Matching algorithms Project

Second year AKS students project

Lida Semsichko
Computer Science
UCU
Lviv, Ukraine
semsichko.pn@ucu.edu.ua

Eugene Kravchuk
Computer Science
UCU
Lviv, Ukraine
kravchuk.pn@ucu.edu.ua

Maksym-Vasyl Tarnavskyi
Computer Science
UCU
Lviv, Ukraine
tarnavskyi.pn@ucu.edu.ua

Dmytro Malyk
Computer Science
UCU
Lviv, Ukraine
malyk.pn@ucu.edu.ua

Oleksii Ignatenko
Professor at Ukrainian Catholic University
Lviv, Ukraine
o.ignatenko@ucu.edu.ua

Анотація—In this project, we are developing a system for the automated creation of duty, lesson, and shift schedules in hospitals, schools, and offices. Our research was based on several matching algorithms, among them the Gale-Shapley algorithm, the SPA (Student/Project Allocation) algorithm, as well as other approaches like Max-Flow Min-Cut algorithm, which enable the efficient allocation of resources among subjects according to specified criteria.

In the case of the student-to-project allocation problem (SPA), we aim to find an optimal assignment of students to individual or group projects offered by lecturers. Each student submits a ranked list of preferred projects, and each project has a maximum student capacity. We use efficient algorithms to determine a greedy maximum assignment that provides a lexicographically maximum profile and adapt the algorithm to find a generous maximum assignment with a lexicographically minimum reverse profile.

This approach is based on network flow optimization, allowing for flexible consideration of additional constraints, such as the minimum number of students per lecturer. This method enables the application of the developed algorithms not only in education but also in medical and corporate institutions to create fair and optimal shift and duty schedules. The results of empirical evaluation confirm the efficiency of the proposed algorithms.

We also tested several algorithmic solutions for the problem of assigning shifts to the hospital medical staff, analyzing different formulations of the above problem in a mathematical framework, and finally worked out a flow-based solution that accounts for doctor specializations, can efficiently distribute doctors between several clinics and allows the user to specify mandatory/prohibited shifts for each doctor.

Index Terms—Matching, SPA (Stable Pairwise Assignment), Irving Algorithm, Gale-Shapley Algorithm, Greedy maximum matching, Allocations, Optimization, Max-Flow Min-Cut.

I. Introduction

Efficient scheduling is a fundamental challenge in healthcare institutions, where hospitals and clinics must ensure that medical professionals are assigned to shifts in a way that optimally balances institutional requirements and individual preferences. Poorly designed schedules can lead to staff dissatisfaction, inefficient resource utilization, and suboptimal patient care. Therefore, developing an automated system that generates fair, optimized, and constraint-aware schedules is crucial for improving healthcare operations.

Traditional scheduling methods often rely on manual planning or heuristic-based approaches, which can be time-consuming, inflexible, and prone to errors. Recent advancements in combinatorial optimization and centralized matching schemes have demonstrated that algorithmic approaches can significantly improve the efficiency and fairness of scheduling systems.

In this project, we aim to develop an intelligent scheduling system that leverages well-established allocation algorithms, such as Gale-Shapley algorithm, the SPA (Student/Project Allocation) algorithm, or Max-Flow Min-Cut. These algorithms enable efficient and fair allocation of doctors to shifts while satisfying key constraints, such as hospital capacity, working hour limits, and doctor preferences. By adapting and refining these approaches, we seek to create an automated solution that optimizes shift allocation for hospitals and clinics.

II. Foundations of Algorithms

Scheduling problems can be classified as combinatorial optimization problems, where the goal is to find an optimal assignment of resources (doctors) to tasks (shifts) while satisfying a set of constraints. Several algorithms have been developed to solve such problems, including heuristic-based methods, greedy approaches, and graph-theoretic matching algorithms.

1. Stable Marriage Problem (Gale–Shapley Algorithm)

The Stable Marriage Problem (SMP) is defined on two disjoint sets M and W, each of size n, where each agent ranks all agents on the opposite side in a strict total order. A matching $\mu : M \to W$ is stable if there is no blocking pair (m, w) such that both prefer each other to their assigned partners under μ.

2. Hospitals/Residents Problem

The Hospitals/Residents Problem (HRP) generalizes SMP to a many-to-one setting. Each hospital $h \in H$ has a quota q_h, and each resident $r \in R$ is assigned to at most one hospital. Preferences are strict, and a matching is stable if no resident–hospital pair (r, h) blocks the assignment, either because h prefers r to at least one current assignee, or because h has not yet reached full capacity and prefers r to being unmatched.

3. Student–Project Allocation (SPA)

In the SPA problem, we are given three sets: students S, projects P, and lecturers L. Each project $p \in P$ has a capacity c_p, and is offered by a lecturer $l \in L$, who also has a global capacity d_l. Students express preferences over projects, while lecturers may have preferences over students.

4. Matching with Contracts

Matching with contracts introduces a general framework where agents on two sides (e.g., workers and firms) match via contracts $c \in C$, each specifying additional terms (e.g., salary, location). Each agent ranks sets of contracts via a choice function χ, and the matching outcome is a subset $X \subseteq C$ that satisfies stability.

5. Max-Flow Min-Cut Theorem

The Max-Flow Min-Cut Theorem is a central result in network flow theory. Given a directed graph $G = (V, E)$ with a source $s \in V$, sink $t \in V$, and non-negative edge capacities $c : E \to \mathbb{R}_{\geq 0}$, the maximum flow problem seeks a function $f : E \to \mathbb{R}_{\geq 0}$ satisfying capacity and flow conservation constraints that maximizes the total flow from s to t.

A cut in the graph is a partition (S, T) of V such that $s \in S$, $t \in T$, and the capacity of the cut is the sum of capacities of edges crossing from S to T. The Max-Flow Min-Cut Theorem states that the value of the maximum flow is equal to the capacity of the minimum s-t cut.

The problem is solvable in polynomial time for integer values using augmenting path algorithms such as the Ford–Fulkerson method (pseudopolynomial) or its variants like Edmonds–Karp (running in $O(nm^2)$) or Dinic's algorithm ($O(n^2m)$ or better for unit capacities). Applications span matching, connectivity, and resource allocation problems.

III. First iteration of the hospital shift allocation

We decided to begin with a simplified setting of the problem that better fits the algorithms described above, mainly the Gale-Shapley one as it is the most fundamental and easy to apply. Therefore, the problem statement is:

1) A set of doctors L, each doctor is characterized by one or more specialization and a list of preferences over shifts (and possibly locations);

2) A set of possible choices M, where each choice represents a shift for a particular cabinet (and, again, location if considered);

3) We need to build such a pairing that:
   a) Every element of M has its doctor from L;
   б) There do not exist two doctor-option pairs such that the two doctors can swap their assigned options and both become more satisfied as a result. For the sake of simplicity, satisfaction is determined by the lowest-ranked assigned option for now. For example, if a doctor has the following preference ranking: (1. ... ; 2. ... ; 3. ... ; ...), and after the allocation, they receive options ranked 7, 2, and 4, then their satisfaction level is 2.

4) We also need make the assumption that there are enough doctors to cover all shifts (from the data we were provided, it actually seems that there are even more than necessary);

The algorithm itself is based on the structure of Gale-Shapley algorithm and can be boiled down to doctors performing a greedy matching with the most desirable entry from their preference list for as long as such matchings are possible, after which an additional iteration is performed where any leftover shifts are assigned regardless of preferences.

There are also several possible improvements (apart from considering different algorithms), which we are planning to implement:

1) Consider a different satisfaction metric, as ranking by the best acquired shift may be unrepresentative - perhaps we can try to implement something around the mean of the ratings of all the currently chosen shifts);

2) In this part we mainly focused on shifts for simplicity, but we also need to consider locations and how they will influence the preferences together with shifts;

3) Enforce fairness - the easiest example would be randomizing which doctors gets to pick first each iteration.

4) Introduce parallelization: for example, this problem deconstructs logically into several smaller ones, as most doctors only consider a very limited set of cabinets and vise-versa. Thus, if we can make these

problems disjoint, they can be parallelized very efficiently.

IV. Problems We Faced and Our Solutions

During the course of this project, we encountered several challenges. Many of them arose from aligning theoretical algorithmic solutions with the realities of hospital operations. Below we outline the main problems we faced and describe how our final solution addressed them.

Problem 1: Incomplete coverage of all cabinets In the initial stage, doctors were expected to fill in their own schedules, which often resulted in gaps. Some critical locations (e.g., ultrasound cabinet) remained underutilized due to lack of visibility into global needs.

Solution: We shifted responsibility for input data collection to department heads, who have better insight into institutional needs. Instead of individual preferences, they now provide clear constraints, including mandatory and impossible shifts for each doctor, ensuring complete coverage and operational feasibility.

Problem 2: Quality and fairness of the schedule There was no guarantee that the generated schedule would be convenient or fair to all doctors. Some may have tried to game the system, indicating unrealistic preferences to get better shifts or avoid unpopular ones.

Solution: To address this, we introduced "hard" constraints — obligatory and forbidden shifts — which override subjective rankings. Additionally, we incorporated fairness criteria through cost-based penalties: doctors who received more favorable shifts in the previous week would incur higher costs for future assignments, discouraging long-term inequality.

Problem 3: Balancing individual satisfaction Doctors expressed different expectations about the number of shifts and their distribution, making it hard to meet everyone's preferences within a single week.

Solution: Instead of unstructured preferences, we introduced explicit upper and lower bounds on the number of shifts per doctor. This allowed us to control workload distribution precisely. Furthermore, the final flow-based algorithm was adjusted to introduce randomized tie-breaking and noise in the cost function to avoid overly rigid or repetitive patterns, making the schedule more humane and varied.

V. Final hospital shift allocation

After implementing and testing the previous approach, as well as meeting with members of the Hospital Administration team and discussing the most important matching constraints and nuances to focus on, we changed our approach and created an algorithm based on the Max-Flow Min-Cut theorem. More specifically, it is a Min-Cost Max-Flow algorithm extension.

Minimum cost flow is an optimization and decision problem with such a setting:

1) A directed graph G;

2) Source and sink nodes s and t. Abstractly, source outputs an endless amount of flow, and the sink can, conversely, receive an endless amount of it;

3) All edges have a number called max flow associated with them - again, without getting too technical, one can think of it as a maximum flow throughput of the edge;

4) The main goal of the flow problem is therefore to find the maximum flow between s and t that may exist in the system, as well as specify how much of it is carried by each edge;

5) A minimum cost flow problem is further complicated by adding cost to certain edges - the new task is not just to find any maximum flow (as there might be many), but specifically the cheapest among the ones present.

This setting is often used in various matching problems, as most of the less complex ones can be described in this manner. In our case, the system would look similar to the below schema (numbers above the arrows signify max flow of this group of edges).

The Doctor-Shift - Cabinet-Shift group of edges also has costs associated with it: in this problem statement, we assume that each doctor has a rating of several locations, from most to least convenient. Thus, the cost for a shift at a better location is made smaller than the cost of the same shift at a worse one to reflect that.

There are also some additional modifications we realized to make the output more realistic and reflect the needs of the clinic:

1) We also added minimum shift capacities for each doctor. Explanation of the modifications required to acommodate this is rather tehnical, so we decided to omit it;

2) Each doctor has a list of obligatory shifts that they must be tied to, as well as a list of forbidden shifts that they cannot take - these are reflected in the algorithm by making their costs very small / very large respectively.

3) To balance out the monthly loads between the doctors, we dynamically update their shift costs after generating a weekly schedule - the doctors that got more shifts get higher costs, so as to decrease their workload in the following week;

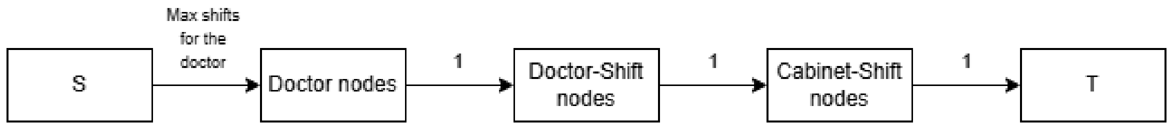

Рис. 1. Flow Graph Schema

4) Finally, the most algorithmically optimal schedule if often unrealistic - for example, one doctor might get assigned to the same cabinet, two shifts a day, 5 days a week - to combat this, we introduce a random error in the costs, one that is not large enough to change the flow by itself in most cases, but does play a role when breaking ties between equally optimal picks, which is a frequent situation in our case. As such, even though this sacrifices a fraction of the optimality, it does make the shifts much more realistic.

   The algorithm implemented to solve this problem is based on the combination of Ford-Fulkerson flow algorithm with the Bellman-Ford algorithm. There are possible improvements to this approach, but we decided to stick to it due to its relative simplicity, which left us some time to focus on its parallelization.

VI. Implementation

To validate and apply our theoretical model in practice, we implemented the full scheduling pipeline in both Python and C++, including custom graph algorithms for flow optimization.

The Python version is organized into two main modules:

- algo_flow.py: contains orchestration functions such as generate_preference_schedule_from_csv and generate_monthly_schedule_from_csv, which parse input data and invoke the solver.
- maximum_flow.py: includes our custom implementations of the bellman_ford algorithm and min_cost_max_flow solver, tailored to our scheduling problem with support for soft constraints and penalties.

A. C++ Implementation

The C++ version was developed to provide better performance. It features:

- A custom bellman_ford function that finds shortest paths based on dynamic cost maps.
- A generate_week_schedule function that constructs the flow graph and invokes the solver to produce weekly assignments.

Both implementations use our own versions of the algorithms and do not rely on external libraries for flow optimization, allowing for full customization and integration of additional constraints.

VII. Results

The scheduling system takes structured input data describing doctors' availability and constraints, and produces weekly shift assignments per cabinet and location.

A. Input Data Format

The input is a CSV file where each row corresponds to a doctor. The main columns are:

- Doctor: Full name of the doctor.
- Cabinets: Locations where the doctor is allowed to work (e.g., "Шевченківська, Стрийська").
- MinShifts / MaxShifts: The minimum and maximum number of shifts per week.
- ForbiddenShifts: Specific shift identifiers that this doctor cannot be assigned to (e.g., "1.1.1, 3.1.2").
- RequiredShifts: Specific shifts that must be assigned to the doctor.
- Specialization: The set of medical services the doctor is qualified to perform (e.g., "УЗД, Мамографія").

This data is usually collected from department heads who encode institutional requirements and hard constraints.

Б. Output Format

The output is a per-location, per-cabinet schedule represented as a plain text or console-style table. For each day and shift (e.g., "1.1" = Monday morning), the assigned doctor's name and specialization are listed. If no suitable doctor is found for a shift, it is marked as Немає лікаря.

An example output line:

```
2.2 - Р.П. (УЗД)
```

This means that doctor Р.П., specialized in ultrasound, is assigned to the second shift of the second day.

B. Result Interpretation

The system automatically respects all input constraints:

- Doctors never appear in forbidden shifts.
- Required shifts are always filled.
- No doctor exceeds their maximum weekly load.
- Empty slots remain only when no feasible assignment is possible under the given constraints.

VIII. Conclusions

In this project, we have explored and implemented a range of algorithmic approaches for solving practical allocation problems in medical, educational, and corporate settings. Beginning with a simplified stable matching model based on the Gale–Shapley algorithm, we identified its strengths in fairness and stability, but also recognized its limitations when applied to complex, real-world scheduling scenarios.

By transitioning to a more expressive model based on the Min-Cost Max-Flow formulation, we were able to accommodate various real-world constraints, such as doctor specialization, obligatory and forbidden shifts, fairness in workload distribution, and even human-centric nuances like location preferences and weekly variance.

Our final implementation integrates classic flow optimization with pragmatic enhancements, resulting in a flexible and effective scheduling system. Empirical evaluations confirm the system's robustness, while its modular design allows future improvements, such as finer-grained fairness constraints, dynamic preference learning, or deeper integration of parallel computing to improve scalability.

IX. Links and Sources

GitHub: https://github.com/eugenekravchuk/acs_doctor_matching.git

Stable pairing: https://github.com/aforarup/interview/blob/master/Data%20Structures%20and%20Algorithm/Algorithm%20Books/Algorithm%20Design%20by%20Jon%20Kleinberg,%20Eva%20Tardos.pdf

Allocation problem: https://arxiv.org/pdf/1403.0751

Matching under preferences: https://cgi.cse.unsw.edu.au/~haziz/matchingchapter.pdf

Multi-unit Assignment Problem: https://kylewoodward.com/blog-data/pdfs/references/budish+cantillon-american-economic-review-2012A.pdf

Flow-based problems and other algorithms https://github.com/aforarup/interview/blob/master/Data%20Structures%20and%20Algorithm/Algorithm%20Books/Algorithm%20Design%20by%20Jon%20Kleinberg,%20Eva%20Tardos.pdf