# Integrating nurse and surgery scheduling

Jeroen Beliën* and Erik Demeulemeester†
*Katholieke Universiteit Leuven, Department of Applied Economics
Naamsestraat 69, B-3000 Leuven, Belgium
Email: jeroen.belien@econ.kuleuven.be
†Katholieke Universiteit Leuven, Department of Applied Economics
Naamsestraat 69, B-3000 Leuven, Belgium
Email: erik.demeulemeester@econ.kuleuven.be

***Abstract*— One common problem at hospitals is the extreme variation in daily (even hourly) workload pressure for nurses. The operating room is considered to be the main engine and hence the main generator of variance in the hospital. It is our belief that integrating the operation room scheduling process with the nurse scheduling process is a simple, yet effective way to achieve considerable savings in staffing costs. The purpose of this paper is threefold. First of all, we present a concrete model that integrates both the nurse and the operating room scheduling process. Secondly, we show how the column generation technique approach, often employed for nurse scheduling problems, can easily cope with this model extension. Thirdly, by means of a large number of computational experiments we provide an idea of the cost saving opportunities and required solution times.**

***Keywords*— nurse scheduling, surgery scheduling, column generation, integer programming.**

## I. Introduction

DURING the last decades, cost pressures on hospitals have increased dramatically. This emphasis on cost containment has forced hospital executives to run their organizations in a more business-like manner. The constant challenge is to provide high-quality service at ever reduced costs. In order to achieve this purpose, inefficient use of resources should be identified and actions should be taken to eliminate these sources of waste. Operations research techniques are increasingly being used to assist in this complicated task.

As nursing services account for an important part of a hospital's annual operating budget, concentrating on this resource can lead to substantial savings. The situation is exacerbated by an acute shortage of nurses in all western countries, said to be 120,000 today and expected to grow to 808,000 by 2020 in the United States (US) alone [24]. Hence, it is of vital importance that nurses are used as much as possible at the right time and at the right place. This goal is hard to achieve because of two reasons. The first one is inherent in service organizations for which human resources outnumber all other types of resources. Unlike machines, staff schedules are restricted by collective agreement requirements. These form an important hindrance for the flexibility with which nurses are scheduled.

A second reason is the presence of variability. Variability is probably the main obstacle to efficient delivery of health care and reducing it is one of the major concerns in current health care management [19]. Compared with industrial environments, hospitals are much more stochastic by nature. One common problem at hospitals is the extreme variation in daily (even hourly) workload pressure for nurses. On days when the workload is too high, the quality of care decreases because it is too costly to staff for peak loads. On days when the workload is too low, there is waste. Fortunately, the situation is not as chaotic as it seems to be at first sight. As pointed out in [19], an important amount of the variability can effectively be managed and reduced by a thorough analysis of the existing system and by appropriate decision-taking. Special emphasis is put on the operating room since it is considered the main engine and hence the main generator of variance in the hospital. It is our believe that integrating the operation room schedule process into the nurse scheduling process is a simple yet effective way to achieve considerable savings in staffing costs.

This paper is organized as follows. In Section II a discussion of the background together with a brief literature review is given. In Section III a general overview of the model together with a branch-and-price solution approach is presented. Section IV provides more details on both pricing problems, while a general overview of the branch-and-price algorithm is given in Section V. Section VI discusses a specific branching scheme. In Section VII some computational issues are discussed and in Section VIII extensive computational results are given. Finally, Section IX draws conclusions and lists some topics for further research.

## II. Background and literature review

Nurse scheduling problems are frequently encountered in the operations research literature. Recently, a good bibliographic survey on medical staff rostering problems has appeared [13]. Several studies in the literature have utilized mathematical programming techniques to assist in finding efficient staff schedules (see e.g. [22], [28], [3], [8], [12], [4]). These problems typically involve some kind of set covering or set partitioning formulation. The main drawback, however, is that these models can have far more variables than can be reasonably attacked directly. Therefore, the linear program (LP) is often solved using column generation (see e.g. [18], [5] and [6], [21], [20]). To the best of our knowledge, all the proposed models consider the nurse scheduling problem as a separate problem, i.e. not related to any other activity in the hospital. In this paper we will describe a more general approach in which the demand constraints are dependent on the operation room schedule and hence become a part of the decision process.

The operations research literature is replete with examples of integer programming techniques being applied to operating room scheduling problems. This work can be categorized based on the stage of the scheduling process to which it applies. Developing operating room (OR) schedules can be seen as a three stage process. In a first stage the available OR time is divided over the different surgeons (or surgical groups). This first phase is also referred to as case mix planning, since it determines for which pathologies capacity will be preserved. Hughes and Soliman [17] propose a linear programming model to solve case mix planning problems. Dexter and Macario [14] argue that OR time should be allocated to maximize OR efficiency instead of ”fixed hours” blocks based on historical utilization data. Blake and Carter [10] propose a methodology that uses two linear goal programming models. One model sets case mix and volume for physicians, while holding service costs fixed; the other translates case mix decisions into a commensurate set of practical changes for physicians.

Once the OR time allocated to each surgical group has been chosen, the second stage involves the development of a master surgery schedule. The master surgery schedule is a cyclic timetable that defines the number and type of operating rooms available, the hours that rooms will be open, and the surgical groups or surgeons who are to be given priority for the operating room time. Compared to case mix planning (first stage) and elective case scheduling (third stage), the literature on master surgery scheduling is rather scant. Blake et al. [11] propose an integer programming model that minimizes the weighted average undersupply of OR hours (i.e. allocating to each surgical group a number of OR hours as close as possible to its target OR hours).

After the development of the master surgery schedule, elective cases can be scheduled. This third stage occurs on a daily base and involves detailed planning of each intervention. Each patient needs a particular surgical procedure, which defines the human (surgeon) and material (equipment) resources to use and the intervention duration. Guinet and Chaabane [15] define this problem as a general assignment problem and propose a primal-dual heuristic to solve it. Weiss [29] deals with the problem of determining the case orderings and presents both analytical and simulation results.

The methodology presented in this paper has some similarities with models for integrating the scheduling of project tasks and employees (Alfares et al. [1] and Alfares and Bailey [2]). Although several authors mention the interdependency between the surgery scheduling process and the development of nurse rosters, as far as we know, no models have been proposed to integrate both areas of decision-making. Litvak and Long [19] underline the negative impact of variability in hospital environments. They consider the operating room as the engine that drives the hospital. Consequently, the activities inside the operation room heavily determine the fluctuations in resource demands throughout the rest of the hospital. A poor operating room schedule could for instance be directly responsible for the occurrence of (contra-productive) peeks in the demand for certain types of resources. The authors distinguish between two types of variability: natural variability and artificial variability. Natural variability is inherent to the uncertain world of health care. This variability arises from uncertainty in patient show-ups, uncertainty in recovery time, uncertainty in the successfulness of therapies etc.... Artificial variability originates from poor scheduling policies. Beliën and Demeulemeester [9] have elaborated this idea. They propose a number of integer programming models for building robust surgery schedules for which the resulting expected bed shortage is minimized.

In this paper the master surgery schedule is being considered as the main generator of the workload of the nurses. In order to couple both scheduling environments, the objective in the surgery schedule process will be to construct a favorable workload distribution for the nurses.

## III. Model description

### A. General idea

Figure 1 contains a schematic overview of the general idea elaborated in this paper.

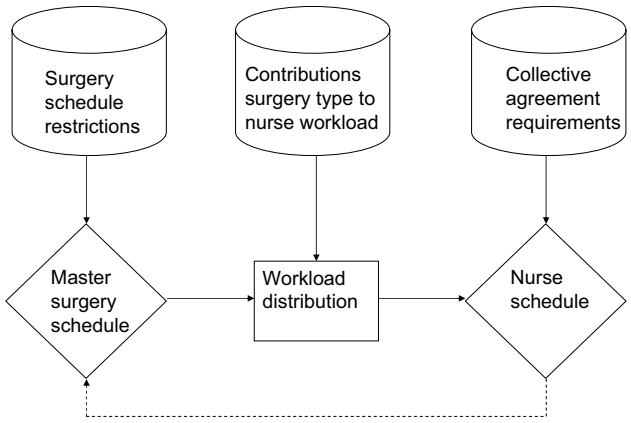


Fig. 1. Schematic overview of the general idea

First have a look at the nurse scheduling process at the right of this figure. The input for the nurse scheduling process consists of the restrictions implied on the individual nurse roster lines on the one hand and the workload distribution over time on the other hand. The workload distribution itself is determined by the master surgery schedule. In order to be able to deduce the workload from the surgery schedule one also has to know the workload contributions of each specific type of surgery. The dotted arrow at the bottom indicates the feedback that could be given from the nurse scheduling process to the surgery scheduling process in order to produce more favorable surgery schedules with respect to the resulting workloads. The freedom in modifying the surgery schedule is however limited, since the master surgery schedule itself is restricted by a set of specific surgery constraints (e.g. capacity and demand constraints). It must be clear, however, that integrating the surgery scheduling process with the nurse scheduling process provides more flexibility in building the nurse schedules, since one has an instrument to make the workload distribution fit for the nurse schedules.

In what follows we will describe a mathematical model for implementing this idea. Therefore, we start with stating the standard nurse scheduling problem and discuss the column generation solution procedure for solving it. Then, we extend this model with the extra decision of the nurse scheduling process and show how the column generation solution procedure can easily cope with this extension. Hereby, we focus on the minimization of the total required number of nurses. The reason for this objective is that it allows for a quantitative measure of the resulting benefits, i.e. the decrease in staffing cost. Obviously, this quantitative benefit can easily be turned into a qualitative benefit by employing the saved nurse(s) on moments when they are most needed.

### *B. The nurse scheduling problem*

The nurse scheduling problem (NSP) consists of generating a configuration of individual schedules over a given time horizon. The configuration of nurse schedules is generated so as to fulfill collective agreement requirements and the hospital staffing demand coverage while minimizing the salary cost. An individual's *roster line* can be viewed as a sequence of *days on* and *days off*, where each day on contains a single *shift* identified by a label such as 'day', 'evening' or 'night'. Each such label coincides with a start and a finish time of the corresponding shift. Furthermore, a day is subdivided into several *demand periods* characterized by fixed starting and ending times. These demand periods do not necessarily coincide with the shifts. However, the demand per shift can easily be determined.

Coverage constraints imply how many nurses of appropriate skills have to be scheduled for each demand period. For ease of exposition and without loss of generalization we consider all nurses equally-skilled throughout the rest of this paper.

Collective agreement requirements are rules that define acceptable schedules for individual nurses in terms of total workload, holidays, weekends off and shift transitions (e.g. a morning shift after a night shift is not allowed). These rules cannot be violated and dramatically reduce the set of feasible individual roster lines. Obviously, when building nurse schedules also a set of individual constraints, often called preference constraints, have to be taken into account. For instance, some nurses prefer to do night shifts, others do not. Again, for ease of exposition and without loss of generalization, we make abstraction of these differences in individual preferences and only consider those restrictions which are stated in the collective agreement rules and consequently apply on all nurses. Hence, we present an integrated model that can be used to find optimal schedules for a homogeneous set of nurses.

In what follows we state the standard set covering model, which is often used for this type of problems. Let $J$ be the set of feasible roster lines $j$ and $I$ be the set of demand periods $i$. Let $d_i \in \Re^+$, $\forall i \in I$, denote the required number of nurses scheduled during period $i$. Furthermore, let $a_{ij}$ be 1 if roster line $j$ contains an active shift during period $i$ and 0 otherwise. The general integer decision variable $x_j$, $\forall j \in J$, indicates the number of individual nurses which are scheduled by roster line $j$.

Then, the nurse scheduling problem (NSP) can be stated as follows:

$$\text{Minimize} \sum_{j \in J} x_j \tag{1}$$

subject to:

$$\sum_{j \in J} a_{ij} x_j \geq d_i \qquad \forall i \in I \tag{2}$$

$$x_j \in \{0, 1, 2, \dots\} \qquad \forall j \in J \tag{3}$$

### C. Solution procedure for the nurse scheduling problem

The integer program (IP) (1)-(3) is solved by first solving the linear programming relaxation and then using a branching scheme to drive the solution into integrality. As the number of possible roster lines an individual can work is usually too large to allow complete, a-priori enumeration, column generation is often applied to solve the LP relaxation. Typically, the pricing step involves the solution of a dynamic programming shortest path problem (also called the *subproblem*) to find the legal column with the most negative reduced cost. Let $\pi_i$, $\forall i \in I$, denote the dual price of constraint (2). Then, the reduced cost of a new column (roster line) $j$ is given by:

$$1 - \sum_{i \in I} a_{ij} \pi_i \tag{4}$$

A brief discussion of the solution procedure for this subproblem is given in Section IV-A. The process of adding new columns continues until no more columns *price out*, i.e. no more columns with negative reduced cost can be found. However, at that point, the solution is not necessarily integral and applying a standard branch-and-bound procedure to the restricted master with its existing columns will not guarantee an optimal (or feasible) solution. Therefore, a branching scheme has to be applied to drive the solution into integrality. After branching, new columns might price out favorably and hence have to be added to the model.

Since it lies not in the scope of this paper to discuss effective branching schemes for the NSP, we will not go into details about this, but instead refer the reader to the specialized literature. Barnhart et al. [7] discuss appropriate branching strategies for solving a mixed integer program (MIP) using column generation. Since NSP (1)-(3) has identical restrictions on subsets (i.e. there are no subsets having a separate convexity constraint), elaborating a branching scheme is a complex issue. Conventional integer programming branching on variables is not effective for reasons of symmetry and also because fixing variables destroys the structure of the subproblem. Vanderbeck and Wolsey [27] developed a general rule in which one is branching on the constraints (see also [26]). The drawback is that the branching constraints cannot be used to eliminate variables and have to be added to the formulation explicitly. Hence, each branching constraint will contribute an additional dual variable to the reduced cost, which complicates the pricing problem.

### D. The generalized nurse scheduling problem

In the NSP the right hand side values of the coverage constraints (i.e. the $d_i$'s in formulation (1)-(3)) are considered to be fixed. Nevertheless, coverage constraints are based on workload estimations which entail the summations of individual patient *workload contributions*. An individual patient workload contribution is determined by the *patient type*. The patient type can generally be described by three dimensions. The first dimension is the type of surgery the patient has undergone. The second is the number of periods the patient has already recovered. The third is the period to which the workload applies. For instance, some pathologies may require increased care during nights.

The number and type of the patients that are present in the hospital at each moment in time is largely determined by the operation room schedule. Obviously, due to emergency cases and uncertainty in patient show-ups, patient recovery times etc..., exact estimations are not possible. However, an in-depth analysis of the operation room schedule enables hospital executives to make a quite accurate prediction of the workload of the nurses. Moreover, they can reshape the workload distribution by modifying the operation room schedule. In the long term case mix planning decisions determine the overall workload. In shorter term the cyclic master surgery schedule determines the workload distribution over time.

The generalized nurse scheduling problem (GNSP) takes into account this extra dimension. Instead of assuming the demand values to be fixed, we consider them to be dependent on the number and type of patients undergoing surgery in the hospital at each moment. By manipulating the master surgery schedule, hospital management can create (and choose between) a number of different workload distributions, further referred to as *workload patterns*. Let $K$ denote the set of possible workload patterns that could be generated by modifying the surgery schedule. These will be obtained by enumerating all possible ways of assigning operating blocks to

the different surgeons, subject to surgery demand and capacity restrictions (for more details see Section IV-B). Each workload pattern $k$ is described by a number of periodic demands $d_{ik} \in \{0, 1, 2, \dots\}$, $\forall i \in I$. Let $z_k$ be 1 if the surgery schedule that corresponds to workload $k$ is chosen and 0 otherwise. Then, the problem can be stated as follows:

$$\text{Minimize} \sum_{j \in J} x_j \tag{5}$$

subject to:

$$\sum_{j \in J} a_{ij} x_j \geq \sum_{k \in K} d_{ik} z_k \qquad \forall i \in I \tag{6}$$

$$\sum_{k \in K} z_k = 1 \tag{7}$$

$$x_j \in \{0, 1, 2, \dots\} \qquad \forall j \in J \tag{8}$$

$$z_k \in \{0, 1\} \qquad \forall k \in K \tag{9}$$

Constraint (7), further referred to as the workload convexity constraint, implies that exactly one workload pattern has to be chosen. In a feasible solution all $z_k$'s but one equal 0. Hence, in constraint (6) only the corresponding $d_{ik}$'s are added in the right hand side values. It is easy to see that the NSP is a special case of the GNSP in which one $z_k$ is fixed to be 1.

### *E. Solution procedure for the generalized nurse scheduling problem*

In this part we show that the column generation approach to solve the LP relaxation of NSP can easily be extended to cope with the GNSP. Similarly to the roster lines, the number of possible workload patterns is usually too large to allow for complete, a-priori enumeration. Also here, the process starts with a limited subset of workload patterns and new patterns (columns) are added as needed. Therefore, a second subproblem has to be solved. The generation of a new workload pattern boils down to the construction of a new master surgery schedule. The subproblem is constrained by a set of specific surgery schedule restrictions. Its objective is the minimization of the reduced cost of a new workload pattern. Let $\gamma$ denote the dual price of the workload pattern convexity constraint (7). Then, the reduced cost of a new workload pattern $k$ is given by:

$$0 - \gamma + \sum_{i \in I} \pi_i d_{ik} \tag{10}$$

Obviously, the appropriate solution approach to price out a new workload pattern strongly depends on the characteristics of the master surgery schedule. In this paper the workload pattern pricing problem is formulated as an IP and solved using a state-of-the-art optimization package (CPLEX). More details on this formulation can be found in Section IV-B.

## IV. Pricing problems

### *A. Generating a new roster line*

Although the generation of a new roster line happens in a standard way (shortest path problem solved with recursive dynamic programming) (see e.g. [12]) and its exact implementation is not really necessary for understanding the general idea of this paper, we briefly discuss the procedure. First, we summarize the restrictions which apply to a roster line.

As already mentioned earlier, this work is only concerned with collective agreement requirements and leaves individual preferences out of consideration. Concretely, we take into account five types of requirements when building a new roster line. First of all, a nurse cannot work more than one shift per day. Secondly, the overall number of *active days*, i.e. days in which the roster line contains an *active shift* (”*day*”, ”*evening*” or ”*night*”), cannot exceed a certain limit. Thirdly, the maximum number of *consecutive* working days is also constrained. The same holds for the maximum number of consecutive rest days. A sequence of working days is further referred to as a *block*. Fourthly, the number of so-called unpopular shifts (night shifts, weekend shifts) is limited per roster line. Fifthly, in a block, certain shift transitions are not allowed. For instance, a nurse cannot switch from, say, a night shift to a morning shift without having a rest first.

Generating a new roster line is typically done using a dynamic programming recursion. To this aim, we define a table giving the minimum cost that can be achieved in days 1 to $d$ by a roster line that, starting from a situation in which on day $d$ a shift $s$ is scheduled and in which between days $d$ to $n$ a certain number of active shifts $f$ occurred, a certain number of unpopular shifts $g$ occurred and a number of consecutive working or rest days $h$ (including day $d$) is assigned. Formally, the entries of the table are of the form

$$\tau(d, f, g, s, h),$$

defined for $d = 1..n$, $f = 0..f_{max}$, $g = 0..g_{max}$, $s \in S$, $h = 0..h_{max}$. Hereby, $n$ denotes the number of days in the scheduling horizon, $f_{max}$ denotes the maximum number of working days in a roster line, $g_{max}$ is the maximum penalty in terms of unpopular shifts, $S$ is the set of shift types (”*day*”, ”*evening*”, ”*night*”, ”*rest*”) and $h_{max}$ is the maximum of both the maximum number

of consecutive working days ($h_1^{max}$) and the maximum number of consecutive rest days ($h_2^{max}$). Let $p_{d,s}$ be the penalty cost for assigning an unpopular shift $(d, s)$. Let $A$ denote the set of allowed shift transitions $(s, s')$ between two consecutive days on. We consider demand periods as being subsets of the shifts, i.e. no demand period can be spread over more than one shift. However, a shift can consist of more demand periods. Let $Q_{(d,s)}$ be the set of demand periods $i$ that fall into shift $(d, s)$. Let $\lambda_{d,s}$ be the total dual cost of a shift $(d, s)$, i.e. $\lambda_{d,s} = \sum_{i \in Q_{(d,s)}} \pi_i$.

The computation of the entries in the table is done by starting at the beginning of the time horizon and working forward by considering an insertion of a shift type $s$ on the next day $d$ of the roster line associated with an entry already computed. Therefore, we make use of recursive algorithm 1.

**Algorithm 1** RECURSION($d, f, g, s, h$)

**if** (d=0) **then**
  return 0; {*beginning of time horizon reached*}
**else if** ($\tau(d, f, g, s, h) \neq 999999999$) **then**
  return $\tau(d, f, g, s, h)$; {*state already visited, can be pruned*}
**else**
  $cost \leftarrow +\infty$;
  $min\_cost \leftarrow +\infty$;
  **for** (all shifts $\bar{s} \in S \backslash \{"rest"\}$) **do**
    **if** ($g + p_{d-1,\bar{s}} \leq g_{max}$) **AND** ($(\bar{s}, s) \in A$) **AND** ($f < f_{max}$) **then**
      **if** ($s \neq "rest"$) **then**
        **if** ($h < h^1_{max}$) **then**
          $cost \leftarrow \lambda_{d,s}$+RECURSION($d-1, f+1, g+p_{d-1,\bar{s}}, \bar{s}, h+1$); {*successive active shift*}
        **end if**
      **else if** ($s = "rest"$) **then**
        $cost \leftarrow$RECURSION($d-1, f+1, g+p_{d-1,\bar{s}}, \bar{s}, 1$); {*start active shift*}
      **end if**
    **end if**
    **if** ($cost < min\_cost$) **then**
      $min\_cost \leftarrow cost$;
    **end if**
  **end for**
  **if** ($s \neq "rest"$) **then**
    $cost \leftarrow \lambda_{d,s}$+ RECURSION($d-1, f, g, "rest", 1$); {*start rest*}
  **else if** ($s = "rest"$) **then**
    **if** ($h < h^2_{max}$) **then**
      $cost \leftarrow$ RECURSION($d-1, f, g, "rest", h+1$); {*successive rest*}
    **end if**
  **end if**
  **if** ($cost < min\_cost$) **then**
    $min\_cost \leftarrow cost$;
  **end if**
  return $\tau(d, f, g, s, h) \leftarrow min\_cost$;
**end if**

Before starting the recursion all entries of table $\tau(d, f, g, s, h)$ are initialized to 999999999. The minimal reduced cost of a new roster line can now easily be calculated by starting the recursion on day $n$ and minimizing over each shift type (see algorithm 2).

Once all the calculations are done, the best new roster line can easily be constructed backward. The overall

**Algorithm 2** FIND-NEW-ROSTER-LINE

{*initialize all entries of* $\tau$}
**for** ($d = 1$ to $n$) **do**
  **for** ($f = 0$ to $f_{max}$) **do**
    **for** ($g = 0$ to $g_{max}$) **do**
      **for** (all shifts $s \in S$) **do**
        **for** ($h = 0$ to $h_{max}$) **do**
          $\tau(d, f, g, s, h) \leftarrow 999999999$;
        **end for**
      **end for**
    **end for**
  **end for**
**end for**
$cost \leftarrow +\infty$;
$min\_cost \leftarrow +\infty$;
{*start the recursion*}
**for** (all shifts $\bar{s} \in S \backslash \{"rest"\}$) **do**
  **if** ($p_{n,\bar{s}} \leq g_{max}$) **then**
    $cost \leftarrow$RECURSION($n, 1, p_{n,\bar{s}}, \bar{s}, 1$); {*end with an active shift*}
  **end if**
  **if** ($cost < min\_cost$) **then**
    $min\_cost \leftarrow cost$;
  **end if**
**end for**
$cost \leftarrow$RECURSION($n, 0, 0, "rest", 1$); {*end with a rest*}
**if** ($cost < min\_cost$) **then**
  $min\_cost \leftarrow cost$;
**end if**

space complexity of the dynamic programming recursion is

$$O(n \cdot f_{max} \cdot g_{max} \cdot |S| \cdot h_{max})$$

whereas the time complexity is (in the case that there are no forbidden shift transitions),

$$O(n \cdot f_{max} \cdot g_{max} \cdot |S| \cdot h_{max} \cdot |S|)$$

since each entry of the table is updated by considering up to $O(|S|)$ other entries.

### *B. Generating a new workload pattern*

Each workload pattern corresponds to a particular surgery schedule. Hence, a new workload pattern can be obtained by building a new surgery schedule. Hereby, the capacity preserved for the different surgeons (or, more generally, surgery groups) is already determined by the case mix planning (first stage, long term) and considered to be fixed in our application. Elective case scheduling (third stage) is also left out of consideration because of two reasons. First of all, the impact of each specific elective case on the workload is rather scant. It is the type of surgery that determines the workload contribution, not the individual case. Secondly, it is very hard to predict the precise impact of the individual cases on the workload contribution at the moment that the nurse rosters have to be built. Often, at that moment, an important part of the elective surgery scheduling is still to be done.

The master surgery schedule is considered to be the tool for manipulating the workload distribution over time. This work is concerned with *cyclic* master surgery schedules. Cyclic schedules are schedules that are repeated after a certain time period (referred to as the cycle time). During such a cycle time there might be a number of time periods during which surgery cannot take place. These periods are referred to as the inactive periods, the others are active. Typically, cycle times are multitudes of weeks in which the weekends are inactive periods.

In our application, a new surgery schedule is built by solving an integer program. To find a new workload pattern with minimal reduced cost given the current set of roster lines and workload patterns, the objective function minimizes the dual price vector of the demand constraints (6) multiplied by the new demands. We deal with two types of constraints. Surgery demand constraints determine how many blocks must be preserved for each surgeon. Capacity constraints ensure that the number of blocks assigned during each period do not exceed the available capacity. Let $y_{rt}$ ($\forall r \in R$ and $t \in T$) be the number of blocks assigned to surgeon $r$ in period $t$. Hereby, $T$ represents the set of active periods and $R$ the set of surgeons. Let $q_r$ be the number of blocks required by each surgeon $r$. Let $b_t$ be the maximal number of blocks available in period $t$. Let $w_{rti} \in \Re^+$ denote the contribution to the workload of demand period $i$ of assigning one block to surgeon $r$ in period $t$. Then, the integer program to construct a new surgery schedule (and at the same time price out a new workload pattern $k$) is as follows:

$$\text{Minimize} \sum_{i \in I} \pi_i d_{ik} \tag{11}$$

subject to:

$$\sum_{t \in T} y_{rt} = q_r \qquad \forall r \in R \tag{12}$$

$$\sum_{r \in R} y_{rt} \leq b_t \qquad \forall t \in T \tag{13}$$

$$\sum_{r \in R} \sum_{t \in T} w_{rti} y_{rt} \leq d_{ik} \qquad \forall i \in I \tag{14}$$

$$y_{rt} \in \{0, 1, 2, \ldots, \min(q_r, b_t)\} \quad \forall r \in R, \forall t \in T \tag{15}$$

$$d_{ik} \in \{0, 1, 2, \ldots\} \qquad \forall i \in I \tag{16}$$

The objective function (11) minimizes the reduced cost of a new workload pattern. Observe that the periodic demands $d_{ik}$ are now an integral part of the decision process, whereas these are merely coefficients in the master problem (5)-(9). Constraint set (12) implies that each surgeon obtains the number of required blocks. Constraint set (13) ensures that the number of blocks assigned does not exceed the available number of blocks in each period. Constraint set (14) triggers the $d_{ik}$'s to the appropriate integer values. Finally, constraint set (15) and (16) define $y_{rt}$ and $d_{ik}$ to be integer.

At first sight, constraint set (16) which requires the periodic demands $d_{ik}$ to be integral, seems to be redundant from a formulation point of view. Indeed, due to constraint (6) and the fact that $a_{ij} \in \{0, 1\}$ and $x_j \in \{0, 1, 2, \ldots\}$ fractional demand values $d_{ik}$ would also be covered by the upper integer number of nurses. The reason why we require the $d_{ik}$'s to be integral is to improve the computational efficiency of the overall branch-and-price algorithm. We come back to this issue in Section VII-A.

## V. Overview of the Branch-and-Price Algorithm

Algorithm 3 contains the pseudocode of the branch-and-price algorithm to solve the GNSP.

The algorithm starts with a heuristic in order to find an initial solution. The heuristic generates only one workload pattern. This is done by building a surgery schedule for which the sum of the resulting quadratic demand values is minimized. The idea is to level the workload distribution as much as possible over the time horizon and as such to avoid the occurrence of peeks in the workload. This approach turned out to be beneficial for the surgery scheduling problem in which the expected shortage of beds has to be minimized (see [9]). The surgery schedule is built with a mixed integer program (MIP) in which the constraints are given by (12)-(15) (replacing the $d_{ik}$'s by $d_i$'s) and the objective is:

$$\text{Minimize} \sum_{i \in I} d_i^2$$

with $d_i$ the required number of nurses in period $i$. To speed up the heuristic, the $d_i$'s are not required to be integral. Instead, we round each $d_i$ to the next upper integer after solution of the quadratic MIP. Given this workload pattern, new roster lines are added until the set of roster lines (one nurse scheduled by each roster line) completely satisfies the coverage constraints. A new roster line is found by solving exactly the same shortest path problem as in Section IV-A, but replacing the dual prices $\pi_i$ by the remaining right hand side values $d_i$. As such each new roster line cuts the peeks in the remaining workload pattern until all demand is covered.

After detection of an initial solution, the objective value is saved as an upper bound and both the surgery schedule and the nurse schedule are registered. The columns making up the initial solution are entered into

**Algorithm 3** BRANCH-AND-PRICE

```
apply heuristic to find initial solution;
if (solution found) then
  register nurse schedule and surgery schedule;
  upper_bound ← best solution found;
  initiate master with the columns making up the initial solution and (|I|+
  1) supercolumns;
else
  upper_bound ← +∞;
  initiate master with |I| + 1 supercolumns;
end if
lower_bound ← −∞;
stop ← FALSE;
while (stop=FALSE) do
  LP_opt_found ← FALSE;
  {solve LP with column generation}
  while (LP_opt_found=FALSE) do
    LP_opt_found ← TRUE;
    improving_roster_line_found ← TRUE;
    while (improving_roster_line_found=TRUE) do
      RC_j ← FIND-NEW-ROSTER-LINE(j);
      if (RC_j < 0) then
        add new roster line to master;
        LP_opt_found ← FALSE;
        LP_opt ← SOLVE-MASTER-LP();
      else
        improving_roster_line_found ← FALSE;
      end if
    end while
    RC_k ← FIND-NEW-WORKLOAD-PATTERN(k);
    if (RC_k < 0) then
      add new workload pattern to master;
      LP_opt_found ← FALSE;
      LP_opt ← SOLVE-MASTER-LP();
    end if
  end while{LP solved to optimality}
  if (fractional_z) then
    expand node; {replace node by two child nodes}
  else if (LP_opt<best_integral_z) then
    best_integral_z ← LP_opt;
  end if
  if (no more nodes) then
    stop ← TRUE;
  else
    explore next node; {best-first}
    lower_bound ← bound_best_node;
    if (lower_bound ≥ upper_bound OR lower_bound ≥ best_integer_z)
    then
      stop ← TRUE;
    end if
  end if
  IP_opt ← SOLVE-MASTER-IP();
  if (IP_opt < upper_bound) then
    upper_bound ← IP_opt;
    register nurse schedule and surgery schedule;
  end if
end while
```

the master together with a number of supercolumns, which are needed to ensure feasibility of the master in each stage of the branch-and-bound algorithm.

The algorithm starts with the LP optimization loop in which iteratively a number of new roster lines and one new workload pattern are added until no more columns price out. Observe that roster lines are added until no more lines with negative reduced cost can be found, whereas only one workload pattern is generated, after which the generation of new roster lines restarts. This approach turned out to be the most successful, given the generally larger computation times to price out a new workload pattern.

Upon detection of the LP optimum, the solution is checked for fractional $z_k$'s (workload patterns). If there still are fractional $z_k$'s, branching is applied in order to drive the solution into an integral $z$ solution (i.e. with only one $z_k$ equal to 1 and all other equal to 0). The algorithm does not branch until an integral $x_j$ (roster line) solution, because branching schemes for the $x_j$ variables are not straightforward to implement and significantly complicate the roster line subproblem. Moreover, it provides no extra value for the extended model, which is the subject of this paper. Instead, we report lower and upper bounds for the required number of nurses to cover demand. The lower bound is the best possible solution with exactly one $z_k$ equal to 1, however one for which the $x_j$'s are not necessarily integral. Hence, the solution represented by the lower bound might not be interpretable in terms of the nurse schedule (e.g. schedule 2.5 nurses following roster line $j$). The upper bound on the other hand is the best found overall integer solution (with also integrality of the $x_j$'s), which is fully interpretable.

In order to increase the lower bound as much as possible, the branch-and-bound tree is traversed in a best-search way. After each move in the tree, the master problem is solved with required integrality on both the $x_j$'s and the $z_k$'s. Because the integral master problem is often computationally very intensive, the MIP optimizer is interrupted after a specified time interval (e.g. 10 seconds). If a better solution is found, the upper bound decreases and as such the gap between the lower and upper bound tightens.

## VI. BRANCHING

For reasons that are explained earlier, this work is only concerned with a branching scheme for driving the $z_k$'s to integrality and leaves the $x_j$'s out of consideration. We apply a constraint branching scheme [23] which works as follows.

First we search for the highest fractional $z_k$. Let this be $z_{k'}$. Then we select another $z_k > 0$, say $z_{k''}$, and take the first period $i$ for which $d_{ik'} \neq d_{ik''}$. If no such period exists, both $z_k$'s represent essentially the same workload patterns and hence one of them can be set to 0 while its fractional value is added to the other one. Suppose we found period $i'$ as the branching period with $d_{i'k'} < d_{i'k''}$. Then, we create two nodes in the branch-and-bound tree. In the left node we imply $d_{i'k} \leq d_{i'k'}$ and in the right node we imply $d_{i'k} \geq d_{i'k'} + 1$. Figure 2 visualizes this branching scheme. Else if $d_{i'k'} > d_{i'k''}$ we imply $d_{i'k} \leq d_{i'k''}$ in the left node and $d_{i'k} \geq d_{i'k''} + 1$ in the right node.

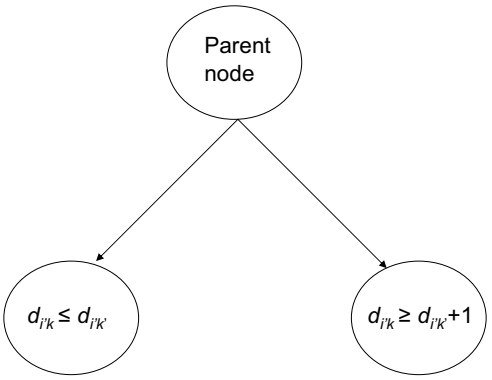


Fig. 2. Binary branching scheme in the case of $d_{i'k'} < d_{i'k''}$

## VII. Computational performance issues

In this section we present some techniques which helped to improve the computational efficiency of the algorithm.

### A. Integral versus fractional demand values

It has already been mentioned at the end of Section IV-B that we imply the $d_{ik}$'s to be integral in the workload pattern pricing problem. Although this is not necessary from a formulation point of view, it has a substantially positive impact on the overall computational efficiency of the algorithm.

Implying integrality of the $d_{ik}$'s affects the computation time in two ways. On the one hand, there is a negative impact, because the pricing problem itself becomes more complex. On the other hand, there is a positive impact as far fewer columns can be found with negative reduced cost. Preliminary results indicate that this positive effect dramatically exceeds the negative effect. Consequently, the master LP is solved much faster when integrality of the $d_{ik}$'s is implied. Moreover, requiring integral demand values in the workload patterns makes the LP optimal solution substantially less fractional in terms of the $x_j$'s. Hence, finding a global optimum (with both integrality on the $z_k$'s and on the $x_j$'s) turns out to be much easier. In our application the gap between the lower and upper bound becomes much smaller.

### B. Upper bound pruning for the workload pattern pricing problem

Basically, we are no longer interested in finding the column with the lowest reduced cost from the moment we know that this reduced cost will be positive anyway. Hence, we can act as if we already found a solution with reduced cost 0 by providing an appropriate upper bound. For the workload pattern subproblem, this observation yields dramatic time savings.

The reduced cost expression (4) consists of a fixed part and a variable part. By setting the upper bound equal to the fixed part with reverse sign, we act as if we found already a new column with reduced cost equal to 0. The reduced cost of a workload pattern is given by $0 - \gamma + \sum_{i \in I} \pi_i d_{ik}$. Consequently, we provide $\gamma$ as an upper bound in the integer program (11)-(16).

Note that, since generating a new roster line is done using a backward dynamic recursion, upper bound pruning cannot be applied here. As an alternative, we wrote an A* algorithm (enumeration approach entailing a forward recursion including both dynamic pruning and pruning based on bound comparisons). Dynamic pruning occurs if a state has already been visited at lower cost. For pruning based on bound comparisons we need an upper and lower bound for the best new roster line. Since the reduced cost of a new roster line is given by $1 - \sum_{i \in I} a_{ij} \pi_i$, we can provide -1 as an initial upper bound in the A* algorithm. Obviously, this bound is decreased each time a better roster line is found. Starting from a certain day, a lower bound on the minimal cost path could be obtained by selecting for each remaining day the shift with the lowest total of corresponding dual prices, i.e:

$$\text{MIN}\Big\{ \underset{s \in S \setminus \{"rest"\}}{\text{MIN}} \{\lambda_{d,i}\}, 0 \Big\} \qquad \forall d$$

and summing up only the $(f_{max} - f)$ lowest values amongst these. In other words, for calculating the lower bound, we relax all constraints but the not-more-than-one-shift-per-day constraint and the maximum number of active days constraint. Preliminary tests, however, indicated that the A* algorithm is outperformed by the backward dynamic recursion. Hence, the time saved from upper bound pruning in the A* algorithm is inferior to

the time won by visiting each state only once in the purely dynamic backward recursion.

### C. Two-phase approach for the workload pattern pricing problem

During the LP optimization loop it is not necessary to find the column with the most negative reduced cost, any column with negative reduced cost will do. Again, particularly for the computationally intensive workload pattern pricing problem, using this observation dramatically decreases the computation times. To guarantee optimality of the LP solution, a two-phase approach is applied for the workload pattern pricing problem. In the first phase, a certain time limit is set for the MIP optimizer. Only if no new workload pattern is found with negative reduced cost within this time limit, the algorithm enters the second phase. In this phase the time limit is undone and the optimizer is required to search until a feasible solution is found with negative reduced cost or it is proven that such a column does not exist.

### D. Lagrange dual pruning

It is well known that Lagrangian relaxation can complement column generation in that it can be used in every iteration of the column generation scheme to compute a lower bound to the original problem with little additional computational effort (see e.g. [25], [27]). If this lower bound exceeds an already found upper bound, the column generation phase can end without any risk of missing the optimum. Using the information from solving the reduced master and the information provided by solving the pricing problem for a new workload pattern $k$, it can be shown (see e.g. [16]) that a lower bound is given by $\delta + RC_k\theta_k$ where $\delta$ is the objective value of the reduced master, $RC_k$ is the reduced cost of a newly found workload pattern $k$ and $\theta_k$ is a binary variable equal to 1 when $RC_k$ is non-negative and set to zero, otherwise. This lower bound is referred to as the Lagrangian lower bound, since it can be shown that it equals the bound obtained by Lagrange relaxation.

Obviously, if the pricing procedure finds a negative reduced cost column during the first phase and hence does not enter the second phase (see Section VII-C) this lower bound cannot be used, because the workload pattern pricing problem has not been solved to optimality.

Using CPLEX, it is very easy to set upper bounds, time limits and limits on the number of feasible solutions. Moreover, it can easily be verified if either the problem has been solved to optimality or optimization has prematurely ended because of an insufficient time limit.

## VIII. Computational results

### A. Test set

To test the algorithm, we started from the same set as the one introduced in [9] for their surgery scheduling application. All surgery scheduling problems in this set involve a cycle time of 7 days. The last two days are not available to allocate OR time (weekend), which is common practice. The problems differ with respect to five factors. These are: (1) the number of time blocks per day, (2) the number of surgeons, (3) the division of requested blocks per surgeon, (4) the number of operated patients per surgeon and finally (5) the length of stay (LOS) distribution. If we consider two settings for each factor and repeat each factor combination three times, we obtain $2^5 * 3 = 96$ test instances. Table I contains the settings for these five factors. Some of the factor settings require some further explanation.

TABLE I
FACTOR SETTINGS IN SURGERY SCHEDULING TEST SET

| Factor setting | Nr. blocks per day | Nr. surgeons | Division req. blocks | Nr. patients per surgeon | LOS |
|---|---|---|---|---|---|
| 1 | 3-6 | 3-7 | evenly distributed | 3-5 | 2-5 |
| 2 | 7-12 | 8-15 | not evenly distributed | 3-12 | 2-12 |

The number of blocks per day is drawn from a uniform distribution with bounds 3 and 6 in the first setting and 7 and 12 in the second setting. A block is defined as the smallest time unit for which a specific operating room can be allocated to a specific surgeon (or surgical group). Note that, due to large set-up time and costs, in real-life applications the number of blocks per day in one operating room is usually 1 or 2, i.e each surgical group has the OR for at least half a day. Hence, considering more blocks can be seen as a way of considering more operating rooms as there is no difference from a computational point of view. The third factor indicates whether or not the requested blocks are evenly distributed among all surgeons; e.g. if there are 20 time blocks and 5 surgeons, each surgeon requires 4 time blocks in the evenly distributed case, whereas in the unevenly distributed case huge differences can occur. For the LOS in factor 5 we simulated exponential distributions (made discrete by use of binomial distributions) with mean dependent on the factor setting.

Next, we generated some weights $w_{rti}$ defining the contributions to the workload of period $i$ of allocating a block to surgeon $r$ in period $t$. These weights

vary linearly with the number of patients of surgeon $r$ operated in period $t$ that are still in the hospital in period $i$. The patient's workload contribution generally decreases the longer the patient has already recovered in the hospital. In our test set the workload demand periods coincide with the shifts. Furthermore, we set the contribution to a "day" shift two times as large as the one to an "evening" shift and four times as large as the one to a "night" shift. Obviously, although attempting to represent realistic scenarios, these contributions are chosen somewhat arbitrarily.

Thirdly, we composed a set of collective agreement rules which apply on individual roster lines. The scheduling horizon amounted to 4 weeks or 28 days ($= n$). The maximum days an active shift could be scheduled ("day", "evening" or "night") was set to 20 ($= f_{max}$). Shifts during the weekends were marked as unpopular shifts: day and evening shifts got a penalty of 1, night shifts got a penalty of 2. The maximum number of consecutive working days was set to 6 (=$h_1^{max} = h_{max}$) and the maximum number of consecutive rest days was set to 3 ($= h_2^{max}$). Furthermore, we distinguished between two scenarios: a hard constrained scenario and a flexible one. Collective agreement rules in the hard constrained scenario differ from those in the flexible scenario on the following two points:

- In the hard constrained scenario, there is only one shift type allowed within each block. In other words, no shift transitions between different shift types can occur without scheduling a rest first. In the flexible scenario, all shift transitions are allowed, except the following three: a "night" shift followed by a "day" shift, a "night" shift followed by an "evening" shift or an "evening" shift followed by a "day" shift.
- In the hard constrained scenario, the maximal penalty with respect to unpopular shifts is set to 4, whereas in the flexible scenario it is set to 8 (=$g_{max}$).

### B. Savings

Table II contains the lower and upper bounds for both the NSP and the GNSP. In the NSP, a surgery schedule is generated randomly. The resulting workload pattern contains the (fixed) right-hand side values of the coverage constraints. Then, the NSP is solved using column generation. In the GNSP, new surgery schedules (and hence resulting workload patterns) are generated during search if needed. We distinguish between the flexible and the hard constrained scenario. To give an idea of the variability, the detailed bounds are provided for the first 9 and the last 9 problems of the problem set. The last line contains the average bounds over the whole set. Observe that the name of each problem (d*ijklm_n*) contains the information about the surgery scheduling subproblem: $i$ stands for the setting of the first factor in Table I (0 for the first setting, 1 for the second), $j$ for the second one, etc…, and $n$ for the iteration number.

From these results one may conclude the following. First have a look at the upper bounds, which are after all the solutions that will be worked with. Although it is not guaranteed that the upper bound will be better (one might be lucky in the NSP and find the same or even a better overall integer solution), the upper bounds for the GNSP are generally better than those for the NSP. We compared them using a one-tailed paired T-test. The extremely small p-values obtained indicate that the differences are statistically significant both for the flexible and for the hard constrained case. The same results are obtained for the lower bounds. Unlike the upper bounds, the GNSP lower bounds are of course guaranteed to be at least as good as the NSP lower bounds.

When comparing the lower bounds for the NSP with the upper bounds for the GNSP, both scenarios entail different conclusions. The average lower bound for the NSP is lower than the average upper bound for the GNSP in the flexible scenario, whereas the reverse is true in the hard constrained scenario. Both differences turned out to be significant using a one-tailed paired T-test (again extremely small p-values). This observation can easily be explained. The stricter the collective agreement rules, the harder it is to nicely fit the nurse rosters into the required workload pattern in the NSP. As the workload pattern can be adapted in the GNSP, the GNSP includes more possible savings in the case of severe collective agreement requirements.

### C. Interpretation of the savings

In the previous section we concluded that integrating the surgery scheduling process with the nurse scheduling process may yield important savings in terms of required nurses to hire. In this section we identify the source of these savings. Therefore, we provide an answer to the question: 'Where lies the waste if one is considering the surgery schedule (and hence the workload distribution) as being fixed?' It turns out that the origin of the waste is twofold.

First of all, an unfavorable workload pattern may contain many workload demands that slightly exceed the workforce of $x$ nurses, but that are dramatically inferior to the workforce of $x + 1$ nurses. In terms of the $d_{ik}$'s one could think of many $d_{ik}$'s having a small decimal part, like e.g. 6.1, 8.2, 4.05 etc… This type of waste is

TABLE II
LOWER AND UPPER BOUNDS FOR THE NSP AND THE GNSP

| | | Flexible scenario | | | | Hard constrained scenario | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | NSP | | GNSP | | NSP | | GNSP | |
| Nr. | Problem | lb | ub | lb | ub | lb | ub | lb | ub |
| 1 | d00000_0 | 15 | 17 | 13 | 15 | 19 | 19 | 16 | 17 |
| 2 | d00000_1 | 26 | 28 | 25 | 27 | 34 | 35 | 31 | 31 |
| 3 | d00000_2 | 25 | 27 | 23 | 25 | 32 | 32 | 28 | 29 |
| 4 | d00001_0 | 40 | 42 | 39 | 41 | 49 | 50 | 47 | 48 |
| 5 | d00001_1 | 45 | 47 | 44 | 46 | 54 | 54 | 52 | 53 |
| 6 | d00001_2 | 94 | 96 | 92 | 94 | 112 | 113 | 109 | 110 |
| 7 | d00010_0 | 34 | 36 | 32 | 35 | 43 | 43 | 40 | 40 |
| 8 | d00010_1 | 40 | 42 | 38 | 40 | 49 | 50 | 47 | 47 |
| 9 | d00010_2 | 28 | 30 | 26 | 27 | 34 | 35 | 32 | 33 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 88 | d11101_0 | 96 | 98 | 94 | 96 | 114 | 115 | 112 | 113 |
| 89 | d11101_1 | 99 | 102 | 97 | 99 | 119 | 120 | 116 | 116 |
| 90 | d11101_2 | 122 | 125 | 119 | 121 | 145 | 146 | 142 | 143 |
| 91 | d11110_0 | 83 | 85 | 80 | 82 | 101 | 102 | 96 | 96 |
| 92 | d11110_1 | 111 | 113 | 109 | 111 | 138 | 139 | 132 | 132 |
| 93 | d11110_2 | 58 | 60 | 56 | 58 | 73 | 74 | 67 | 68 |
| 94 | d11111_0 | 252 | 254 | 249 | 252 | 303 | 304 | 296 | 297 |
| 95 | d11111_1 | 119 | 122 | 116 | 119 | 143 | 144 | 139 | 140 |
| 96 | d11111_2 | 135 | 137 | 131 | 133 | 162 | 163 | 156 | 157 |
| | Average | 70.18 | 72.43 | 68.33 | 70.44 | 86.07 | 86.73 | 81.91 | 82.61 |

referred to as the waste due to the workforce surplus per shift. In many hospitals this kind of waste is taken care of by simply scheduling $x$ nurses instead of $x+1$ nurses during those shifts. The result is a group of overworked nurses and an almost for sure decrease in the quality of care. This illustrates how the GNSP approach can also be very useful for optimizing qualitative instead of quantitative objectives.

Secondly, waste also originates from the inflexibility of the roster lines, due to strict general agreement requirements. Because of this, no set of roster lines can be found that perfectly fit with the workload demand. This source of waste is further referred to as waste due to the inflexibility of roster lines.

Table III gives an overview of the importance of both sources of waste. Hereby, we again distinguish between the flexible scenario and the hard constrained scenario. For each scenario there are three columns. The first column contains the total waste in terms of overstaffing in the NSP compared with the GNSP. These numbers are obtained by subtracting the upper bounds for the GNSP from those for the NSP. The second and third column indicate the parts of this total waste that are due to the workforce surplus per shift and to the inflexibility of roster lines. These numbers can easily be calculated as follows. Firstly, for both the NSP and the GNSP we make the sum of the (integral) demands of the chosen workload pattern. Call this number the total required workforce ($= \sum_{i \in I} d_i$ for the NSP and $\sum_{i \in I} \sum_{k \in K} d_{ik} z_k$ for the GNSP). Next, divide this number by the workforce per nurse ($= f_{max}$ in our application). This gives the minimal number of nurses that would be needed and can be obtained in the case of fully flexible roster lines. The difference between these numbers for the NSP and GNSP is the waste due to the workforce surplus per shift. The difference between the total waste and the waste due to the workforce surplus per shift is the waste due to the inflexibility of roster lines. Observe that these wastes may be negative (e.g. the waste due to workforce surplus per shift for problem d00000_2 is -1). This situation occurs when the gain with respect to one source of waste is so large that the best found solution for the GNSP includes a limited sacrifice with respect to the other source of waste.

The results in Table III clearly indicate that the importance of the source of waste strongly depends on the strictness of the general agreement requirements. The stricter these requirements are, the larger is the share of the waste due to the inflexibility of the roster lines.

TABLE III
INTERPRETATION OF THE SAVINGS

| | | Flexible scenario | | | Hard constrained scenario | | |
|---|---|---|---|---|---|---|---|
| Nr. | Problem | Total waste | Waste due to workforce surplus per shift | Waste due to inflexibility of roster lines | Total waste | Waste due to workforce surplus per shift | Waste due to inflexibility of roster lines |
| 1 | d00000_0 | 2 | 1.2 | 0.8 | 2 | 1.2 | 0.8 |
| 2 | d00000_1 | 1 | 1.2 | -0.2 | 4 | 1.4 | 2.6 |
| 3 | d00000_2 | 1 | 2 | -1 | 3 | 1 | 2 |
| 4 | d00001_0 | 1 | 1.2 | -0.2 | 2 | 0.6 | 1.4 |
| 5 | d00001_1 | 2 | 1 | 1 | 1 | 0.2 | 0.8 |
| 6 | d00001_2 | 2 | 1.6 | 0.4 | 3 | 0 | 3 |
| 7 | d00010_0 | 1 | 1.4 | -0.4 | 3 | 1 | 2 |
| 8 | d00010_1 | 1 | 1.6 | -0.6 | 3 | 1.6 | 1.4 |
| 9 | d00010_2 | 1 | 1.8 | -0.8 | 2 | -0.6 | 2.6 |
| … | … | … | … | … | … | … | … |
| 88 | d11101_0 | 2 | 1.4 | 0.6 | 2 | 0.6 | 1.4 |
| 89 | d11101_1 | 2 | 1.8 | 0.2 | 4 | 0.2 | 3.8 |
| 90 | d11101_2 | 1 | 2.2 | -1.2 | 3 | 0.2 | 2.8 |
| 91 | d11110_0 | 2 | 1.6 | 0.4 | 6 | 0.8 | 5.2 |
| 92 | d11110_1 | 2 | 0.8 | 1.2 | 7 | 0.6 | 6.4 |
| 93 | d11110_2 | 1 | 2 | -1 | 6 | 1.8 | 4.2 |
| 94 | d11111_0 | 2 | 1.2 | 0.8 | 7 | 0.2 | 6.8 |
| 95 | d11111_1 | 2 | 1.8 | 0.2 | 4 | -0.6 | 4.6 |
| 96 | d11111_2 | 1 | 2 | -1 | 6 | 0.6 | 5.4 |
| | Average | 1.58 | 1.43 | 0.16 | 4.11 | 0.28 | 3.84 |

### *D. Computational results*

Table IV and Table V contain the computational results for the flexible respectively hard constrained scenario. For the NSP, both the computation time and the number of generated roster lines are given. For the GNSP also the number of generated demand patterns and the number of nodes in the branch-and-bound tree are provided.

Obviously, the required computation times for the GNSP exceed those for the NSP. However, taking into account the explosion of the feasible solution space for the GNSP compared to the NSP, the increase in computation time is rather small. We can conclude that column generation is an excellent technique for solving the GNSP.

If we compare the flexible scenario with the hard constrained scenario, a couple of things attract the attention. First of all, observe that for the NSP the computation times for the flexible scenario surpass those for the hard constrained scenario, whereas for the GNSP the computation times for the hard constrained scenario exceed those for the flexible scenario. For the NSP this difference is statistically significant (extremely small p-value for a two-tailed paired T-test) and easy to explain. In the flexible scenario much more legal roster lines exist and hence much more roster lines with negative reduced cost are found during the search process (on average 207.25 versus 106.07). Moreover, the time needed to price out a new roster line is also larger since the feasible state space contains more legal states.

For the GNSP the difference in computation time is not statistically significant at the 5% level (p-value of 0.113 for a two-tailed paired T-test). As again the number of generated roster lines is significantly smaller (very small p-value for a two-tailed paired T-test), the higher computation times for the constrained scenario must be produced by the higher number of generated workload patterns and the higher number of nodes in the branch-and-bound tree. The differences in number of generated workload patterns and in nodes in the branch-and-bound tree are found to be significant (very small p-values for two-tailed paired T-tests). This can easily be explained as follows. In the flexible scenario, it is unlikely that an extra workload pattern improves the overall solution. Thanks to the flexibility in the roster lines, an already very good solution can be found using a limited set of workload patterns. In the hard constrained case on the other hand, the inflexibility of the roster lines might

obstruct the detection of a good solution. In this case, it is far more likely that adding a new workload pattern improves the overall solution. We can conclude that the GNSP is easier to solve if the collective agreement requirements are less strict, whereas the reverse is true for the NSP.

As a final remark we note that a large part of the computation time goes to the calculation of an overall feasible solution in order to detect an upper bound after each move in the branch-and-bound tree in the GNSP and at the end of the column generation process in the NSP.

TABLE IV
COMPUTATIONAL RESULTS FOR THE FLEXIBLE SCENARIO

| | | NSP | | GNSP | | | |
|---|---|---|---|---|---|---|---|
| | | | Roster | | Roster | Workload | |
| Nr. | Problem | Time (s) | lines | Time (s) | lines | patterns | Nodes |
| 1 | d00000_0 | 43484 | 150 | 44422 | 183 | 2 | 0 |
| 2 | d00000_1 | 44063 | 174 | 51000 | 196 | 2 | 0 |
| 3 | d00000_2 | 46423 | 235 | 45438 | 213 | 2 | 0 |
| 4 | d00001_0 | 44078 | 173 | 46000 | 221 | 2 | 0 |
| 5 | d00001_1 | 43829 | 167 | 45172 | 190 | 2 | 0 |
| 6 | d00001_2 | 44844 | 212 | 48829 | 238 | 3 | 0 |
| 7 | d00010_0 | 45266 | 211 | 70359 | 274 | 2 | 0 |
| 8 | d00010_1 | 46311 | 237 | 185623 | 535 | 17 | 8 |
| 9 | d00010_2 | 44594 | 208 | 166892 | 640 | 32 | 13 |
| … | … | … | … | … | … | … | … |
| 88 | d11101_0 | 44390 | 213 | 47984 | 243 | 2 | 0 |
| 89 | d11101_1 | 44953 | 228 | 52031 | 257 | 2 | 0 |
| 90 | d11101_2 | 44734 | 230 | 56438 | 280 | 2 | 0 |
| 91 | d11110_0 | 46203 | 252 | 358811 | 555 | 30 | 15 |
| 92 | d11110_1 | 45265 | 238 | 1765257 | 815 | 128 | 59 |
| 93 | d11110_2 | 47359 | 200 | 423125 | 507 | 28 | 14 |
| 94 | d11111_0 | 46360 | 347 | 69266 | 381 | 2 | 0 |
| 95 | d11111_1 | 45719 | 243 | 59063 | 319 | 2 | 0 |
| 96 | d11111_2 | 45048 | 237 | 251970 | 512 | 14 | 6 |
| | Average | 44146.04 | 207.25 | 99008.57 | 310.31 | 5.93 | 1.95 |

TABLE V
COMPUTATIONAL RESULTS FOR THE HARD CONSTRAINED SCENARIO

| | | NSP | | GNSP | | | |
|---|---|---|---|---|---|---|---|
| | | | Roster | | Roster | Workload | |
| Nr. | Problem | Time (s) | lines | Time (s) | lines | patterns | Nodes |
| 1 | d00000_0 | 453 | 46 | 66953 | 263 | 8 | 4 |
| 2 | d00000_1 | 500 | 70 | 55359 | 304 | 18 | 6 |
| 3 | d00000_2 | 422 | 64 | 11781 | 111 | 2 | 0 |
| 4 | d00001_0 | 468 | 77 | 609 | 81 | 2 | 0 |
| 5 | d00001_1 | 453 | 74 | 687 | 95 | 3 | 0 |
| 6 | d00001_2 | 672 | 120 | 782 | 127 | 2 | 0 |
| 7 | d00010_0 | 4250 | 113 | 216064 | 470 | 79 | 43 |
| 8 | d00010_1 | 953 | 113 | 323236 | 448 | 129 | 47 |
| 9 | d00010_2 | 750 | 80 | 201970 | 459 | 102 | 39 |
| … | … | … | … | … | … | … | … |
| 88 | d11101_0 | 2125 | 122 | 1656 | 130 | 2 | 0 |
| 89 | d11101_1 | 1531 | 126 | 2625 | 146 | 2 | 0 |
| 90 | d11101_2 | 1610 | 149 | 2109 | 159 | 2 | 0 |
| 91 | d11110_0 | 1938 | 123 | 456191 | 439 | 58 | 17 |
| 92 | d11110_1 | 1500 | 152 | 1228851 | 508 | 92 | 45 |
| 93 | d11110_2 | 5438 | 101 | 102470 | 310 | 10 | 1 |
| 94 | d11111_0 | 8000 | 251 | 12265 | 264 | 2 | 0 |
| 95 | d11111_1 | 4859 | 143 | 19359 | 185 | 2 | 0 |
| 96 | d11111_2 | 4922 | 153 | 1809557 | 600 | 221 | 83 |
| | Average | 1215.52 | 106.07 | 153927.85 | 226.05 | 28.08 | 10.81 |

## IX. CONCLUSIONS AND FURTHER RESEARCH

This paper presents an integrated approach for building nurse and surgery schedules. It has been shown how the column generation technique, often employed for solving nurse scheduling problems, can easily be extended to cope with this integrated approach. The approach involves the solution of two types of pricing problems, the first one is solved with a standard dynamic programming recursion, the second one by aims of a state-of-the-art mixed integer programming optimizer. A constraint branching scheme has been proposed to drive the solution into integrality with respect to the workload patterns while the integrality of the roster lines was left out of the scope of this paper. Finally, some techniques were presented that helped to improve the computational efficiency of the branch-and-price algorithm.

Our computational results indicate that considerable savings could be achieved by using this approach to build nurse and surgery schedules. We simulated problems for a large range of surgery scheduling instances and distinguished between a flexible and a hard constrained scenario with respect to the collective agreement requirements. Our conclusions can be summarized as follows. First of all, column generation is a good technique to

deal with the extra problem dimension of modifying surgery schedules. Secondly, the obtained gains originate from two sources of waste: waste due to the workforce surplus per shift and waste due to the inflexibility of roster lines. Thirdly, unlike the NSP, the GNSP turns out to become harder to solve when the collective agreement requirements are more strict.

Obviously, in real-life hospital environments it is not so easy to modify the master surgery schedule. As the surgery schedule can be considered to be the main engine of the hospital, it not only has an impact on the workload distribution for nurses, but also on several other resources throughout the hospital. Think for instance about anaesthetists, equipment, radiology, laboratory tests and consultation. This observation yields a negative as well as a positive note for the reasoning in this paper. The negative note is that the possible savings obtained through integrating the nurse and the surgery scheduling process are in real-life probably much smaller, due to the smaller flexibility with which surgery schedules can be modified. The positive note is that not only savings in nurse staffing costs are possible, but also in other related resource types, by integrating the scheduling of these resources with the surgery scheduling process. This is probably the main contribution of this paper. This work clearly shows the benefits of integrating scheduling processes in health care environments and moreover proposes a methodology for implementing the heart of a supporting ICT infrastructure.

Possible topics for further research include the application of this approach in a real-world environment involving a detailed report on the experienced merits and pitfalls. From a theoretical point of view, it would be interesting to elaborate this technique for one or more of the other resource types stated above.

## ACKNOWLEDGEMENTS

We acknowledge the support given to this project by the Fonds voor Wetenschappelijk Onderzoek (FWO) - Vlaanderen, Belgium under contract number G.0463.04.

## REFERENCES

[1] H. Alfares and J. Bailey, “Integrated project task and manpower scheduling,” *IIE Transactions*, vol. 29, pp. 711–718, 1997.

[2] H. Alfares, J. Bailey, and W. Lin, “Integrating project operations and personnel scheduling with multiple labor classes,” *Production Planning & Control*, vol. 10, pp. 570–578, 1999.

[3] M. N. Azaiez and S. S. Al Sharif, “A 0-1 goal programming model for nurse scheduling,” *Computers and Operations Research*, vol. 32, pp. 491–507, 2005.

[4] J. F. Bard, C. Binici, and A. H. deSilva, “Staff scheduling at the United States Postal Service,” *Computers and Operations Research*, vol. 30, pp. 745–771, 2003.

[5] J. F. Bard and H. W. Purnomo, “A column generation-based approach to solve the preference scheduling problem for nurses with downgrading,” *Socio-Economic Planning Sciences*, vol. 39, pp. 193–213, 2005.

[6] ——, “Preference scheduling for nurses using column generation,” *European Journal of Operational Research*, vol. 164, pp. 510–534, 2005.

[7] C. Barnhart, E. L. Johnson, G. L. Nemhauser, M. W. P. Savelsbergh, and P. H. Vance, “Branch-and-price: Column generation for solving huge integer programs,” *Operations Research*, vol. 46, pp. 316–329, 1998.

[8] N. Beaumont, “Scheduling staff using mixed integer programming,” *European Journal of Operational Research*, vol. 98, pp. 473–484, 1997.

[9] J. Beliën and E. Demeulemeester, “Integer programming for building robust surgery schedules,” Katholieke Universiteit Leuven, Department of Applied Economics, Research Report OR 0446, 2004.

[10] J. T. Blake and M. W. Carter, “A goal programming approach to strategic resource allocation in acute care hospitals,” *European Journal of Operational Research*, vol. 140, pp. 541–561, 2002.

[11] J. T. Blake, F. Dexter, and J. Donald, “Operating room manager’s use of integer programming for assigning block time to surgical groups: A case study,” *Anesthesia and Analgesia*, vol. 94, pp. 143–148, 2002.

[12] A. Caprara, M. Monaci, and P. Toth, “Models and algorithms for a staff scheduling problem,” *Mathematical Programming*, vol. 98, pp. 445–476, 2003.

[13] B. Cheang, H. Li, A. Lim, and B. Rodrigues, “Nurse rostering problems - A bibliographic survey,” *European Journal of Operational Research*, vol. 151, pp. 447–460, 2003.

[14] F. Dexter and A. Macario, “Changing allocations of operating room time from a system based on historical utilization to one where the aim is to schedule as many surgical cases as possible,” *Anesthesia and Analgesia*, vol. 94, pp. 1272–1279, 2002.

[15] A. Guinet and S. Chaabane, “Operating theatre planning,” *Int. J. Production Economics*, vol. 85, pp. 69–81, 2003.

[16] E. W. Hans, “Resource loading by branch-and-price techniques,” Ph.D. Dissertation, Twente University Press, Enschede, The Netherlands, 2001.

[17] W. L. Hughes and S. Y. Soliman, “Short-term case mix management with linear programming,” *Hospital and Health Services Administration*, vol. 30, pp. 52–60, 1985.

[18] B. Jaumard, F. Semet, and T. Vovor, “A generalized linear programming model for nurse scheduling,” *European Journal of Operational Research*, vol. 107, pp. 1–18, 1998.

[19] E. Litvak and M. C. Long, “Cost and quality under managed care: Irreconcilable differences?” *The American Journal of Managed Care*, vol. 6, pp. 305–312, 2000.

[20] A. J. Mason and M. C. Smith, “A nested column generator for solving rostering problems with integer programming,” in *International Conference on Optimisation: Techniques and Applications*, 1998, pp. 827–834.

[21] A. Mehrotra, K. E. Murphy, and M. A. Trick, “Optimal shift scheduling: A branch-and-price approach,” *Naval Research Logistics*, vol. 47, pp. 185–200, 2000.

[22] H. E. Miller, W. P. pierskalla, and G. J. Rath, “Nurse scheduling using mathematical programming,” *Operations Research*, vol. 24, pp. 857–870, 1976.

[23] D. M. Ryan and B. A. Foster, “An integer programming approach to scheduling,” in *Computer Scheduling of Public Transport Urban Passenger Vehicle and Crew Scheduling*, A. Weren, Ed. North-Holland, Amsterdam, 1981, pp. 269–280.

[24] USDHHS, *Projected supply, demand and shortages of registered nurses: 2000-2020*. National Center for Health Work-

force Analysis. US Department of health and Human Services, Rockville, MD, 2002.

[25] M. Van den Akker, H. Hoogeveen, and S. L. van de Velde, "Combining column generation and lagrangian relaxation to solve a single-machine common due date problem," *INFORMS Journal on Computing*, vol. 14, pp. 37–51, 2002.

[26] F. Vanderbeck, "On Dantzig-Wolfe decomposition in integer programming and ways to perform branching in a branch-and-price algorithm," *Operations Research*, vol. 48, pp. 111–128, 2000.

[27] F. Vanderbeck and L. A. Wolsey, "An exact algorithm for IP column generation," *Operations Research Letters*, vol. 19, pp. 151–159, 1996.

[28] D. M. Warner, "Scheduling nursing personnel according to nursing preferences: A mathematical programming approach," *Operations Research*, vol. 24, pp. 842–856, 1976.

[29] E. N. Weiss, "Models for determining estimated start times and case orderings in hospital operating rooms," *IIE Transactions*, vol. 22, pp. 143–150, 1990.