has an upper bound of 4 degrees. In this instance, the algorithm evaluates potential instances of two disjoint cycles. This is done by considering potential disjoint edges which can be added between nodes of degree 1 or 2. If adding up to two disjoint edges results in two disjoint cycles in the graph, the graph is still partitionable as if the two disjoint cycles existed, due to the limited degree of nodes in each potential disjoint edge added.

The algorithm first checks whether two disjoint cycles already exist within the graph and forms a partition based on the two cycles. If no cycles exist at all, there is no satisfactory partition. However, if one cycle is identified, the algorithm checks whether a disjoint cycle can be created by adding disjoint edges between nodes of degree 1-2. It does this by collecting all of the nodes of degree 1 or 2 which can be connected by potential new edges. While maintaining the set of candidate nodes with degrees of 1 or 2 in the original graph, it removes the cycle already identified from the graph. Then, it attempts to add up to two disjoint edges between the candidate nodes to form a cycle in the remaining graph. If a new potential cycle is identified, the satisfactory partition consists of the nodes in each (potential) cycle.

6 Non-star trees

Bazgan et al. trivially stated that for a tree that is not a star, there must exist a satisfactory partition. However, they do not indicate what the satisfactory

partition is nor how to find such a partition. We fill this gap and describe a solution for partitioning for non-star trees.

Algorithm: Consider the tree as a rooted tree, where every node has at most 1 parent. We can consider a partition as V_1 contains a leaf node, its parent, and the children of its parent. V_2 is the rest of the nodes in the graph.

Claim: G is a tree and not a star. Let V_1 be a set in G that only contains all the leaves that share a common neighbor (i.e., the leaves' parent), and the neighbor itself . Let V_2 be the rest of of the vertices. (V_1, V_2) is a satisfactory partition.

Proof. We want to prove that

$$\forall v_1 \in V_1, d_{V_1}(v_1) \geq d_{V_2}(v_1)$$

and

$$\forall v_2 \in V_1, d_{V_1}(v_2) \leq d_{V_2}(v_2)$$

First, by the definition of a tree, there exists at least 1 leaf node in the graph (a node with only one neighbor).

Let V_L be all the leaf nodes in V_1.

Since each leaf in V_L has only one neighbor, $\forall v_l \in V_L$, $d_{V_1}(v_l) = 1$ and $d_{V_2}(v_l) = 0$.

Let the parent node of V_1 be denoted as v_n. As all non-root nodes in a tree have exactly one parent and G is a non-star tree, v_n must have a parent as well and a grandparent. Further, v_n has exactly one parent which lies outside of V_1. This implies that v_n has exactly 1 neighbor in V_2 and at least 1 neighbor in V_1, satisfying $\forall v_1 \in V_1, d_{V_1}(v_1) \geq d_{V_2}(v_1)$.

The parent of v_n must have a parent, and its parent must be in V_2. Also, v_n is its only neighbor in V_1. Thus, the parent of v_n also satisfies the condition for satisfactory partition.

For the rest of the nodes in V_2, since they have no neighbors in V_1, we can guarantee that they satisfy the satisfactory parition condition.

Hence, the claim is true. □

7 Conclusions and future research

We provided an implementation of several sub-cases of the satisfactory partition problem which are solvable as proved by Bazgan et al. There are several possible extensions of this work:

1. Implementing Bazgan et al.'s specific cases for 3 and 4-regular graphs with degree less than 10

2. Change the requirements of the satisfactory partition problem. Instead of a "majority threshold" that requires that the node has at least as many neighbors in its subset as outside, the problem has been reformulated for varying thresholds. It has been found that instances where the threshold is

a function of the node's number of neighbors is solvable, whereas constant size thresholds are NP-complete (Ciccarelli et al., 2023).

References

[1] Bazgan, C., Tuza, Z., Vanderpooten, D. (2006). The satisfactory partition problem. *Discrete Applied Mathematics, 154*(8), 1236-1245. https://doi.org/10.1016/j.dam.2005.10.014

[2] Ciccarelli, F., Di Ianni, M., Palumbo, G. (2023). A note on the satisfactory partition problem: Constant size requirement, *Information Processing Letters, 179*, 321-324. https://doi.org/10.1016/j.ipl.2022.106292