Les Cahiers du GERAD ISSN: 0711–2440

An online stochastic algorithm for a dynamic nurse scheduling problem

A. Legrain, J. Omer,
S. Rosat

G–2016–78

Octobre 2016
Revised: October 2017

GERAD HEC Montréal
3000, chemin de la Côte-Sainte-Catherine
Montréal (Québec) Canada H3T 2A7

Tél. : 514 340-6053
Téléc. : 514 340-5665
info@gerad.ca
www.gerad.ca

A sampling method based on column generation for the nurse scheduling problem under uncertainty

Antoine Legrain [a,b]

Jérémy Omer [c,d]

Samuel Rosat [b,d]

[a] *CIRRELT, Université de Montréal Montréal (Québec), Canada, H3C 3J7*

[b] *Department of Mathematics and Industrial Engineering, Polytechnique Montréal (Québec) Canada, H3C 3A7*

[c] *INSA, 35708 Rennes, France*

[d] *GERAD, HEC Montréal, Montréal (Québec), Canada, H3T 2A7*

antoine.legrain@polymtl.ca
jeremy.omer@insa-rennes.fr
samuel.rosat@polymtl.ca

Octobre 2016
Revised: October 2017
Les Cahiers du GERAD
G–2016–78

Abstract: In this paper, we focus on the problem that has been described in the second international nurse rostering competition: a personalized nurse scheduling problem under uncertainty. The schedules must be computed week by week over a planning horizon of one or two months depending on the instances. We present the work that was submitted to this competition and which was awarded the second prize.

At each week, a dynamic algorithm is fed with the staffing demand and nurse preferences for the week, and it computes an irrevocable weekly schedule for all nurses without knowledge on future inputs. The challenge is to obtain a feasible and near-optimal schedule at the end of the horizon.

The proposed online stochastic algorithm draws inspiration from the primal-dual algorithm for online optimization and the sample average approximation, and it builds upon an existing static nurse scheduling software. The procedure generates a small set of candidate schedules, evaluates each of them on few scenarios, and keeps the one with the best evaluation. Numerical results show that this algorithm is very robust, since it has been able to produce feasible and near optimal solutions on most of the proposed instances ranging from 30 to 120 nurses over a horizon of 4 or 8 weeks. Finally, the code of our implementation is open source and shared on a public repository.

Keywords: Stochastic programming, nurse rostering, dynamic problem, sample average approximation, primal-dual algorithm, scheduling

1 Introduction

Western countries are all facing a crisis in the nursing services that is mainly due to the overall aging of the population. In the United Kingdom, nurses could go on strike for the first time. Nagesh [15] says that "It's a message to all parties that the crisis in nursing recruitment must be put center stage in this election". In the United-States, "Inadequate staffing is a nationwide problem, and with the exception of California, not a single state sets a minimum standard for hospital-wide nurse-to-patient ratios." [20]. In this context, the rotation ratio of nurses is extremely high, and hospitals try to retain their nurses as much as possible. However, nurses tend to often change positions, because the work conditions are usually difficult and schedules are not convenient for the newly hired nurses (mostly due to seniority priority in the collective agreements). Consequently, providing high quality schedules for all the nurses is a major challenge for the hospitals that are also bound to provide expected levels of service.

The nurse scheduling problem (NSP) is thus a key problem of the healthcare management science that has been widely studied for more than two decades (refer to [5] for a literature review of the field). The NSP aims at building a schedule for the nurses over a certain period of times (typically two weeks or one month) while ensuring a certain level of service and respecting the collective agreements. However, in practice, nurses often know their needs for days-off no more than one week ahead of time. To be able to satisfy these preferences, managers try to update the schedules already computed. If managers were able to compute these schedules on a weekly basis while ensuring the respect of monthly constraints (e.g., the workload of full-time and part-time nurses), this would increase the quality of the schedules proposed to the nurses, and thus retain them longer in the service.

The second International Nurse Rostering Competition (INRC–II) [6] has presented a challenge that exactly proposed to solve the NSP in a dynamic fashion. The competition presents a problem with a wide variety of constraints that are close to the reality. In this paper, we present the work we submitted to the competition and which was awarded the second prize.

1.1 Literature review

Dynamic problems must be solved iteratively without full knowledge of the future. At each stage, new information is revealed and one needs to compute a new solution based on the solutions of the previous stages that are irrevocably fixed. The optimal solution of the problem is the same as that of the static (i.e. *offline*) counterpart, where all the information is known beforehand, and the challenge is to approach this solution even though information is revealed dynamically (i.e. *online*).

Four main techniques have been developed to do this: computing an offline policy (Markov decision processes [19] are mainly used), following a simple online policy (Online optimization [4] studies these algorithms), optimizing the current and future decisions (Stochastic optimization [3] handles the remaining uncertainty), or reoptimizing the system for each new stage (Online stochastic optimization [22] provides a general framework for designing these algorithms).

Markov decision processes decompose the problem into two different sets (states and actions) and two functions (transition and reward). An offline policy is then computed for each state, and will be applied at each stage depending on the current state. Such techniques are overwhelmed by the combinatorial explosion of problems such as the NSP, even if approximate dynamic programming [18] proposes ways to deal with the exponential growth of the size of the state space. This technique has been successfully applied to financial optimization [1], booking [17], and routing [16] problems. The main feature of this method is to invest all the computational effort beforehand, so this technique relies essentially on the probability model that infers the future events.

Online algorithms aim at solving problems where decisions are made in real-time, such as online advertisement, revenue management or online routing. As nearly no computation time is available, researchers have studied these algorithms to ensure a worse case or expected bound on the final solution. For instance, Buchbinder [4] proposes a primal-dual algorithm for a wide range of problems such as set covering, routing,

and resource allocation problems, which ensures a competitive-ratio (i.e., a bound on the worst-case scenario) for each of these applications. Although these techniques can solve very large models, they cannot solve rich scheduling problems in their own as they do not provide the tools for handling complex constraints.

Stochastic optimization [3] is a wide field that tackles many optimization problems from the scheduling of operating rooms [7] to the optimization of electricity production [9]. This field studies the minimization of a statistical function such as the expected value assuming that the probability distribution of the uncertain data is given. In this framework, it is usual to consider multi-stage problems with recourse, where first-level decisions must be taken right away, and recourse actions can be executed when uncertain data is revealed. The value of the recourse function is often approximated with cuts that are dynamically computed from the dual solutions of some subproblems obtained with Benders' decomposition. However, for combinatorial problems, these Benders-based decomposition methods converge slowly, because the dual solutions do not always contain the right information. Thus, the solution of such problems require an important amount of computational time that is not always available. To overcome this difficulty, the sample average approximation (SAA) [10] proposes a scheme to approximate roughly the uncertainty with a small set of scenarios during the solution, and to evaluate the solution accurately by considering a larger number of scenarios.

Finally, online stochastic optimization [22] is a framework oriented towards the solution of real problems from industry. The main idea is to decompose the solution in three steps: sampling scenarios of the future, solving each one of them, and finally computing the decisions of the current stage based on the solution of each scenario. Such techniques have been successfully applied to solve large scale problems as on-demand transportation system design [2] or online scheduling of radiotherapy centers [12]. Their main strength is that any algorithm can be used to solve the scenarios.

1.2 Contributions

The INRC–II challenges the candidates to compute a weekly schedule in a very limited computational time (less than 5 minutes), with a wide variety of rich constraints, and with important correlations between the weeks. Due to the important complexity of this dynamic NSP, none of the tools presented in the literature review allows to solve this problem by themselves.

We describe an online stochastic algorithm that draws inspiration from the primal-dual algorithms and the SAA. Each of these techniques answer to one of the key features of the competition:

- The online stochastic algorithm offers a framework to solve rich combinatorial problems;
- The primal-dual algorithm speeds up the solution by inferring quickly the impact of some decisions;
- The SAA handles efficiently the important correlations between weeks without increasing tremendously the computational time.

Finally, the algorithm is based on a free and open-source software that has been developed during the competition to solve the static version of the NSP. It has already been described in [14], and is summarized in Section 3.

We emphasize that the algorithm described in this article has been developed in a time-constrained environment, thus forcing the authors to balance their efforts between the different parts of the software. The resulting code is publicly shared on a public Git repository [13] for reproduction of the results, future comparisons, improvements and extensions. The remainder of the article is organized as follows. In Section 2, we give a detailed description of the NSP as well as the dynamic features of the competition. In Section 3, we state a static formulation and summarize the main features of the algorithm we developed to solve it. In Section 4, we present the dynamic formulation of the NSP, the design of the algorithm, and the articulation of all its components. In Section 5, we give some details on the implementation of the algorithm, discuss our results during the competition, and compare them to those obtained by the other finalist teams. Our concluding remarks appear in Section 6.

2 The Nurse Scheduling Problem

The formulation of the NSP that we consider is the one proposed by Ceschia et al. [6] in the INRC–II, so the description that we recall here is similar to the one given by the organizers. The constraints and the objective of the scheduling problem are first presented, and then we discuss the challenges associated to the uncertainty introduced in the problem.

The NSP aims at computing the schedule of a group of nurses over a given horizon while respecting a large set of soft and hard constraints. The soft constraints may be violated at the expense of a penalty in the objective, whereas hard constraints cannot be violated in a feasible solution. The dynamic version of the problem considers that the planning horizon is divided into one week-long stages and that the demand for nurses at each stage is known only at the beginning of the stage. Therefore, the schedule of each stage must be computed individually without full knowledge of the future demand.

A schedule for a nurse is decomposed into work and rest periods and the complete schedules of all the nurses must satisfy the set of constraints presented in Table 1. Each nurse can perform different skills (e.g., Head nurse, Nurse) and each day is divided into shifts (e.g., Day, Night). Furthermore, each nurse has signed a contract with their employers that determines their work status (e.g., Full-time, Part-time) and consequently some work agreements on the number of days and weekends worked within a month, and the minimum and maximum duration of work and rest periods. For the sake of nurses' health and personal life and to ensure a sufficient level of awareness, some successions of shifts are forbidden. For instance, a night shift cannot be followed by a day shift without at least one resting day. The employers also need to ensure a certain quality of service by scheduling a given number of nurses with the right skills for each shift and day. Finally, the length of the schedules (i.e., the planning horizon) can be one or two months (four or eight weeks to be more accurate).

Table 1: Constraints handled by the software.

Hard constraints	
H1	**Single assignment per day**: A nurse can be assigned at most one shift per day.
H2	**Under-staffing**: The number of nurses performing a skill on a shift must be at least equal to the minimum demand for this shift.
H3	**Shift type successions**: A nurse cannot work certain successions of shifts on two consecutive days.
H4	**Missing required skill**: A nurse can only cover the demand of a skill that he/she can perform.
Soft constraints	
S1	**Insufficient staffing for optimal coverage**: The number of nurses performing a skill on a shift must be at least equal to an optimal demand. Each missing nurse is penalized according to a unit weight but extra nurses above the optimal value are not considered in the cost.
S2	**Consecutive assignments**: For each nurse, the number of consecutive assignments should be within a certain range and the number of consecutive assignments to the same shift should also be within another certain range. Each extra or missing assignment is penalized by a unit weight.
S3	**Consecutive resting days**: For each nurse, the number of consecutive resting days should be within a certain range. Each extra or missing resting day is penalized by a unit weight.
S4	**Preferences**: Each assignment of a nurse to an undesired shift is penalized by a unit weight.
S5	**Complete week-end**: A given subset of nurses must work both days of the week-end or none of them. If he/she works only one of the two days Saturday or Sunday, it is penalized by a unit weight.
S6	**Total assignments**: For each nurse, the total number of assignments (worked days) must be within a given range. The difference (in either direction), multiplied by a unit weight, is added to the objective function.
S7	**Total working week-ends**: For each nurse, the number of week-ends with at least one assignment must be less than or equal to a given limit. The number of worked week-ends over that limit multiplied by a unit weight is added to the objective function.

We describe the four hard and the seven soft constraints in Table 1. The hard constraints are typical for workforce scheduling problems (especially the first two). These are the minimum sets of constraints that one must consider solving a scheduling problem: ensuring one assignment by day to each worker, fulfilling a given demand in terms of number of employees, forbidding some shift successions that cannot be performed by an employee for health and security reasons, and ensuring a required level of qualification of the workers. Moreover, several soft constraints should also be respected as much as possible to maximize the quality of service and retain the nurses within the unit. This last part is especially important in the current context

where most developed countries are facing important shortages in terms of nurses. The quality of the schedules (alternation of work and rest periods, numbers of worked days and weekends, respect of nurses' preferences) are primordial in order to keep the most qualified employees. These specificities make the NSP one of the most difficult workforce scheduling problem in the literature, because a personalized roster needs to be computed for each nurse. Furthermore, the main feature of this benchmark is that most constraints are soft. This eases the search for a feasible solution but it makes the pursuit of optimality more difficult.

The goal of the dynamic NSP is to sequentially build weekly schedules so as to minimize the total cost of the schedule and ensure its feasibility over the complete planning horizon. The main difficulty is to reach a feasible (i.e., managing the global hard constraints **H3**) and near-optimal (i.e., managing the global soft constraints **S6** – **S7** as well as consecutive constraints **S2** – **S3**) schedule without knowing the future demands and nurses' preferences. Indeed, the hard constraints **H1**, **H2**, and **H4** deal with local features that do not impact the following days. Each of these constraints concern either only one day (i.e., one assignment per day **H1**) or even one shift (i.e., the demand for a shift **H2** and the requirement that a nurse must possess the required skill **H4**). In the same way, soft constraints **S1**, and **S4** – **S5** are included in the objective with local costs that either depend on one shift, day or weekend. To summarize, the proposed algorithm must handle global requirements and the border effects between weeks that are induced by the dynamic process. These effects are propagated to the following week/stage through either the initial state or the number of worked days and weekends of the current stage.

3 The static nurse scheduling problem

The solution of this problem is not the focus of the paper as we will build upon the rotation-based algorithm that we describe in [14] for the computation of offline nurses' schedules over a given time horizon. This software solves the NSP with a branch-and-price algorithm [8]. The main idea is to generate a *roster* for each nurse, i.e., a sequence of work and rest periods covering the planning horizon. Each individual roster satisfies constraints **H1**, **H3** and **H4**, and the rosters of all the nurses satisfy **H2**. A *rotation* is a list of shifts from the roster that are performed on consecutive days, and preceded and followed by a resting day. Moreover, a rotation does **not** contain any information about the skills performed on its shifts. A rotation is called *feasible* (or *legal*) if it respects the single assignment and succession constraints **H1** and **H3**. A roster is therefore a sequence of rotations, separated by nonempty rest periods, to which skills are added (see Example 1).

Example 1 *Consider the following single-week roster:*

Day	*0*	*1*	*2*	*3*	*4*	*5*	*6*
Shift	*Night*	*Night*	*Rest*	*Rest*	*Early*	*Day*	*Rest*
Skill	*N*	*N*	-	-	*HN*	*N*	-

,

where N stands for nurse and HN for head nurse. The rotations of that roster, highlighted on the table above, are $\{(0, Night), (1, Night)\}$ *and* $\{(4, Early), (5, Day)\}$.

The static model presented in [14] is based on the enumeration of possible rotations by column generation. As is standard for column generation algorithms, a restricted master problem is solved to find the best fractional roster using a small set of rotations, and a subproblem allows to find rotations that could be added to improve the current solution or prove optimality. These subproblems are shortest path problems with resource constraints whose networks are described in detail in [14]. To get an integer solution, this process is embedded within a branch-and-bound procedure. The rest of the section focuses on the master problem, because its description will be useful for the presentation of the online algorithm. As a consequence, we will assume that every rotation of every nurse is available, which conceals the role of the subproblem. It is also worth mentioning that the software is based only on open-source libraries from the COIN-OR project: the BCP framework for branch-cut and price and the linear solver CLP.

We consider a set $\mathcal{N}$ of nurses over a planning horizon of M weeks (or $K = 7M$ days). The sets of all shifts and skills are respectively denoted Σ and $\mathcal{S}$. The nurse's type corresponds to the set of skills he or she

can use. For instance, most head nurses can fill head nurses demands, but they can also fill nurses demands in most cases. All the nurses of type $t \in \mathcal{T}$ (e.g., Nurse or Head nurse) are gathered within the subset $\mathcal{N}_t$. For the sake of readability, indices are standardized in the following way: nurses are denoted as $i \in \mathcal{N}$, weeks as $m \in \{1 \dots M\}$, days as $k \in \{1 \dots K\}$, shifts as $s \in \mathcal{S}$ and skills as $\sigma \in \Sigma$. We use (k, s) to denote the shift s of day k. All other data is summarized in Table 2.

Table 2: Summary of the input data.

Nurses	
L_i^-, L_i^+	min/max total number of worked days over the planning horizon for nurse i
CR_i^-, CR_i^+	min/max number of consecutive days-off for nurse i
B_i	max number of worked week-ends over the planning horizon for nurse i
Demand	
D_σ^{sk}	min demand in nurses performing skill σ on shift (k, s)
O_σ^{sk}	optimal demand in nurses performing skill σ on shift (k, s)
Initial state	
CD_i^0	initial number of ongoing consecutive worked days for nurse i
CS_i^0	initial number of ongoing consecutive worked days on the same shift for nurse i
s_i^0	shift worked on the last day before the planning horizon for nurse i
CR_i^0	initial number of ongoing consecutive resting days for nurse i

Remark 1 (Initial state) *Obviously, if $CR_i^0 > 0$, then $CD_i^0 = CS_i^0 = 0$, and vice-versa, because the nurse was either working or resting on the last day before the planning horizon. Moreover, s_i^0 only matters if the nurse was working on that day. The total number of worked days and worked week-ends of a nurse is set at zero (0) at the beginning of the planning horizon.*

The master problem (1) must assign a subset of rotations to each nurse while ensuring at the same time that the rotations are compatible and the demand is filled. The cost function is shaped by the penalties of the soft constraints as no other cost is taken into account in the problem proposed by the competition. For any soft constraint **SX**, its associated unit weight in the objective function is denoted as c_X.

Let $\mathcal{R}_i$ be the set of all feasible rotations for nurse i. The rotation j of nurse i has a cost c_{ij} (i.e., the sum of the soft penalties **S2**, **S4** and **S5**) and is described by the following parameters: a_{ij}^{sk}, a_{ij}^{k}, and b_{ij}^{m} which are equal to 1 if nurse i works respectively on shift (k, s), on day k, and weekend m, and 0 otherwise. Finally, f_{ij}^- and f_{ij}^+ represent the first and last worked days of this rotation.

Let x_{ij} be a binary decision variable which is equals to 1 if rotation j is part of the schedule of nurse i. The binary variables r_{ikl} and r_{ik} measure if constraint **S3** is violated: they are respectively equal to 1 if nurse i has a rest period from day k to $l-1$ including at most CR_i^+ consecutive days (cost: c_3^{ikl}), and if nurse i rests on day k and has already rested for at least CR_i^+ consecutive days before k. The integer variables w_i^+ and w_i^- count the number of days worked respectively above L_i^+ and below L_i^- by nurse i. The integer variable v_i also counts the number of weekends worked above B_i by nurse i. Finally, the integer variables n_σ^{sk}, $n_{t\sigma}^{sk}$, and z_σ^{sk} measures respectively the number of nurses performing skill σ, the number of nurses of type t performing skill σ, and the number of missing nurses performing skill σ on shift (k, s).

$$\min \sum_{i \in \mathcal{N}} \Bigg[\underbrace{\sum_{j \in \mathcal{R}_i} c_{ij} x_{ij}}_{\textbf{S2},\textbf{S4},\textbf{S5}} + \underbrace{\sum_{k=1}^{CR_i^+} \left(c_3 r_{ik} + \sum_{l=k+1}^{\min(K+1, k+CR_i^+)} c_3^{ikl} r_{ikl} \right)}_{\textbf{S3}} + \underbrace{c_6 (w_i^+ + w_i^-)}_{\textbf{S6}} + \underbrace{c_7 v_i}_{\textbf{S7}} \Bigg]$$
$$+ \underbrace{c_1 \sum_{k=1}^{K} \sum_{s \in \mathcal{S}} \sum_{\sigma \in \Sigma} z_\sigma^{sk}}_{\textbf{S1}} \tag{1a}$$

subject to:

$$[\mathbf{H1},\mathbf{H3}]: \quad \sum_{l=k+1}^{\min(K+1,k+CR_i^+)} r_{ikl} - \sum_{j\in\mathcal{R}_i: f_{ij}^+=k-1} x_{ij} = 0, \qquad \forall i\in\mathcal{N}, \forall k=2\dots K \tag{1b}$$

$$[\mathbf{H1},\mathbf{H3}]: \quad r_{ik} - r_{i(k-1)} + \sum_{j\in\mathcal{R}_i: f_{ij}^-=k} x_{ij} - \sum_{l=\max(1,k-CR_i^+)}^{k-1} r_{ilk} = 0, \qquad \forall i\in\mathcal{N}, \forall k=2\dots K \tag{1c}$$

$$[\mathbf{H1},\mathbf{H3}]: \quad \sum_{l=\max(1,K+1-CR_i^+)}^{K} r_{ilK} + r_{iK} + \sum_{j: f_{ij}^+=K} x_{ij} = 1, \qquad \forall i\in\mathcal{N} \tag{1d}$$

$$[\mathbf{S6}]: \quad \sum_{j\in\mathcal{R}_i}\sum_{k=1}^{K} a_{ij}^k x_{ij} + w_i^- \geq L_i^-, \qquad \forall i\in\mathcal{N} \tag{1e}$$

$$[\mathbf{S6}]: \quad \sum_{j\in\mathcal{R}_i}\sum_{k=1}^{K} a_{ij}^k x_{ij} - w_i^+ \leq L_i^+, \qquad \forall i\in\mathcal{N} \tag{1f}$$

$$[\mathbf{S7}]: \quad \sum_{j\in\mathcal{R}_i}\sum_{m=1}^{M} b_{ij}^m x_{ij} - v_i \leq B_i, \qquad \forall i\in\mathcal{N} \tag{1g}$$

$$[\mathbf{H2}]: \quad \sum_{t\in\mathcal{T}_\sigma} n_{t\sigma}^{sk} \geq D_\sigma^{sk}, \qquad \forall s\in\mathcal{S}, k\in\{1\dots K\}, \sigma\in\Sigma \tag{1h}$$

$$[\mathbf{S1}]: \quad \sum_{t\in\mathcal{T}_\sigma} n_{t\sigma}^{sk} + z_\sigma^{sk} \geq O_\sigma^{sk}, \qquad \forall s\in\mathcal{S}, k\in\{1\dots K\}, \sigma\in\Sigma \tag{1i}$$

$$[\mathbf{H4}]: \quad \sum_{i\in\mathcal{N}_t, j} a_{ij}^{sk} x_{ij} - \sum_{\sigma\in\Sigma_t} n_{t\sigma}^{sk} = 0, \qquad \forall s\in\mathcal{S}, k\in\{1\dots K\}, \sigma\in\Sigma \tag{1j}$$

$$x_{ij}\in\mathbb{N}, z_\sigma^{sk}, n_{t\sigma}^{sk}\in\mathbb{R}, \qquad \forall i\in\mathcal{N}, j\in\mathcal{R}_i, s\in\mathcal{S}, k\in\{1\dots K\}, t\in\mathcal{T}, \sigma\in\Sigma \tag{1k}$$

$$r_{ikl}, r_{ik}, w_i^+, w_i^-, v_i \geq 0, \qquad \forall i\in\mathcal{N}, k\in\{1\dots K\}, l=k+1\dots\min(K+1,k+CR_i^+) \tag{1l}$$

where Σ_t is the set of skills mastered by a nurse of type t (e.g., head nurses have the skills head nurse and nurse), and $\mathcal{T}_\sigma$ is the set of nurse type that masters skill σ (e.g., head nurse skill can be only provided by head nurses).

The objective function (1a) is composed of 5 parts: the cost of the chosen rotations in terms of consecutive assignments and preferences (**S2**, **S4**, **S5**), the minimum and maximum consecutive resting days violations (**S3**), the total number of working days violation (**S6**), the total number of worked week-ends violation (**S7**), and the insufficient staff for optimal coverage (**S1**). Constraints (1b)–(1d) are the flow constraints of the rostering graph (presented in Figure 1) of each nurse $i\in\mathcal{N}$. Constraints (1e) and (1f) measure the distance between the number of worked days and the authorized number of assignments: the variable w_i^+ represents the number of missing days when the minimum number of assignments, L_i^-, is not reached, and w_i^- is the number of assignments over the maximum allowed when the total number of assignments exceeds L_i^+. Constraints (1g) measure the number of weekends worked exceeding the maximum B_i. Constraints (1h) ensure that enough nurses with the right skill are working on each shift to meet the minimal demand. Constraints (1i) measure the number of missing nurses to reach the optimal demand. Constraints (1j) ensure a valid allocation of the skills among nurses of a same type for each shift. Constraints (1k) and (1l) ensure the integrality and the nonnegativity of the decision variables.

A valid sequence of rotations and rest periods can also be represented in a rostering graph whose arcs correspond to rotations and rest periods and whose vertices correspond to the starting days of these rotations and rest periods. Figure 1 shows an illustration of a rostering graph for some nurse i, and highlights the border effects. Nurse i has been resting for one day in her/his initial state, so the binary variable r_{i14} has a

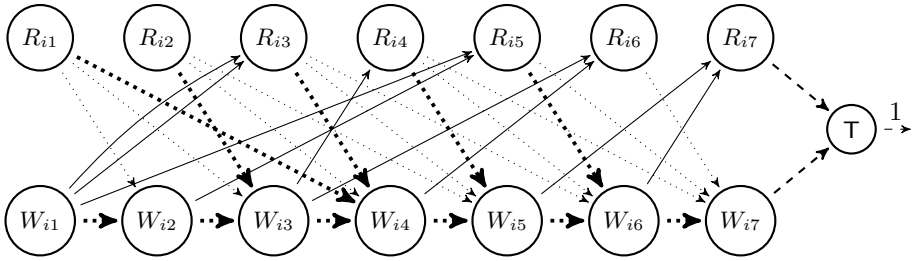

Figure 1: Example of a rostering graph for nurse $i \in \mathcal{N}$ over a horizon of $K = 7$ days, where the minimum and maximum number of consecutive resting days are respectively $CR_i^- = 2$ and $CR_i^+ = 3$, and the initial number of consecutive resting days is $CR_i^0 = 1$. The rotation arcs (x_{ij}) are the plain arrows, the rest arcs (r_{ikl} and r_{ik}) are the dotted arcs, and the artificial flow arcs are the dashed arrows. The bold rest arcs have a cost c_3 of and the others are free.

cost c_3 instead of zero, but the binary variable r_{i67} has a zero cost, because nurse i could continue to rest on the first days of the following week. If variable r_{i67} is set to one, nurse i will then start the following with one resting day as initial state. Finally, if nurse i was working in her/his initial state, the penalties associated to this border effect would be included in the cost of either the first rotation if the nurse continues to work, or the first resting arcs r_{i1k} if the nurse starts to rest.

4 Handling the uncertain demand

The main purpose of this section is to present the dynamic model used for the NSP, and then show how we can design an efficient algorithm to compute near-optimal schedules in a very limited amount of computational time. We propose a dynamic math-heuristic based on a primal-dual algorithm [4] and wrapped into a sample average approximation (SAA) [11]. As previously stated, the dynamic algorithm should especially take into account the global hard (i.e. **H3**) and soft (i.e. **S6** and **S7**) constraints to reach a feasible and near-optimal global solution.

4.1 The dynamic NSP

As the proposed dynamic algorithm must handle the border effects between weeks, we propose to reformulate Formulation (1) to gather decisions of the same week, abstract most of the individual constraints, and highlight these challenges. The resulting weekly Formulation (2) clusters together all the local constraints for all nurses in a weekly schedule j for each week, enumerates all the possible schedules, and models only the border effects. This formulation is not solved in practice, but we use it to present and build the proposed online stochastic algorithm.

The binary variable y_j^m equals 1 if schedule j is chosen for week m. As in a rotation, a global weekly schedule $j \in \mathcal{R}$ is described by a weekly cost c_j and the parameters a_{ij} and b_{ij} that respectively count the number of days and weekends worked by nurse i. The variables w_i^+, w_i^-, and v_i are defined as in Formulation (1).

$$\min \underbrace{\sum_{j\in\mathcal{R}}\sum_{m=1}^{M} c_j y_j^m}_{\mathbf{S1-S5}} + \underbrace{c_6 \sum_{i\in\mathcal{N}} (w_i^+ + w_i^-)}_{\mathbf{S6}} + \underbrace{c_7 \sum_{i\in\mathcal{N}} v_i}_{\mathbf{S7}} \tag{2a}$$

subject to:

$$[\mathbf{H1}-\mathbf{H4}, \mathbf{S1}-\mathbf{S5}]: \quad \sum_{j\in\mathcal{R}} y_j^m = 1, \quad \forall m \in \{1\dots M\} \qquad [\alpha^m] \tag{2b}$$

$$[\mathbf{H3}, \mathbf{S2}, \mathbf{S3}]: \quad \sum_{j' \in \mathcal{C}_j} y_{j'}^{m+1} \geq y_j^m, \quad \forall j \in \mathcal{N}, m = 1 \dots M-1 \qquad [\delta_j^m] \qquad (2c)$$

$$[\mathbf{S6}]: \quad \sum_{j \in \mathcal{R}} \sum_{m=1}^{M} a_{ij} y_j^m + w_i^- \geq L_i^-, \quad \forall i \in \mathcal{N} \qquad [\beta_i^-] \qquad (2d)$$

$$[\mathbf{S6}]: \quad \sum_{j \in \mathcal{R}} \sum_{m=1}^{M} a_{ij} y_j^m - w_i^+ \leq L_i^+, \quad \forall i \in \mathcal{N} \qquad [\beta_i^+] \qquad (2e)$$

$$[\mathbf{S7}]: \quad \sum_{j \in \mathcal{R}} \sum_{m=1}^{M} b_{ij} y_j^m - v_i \leq B_i, \quad \forall i \in \mathcal{N} \qquad [\gamma_i] \qquad (2f)$$

$$y_j^m \in \{0,1\}, \quad \forall j \in \mathcal{R}, m \in \{1 \dots M\} \qquad (2g)$$

$$w_i^+, w_i^-, v_i \geq 0, \quad \forall i \in \mathcal{N} \qquad (2h)$$

The objective (2a) is decomposed into a weekly cost for the schedule and global penalties. Constraints (2b) ensure that exactly one schedule is chosen for each week. Constraints (2c) hide the succession constraints by summarizing them into a filtering constraint between consecutive schedules. These constraints simplify the resulting formulation, but will not be used in practice as their number is not tractable (see below). Constraints (2d)–(2f) measure the penalties associated with the number of worked days and weekends. Constraints (2g)–(2h) are respectively integrality and nonnegativity constraints. The greek letters indicated between brackets (α, β, δ and γ) denote the dual variables associated with these constraints.

Constraints (2c) model the sequential aspect of the problem. Indeed, this formulation is solved stage by stage in practice, and thus the solution of stage m is fixed when solving the next stage $m+1$. This means that when computing the schedule of week $m+1$, the binary variables y_j^m are all null except one $y_{j_m}^m$ that corresponds to the schedule chosen for week m. All the constraints (2c) corresponding to $y_j^m = 0$ can be removed, and only one is kept: $\sum_{j' \in \mathcal{C}_{j_m}} y_{j'}^{m+1} \geq 1$. In this constraint, the set $\mathcal{C}_{j_m}$ contains all the schedules compatible with j_m, i.e., the schedules that are feasible and correctly priced when schedule j_m is used for setting the initial state of stage $m+1$. To summarize, constraints (2c) can be seen as filtering constraints that hide the difficulties associated to the border effects induced by the constraints **H3**, **S2**, and **S3**.

The main challenge of the dynamic NSP is to correctly handle Constraints (2c)–(2h) to maximize the chance of building a feasible and near-optimal solution at the end of the horizon. The proposed dynamic procedure for generating and evaluating the computed schedules at each stage draws inspiration from the SAA. Algorithm 1 summarizes the whole iterative process over all the stages. Each generated schedule is evaluated

Algorithm 1: A sample average approximation based algorithm

```
for each stage m = 1 ... M − 1 do
    Initialize the set of candidate schedules of stage m: S^m = ∅
    Initialize the generation algorithm with the chosen schedule j_{m−1} of the previous stage m − 1 (i.e., set the initial state)
    Sample a set Ω^m of future demands for the evaluation
    while there is enough computational time do
        Generate a weekly schedule j for stage m
        Initialize the evaluation algorithm with schedule j (i.e., set the initial state)
        for each scenario ω ∈ Ω^m do
            Evaluate schedule j over scenario ω
        end for
        Store the schedule (S^m := S^m ∪ {j}) and its score (e.g., its average evaluation cost)
    end while
    Choose the schedule j_m ∈ S^m with the best score
end for
Compute the best schedule for the last stage M with the given computational time
```

before generating another new one, because a schedule without evaluation is worthless. This generation-evaluation step is repeated until the time limit is reached. Note that the last stage M is solved by an offline algorithm (e.g., the software presented in Section 3), because the demand is totally known at this time.

The two following subsections describe each one of the main steps:

1. The generation of a schedule with an offline procedure that takes into account a rough approximation of the uncertainty;
2. The evaluation of a schedule for a demand scenario that measures the impact on the remaining weeks. (This step also computes an evaluation score of a schedule based on the sampled scenarios.)

4.2 Generating a candidate schedule

In a first attempt to generate a schedule, a primal-dual algorithm inspired from [4] is proposed. However, this procedure doesn't handle all the correlations between the weekly schedules (i.e., Constraints (2c)). The proposed primal-dual algorithm is then adapted to better take into account the border effects between the weeks, and make use of every available insight on the following weeks.

4.2.1 A primal-dual algorithm

Primal-dual algorithms for online optimization aim at building pairs of primal and dual solutions dynamically. At each stage, some primal decisions are irrevocably made and the dual solution needs to be updated so as to remain feasible. The current dual solution drives the algorithm to better primal decisions by using those dual values as multipliers in a Lagrangian relaxation. The goal is to obtain a pair of feasible primal and dual solutions that satisfy the complementary slackness property at the end of the process.

We use a similar primal-dual algorithm to solve the online problem associated to Formulation (2). In this dynamic process, we wish to sequentially solve a restriction of Formulation (2) to week m for all stages $m \in \{1, \dots, M\}$ with a view to reaching an optimal solution of the complete formulation. This process raises an issue though: how can we take into account the constraints (4c)–(4e) in a restriction to a single week? To do this, the primal-dual algorithm uses dual information from stage m to compute the schedule of stage $m+1$ by solving the following Lagrangian relaxation of Formulation (2):

$$\min \quad \sum_{j \in \mathcal{R}} [\underbrace{c_j}_{\mathbf{S1},\mathbf{S2},\mathbf{S3},\mathbf{S4},\mathbf{S5}} + \underbrace{\sum_{i \in \mathcal{N}} (\hat{\beta}_i^+ - \hat{\beta}_i^-) a_{ij}}_{\mathbf{S6}} + \underbrace{\sum_{i \in \mathcal{N}} \hat{\gamma}_i b_{ij}}_{\mathbf{S7}}] y_j^{m+1} \tag{3a}$$

$$\text{s.t.:} \quad [\mathbf{H1} - \mathbf{H3}] : \sum_{j \in \mathcal{C}_{j_m}} y_j^{m+1} = 1 \tag{3b}$$

$$y_j^{m+1} \in \{0,1\}, \ \forall j \in \mathcal{R} \tag{3c}$$

where $\hat{\beta}_i^-, \hat{\beta}_i^+, \hat{\gamma}_i \geq 0$ are multipliers respectively associated with Constraints (2d)–(2f), and both Constraints (2b) and (2c) that guarantee the feasibility of the weekly schedules are aggregated under Constraint (3b). More specifically, any new assignment for nurse i will be penalized with $\hat{\beta}_i^- - \hat{\beta}_i^+$ and worked week-ends will cost an additional $\hat{\gamma}_i$. It is thus essential to set these multipliers to values that will drive the computation of weekly schedules towards efficient schedules over the complete horizon. For this, we consider the dual of the linear relaxation of Formulation (2):

$$\max \quad \sum_{m=1}^{M} \alpha^m + \sum_{i \in \mathcal{N}} (L_i^- \beta_i^- - L_i^+ \beta_i^+ - B_i \gamma_i) \tag{4a}$$

$$\text{s. t.:} \quad \alpha^m + \sum_{i \in \mathcal{N}} (a_{ij} \beta_i^- - a_{ij} \beta_i^+ - b_{ij} \gamma_i) - \delta_j^m + \sum_{j' \in \mathcal{C}_j^{-1}} \delta_{j'}^{m-1} \leq c_j, \quad \forall j \in \mathcal{R}, \forall m \qquad [y_j^m] \tag{4b}$$

$$\beta_i^- \leq c_6, \quad \forall i \in \mathcal{N} \qquad [w_i^-] \tag{4c}$$

$$\beta_i^+ \leq c_6, \quad \forall i \in \mathcal{N} \qquad [w_i^+] \tag{4d}$$

$$\gamma_i \leq c_7, \quad \forall i \in \mathcal{N} \qquad [v_i] \tag{4e}$$

$$\beta_i^+, \beta_i^-, \gamma_i, \delta_j^m \geq 0, \quad \forall j \in \mathcal{R}, m \in \{1 \dots M\}, \forall i \in \mathcal{N} \tag{4f}$$

where set $\mathcal{C}_j^{-1}$ contains all the schedules with which schedule j is compatible. The dual variables $\alpha^m, \delta_j^m, \beta_i^-, \beta_i^+$ and γ_i are respectively associated with Constraints (2b), (2c), (2d), (2e), and (2f), the variables δ_j^0 are set to zero to get a unified formulation. The variables in brackets denote the primal variables associated to these dual constraints. At each stage, the primal-dual algorithm sets the values of the multipliers so that they correspond to a feasible and locally optimal dual solution, and uses this solution as Lagrangian multipliers in Formulation (2). Another point of view is to consider the current primal solution at stage m as a basis of the simplex algorithm for the linear relaxation of Formulation (2). The resolution of stage $m+1$ corresponds to the creation of a new basis: Formulation (3) seeks a candidate pivot with a minimum reduced cost according to the associated dual solution.

In choosing the dual variables, a second goal (after seeking a feasible dual solution) is to satisfy complementary conditions between the current primal solution at stage m and the dual solution computed for stage $m+1$. In the computation of a dual solution, the variables α^m and δ_j^m do not need to be explicitly considered, because they will not be used in Formulation (3). What is more, focusing on stage m, the only dual constraints that involve α^m and δ_j^m (4b), can be satisfied for any value of $\hat{\beta}_i^-, \hat{\beta}_i^+$ and $\hat{\gamma}_i$ by setting $\delta_j^m = 0, \forall j \in \mathcal{R}$, and

$$\alpha^m = \min_{j \in \mathcal{R}} \{c_j - \sum_{i \in \mathcal{N}} (a_{ij}\hat{\beta}_i^- - a_{ij}\hat{\beta}_i^+ - b_{ij}\hat{\gamma}_i)\}.$$

Observe that the expression of the objective function of Formulation (2) ensures that the only schedule variable satisfying $y_{j_m}^m > 0$ will be such that $j_m \in \text{argmin}_{j \in \mathcal{R}} \{c_j - \sum_i (a_{ij}\beta_i^- - a_{ij}\beta_i^+ - b_{ij}\gamma_i)\}$, so complementarity is achieved.

To set the values of $\hat{\beta}_i^-, \hat{\beta}_i^+$ and $\hat{\gamma}_i$, we first observe that complementary conditions are satisfied if

$$\hat{\beta}_i^- = \begin{cases} c_6 & \text{if } \sum_{j,m} a_{ij} y_j^m < L_i^- \\ 0 & \text{otherwise} \end{cases} \quad \hat{\beta}_i^+ = \begin{cases} c_6 & \text{if } \sum_{j,m} a_{ij} y_j^m \geq L_i^+ \\ 0 & \text{otherwise} \end{cases} \quad \hat{\gamma}_i = \begin{cases} c_7 & \text{if } \sum_{j,m} b_{ij} y_j^m \geq B_i \\ 0 & \text{otherwise} \end{cases}$$

Since the history of the nurses are initialized with zero assignment and week-end worked, we initially set $\hat{\beta}_i^- = c_6$ and $\hat{\beta}_i^+ = \hat{\gamma}_i = 0$ to satisfy complementarity. We then perform linear updates at each stage m, using the characteristics of the schedule j_m chosen for the corresponding week:

$$\hat{\beta}_i^- = \max\left(0, \hat{\beta}_i^- - c_6 \frac{a_{ij_m}^m}{L_i^-}\right), \quad \hat{\beta}_i^+ = \min\left(c_6, \hat{\beta}_i^+ + c_6 \frac{a_{ij_m}^m}{L_i^+}\right), \quad \hat{\gamma}_i = \min\left(c_7, \hat{\gamma}_i + c_7 \frac{b_{ij_m}^m}{B_i}\right).$$

These updates do not maintain complementarity at each stage but allow for a more balanced penalization of the number of assignments and worked week-ends. What is more, the variations of $\hat{\beta}_i^-, \hat{\beta}_i^+$ and $\hat{\gamma}_i$ ensure that constraints (2b) remain feasible for the previous stage, even though complementarity may be lost.

In most theoretical descriptions of the online primal-dual algorithm, the authors use non-linear updates to be able to derive a competitive-ratio. However, no competitive-ratio is sought by this approach and linear updates are easier to design. Non-linear updates could be investigated in the future.

Algorithm 2 summarizes the primal-dual algorithm. It estimates the impact of a chosen schedule on the global soft constraints through their dual variables. As it is, it gives poor results in practice though. The reason is that the information obtained through the dual variables does not describe precisely the real problem. At the beginning of the algorithm, the value of the dual variables pushes the nurses to work as much as possible. Consequently, the nurses work too much at the beginning and cannot cover all the necessary shifts at the end of the horizon. Furthermore, the expected impact of the filtering Constraints (2c) are totally ignored for the moment. Especially, the shift type succession constraints **H3** imply many feasibility issues at the border between two weeks as Formulation (2) is solved sequentially with this primal-dual algorithm. The following two sections describe how this initial implementation was adapted to cope with these issues.

4.2.2 Sampling a second week demand for feasibility issues

Preliminary results have shown that Algorithm 2 has some feasibility issues due to constraints **H3** on forbidden shift successions between the last day of one week and the first day of the following one. In other words, there should be some way to capture border effects during the computation of a weekly schedule.

Algorithm 2: Primal-dual algorithm

$\hat{\beta}_i^- = c_6, \hat{\beta}_i^+ = \hat{\gamma}_i = 0, \forall i \in \mathcal{N}$
for each stage m **do**
  Solve Formulation (3) with a deterministic algorithm
  Update $\hat{\beta}_i^- = \max\left(0, \hat{\beta}_i^- - c_6 \frac{a^m_{ij_m}}{L_i^-}\right), \forall i \in \mathcal{N}$
  Update $\hat{\beta}_i^+ = \min\left(c_6, \hat{\beta}_i^+ + c_6 \frac{a^m_{ij_m}}{L_i^+}\right), \forall i \in \mathcal{N}$
  Update $\hat{\gamma}_i = \min\left(c_7, \hat{\gamma}_i + c_7 \frac{b^m_{ij_m}}{B_i}\right), \forall i \in \mathcal{N}$
end for

Instead of solving each stage over one week, we solve Formulation (3) over two weeks and keep only the first week as a solution of the current stage. The compatibility constraints (2c) between the two weeks are now included in this two-weeks model. In this approach, the data about the first week is available since it corresponds to the week that must be scheduled at this stage, but there is no data about any future week. The demand relative to the second week is thus sampled as described in Section 4.4. By providing a scenario for a potential future demand, the second week schedule ensures that the first week schedule ends with assignments that are at least compatible in this scenario, thus increasing the probability of building a feasible schedule over the complete horizon.

What is more, for two different samples of the second week demand, the two-weeks version of Formulation (3) should lead to two different solutions for the first week. As a consequence, we can solve the model several times to generate different candidate schedules for the week under consideration. As described in Algorithm 1, we use this property to generate new candidates until time limit is reached.

4.2.3 Global bounds to reduce staff shortages

Preliminary results have also shown that Algorithm 2 creates many staff shortages in the last weeks. Our intent is thus to bound the number of assignment and worked weekends in the early stages to avoid the later shortages. The naive approach is to resize constraints (2d)–(2f) proportionally to the length of the demand considered in Formulation (3) (i.e., two weeks in our case). However, it can be desirable to allow for important variations in the number of assignments to a given nurse from one week to another, and even from one pair of weeks to another. Stated otherwise, it is not optimal to build a schedule that can only draw one or two weeks-long patterns as would be the case for less constrained environments. A simple illustration arises by considering the constraints on the maximum number of worked weekends. To comply with these constraints, there cannot be some nurses that always work on weekends and others that never do. Coupled with the other constraints, this results necessarily in complex and irregular schedules. Consequently, bounding the number of assignments individually would forbid valuable schedules.

Instead, we propose to bound the number of assignments and worked weekends for sets of similar nurses in order to both stabilize the total number of worked days within this set and allow irregularities in the individual schedules. We choose to cluster the nurses with the same contract, because they have the same minimum and maximum bounds for their soft constraints. Hence, for each stage m, we add one set of constraints similar to (2d)–(2f) for each contract. In the constraints associated with contract $\kappa \in \Gamma$, the left hand-sides are resized proportionally to the number of nurses with contract κ and the number of weeks in the demand horizon. Let L_κ^{m-}, L_κ^{m+}, and B_κ^m be respectively the minimum and maximum total number of assignments, and the maximum total number of worked weekends over the two-weeks demand horizon for the nurses with contract κ. We define these global bounds as:

- $L_\kappa^{m-} = 7 * \frac{2}{M-m+1} \sum_{i:\kappa_i=\kappa} \max(0, L_{\kappa_i}^- - \sum_{m'=1}^{m'<m} \sum_j a_{ij} y_j^{m'})$,
- $L_\kappa^{m+} = 7 * \frac{2}{M-m+1} \sum_{i:\kappa_i=\kappa} \max(0, L_{\kappa_i}^+ - \sum_{m'=1}^{m'<m} \sum_j a_{ij} y_j^{m'})$,
- $B_\kappa^m = \frac{2}{M-m+1} \sum_{i:\kappa_i=\kappa} \max(0, B_{\kappa_i} - \sum_{m'=1}^{m'<m} \sum_j b_{ij} y_j^{m'})$,

where κ_i is the contract of a nurse i.

Finally, the objective (3a) is modified to take into account the new slack variables $w_\kappa^{m-}, w_\kappa^{m+}, v_\kappa^m$ associated to the new soft constraints. The costs of these slack variables is set to make sure that violations of the soft constraints are not penalized more than once for an individual nurse. For instance, instead of counting the full cost c_6 for variable w_κ^{m+}, we compute its cost as $(c_6 - \max_{i|\kappa_i=\kappa}(\beta_i^+))$. This guarantees that an extra assignment is never penalized with more than c_6 for any individual nurse. The cost of the variables w_κ^{m+} and v_κ^m have been modified in the same way for analogous reasons.

Formulation (5) summarizes the final model used for the generation of the schedules. We recall that the variables y_j^m are now selecting a schedule j which covers a two weeks demand, and that this formulation is in fact solved by a branch-and-price algorithm that selects rotations instead of weekly schedules.

$$\min \quad \sum_{j\in\mathcal{R}} \left[c_j + \sum_{i\in\mathcal{N}} \Big((\beta_i^+ - \beta_i^-)a_{ij} + \gamma_i b_{ij}^m\Big)\right] y_j^m$$

$$+ \sum_{\kappa\in\Gamma} \Big[(c_6 - \max_{i:\kappa_i=\kappa}(\beta_i^-))w_\kappa^{m-} + (c_6 - \max_{i:\kappa_i=\kappa}(\beta_i^+))w_\kappa^{m+}) + (c_7 - \max_{i:\kappa_i=\kappa}(\gamma_i^m))v_\kappa^m)\Big] \tag{5a}$$

$$\text{s.t.: } [\mathbf{H1}, \mathbf{H2}, \mathbf{H3}, \mathbf{H3}]: \quad \sum_{j\in\mathcal{R}} y_j^m = 1, \tag{5b}$$

$$[\mathbf{S6}]: \quad \sum_{i:\kappa_i=\kappa} \sum_{j\in\mathcal{R}} a_{ij} y_j^m + w_\kappa^{m-} \geq L_\kappa^{m-}, \quad \forall \kappa \in \Gamma \tag{5c}$$

$$[\mathbf{S6}]: \quad \sum_{i:\kappa_i=\kappa} \sum_{j\in\mathcal{R}} a_{ij} y_j^m + w_\kappa^{m+} \leq L_\kappa^{m+}, \quad \forall \kappa \in \Gamma \tag{5d}$$

$$[\mathbf{S7}]: \quad \sum_{i:\kappa_i=\kappa} \sum_{j\in\mathcal{R}} b_{ij} y_j^m - v_\kappa^m \leq B_\kappa^m, \quad \forall \kappa \in \Gamma \tag{5e}$$

$$y_j^m \in \{0,1\}, \quad \forall j \in \mathcal{R} \tag{5f}$$

$$w_\kappa^{m-}, w_\kappa^{m+}, v_\kappa^m \geq 0, \quad \forall \kappa \in \Gamma \tag{5g}$$

To conclude, Formulation (5) allows to control the impact of a schedule on the future through two mechanisms: the problem is solved over two weeks to diminish the border effects that may lead to infeasibility, and the costs are modified to globally limit the penalties due to constraints **S6** and **S7**. Furthermore, this formulation can generate different schedules fort the first week by considering different samples for the second week demand.

4.3 Evaluating candidate schedules

In the spirit of the SAA, the first-week schedules generated by Formulation (5) are evaluated to be ranked. The evaluation should measure the expected impact of each schedule on the global solution (i.e. over M weeks). This impact can be measured by solving a NSP several times over different sampled demands for the remaining weeks.

Let Ω^m be the set of scenarios of future demands for weeks $m+1 \dots M$, and assume that a schedule j has been computed for week m. To evaluate schedule j, we wish to solve the NSP for each demand $\omega \in \Omega^m$ by using j to set the initial history of the NSP. Denoting $V_{j\omega}^m$ the value of the solution, we can infer that the future cost $c_{j\omega}^m$ of schedule j in scenario ω is equal to $c_j + V_{j\omega}^m$: the actual cost of the schedule plus the resulting cost for scenario ω. Then, a score that takes into account all the future costs $(c_{j\omega}^m)_{\omega\in\Omega^m}$ of a given schedule j is computed. Several functions have been tested and preliminary results have shown that the expected value was producing the best results. Finally, the schedule j_m with the best score (i.e., the expected future cost) is retained.

However, computing the value $V_{j\omega}^m$ raises two main issues. First, the NSP is an integer program for which it can be too time-consuming to even find a feasible solution. We thus use the linear relaxation of this problem to compute an estimation of the future cost. This simplification decreases drastically the computational

time, and it can also detect feasibility issues at the border between weeks m and $m+1$. The second issue is that over a long time horizon, even the linear relaxation of the NSP cannot be solved in sufficiently small computational time. We thus restrict the evaluation to scenarios of future demands that are at most two weeks long. More specifically, the scenarios are one week-long for the stage $M-1$ and two weeks long for the previous stages. We observed that this restriction allows to keep the solution time in a reasonable interval while giving a good measure of the impact of the schedule j on the future.

To summarize, the value $V_{j\omega}^m$ is computed by solving the linear relaxation of Formulation (1) for a two-week demand ω, and the initial state is set by using the schedule j. Finally, the parameters L_i^-, L_i^+, and B_i are proportionally resized over two weeks, as follows.

- $L_i^{(m+1)-} = \lfloor 7 * \frac{2}{M-m} \max(0, L_i^- - \sum_{m'=1}^{m} \sum_j a_{ij}^{m'} y_j^{m'}) \rfloor$;
- $L_i^{(m+1)+} = \lceil 7 * \frac{2}{M-m} \max(0, L_i^+ - \sum_{m'=1}^{m} \sum_j a_{ij}^{m'} y_j^{m'}) \rceil$;
- $B_i^{m+1} = \lceil \frac{2}{M-m} \max(0, B_i - \sum_{m'=1}^{m} \sum_j b_{ij}^{m'} y_j^{m'}) \rceil$.

As already stated, the number of evaluation scenario included in Ω^m is kept low (e.g., $|\Omega^m| = 5$) to meet the requirements in computational time. These scenarios are sampled as described in the next section.

4.4 Sampling of the scenarios

The competition data does not provide any knowledge about past demands, potential probability distributions of the demand, nor any other type of information that could help for sampling scenarios of demand. It is thus impossible to build complex and accurate prediction models for the future demand. At a given stage m, the algorithm has absolutely no knowledge about the future realizations of the demand, so the sampling can only be based on the current and past observations of the weekly demands on stages $1, \ldots, m$. To build scenarios of future demand, we simply perturb these observations with some noise that is uniformly distributed within a small range (typically one or two nurses) and mix randomly these observations (e.g., pick the Monday of one observation and the Tuesday from another one). The future preferences are not sampled in the scenarios, because they cannot lead to an infeasible solution, they do not induce border effects, and they have small costs when compared to the other soft constraints. The goal of the sampling method is only to obtain some diversity in the scenarios used to generate different candidate schedules and in those used to evaluate the candidate schedules. Assuming that the demands will not change dramatically from one week to another, this allows for additional robustness and efficiency in many situations.

4.5 Summary of the primal-dual-based sample average approximation

Algorithm 3 provides a detailed description of the overall algorithm we submitted to the INRC–II. It generates several schedules with a primal-dual algorithm and evaluates them over a set Ω^m of future demands. The evaluation step increases the probability of selecting a globally feasible schedule, that has already been feasible for several resolutions of the linear relaxation of Formulation (1). The performances of this algorithm are discussed in Section 5.

5 Experimentations

This section presents the results obtained at the INRC–II. The competition was organized in two rounds. In the selection round, each team had to submit their best results on a benchmark of 28 instances that were available to the participants before submitting the codes. The organizers then retained the best eight teams for the final round where they tested the algorithms against a new set of 60 instances. The algorithm described above ranked second in both rounds.

The instances used during each round are summarized in Tables 3 and 4. They mix relatively small instances (35 nurses over 4 weeks) to really big instances (120 nurses over 8 weeks) that are very difficult to solve even in a static setting [14]. The algorithms of the participants all had the same limited computational

Algorithm 3: A primal-dual-based sample average approximation

$\beta_i^- = c_6, \beta_i^+ = \gamma_i = 0, \forall i \in \mathcal{N}$
for each stage $m = 1 \dots M-1$ **do**
  Initialize the set of candidate schedules of stage m: $S^m = \emptyset$
  Initialize the generation model using the chosen schedule of the previous stage $m-1$ (i.e., set the initial state)
  Sample a set Ω^m of future demands for the evaluation
  while there is enough computational time **do**
    Sample a second week demand for the generation
    Solve Formulation (5) with a deterministic algorithm to build a two-weeks schedule (j^1, j^2)
    Store the schedule of the first week: $S^m := S^m \cup \{j^1\}$
    Initialize the evaluation model with schedule j^1 (i.e., set the initial state)
    for each demand $\omega \in \Omega^m$ **do**
      Compute the value $V_{j\omega}^m$ as the optimal value of the linear relaxation of Formulation (1) over two weeks if $m < M-1$, and one week otherwise
    end for
  end while
  Choose the schedule j_m with the best average evaluation cost
  Update the dual variables as in Algorithm 2
end for
Compute the best schedule for the last stage M with the given computational time

time to solve each stage (depending on the number of nurses and on the computer used for the tests). The solution obtained at each week was used as an initial state for the schedule of the following week. If an algorithm reached an infeasible schedule, the iterative process was stopped. The final rank of each team was computed as the average rank over all the instances.

Table 3: Instances used for the selection

		Number of nurses						
		30	40	50	60	80	100	120
Number of weeks	4	2	2	2	2	2	2	2
	8	2	2	2	2	2	2	2
Computational time per week (s)		35	60	90	120	150	240	300

Table 4: Instances used for the final

		Number of nurses		
		35	70	110
Number of weeks	4	10	10	10
	8	10	10	10
Computational time per week (s)		45	150	270

In this section, we will focus our discussion on the quality of the results obtained with our algorithm. More details about the competition and the final results can be found on the competition website: http://mobiz.vives.be/inrc2/.

5.1 Algorithm implementation

Algorithm 3 depends on how future demands are sampled, on the number of scenarios used for the evaluation, and last but not least, on the scheduling software.

The algorithm uses only five scenarios of future demands for the evaluation. Indeed, the algorithm must divide its short available computational time between the generation and the evaluation of the schedules. The first step aims at computing the best schedule according to the current demand while the second step seeks a robust planning that yields promising results for the following stages (high probability to remain feasible and near-optimal). In order to generate several schedules (at least two) in the granted computational time, the number of demand scenarios must remain small. Moreover, since the demand scenarios we generate are

not based on accurate data, but only on a learned distribution, there is no guarantee that a larger number of scenarios would provide a better evaluation. In fact, we tested different configurations (3 to 10 scenarios used for the evaluation), and they all gave approximately the same results (the best results were obtained for 4 to 6 scenarios).

The code is publicly shared on the Git repository [13]. The scheduling software is implemented in C++ and is based on the branch-cut-and-price framework of the COIN-OR project: BCP. The choice of this framework has been motivated by the competition requirement of using free and open-source libraries in the competition. The pricing problems are modeled by a shortest path with resource constraints and solved with the Boost library. The solution algorithm is not parallelized and it uses the branching strategy 'two dives' described in [14]. This strategy 'dives' two times in the branching tree and keeps the best feasible solution. If no solution is found after two dives (which was never the case), the algorithm continues until it reaches either the time limit or a feasible solution.

5.2 Selection instances

Figure 2 shows the results of the best three finalist teams on the 28 instances used for the selection. These results are presented as a cumulative distribution of their respective relative gap from the best solutions (e.g., a zero gap means that the algorithm has produced the best solution). The first team has obtained the best results by far: they outperform all the others teams, and have either the first or second position on all the instances (excepting one third position). Their algorithm is also based on a mixed integer programming approach that computes weekly schedules, but they directly model the problem using a large flow network where a state-expansion is used to consider the soft constraints. For the time being, only a brief description of the algorithm is available in [21]. Algorithm 3 obtains a fair second position and competes with the best algorithm, since it also ranks first or second on every instance but two, for which it ranks third. Finally, the third team is significantly behind the first two. As highlighted by Figure 2, the solutions found by their algorithms exhibit at least a 9% relative gap with respect to the best solution.

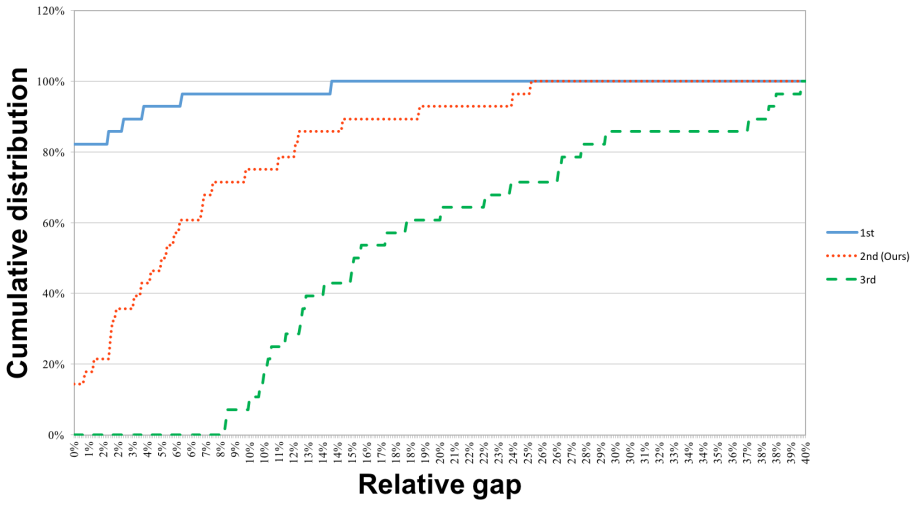

Figure 2: Cumulative distribution of the relative gap on the selection instances.

It is also important to note that these algorithms are randomized, because they are all based on random sampling of future demands. For the first phase of the competition, the participants had to provide the random seeds they used to obtain their results. During the second phase, the organizers executed the code of each team ten times with different arbitrary random seeds on each instance.

Figure 3 shows the distribution of the objective value for 180 observations of the solution of an instance with 80 nurses over 8 weeks using Algorithm 3. The values of the solution are within a $[-6\%, +7\%]$ range.

Because of these variations, we run the algorithm many times on each of the instances used for the selection to submit only the best ones and increase our chance of qualification. Most teams must have used the same technique, since the ranking between the selection and the final rounds did not really change.

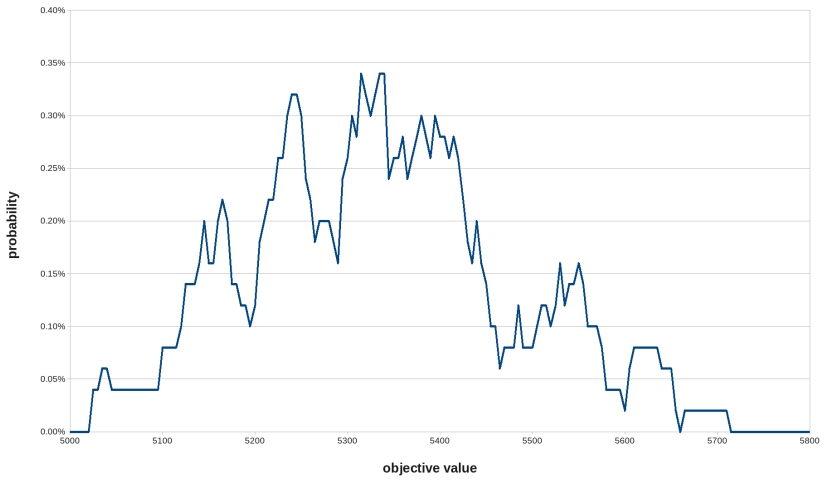

Figure 3: Distribution of the objective value for the selection instances.

5.3 Final instances

The final results respect the same ranking as the one obtained after the selection. However, these comparisons are more fair, since the results were computed by the organizers, so that the teams were not able to select the solutions they submitted. This configuration evaluates in a better way the proposed algorithms, and especially their robustness. The organizers have even run 10 times the algorithms on each of the 60 final instances, and have thus compared the proposed software over 600 tests.

Figure 4 presents the relative gaps obtained on the final instances. The two first teams are really close and their algorithms highlight two distinct features of the competition. The winning team's algorithm builds the best schedules for about 65% of the instances, but our algorithm appears to be more robust. Indeed, our algorithm was able to produce a feasible schedule in every test but one while the winning algorithm could not build a feasible schedule in 5% of the tests (i.e., 34 tests). This comparison also highlights the balance that needs to be found between the time spent in the generation of the best possible schedules and their evaluation, since this second phase provides a measure of their robustness to future demands.

Figure 5 shows the cumulative distribution of the relative gap of the winning team solutions from ours as a function of the number of nurses. It is clear that once the instances exceed a certain size (i.e., 110 nurses), the quality of the solution of the winning team decreases. Indeed, in [21], the winning team comments that their algorithms was simply unable to find feasible integer solutions for some week demands of these instances, showing that the method experiments difficulties in scaling up. Furthermore, this algorithm was also not able to find feasible solutions for an important number of the small instances (i.e., 35 nurses). As a possible explanation, we have observed that it is more difficult to find feasible solutions for these instances, because they leave less flexibility for creating the schedules. Stated otherwise, the proportion of feasible schedules that meet the minimum demand is much smaller for the smallest instances used in the competition.

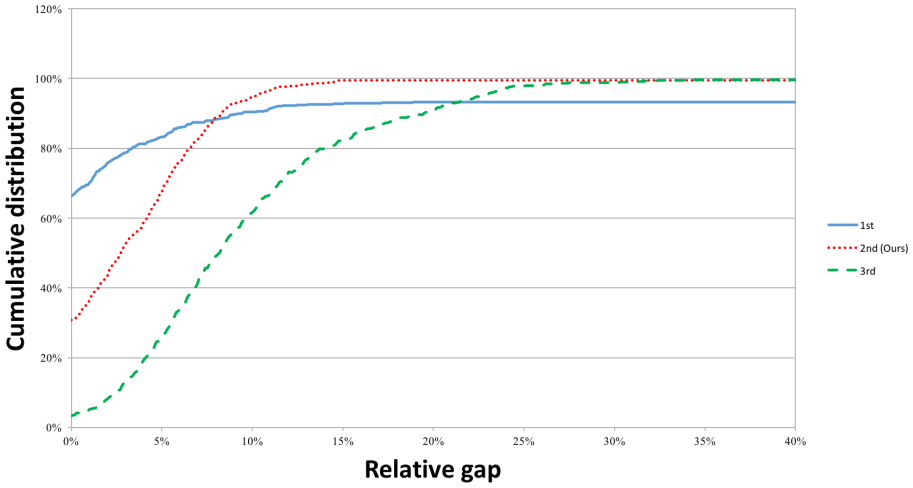

Figure 4: Cumulative distribution of the relative gap on the final instances.

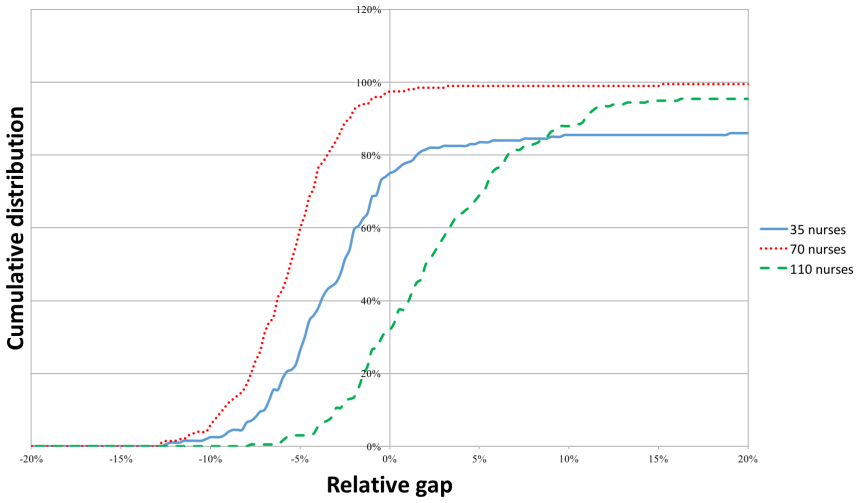

Figure 5: Cumulative distribution of the relative gap as a function of the number of nurses.

6 Conclusions

This article deals with the nurse scheduling problem as described in the context of the international competition INRC–II. The objective is to sequentially build, week by week, the schedule of a set of nurses over a planning horizon of several weeks. In this dynamic process, the schedule computed for a given week is irrevocably fixed before the demand and the preferences for the next week are revealed. The main difficulty is to correctly handle the border effects between weeks and the global soft constraints to compute a feasible and near-optimal schedule for the whole horizon.

Our main contribution is in the design of a robust online stochastic algorithm that performs very well for a wide range of instances (from 30 to 120 nurses over a four or eight weeks horizon). The proposed algorithm embeds a primal-dual algorithm within a sample average approximation. The primal-dual procedure generates candidates schedules for the current week, and the sample average approximation allows to evaluate each of them and retain the best one. The resulting implementation is shared on a public repository [13] and builds upon an open source static nurse scheduling software.

The designed algorithm has been awarded the second prize of the INRC–II competition. The results show that, even if this procedure did not compute the best schedules for a majority of instance, it was the most robust one. Indeed, it has obtained feasible solutions for almost every instance of the competition while providing high-quality schedules.

Despite the limits of this algorithm, our intent with this article was to present the exact implementation that has been submitted to the competition. There is place for improvements that could be developed in the future. For instance, the primal-dual algorithm could be enhanced with non-linear updates, the new features developed in the static nurse scheduling software could be tested, or the bounding constraints added in the primal-dual algorithm could be refined.

References

[1] Bäuerle, N., and Rieder, U. Markov Decision Processes with Applications to Finance. Springer Science & Business Media, 2011.

[2] Bent, R. W., and Van Hentenryck, P. Scenario-based planning for partially dynamic vehicle routing with stochastic customers. Operations Research 52, 6 (2004), 977–987.

[3] Birge, J. R., and Louveaux, F. Introduction to Stochastic Programming, 2nd ed. Springer Series in Operations Research and Financial Engineering. Springer, New York, NY, USA, 2011.

[4] Buchbinder, N. Designing Competitive Online Algorithms via a Primal-Dual Approach. PhD thesis, Technion, 2008.

[5] Burke, E. K., De Causmaecker, P., Berghe, G. V., and Van Landeghem, H. The state of the art of nurse rostering. Journal of Scheduling 7, 6 (2004), 441–499.

[6] Ceschia, S., Dang, N. T. T., De Causmaecker, P., Haspeslagh, S., and Schaerf, A. The second international nurse rostering competition. In PATAT 2014: Proceedings of the 10th International Conference of the Practice and Theory of Automated Timetabling (2014), pp. 554–556.

[7] Denton, B. T., Miller, A. J., Balasubramanian, H. J., and Huschka, T. R. Optimal allocation of surgery blocks to operating rooms under uncertainty. Operations research 58, 4-Part-1 (2010), 802–816.

[8] Desaulniers, G., Desrosiers, J., and Solomon, M. M. Column generation, vol. 5. Springer Science & Business Media, 2006.

[9] Fleten, S.-E., and Kristoffersen, T. K. Short-term hydropower production planning by stochastic programming. Computers & Operations Research 35, 8 (2008), 2656 – 2671. Queues in Practice.

[10] Kleywegt, A. J., Shapiro, A., and Homem-de Mello, T. The sample average approximation method for stochastic discrete optimization. SIAM Journal on Optimization 12, 2 (2002), 479–502.

[11] Kleywegt, A. J., Shapiro, A., and Homem-de Mello, T. The sample average approximation method for stochastic discrete optimization. SIAM Journal on Optimization 12, 2 (2002), 479–502.

[12] Legrain, A., Fortin, M.-A., Lahrichi, N., and Rousseau, L.-M. Online stochastic optimization of radiotherapy patient scheduling. Health Care Management Science 3, 3 (2014), 1–14.

[13] Legrain, A., Omer, J., and Rosat, S. Dynamic nurse scheduler. https://github.com/legraina/DynamicNurseScheduler.git, 2017.

[14] Legrain, A., Omer, J., and Rosat, S. A rotation-based branch-and-price approach for the nurse scheduling problem, 2017.

[15] Nagesh, A. Nurses could go on strike for the first time in british history. http://metro.co.uk/2017/05/14/nurses-could-go-on-strike-for-the-first-time-in-british-history-6636606, 2017.Accessed:2017-05-23.

[16] Novoa, C., and Storer, R. An approximate dynamic programming approach for the vehicle routing problem with stochastic demands. European Journal of Operational Research 196, 2 (2009), 509–515.

[17] Patrick, J., Puterman, M. L., and Queyranne, M. Dynamic multipriority patient scheduling for a diagnostic resource. Operations Research 56, 6 (2008), 1507–1525.

[18] Powell, W. B. Approximate Dynamic Programming: Solving the Curses of Dimensionality. John Wiley & Sons, 2007.

[19] Puterman, M. L. Markov Decision Processes: Discrete Stochastic Dynamic Programming. John Wiley & Sons, 2014.

[20] Robbins, A. We need more nurses. http://www.nytimes.com/2015/05/28/opinion/we-need-more-nurses.html?_r=0, 2015. Accessed: 2017-05-23.

[21] Römer, M., and Mellouli, T. A direct MILP approach based on state-expanded network flows and anticipation for multi-stage nurse rostering under uncertainty. In Proceedings of the 11th International Confenference on Practice and Theory of Automated Timetabling (PATAT-2016) (2016).

[22] Van Hentenryck, P., and Bent, R. Online Stochastic Combinatorial Optimization. The MIT Press, 2009.