Optimization of University Course Assignment System Using Graph Theory

Vijay Dharmaji, Sarang S, Krupali Jadhav

November 2023

Optimization of University Course Assignment System Using Graph Theory

Abstract

The report explores the optimization of University Course Assignment Systems through Graph Theory principles, focusing on faculty categorization, varied course loads, and individual course preferences. Employing bipartite graphs, matching, and assignment algorithms, it seeks to maximize faculty satisfaction while adhering to category-based constraints and individual course preferences. It addresses the minimum-cost path algorithm and presents a method for enumerating maximal matchings in bipartite graphs. Overall, the report aims to get close to an ideal solution for the course assignment by aligning faculty preferences with departmental constraints, ensuring adaptability, and providing a scalable solution for varying scenarios.

Introduction

The focus of this project is the optimization of the University Course Assignment System within a department, employing principles of Graph Optimization. The faculty body is segmented into three distinct categories: "x1," "x2," and "x3," each assigned varying course loads - 0.5, 1, and 1.5 courses per semester, respectively. The existing system allows faculty members the flexibility to enroll in multiple courses in a semester, and reciprocally, a single course may be assigned to multiple faculty members. Shared courses contribute 0.5 to each professor's load. Furthermore, faculty members maintain individual preference lists of courses, ranked based on personal preferences, with the most favored courses occupying the top positions. Importantly, there exists no prioritization among faculty members within the same category. The central research objective is the development of an assignment scheme that maximizes the number of courses assigned to faculty, harmonizing with both their preferences and category-based constraints ("x1," "x2," "x3"). The challenge lies in ensuring that a course is assigned to a faculty member only if it is present in their preference list. Distinguished by its flexibility in faculty course enrollments, this problem deviates from conventional Assignment problems. Potential adaptations encompass

adjusting the maximum number of courses ("y") for each professor category, allowing for deviations instead of strict adherence. Additionally, there's room for innovation by extending the number of professor categories beyond the current three, contributing to a more versatile and generalized solution. This research aims to address these intricacies, culminating in an advanced University Course Assignment System that maximizes faculty satisfaction, aligns with departmental constraints, and exhibits adaptability to a range of scenarios.

Objectives

1. Optimization Scheme Development: Formulate an optimization algorithm that maximizes the number of assigned courses to faculty members. Ensure that the assignment scheme aligns with faculty preferences and category-based constraints.

2. Flexibility Enhancement: Investigate modifications to allow flexibility in the number of courses faculty members can take, going beyond traditional assignment problem constraints.

3. Preference Integration: Develop a mechanism to integrate faculty preference lists into the assignment algorithm, prioritizing highly preferred courses.

4. Scalability and Generalization: Assess the scalability of the optimization algorithm to accommodate varying department sizes. Explore possibilities for generalizing the solution to handle an extended range of professor categories.

5. User-Friendly Interface: Design an intuitive and user-friendly interface for administrators to input data and interpret optimization results.

6. Performance Evaluation: Conduct rigorous testing and evaluation of the developed optimization scheme under diverse scenarios.

7. Documentation and Reporting: Provide comprehensive documentation outlining the algorithm, its implementation, and usage guidelines. Generate a detailed report summarizing the project, methodologies, findings, and recommendations.

Bipartite Graph and Matching

Bipartite Graph

Bipartite graphs are a type of graph where the vertices can be split into two disjoint sets such that every edge connects a vertex from one set to a vertex in the other set—there are no edges within a set. Mathematically, a graph G = (V, E) is bipartite if its vertex set V can be partitioned into two non-empty sets

V1 and V2, such that every edge in E connects a vertex in V1 to a vertex in V2.

Matching and Perfect Matchings

A matching in a graph refers to a set of edges without common vertices. In other words, it's a subset of edges in which no two edges share a common endpoint. A perfect matching in a graph is a matching where every vertex in the graph is incident to exactly one edge in the matching. For a graph with n vertices, a perfect matching contains n/2 edges assuming n is even. In the case of an odd number of vertices, it's not possible to form a perfect matching.

Converting the Problem to a Graph Theory Question

Algorithm Application Prerequisites:

The algorithms discussed necessitate the transformation of the problem into a graph-theoretic framework that complies with their specific constraints. Specifically, these algorithms operate on m-source, n-destination bipartite weighted graphs.

Bipartite Graph Conversion:

Conversion into a bipartite graph involves designating professors as the source and courses as the index. Each professor's slot is considered as a fractional course load:

- Professors in category "x1" are represented as 1 professor.
- Professors in category "x2" with a course load of 1 are represented as 2 professors ($\frac{1}{0.5} = 2$).
- Professors in category "x3" with a course load of 1.5 are represented as 3 professors ($\frac{1.5}{0.5} = 3$).

Courses are similarly divided into 2 units of a 0.5 course load each.

Maintaining Load Equivalence:

Equivalence in course loads between professors and courses is ensured by verifying that the number of split professor course loads is lesser than the split course loads.

Consideration of Professor Preferences:

Edges within the graph are exclusively allowed between professors and their preferred courses. The weight assigned to each edge is determined by the rank of the course's preference for that specific professor.
This outlines the process of transforming the problem into a suitable bipartite graph, considering fractional professor course loads, maintaining load equilibrium, and incorporating professor-course preferences into the graph structure.

Assignment Problem

An instance of the assignment problem is specified by a complete undirected bipartite graph with an assignment with an assignment of non negative weights to each edge. The graph, represented as G = (V, E), is defined with its vertex set V formed by the combination of two disjoint sets: X (the sources) and Y (the destinations), where the cardinality of X is smaller than that of Y. The set of edges E comprises all pairs x, y where x belongs to X and y belongs to Y. Each edge e=x, y has a cost denoted as c(e) or c(x, y). A matching in G is a subset M of E in which each vertex is connected to at most one edge in M. The total cost of this matching, indicated as c(M), sums up the costs of its individual edges. A matching M is considered complete if every source vertex is connected to at least one edge in M. The assignment problem seeks to find a complete matching with the lowest total cost.

Mathematically: (as a constraint matching problem)

Constraint 1: Matching

$$\sum_{j} x_i y_j \leq 1$$

$$\sum_{i} x_i y_j \leq 1$$

This ensures that all vertices have only 1 edge

Constraint 2: Minimum Cost:

$$C(M) = \sum_{e \in M} c(e)$$

$$Suchthat, C(M) = \min(C(M_i)) \quad \forall M_i \subseteq E$$

Augmenting Paths

When considering a matching M, a vertex v is termed "free" if it isn't connected to any edge in M. Within a graph G, a path is referred to as "alternating" if

the edges it comprises alternate between being in M and not in M. When such a path exists between free vertices and doesn't repeat vertices, it's labeled an "augmenting path." If P represents an augmenting path, then one of its endpoints is a source vertex, while the other is a destination vertex.

If M is a matching and P is an augmenting path, then their symmetric difference is also an augmenting path and $|M\Delta P| = |M| + 1$. The cost of the augmenting path P is $c(M\Delta P) - c(M)$.

Lemma 1. If M is of minimum cost among matchings of cardinality k and P is of minimum cost among augmenting paths relative to M, then $M\Delta P$ is of minimum cost among matchings of cardinality $k + 1$.
Lemma 1 is the basis of the following algorithm to solve the assignment problem.

Assignment Algorithm - Version 1

```
begin
    M <- ∅;
    while |M| < |X| do
    begin
        let P be a minimum-cost augmenting path relative to M;
        M <- M  P
    end;
end
```

The problem now reduces to finding the augmenting path P relative to M.

The augmenting paths relative to a matching M can be determined using an associated directed graph G = (V,E) called an augmentation graph. The set of directed edges E consists of forward edges and backward edges, specified as follows:

The problem now reduces to finding the augmenting path P relative to M.

The augmenting paths relative to a matching M can be determined using an associated directed graph $G' = (V, E')$ called an augmentation graph. The set of directed edges E' consists of forward edges and backward edges, specified as follows:

- if $\{x, y\} \in E$, $x \in X$ and $y \in Y$ then:
- if $\{x, y\} \notin M$ then $\langle x, y\rangle$ is a forward edge;
- if $\{x, y\} \in M$ then $\langle y, x\rangle$ is a backward edge.

A minimum-cost augmenting path P is found as follows: To each forward edge $\langle x, y\rangle$ assign cost $c(x, y)$; to each backward edge $\langle y, x\rangle$ assign cost $-c(x, y)$; let P be a minimum-cost directed path from a free source to a free destination; then

$$P = \{\{x, y\} | \langle x, y\rangle \in \hat{P} or \langle y, x\rangle \in \hat{P}\}.$$

Assignment Algorithm - Version 2

The enhancement relies on the rapid computation of a minimum-cost path in a directed graph from one specific set of vertices to another, given that all edge costs are positive. The improvement strategy involves altering the costs of edges within the augmentation network so that:

- The minimum-cost directed path(s) from a free vertex in set X to a free vertex in set Y remains unchanged.
- Ensuring all edge costs are non negative

This modification is achieved by assigning each vertex v a "potential" denoted as $\alpha(v)$, which has an additive effect on the costs of all edges incident with vertex v.

Assignment Algorithm - Version 2

```
begin
    M <- ∅
    for v in V do (v) <- 0
    while |M| < |X| do
        begin
            form the augmentation network G;
            for each {x,y} in E-M assign directed edge <x,y> the cost
                c(x,y) = c(x,y)+(x)-(y);
            for each {x,y} in M assign directed edge <y,x> the cost 0;
            for all v in V do
                begin
                    let (v) be the minimum cost of a directed path in G which begins
                        at a free source and ends either at v or at a free destination;
                    (v) <- (v)+(v)
                end
            let P be a minimum-cost directed path in G from a free source to a
                free destination;
            let P be the set of edges in G corresponding to edges in P;
            M <- M  P
        end
end
```

A Minimum Cost Path Using Priority Queues:

- The calculation of path P and quantities $\gamma(v)$ involves a variation of the problem seeking the minimum cost path from a single source.
- The input data for this problem includes:

- A directed graph $G = (V, E)$
- A set $S \subseteq V$ comprising start vertices
- A set $T \subseteq V$ consisting of target vertices where $S \cap T = \emptyset$ and there exists a path from S to T
- Non-negative costs $d(u, v)$ for each edge (u, v)

- For every $v \in V$:
  - $OUT(v)$ refers to the set of edges in G directed out of vertex v
  - $\gamma(v)$ represents the minimum cost of a path from a vertex in S to a vertex in $T \cup \{v\}$
- The subsequent algorithm computes $\gamma(v)$ along with the associated minimum cost.

Minimum-Cost Path Algorithm - Verision 1

```
begin
    PATHSET <- ∅;
    EDGE <- ∅;
    R <- S;
    for u in R do (u) <- 0; EDGE <- EDGE  OUT(u);
    while R  T = ∅ do
        begin
            choose <x,y>  EDGE so that (x) + d(x,y) =
                min [(u) + d(u,v)]; <u,v>EDGE
            EDGE <- EDGE - {<x,y>};
            if y  R then
                begin
                    PATHSET <- PATHSET  {<x,y>};
                    R <- R  {y};
                    (y) <- (x) + d(x,y);
                    EDGE <- EDGE  OUT(y)
                end
        end;
    for v in R do (v) <- (y);
end
```

Algorithms for enumerating all perfect, maximum and maximal matching in bipartite graphs

Consider a graph G that consists of two distinct sets of vertices, $V1$ and $V2$, without any directional edges connecting them. The set of edges, denoted as E, exists solely between vertices in $V1$ and $V2$.

Matching: A matching M in G is a set of edges where no two edges share the same endpoints. These edges within a matching are referred to as matching edges.

Coverage: A vertex is considered covered by a matching if it is incident to at least one matching edge; otherwise, it remains uncovered.

The objective of the paper is to enumerate all possible maximal matchings of a given bipartite graph as defined previously. The algorithm takes $O(\log(n))$ per perfect matching.

Algorithm:

In a bipartite graph $G = (V1 \cup V2, E)$, the collection of all perfect matchings is denoted as $M(G)$. When you remove a set of edges E' from the graph G, the resulting graph is denoted as G'. The process of enumerating perfect matchings in G involves splitting the problem into two subproblems: $G \setminus E1$ and $G \setminus E2$. The presence of a perfect matching in $G \setminus Ei$ is determined by ensuring the intersection between M (perfect matchings) and Ei is empty. To discover distinct perfect matchings M and M', two edge sets, $E1$ and $E2$, are established. $E1$ includes an edge $e \in M \setminus M'$, while $E2$ includes an edge $e \in M' \setminus M$. Locating a perfect matching M takes $O(\frac{|V|}{2|E|})$ time.

The process of finding an alternative perfect matching M' involves utilizing alternating cycles. These cycles consist of edges alternating between those in M and those not in M. $DG(G, M)$ is a directed graph derived from G and the matching M. Vertices in this graph correspond to V. Arcs in $DG(G, M)$ orient edges from $V1$ to $V2$ if they belong to M, and reverse edges from $E \setminus M$ to M. Directed cycles within $DG(G, M)$ reveal alternating cycles within G. The alternation is evident in the appearance of arcs from M and non-M arcs along the cycle in G.

ALGORITHM Basic_Algorithm(G)

1. If (G includes no perfect matching) then stop.
2. $M :=$ (a perfect matching of G)
3. Call **Basic_Algorithm_Iter** (G, M)

ALGORITHM Basic_Algorithm_Iter(G, M)

1. Construct $DG(G, M)$.
2. Find an alternating cycle C by finding a directed cycle of $DG(G, M)$.
3. If (no directed cycle exists) then output M; stop
4. $M' :=$ the perfect matching obtained from M and C
5. $e :=$ an edge in $M \setminus M'$; $v :=$ an endpoint of e

6. $E_1 := \{e\}$; $E_2 := \{alltheedgesincidenttovexceptfore\}$
7. Call **Basic_Algorithm_Iter**$(G \backslash E_2, M)$
8. Call **Basic_Algorithm_Iter**$(G \backslash E_1, M')$

Conclusion

This report delves into the enhancement and optimization of the University Course Assignment System utilizing Graph Theory principles. The project aims to devise an assignment scheme that maximizes faculty satisfaction, adhering to faculty preferences and category-based constraints ("x1," "x2," "x3"). The existing system permits faculty to enrol in multiple courses while allowing a single course to be assigned to multiple faculty members, contributing 0.5 to each professor's load when shared. Each faculty maintains individual preference lists of courses, ranked based on personal preferences. The primary challenge lies in ensuring that courses are assigned to faculty only if they are on their preference list, requiring an advanced optimization algorithm.

The objectives encompass developing an optimization algorithm aligned with faculty preferences and constraints, integrating flexibility beyond traditional assignment problem constraints, and ensuring scalability for varying department sizes and professor categories. The report details a comprehensive methodology for developing a user-friendly interface, performance evaluation, and extensive documentation outlining the algorithm's implementation.

The report introduces the fundamental concepts of Bipartite Graphs, Matchings, Perfect Matchings, and the Assignment Problem. It presents two versions of the Assignment Algorithm, detailing their step-by-step execution and augmentation. The second version involves modifying edge costs within an augmentation network by introducing vertex potentials and optimizing the computation of a minimum-cost path.

Furthermore, it introduces an algorithm for finding a minimum-cost path in a directed graph using priority queues, crucial for determining augmenting paths in the assignment problem. The document culminates with an exploration of enumerating all possible maximal matchings in a bipartite graph, delineating an algorithm for identifying distinct perfect matchings and their alterations.

This report serves as a guide for the optimization of University Course Assignment Systems using Graph Theory.

References

- Algorithms for enumerating all perfect, maximum and maximal matchings in bipartite graphs - Takeaki Uno
- An Algorithm to Solve the mxn Assignment Problem - R.M Karp