Abstract

1 Introduction

On-call schedules for a fixed number of health-care providers are central to the efficient running of hospitals. Hospital departments provide services where patient needs, and thus the system's demands, often exceed the available supply. For example, it is important that a hospital department allocates its resources, such as the availability of a finite number of clinicians, optimally, to ensure the best possible service for its patients. Carefully allocated on-call schedules are meant to simultaneously ensure sufficient resources are provided to patients while not overworking clinicians to prevent costly mistakes [ref]. It is common practice for on-call schedules to be created manually. Yet manually-created schedules are prone to errors and potential for biases [ref]. First, when there is a large number of clinicians in a single department, or the constraints that need to be satisfied by the department are very complex, a manual method may not provide an optimal schedule. Second, such methods are likely to overlook certain constraints that must be maintained to have an operational department, such as XXXX. Third, manual scheduling is often time-consuming for the person developing the schedule. For these reasons, it is important to develop automated methods that can generate optimal schedules that satisfy the given constraints of the hospital department.

Automated methods to optimize schedules have been studied and applied in many industries, including transportation [??], manufacturing [??], [...]. Of special interest to a clinician on-call scheduling problem are the approaches to schedule nurses, who often work in shifts. In the nurse scheduling problem, the goal is to find an optimal assignment of nurses to shifts that satisfies all of the hard constraints, such as hospital regulations, and as many soft constraints as possible, which may include nurse preferences. A wide variety of approaches, including exact and heuristic approaches, have been used to solve the nurse scheduling problem: integer linear programming [??], network flows [??], genetic algorithms [??], simulated annealing [??], and artificial intelligence [??].

[...] Many of these approaches were designed to satisfy the requirements of a specific hospital department which causes a large number of variables and constraints to be incorporated into the problem formulation. While these department-specific approaches allow end-users to find precise schedules that satisfy the needs of the department and the preferences of the nurses and clinicians in that department, they are difficult to readily adapt to other departments in the same hospital or other hospitals. Moreover, the large number of variables and constraints also leads to computational complexity issues [ref], especially when using exact methods for finding the solution. In this paper, we tackle a version of the nurse scheduling problem arising from a case study of one clinical division, providing two different services simultaneously (general infectious

disease (ID) consults; and HIV consults service) at St. Michael's Hospital in Toronto, Canada. Our goal is to (1) present a simple formulation for the problem developed and tested at the hospital after switching from a manual approach to scheduling; and (2) analyze the performance of integer linear programming in solving difficult instances of the problem and compare the results with those of the manual approach; and (3) describe the adaptability of the formulation as a basic framework for solving similar problems in other departments.
We begin by describing the problem, then...

[...]

2 Problem

At St. Michael's Hospital, the (???) division offers general ID and HIV consultation throughout the whole year, during regular work weeks as well as weekends and holidays. The clinicians in the division typically receive a schedule in advance, outlining their on-call service dates for the full year. In the yearly schedule each clinician is assigned to blocks of regular work weeks, as well as weekends. Each block corresponds to two consecutive work weeks. Apart from long (holiday) weekends, a work week starts on Monday at 8 A.M. and ends on Friday at 5 P.M. Conversely, weekend service starts on Friday at 5 P.M., and goes on until the start of the next work week on Monday at 8 A.M. During the weekend, all clinicians in the department provide both ID and HIV consultation, while during the week some clinicians only provide one of the two services.

In order to provide quality service and ensure patients get the best care they can, it is important to prevent under- and over-working of clinicians. Several constraints are placed on the assignments in hopes of preventing such issues. Firstly, each clinician has limits on the number of blocks they can and must work during the year, depending on the type of consultation. For instance, a clinician might have to provide 3-5 blocks of general ID consultation as well as 2-3 blocks of HIV consultation throughout the year. These limits may change from year to year as the number of clinicians in the department changes. Moreover, the schedule does not assign a clinician to work for two blocks or two weekends in a row. It is especially important to prevent over-working during weekends, as the demand for the on-call service increases drastically. Hence, the schedule attempts to distribute both regular and holiday weekends equally among all clinicians.

Apart from maintaining a balanced work load among clinicians, the schedule also tries to accommodate their preferences. Clinicians provide their requests for time off ahead of schedule generation so that they can be integrated into the schedule. They are free to specify days, weeks or weekends off, with the understanding that any blocks overlapping with their request will be assigned to a different clinician, if possible. For example, if a clinician only requests Monday and Tuesday off, the schedule will generally avoid assigning the entire block to that clinician. Clinicians also prefer to have their weekend and block assignments close together, so the schedule tries to take this into account when distributing assignments.

[...]

3 Methods

4 Results

5 Simulations

6 Discussion