European Journal of Operational Research 257 (2017) 439–455

Contents lists available at ScienceDirect

European Journal of Operational Research

journal homepage: www.elsevier.com/locate/ejor

Discrete Optimization

Propagating logic-based Benders' decomposition approaches for distributed operating room scheduling

Vahid Roshanaei[a,*], Curtiss Luong[a], Dionne M. Aleman[a,b,c], David Urbach[d]

[a] *Department of Mechanical and Industrial Engineering, University of Toronto, 5 King's College Road, Toronto, ON M5S 3G8, Canada*
[b] *Institute of Health Policy, Management and Evaluation, University of Toronto, 155 College Street, Suite 425, Toronto, ON M5S 3E3, Canada*
[c] *Techna Institute, University Health Network, 124-100 College Street, Toronto, ON M5G 1P5, Canada*
[d] *Division of General Surgery, Toronto General Hospital, University Health Network, 200 Elizabeth Street, Toronto, ON M5G 2C4, Canada*

ARTICLE INFO

Article history:
Received 8 November 2015
Accepted 8 August 2016
Available online 12 August 2016

Keywords:
Distributed operating room scheduling
Logic-based Benders' decomposition
Cut strength
Propagation
Implementation

ABSTRACT

We develop three novel logic-based Benders' decomposition (LBBD) approaches and a cut propagation mechanism to solve large-scale location-allocation integer programs (IPs). We show that each LBBD can be implemented in four different possible ways, yielding distinct LBBD variants with completely different computational performances. LBBDs decompose the IP model into an integer location and knapsack-based allocation master problem and multiple packing IP sub-problems. We illustrate the performance of our LBBDs on the distributed operating room (OR) scheduling (DORS) problem, where patients and ORs are collaboratively scheduled across a network of hospitals, and the goal is to select patients with the highest priority scores and schedule them in the current planning horizon, while determining the number of surgical suites and operating rooms required to accommodate the schedule at minimum cost. We quantitatively demonstrate that the new Benders' cuts, cut propagation, and implementation can individually or collectively yield a computational time impact of at least two orders of magnitude. We also demonstrate that our LBBDs are 10–100x faster than IP+Gurobi and are more successful at finding optimal solutions. Finally, we show that DORS increases the average OR utilization across the network to more than 95%.

1. Introduction

We propose an optimal framework for solving large-scale location-allocation integer programs (IPs) via logic-based Benders' decomposition (LBBD). LBBD (Hooker & Ottosson, 2003) is a versatile decomposition technique applied successfully to a wide variety of mixed IP (MIP) and constraint programming (CP) problems, including multi-factory planning and scheduling (Hooker, 2007), location-allocation (Fazel-Zarandi & Beck, 2012), single- and multi-stage parallel machine scheduling (Harjunkoski & Grossmann, 2002), parallel machine scheduling with sequence-dependent setup time (Tran, Araujo, & Beck, 2016), two-dimensional bin-packing (Pisinger & Sigurd, 2007), and healthcare worker routing and scheduling (Cire & Hooker, 2012).

LBBD generalizes classical Benders' decomposition by incorporating an optimization master problem (MP) and one or multiple optimization subproblems (SPs) of any mathematical structure. Unlike classical Benders' decomposition, which relies on the solution of its SP dual to derive a Benders' cut, LBBD provides no standard scheme for generating Benders' cuts and cuts must be devised for each problem class uniquely. Hooker and Ottosson (2003) maintain that the notion of a dual must be definable for any SPs, not just for linear ones. Thus, a generalized dual can be defined as an inference dual, which is responsible for inferring the tightest possible bound from the SP constraint set. The SP inference dual solution is used as the tightest bound on the optimal value. The solution of the dual inference is incorporated into the LBBD cuts to guide the master search towards feasibility or optimality by providing a feedback on the quality of the previous MP solution. The LBBD iterative process continues until the MP and SPs converge in value.

LBBD mitigates the complexity of a mathematical model by decomposing it into a relatively easy MP and one or more SPs. In location-allocation problems, the LBBD approach mostly uses an MIP model for assigning tasks to facilities in the MP, and then performs feasible sequencing, routing, or packing among tasks assigned to each facility via either CP or mathematical programming in the SPs. The MP in LBBD is a relaxation of the original problem, and is optimized over primary variables, the values of

* Corresponding author.
E-mail addresses: vroshana@mie.utoronto.ca (V. Roshanaei), curtiss@mie.utoronto.ca (C. Luong), aleman@mie.utoronto.ca (D.M. Aleman), david.urbach@uhn.on.ca (D. Urbach).

http://dx.doi.org/10.1016/j.ejor.2016.08.024
0377-2217/© 2016 Elsevier B.V. All rights reserved.

which are used as input to the SPs, which are optimization problems over secondary variables. SPs include the original constraints that were relaxed in the MP. In minimization problems, the MP provides a lower bound for the SPs and SP solutions are upper bounds for the entire problem. In case of a mismatch between the solution of the MP and SPs, Benders' cuts are developed from SPs and incorporated into the MP. Benders' cuts encompass all the necessary remedial strategies to remove solutions from the MP that are unlikely to satisfy the SP constraints. This give-and-take process is continued until the solution of the MP is feasibly packable (sequenceable) in each SP and the MP objective value (lower bound) and SP objective values (upper bounds) converge.

The three challenges in developing LBBD approaches are the incorporation of a tight SP relaxation into the MP to direct the master search towards solutions that are likely to satisfy the SPs, the determination of the frequency of solving SPs (Beck, 2010; Thorsteinsson, 2001), and the generation of strong Benders' cuts (Hooker, 2007). We discuss these concerns and provide two new perspectives: (1) how to implement LBBD when multiple hard SPs are involved; and (2) how to infer and remove infeasible possible future solutions based on structural similarities among SP constraints. Improper LBBD implementation can yield computational performance comparable to commercial solvers (Heinz & Beck, 2012), while proper implementation can yield up to three orders of magnitude computational time improvement over commercial solvers (Cire, Coban, & Hooker, 2013).

The SP relaxation should neither be so tight as to stifle the MP convergence towards an optimal solution, nor so loose as to lead the MP to a poor optimal solution. SPs are mostly solved given the MP optimal solution in each LBBD iteration. MP difficulty may result in the allotted time limit being reached before an MP optimal solution is found. Thus, no optimality gap is obtained at the end of time limit if the MP solution is not verified by SPs. An effective approach to remedy this problem is to solve SPs for each (Thorsteinsson, 2001) or some (Beck, 2010) MP incumbents if SPs are easy to solve. While developing Benders' cuts, attention should be given to strike a balance among frequency, strength, and the computational efforts to derive Benders' cuts. While only one Benders' cut is required to eliminate the MP solution, almost all LBBD literature has chosen to pass back Benders' cuts from all SPs. Strong cuts are not necessarily productive if the computational effort (SP time) to generate them is too computationally burdensome. We show that relatively weaker cuts can sometimes enhance LBBD performance significantly. Benders' cut strength can be defined in at least three different ways: (1) the ability to remove a large number of master solutions (Fazel-Zarandi & Beck, 2012); (2) the ability to heuristically find a subset of items that have no feasible schedule and remove at least one of these items from the primary variables (Hooker, 2007); and/or (3) the ability to exploit the structural similarity among SPs and infer infeasibility of some SPs from the structure of other SPs (i.e., ability to employ a cut propagation mechanism). In this study, we develop new Benders' cuts using the first and the third definitions due to their better performances on our problem.

We develop three new approaches to generate Benders' cuts of different strengths for LBBD, and we develop a cut propagation mechanism applied to each LBBD variant. We show that each LBBD can be implemented in at least four possible ways, giving rise to various combinatorial LBBDs with different computational performances and robustness. As a case study, we apply our LBBDs to a healthcare operations scheduling problem, called deterministic distributed operating room (OR) scheduling (DORS), where patients and ORs are collaboratively scheduled across multiple hospitals.

2. Literature review

LBBD has recently garnered considerable interest as substantial speedups over existing mathematical models can be achieved (Fazel-Zarandi & Beck, 2012; Harjunkoski & Grossmann, 2002; Hooker & Ottosson, 2003; Tran et al., 2016; Tran & Beck, 2012). These methods have demonstrated promise in tackling both MIP problems (optimization must be achieved) and constraint programming problems (feasibility must be achieved) (Harjunkoski & Grossmann, 2002). Additionally, combining an MIP assignment MP and an MIP sequencing SP has also proven to be a promising LBBD approach (Harjunkoski & Grossmann, 2002).

Although the LBBD MP may be solved sub-optimally (Cire & Hooker, 2012), it is customary to solve it to optimality and the SPs are solved given the MP optimum. LBBD does not guarantee a global feasible solution if the MP does not reach optimality. To resolve this issue, Thorsteinsson (2001) developed an LBBD-based exact technique called branch-and-check, which solves SPs for each MP incumbent instead of only the MP optimum and provides the optimality gap for each MP incumbent that can be verified by SPs. Beck (2010) proposed a novel branch-and-check approach which limits SP checking to MP incumbents whose optimality gap is less than 15% and showed that his approach is more effective than other branch-and-check variants (Bockmayr & Pisaruk, 2006; Sadykov, 2008; Sadykov & Wolsey, 2006; Thorsteinsson, 2001). To the best of our knowledge, despite varying frequencies of SP checking in LBBD and branch-and-check approaches, all SPs are solved for any MP solution (feasible or optimal) to obtain Benders' feasibility and/or optimality cuts. Some LBBDs incorporate only feasibility (Harjunkoski & Grossmann, 2002) or optimality (Fazel-Zarandi & Beck, 2012) Benders' cuts, while some others incorporate both feasibility and optimality cuts (Hooker, 2007). To the best of our knowledge, there is no study in the literature that systematically ascertains which of these approaches yield faster convergence.

The recognition of the benefits of distributed scheduling has recently spurred studies in several fields. In the manufacturing literature, the distributed permutation flowshop (Naderi & Ruiz, 2010) and job shop (Naderi & Azab, 2014) scheduling problems have been studied using both mathematical programming (Naderi & Azab, 2014; Naderi & Ruiz, 2010) and heuristics (Naderi & Azab, 2014; Naderi & Ruiz, 2010; 2014). Behnamian and Fatemi Ghomi (2013) also studied the problem of parallel machine scheduling in a heterogeneous multi-factory network, and developed an MIP model and two heuristics. However, these previous studies use sub-optimal techniques and cannot optimally solve industrial-scale problems. For example, Naderi and Ruiz (2010) developed 14 heuristics for distributed permutation flowshop scheduling and compared their relative percentage deviations from their best-performing algorithm, but did not provide any optimality gap information for large-scale test problems. However, none of their techniques are directly applicable to our research due to their machine shop layout which disposes machines serially and all jobs have to see all machines linearly, compared to our scheduling setting which can be represented as a related parallel machine scheduling.

Distributed task allocation and scheduling in multiple facilities has also been studied using LBBD, which found optimal solutions for large test cases and was faster than existing mathematical models by several orders of magnitude over makespan and tardiness objective functions (Hooker, 2007). The problem of facility location and customer-to-facility-and-truck allocation under facility capacity and truck travel distance constraints has been studied (Albareda-Sambola, Fernández, & Laporte, 2009), where the MP is a facility location and customer-to-facility assignment to find the minimum required vehicles to satisfy the customer demands, the SP is customer-to-truck allocation, and the MP and SP are integrated via a nested Tabu search algorithm. The algorithm finds optimal or near-optimal solutions, when optimal solutions exist via CPLEX.

Fazel-Zarandi and Beck (2012) developed an optimal LBBD framework, consisting of a location-allocation MP solved via CPLEX and customer packing SP solved via a CP optimizer. The authors connected the MP and SP through a Benders' packing cut, which is a modified version of the Benders' scheduling cut initially developed for the minimum makespan problem by Hooker (2007). The resulting LBBD found optimal solutions up to three orders of magnitude faster than the original IP model of Albareda-Sambola et al. (2009) and was faster than the Tabu search approach (Albareda-Sambola et al., 2009) on medium-to-large instances of the problem. The LBBD was tested for scenarios in which the SP runtime is less than 1% of the total computational time, and has not been tested when SPs constitute a significant portion of total CPU time. However, there is a systematic issue with the Benders' optimality cut in Fazel-Zarandi and Beck (2012), which may lead the LBBD to accept an infeasible solution as the optimal solution. We address and resolve this issue in Section 4.4.3.

While DORS is similar to the location-allocation problem of Albareda-Sambola et al. (2009) and Fazel-Zarandi and Beck (2012), it differs in a number of ways. Unlike the strategic location-allocation problem of Albareda-Sambola et al. (2009) and Fazel-Zarandi and Beck (2012) that is solved once, DORS is a weekly OR scheduling problem similar to Fei, Chu, and Meskens (2009), Fei, Meskens, and Chu (2010), Marques, Captivo, and Margarida (2012) and Hashemi Doulabi, Rousseau, and Pesant (2016) with different numbers of patients with varying surgical lengths in each week. Unlike (Albareda-Sambola et al., 2009; Fazel-Zarandi & Beck, 2012) that serves all of its customers, DORS selectively schedules patients based on health status scores and OR capacity. We tailor our LBBDs to account for these structural differences and we show that an LBBD implementation similar to the single-cut LBBD of Fazel-Zarandi and Beck (2012) performs poorly as SP difficulty increases.

Deterministic single-hospital OR scheduling problem is well-studied (Augusto, Xie, & Perdomo, 2008; 2010; Cardoen, Demeulemeester, & Belien, 2009a; 2009b; Denton, Miller, Balasubramanian, & Huschka, 2010; Fei et al., 2009; Fei et al., 2010; Jebali, Alouane, & Ladet, 2006; Silva, de Souza, Saldanha, & Burke, 2015; Wang, Meskens, & Duvivier, 2015) and a wide variety of solution techniques including heuristics (Fei et al., 2009; Herring & Herrmann, 2012; Liu, Chu, & Wang, 2011; Tanfani & Testi, 2010), meta-heuristics (Fei et al., 2010; Rizk & Arnaout, 2012), approximate methods (Astaraky & Patrick, 2015; Lin, Muthuraman, & Lawley, 2011), and exact methods (Augusto et al., 2008; Cardoen, Demeulemeester, & Belien, 2009b; Lamiri, Xie, & Zhang, 2008; Lin et al., 2011; Range, Lusby, & Larsen, 2014) have been proposed. In a healthcare context, the problem of distributed master surgical scheduling (MSS), consisting of allocating surgeons to OR blocks, has been studied in a network of hospitals and solved via existing mathematical programming solvers (Santibanez, Begen, & Atkins, 2007). The authors determine the number of procedures from each surgical group that needs to be assigned to each OR in a hospital-day and do not explicitly model patient to hospital-day-OR allocation. Unlike distributed MSS, DORS has not been investigated in the OR scheduling literature. In order to characterize DORS, we formulate the problem as a novel IP to account for the selective scheduling of patients based on priority scores. None of the methods developed in the previous papers can be directly used to solve DORS because they have been tailored for their single-hospital structures.

To the best of our knowledge and according to two recently published review papers on OR scheduling (Cardoen, Demeulemeester, & Belien, 2010; Guerriero & Guido, 2011), no studies have combined a location-allocation MP and packing SPs in deterministic OR scheduling as we propose. We thus contribute to the developments of LBBDs with a location-allocation MP and multiple packing SPs as follows:

- Given an existing decomposition scheme (Fazel-Zarandi & Beck, 2012), we develop novel LBBD variants for DORS based on new implementations (when and how to solve SPs), new Benders' cuts (which types of Benders' cuts to incorporate into the MP), and a cut propagation mechanism (how to remove future infeasible and sub-optimal solutions by learning from other SPs).
- Unlike the existing similar LBBD in literature (Fazel-Zarandi & Beck, 2012), which uses only Benders' optimality cuts and may converge to an infeasible solution, we employ Benders' feasibility cuts when SPs are infeasible and Benders' optimality cuts when SPs are sub-optimal (i.e., when the SP solution is optimal but greater than the MP minimization solution). We prove the validity of our Benders' feasibility and optimality cuts.
- We incorporate easy-to-compute feasibility cuts of different strengths for infeasible SPs and show that they can communicate the same amount of information to the MP as strongly as optimality cuts in certain circumstances.
- We incorporate a cut propagation mechanism into our LBBDs which exploits the structure of the infeasible or sub-optimal SPs to eliminate similar infeasible or sub-optimal solutions that the MP may generate for other SPs in future iterations.
- We show that LBBDs with multiple SPs can be implemented in four different ways, and empirically demonstrate that the choice of LBBD implementation can exert a computational influence of up to two orders of magnitude on the computational performance and a significant impact on robustness (ability to solve).
- We show that for both easy and hard SPs, a tailor-made LBBD with customized Benders' cuts and implementation is needed.
- We show that bi-cut (both feasibility and optimality cuts) LBBDs are more robust than single-cut (only optimality cuts) LBBDs for hard SPs. We also show that bi-cut LBBDs with strong feasibility cuts outperform those LBBDs with weak feasibility cuts.

3. Problem definition

DORS is a centralized multi-hospital priority-based approach to elective surgery scheduling, consisting of a set of collaborating hospitals ($h \in \mathcal{H}$), ORs in each hospital ($r \in \mathcal{R}_h$), days in the planning horizon ($d \in \mathcal{D}$), and patients ($p \in \mathcal{P}$). DORS takes into account patients' wait times (α_p) and urgency scores (ρ_p) and schedules them based on their health status scores, which we define as the product of the wait time and urgency score. Associated with each surgery is a total booked time T_p (preparation time + surgery time + cleaning time). DORS schedules all or a subset of the existing patients in the hospitals' wait lists, while taking into account the number of ORs in the network ($|\mathcal{H}| \times |\mathcal{D}| \times |\mathcal{R}_h|$) and the daily availability time of ORs in each hospital (B_{hd}). Any patient not scheduled in the current planning horizon is added to the pool of patients to be scheduled for the next planning horizon.

To give scheduling priority to highly critical elective patients, DORS considers a threshold (Γ) based on which the existing pool of patients is divided into two sets: mandatory patients ($p \in \mathcal{P}'$) whose health status score is greater than the threshold and optional patients ($p \in \mathcal{P} \setminus \mathcal{P}'$) whose health status score is less than the threshold. The health status scores of the set of mandatory patients ($\mathcal{P}'$) correspond to elective priority patients on whom surgical operation have to be performed in less than a week (Marques et al., 2012). The threshold Γ is a discretionary parameter chosen

Table 1
IP notation.

Sets:	
$\mathcal{P}$	Set of patients, $p \in \mathcal{P}$
$\mathcal{P}'$	Set of mandatory patients, $\mathcal{P}' = \{p \mid \rho_p(\lvert\mathcal{D}\rvert - \alpha_p) \leq -\Gamma\}$
$\mathcal{H}$	Set of hospitals, $h \in \mathcal{H}$
$\mathcal{D}$	Set of days in the planning horizon, $d \in \mathcal{D}$
$\mathcal{R}_h$	Set of ORs in each hospital's surgical suite, $r \in \mathcal{R}_h$
Parameters:	
G_{hd}	Cost of opening the surgical suite in hospital h on day d
F_{hd}	Cost of opening an OR in hospital h on day d
B_{hd}	Regular operating hours of each OR on day d in hospital h
T_p	Total booked time (preparation time + surgery time + cleaning time) of patient p
ρ_p	Health status score assigned to patient p
α_p	Number of days elapsed from the referral date of patient p
κ_1	Waiting cost for scheduled patients
κ_2	Waiting cost for unscheduled patients
Γ	Health status threshold above which patients have to be operated
Variables:	
x_{hdp}	1 if patient p is assigned to hospital h on day d, 0 otherwise
u_{hd}	1 if the surgical suite in hospital h is opened on day d, 0 otherwise
y_{hdr}	1 if OR r of surgical suite of hospital h is opened on day d, 0 otherwise
x_{hdpr}	1 if patient p is assigned to OR r of hospital h on day d, 0 otherwise
w_p	1 if patient p is not scheduled this horizon, 0 otherwise

such that there are sufficient resources to operate on all mandatory patients in the current planning horizon, thus ensuring feasibility. DORS considers a fixed daily deterioration cost (κ_1, health status score multiplied by daily deterioration cost) for all patients and attempts to schedule patients with the highest deterioration costs at the beginning of the current planning horizon and low-priority patients at the end of the current planning horizon. The daily deterioration cost of patients not scheduled in the current planning horizon is $\kappa_2 < \kappa_1$. DORS trades off the health deterioration costs of patients with the costs of providing required resources, including the fixed costs of opening surgical suites and ORs. Given the existing number of surgical suites (hospitals) and ORs within each surgical suite, the amount of OR availability, and health status scores of patients (deterioration costs), DORS determines an optimal allocation of mandatory patients to hospital-day-ORs and also possible allocation of optional patients if OR capacity permits. DORS minimizes the fixed cost of opening surgical suites and ORs within each surgical suite and the deterioration costs for scheduled and unscheduled patients. The IP notation is shown in Table 1.

We formulate DORS as an IP model, which is a combination of daily surgical suite and OR closing/opening (Denton et al., 2010; Roland, Martinelly, Riane, & Pochet, 2010), knapsack (Fei et al., 2009; Hashemi Doulabi et al., 2016; Jebali et al., 2006; Marcon, Kharraja, & Simonnet, 2003; Marques et al., 2012), and bin-packing (Day, Garfinkel, & Thompson, 2012; Denton et al., 2010; Houdenhoven, van Oostrum, Hans, Wullink, & Kazemier, 2007; Houdenhoven et al., 2008; Roland et al., 2010; Vijayakumar, Parikh, Scott, Barnes, & Gallimore, 2013) problems with decision variables to make patient-hospital-OR-day assignments. The process of selecting patients in DORS reduces to an integer knapsack problem in which a pool of patients with known surgical times are fit into ORs so that costs are minimized and the total priority score of the selected patients is maximized. The priority score of each patient is the number of days elapsed from the referral date of the patient multiplied by his/her health status score. While it is common for wait lists to exceed surgical suite capacity, for the sake of generality and completeness, we keep the decision variables regarding opening and closing of surgical suites and ORs.

We make the following assumptions that are common in OR scheduling literature (Cardoen et al., 2010; Guerriero & Guido, 2011): (1) all ORs and surgeons are functionally identical within each surgical specialty; (2) master surgical scheduling along with nurse and anesthesiologist assignments to ORs have been performed a priori; (3) each OR block is assigned to only one surgeon in a day (therefore, patient-to-OR assignment is equivalent to patient-to-surgeon assignment); (4) hospitals have an equal number of ORs because each hospital in our real-world dataset has the same number of ORs; (5) the collaborative scheduling is done within only one surgical specialty; (6) surgical durations are considered to be deterministic for computational simplicity; and (7) each single surgical duration (T_p) is less than the maximum OR session length (B_{hd}). We will later discuss how relaxing these assumptions affects our IP model and LBBD approaches. The IP is as follows:

$$\begin{aligned}
\text{minimize}\quad & \sum_{h\in\mathcal{H}}\sum_{d\in\mathcal{D}} G_{hd}u_{hd} + \sum_{h\in\mathcal{H}}\sum_{d\in\mathcal{D}}\sum_{r\in\mathcal{R}_h} F_{hd}y_{hdr} \\
& + \sum_{h\in\mathcal{H}}\sum_{d\in\mathcal{D}}\sum_{p\in\mathcal{P}}\sum_{r\in\mathcal{R}_h} \kappa_1[\rho_p(d-\alpha_p)x_{hdpr}] \\
& + \sum_{p\in\mathcal{P}\setminus\{\mathcal{P}'\}} \kappa_2[\rho_p(\lvert\mathcal{D}\rvert + 1 - \alpha_p)w_p]
\end{aligned} \tag{IP}$$

$$\text{subject to}\quad \sum_{h\in\mathcal{H}}\sum_{d\in\mathcal{D}}\sum_{r\in\mathcal{R}_h} x_{hdpr} = 1 \qquad \forall p \in \mathcal{P}' \tag{1}$$

$$\sum_{h\in\mathcal{H}}\sum_{d\in\mathcal{D}}\sum_{r\in\mathcal{R}_h} x_{hdpr} + w_p = 1 \qquad \forall p \in \mathcal{P}\setminus\{\mathcal{P}'\} \tag{2}$$

$$\sum_{p\in\mathcal{P}} T_p x_{hdpr} \leq B_{hd}y_{hdr} \qquad \forall h\in\mathcal{H};\ d\in\mathcal{D};\ r\in\mathcal{R}_h \tag{3}$$

$$y_{hdr} \leq y_{hd,r-1} \qquad \forall h\in\mathcal{H};\ d\in\mathcal{D};\ r\in\mathcal{R}_h\setminus\{1\} \tag{4}$$

$$y_{hdr} \leq u_{hd} \qquad \forall h\in\mathcal{H};\ d\in\mathcal{D};\ r\in\mathcal{R}_h \tag{5}$$

$$x_{hdpr} \leq y_{hdr} \qquad \forall h\in\mathcal{H};\ d\in\mathcal{D};\ p\in\mathcal{P};\ r\in\mathcal{R}_h \tag{6}$$

$$u_{hd}, y_{hdr}, x_{hdpr} \in \{0,1\} \qquad \forall h\in\mathcal{H};\ d\in\mathcal{D};\ p\in\mathcal{P};\ r\in\mathcal{R}_h$$

$$w_p \in \{0,1\} \qquad \forall p\in\mathcal{P}\setminus\{\mathcal{P}'\}.$$

The objective function minimizes the costs associated with opening surgical suites and ORs, and the waiting costs of patients scheduled to be operated on during the planning horizon and of those who remain on the waiting lists. Constraint (1) ensures that mandatory patients are operated on in the current planning horizon. Constraints (2), (5), and (6) ensure consistency between the variables. Constraint (3) ensures that the availability time of each OR is not violated, and Constraint (4) breaks the symmetry among ORs of each hospital on each day, since all ORs are identical. Although Constraint (6) may seem redundant from modeling perspective due to the presence of Constraint (3), it is used to strengthen the LP relaxation of the IP model (Albareda-Sambola et al., 2009; Denton et al., 2010; Fazel-Zarandi & Beck, 2012). If $u_{hd} = 0$, the surgical suite is closed in that day and those ORs can be reallocated to emergency surgeries or other surgical specialties, mitigating surgical demand fluctuations and uncertainties.

4. Logic-based Benders' decomposition

LBBD (Hooker & Ottosson, 2003), illustrated in Fig. 1 for DORS, is a decomposition-based row generation technique of which classical Benders' decomposition (Benders, 1962) is a special case. Like classical Benders' decomposition, LBBD decomposes a model into an MP (consisting of primary variables) and one or more SPs (consisting of secondary variables). The MP is a relaxation of the original model, considering only the primary variables (u_{hd}, $x_{hdp} \in \{0, 1\}$, and $y_{hd} \geq 0$ and integer) and constraints relating to those variables. The SP results from formulating an optimization over the secondary variables (x_{pr} and $y_r \in \{0, 1\}$) while fixing primary variables to values computed by the MP. Depending on the solution of the SP and its relationship with the MP solution, feasibility or op-

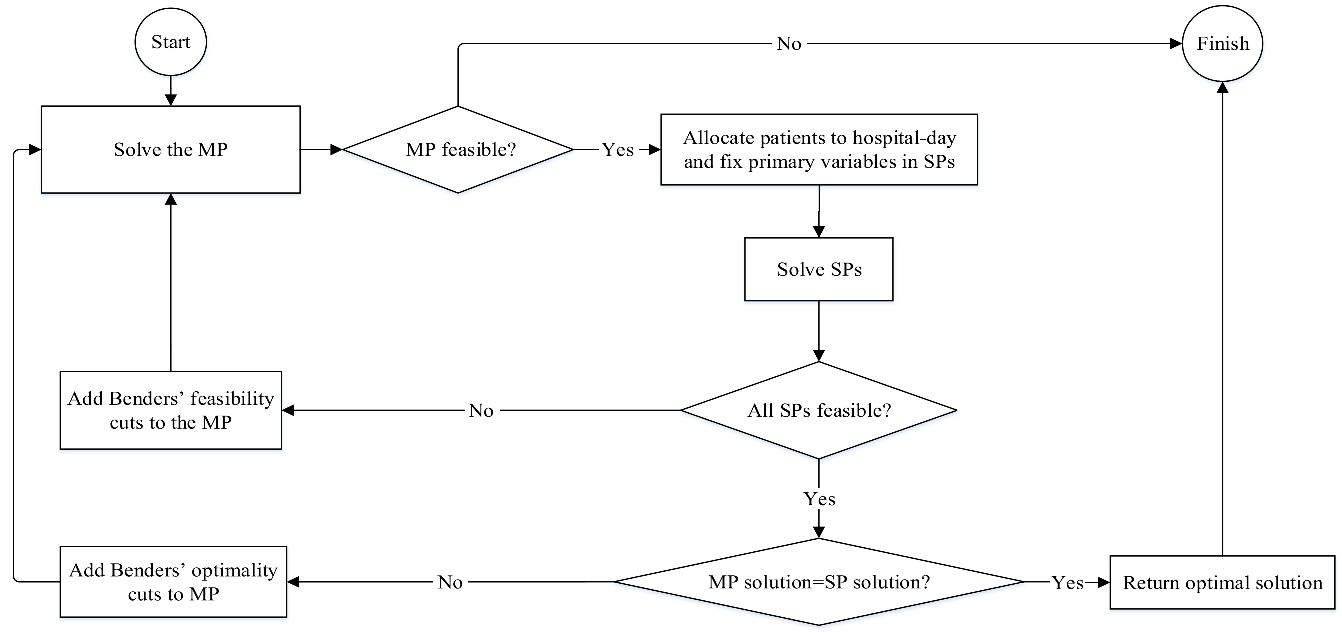

Start
Solve the MP
MP feasible?
No
Yes
Finish
Allocate patients to hospital-day and fix primary variables in SPs
Solve SPs
All SPs feasible?
No
Add Benders' feasibility cuts to the MP
Yes
MP solution=SP solution?
No
Yes
Add Benders' optimality cuts to MP
Return optimal solution

Fig. 1. LBBD flowchart for DORS.

timality cuts are developed and added to the MP. While the MP is always theoretically feasible due to the choice of Γ, if the MP is not optimally solvable within the time limit, we cannot guarantee a globally feasible or optimal solution. The optimization process is terminated if the MP is infeasible, and no scheduling solution is returned.

In LBBD, the Benders' cuts are derived by tackling the inference dual of the SP, of which the linear programming (LP) dual is a special case. The solution of the inference dual is proof of optimality inferred from the constraints of the SP when the primary variables of the MP are fixed to their either feasible or optimal trial values. The equality of the inference dual solution (SP optimum) and the MP optimum is proof of optimality; otherwise, a Benders' cut has to be developed for the SPs whose integer/fractional feasible solution is greater than the MP solution. Although a cut developed by the SP optimum is the strongest possible cut, SP optimality is not required, and may be detrimental to overall performance if the SP is computationally intensive. Instead, a Benders' cut may incorporate an inference dual solution (an SP optimum or any upper bound to the MP solution). The Benders' cut incorporates the inference dual solution at each iteration to derive a valid bound on the optimal value for the entire problem. LBBD is executed by iteratively solving the MP and SPs, adding Benders' cuts from SPs to the MP until the MP solution converges to the SP solutions. We exploit the decomposability of DORS to develop four bi-cut and two single-cut LBBDs, consisting of a location-allocation MP and a set of independent OR-packing SPs. For the generic LBBD procedure, readers are referred to Hooker and Ottosson (2003).

4.1. Location-allocation master problem

The MP allocates patients to different surgical suites of hospital-days, leaving patient-to-OR allocations within the surgical suites to be determined by the SPs. The MP also determines the number of surgical suites to open and computes a lower bound on the minimum number of ORs required for each hospital-day. The MP is obtained through a relaxation of IP in which the four-indexed allocation binary variable (x_{hdpr}) is transformed into a three-indexed allocation binary variable (x_{hdp}), and the binary variable for deciding whether to open an OR in each hospital-day (y_{hdr}) is removed. Since we do not include y_{hdr} in our MP, constraints regarding the symmetry breaking among ORs are not included in the MP, but are instead in the SP. To determine the number of ORs required to operate patients in each hospital-day, we introduce y_{hd}, an integer variable reflecting the lower bound on the number of ORs that can be represented without considering specific patient-to-OR allocations. The MP is formulated as follows:

$$\begin{aligned}\text{minimize}\quad & \sum_{h\in\mathcal{H}}\sum_{d\in\mathcal{D}} G_{hd}u_{hd} + \sum_{h\in\mathcal{H}}\sum_{d\in\mathcal{D}} F_{hd}y_{hd}\\ & + \sum_{h\in\mathcal{H}}\sum_{d\in\mathcal{D}}\sum_{p\in\mathcal{P}} \kappa_1[\rho_p(d-\alpha_p)x_{hdp}]\\ & + \sum_{p\in\mathcal{P}\setminus\{\mathcal{P}'\}} \kappa_2[\rho_p(|\mathcal{D}|+1-\alpha_p)w_p] \end{aligned}\tag{MP}$$

$$\text{subject to}\quad \sum_{h\in\mathcal{H}}\sum_{d\in\mathcal{D}} x_{hdp} = 1 \qquad \forall p\in\mathcal{P}' \tag{7}$$

$$\sum_{h\in\mathcal{H}}\sum_{d\in\mathcal{D}} x_{hdp} + w_p = 1 \qquad \forall p\in\mathcal{P}\setminus\{\mathcal{P}'\} \tag{8}$$

$$x_{hdp} \le u_{hd} \qquad \forall h\in\mathcal{H};\ d\in\mathcal{D};\ p\in\mathcal{P} \tag{9}$$

$$\sum_{p\in\mathcal{P}} T_p x_{hdp} \le |\mathcal{R}_h| B_{hd} u_{hd} \qquad \forall h\in\mathcal{H};\ d\in\mathcal{D} \tag{10}$$

$$T_p x_{hdp} \le B_{hd} \qquad \forall h\in\mathcal{H};\ d\in\mathcal{D};\ p\in\mathcal{P} \tag{11}$$

$$y_{hd} \ge \frac{\sum_{p\in\mathcal{P}} T_p x_{hdp}}{B_{hd}} \qquad \forall h\in\mathcal{H};\ d\in\mathcal{D} \tag{12}$$

$$y_{hd} \le |\mathcal{R}_h| \qquad \forall h\in\mathcal{H};\ d\in\mathcal{D} \tag{13}$$

$$y_{hd} \in \mathbb{Z}^+ \qquad \forall h\in\mathcal{H};\ d\in\mathcal{D}$$

$$u_{hd}, x_{hdp} \in \{0,1\} \qquad \forall h\in\mathcal{H};\ d\in\mathcal{D};\ p\in\mathcal{P}$$

$$w_p \in \{0,1\} \qquad \forall p\in\mathcal{P}\setminus\{\mathcal{P}'\}.$$

Constraints (7)–(9) are translated directly from IP; they are sufficient to construct a patient-hospital-day allocation and to ensure that surgical suites are correctly opened on days that they

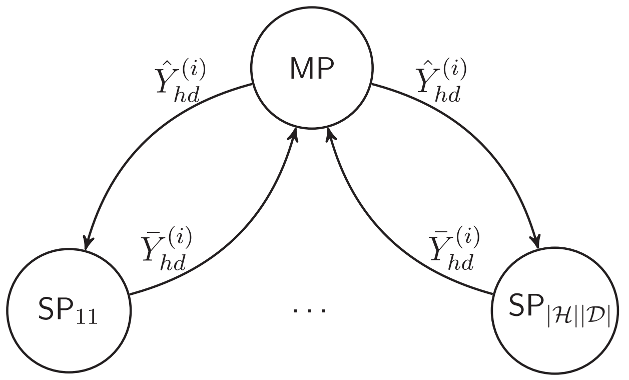


Fig. 2. The linking variable between the MP and SPs. An MP solution is considered optimal if $\hat{Y}^{(i)}_{hd}$ (MP optimum) = $\bar{Y}^{(i)}_{hd}$ (SP optimum) for all SPs.

are required. Because limiting the search space of the MP through the incorporation of an SP relaxation has been shown to be an effective LBBD strategy (Hooker, 2007), we introduce Constraints (10)–(12) as SP relaxations, representing the relaxation of packing Constraint (3) in IP. Constraint (10) indicates that the surgical caseload assigned to each hospital does not exceed the available OR time for each hospital-day. Similarly, Constraint (11) ensures that no single surgical case lasts longer than the OR availability time of a hospital-day to which it is assigned. Constraint (12) computes a lower bound on the number of ORs required for each hospital-day SP given the caseload assigned due to patient-hospital-day allocation decisions. The relationship between Constraints (9) and (10) is similar to the relationship between Constraints (3) and (6) to strengthen the MP LP relaxation (Albareda-Sambola et al., 2009; Fazel-Zarandi & Beck, 2012). In iteration i of LBBD, the lower bound on the minimum number of ORs and the set of patients that the MP determines for hospital h on day d are denoted $\hat{Y}^{(i)}_{hd}$ and $\hat{\mathcal{P}}^{(i)}_{hd}$, respectively. The linking variable between the MP and SPs is shown in Fig. 2.

4.2. OR allocation sub-problems

The optimized solution of the MP model $(\hat{Y}^{(i)}_{hd}, \hat{\mathcal{P}}^{(i)}_{hd})$ is fed into the SPs (one for each open surgical suite in hospital-day), which then simultaneously determine the minimum number of ORs $(\bar{Y}^{(i)}_{hd})$ and optimal patient-OR allocation $(\bar{x}^{(i)}_{pr})$ for each hospital-day, while simultaneously assigning each patient to exactly one OR through the binary variables x_{pr}, along with opening/closing ORs with the binary variables y_r. Our SP represents a standard bin-packing problem with symmetry breaking constraints among ORs in each hospital-day. The allocation of ORs to an open surgical suite of hospital $h \in \mathcal{H}$ on $d \in \mathcal{D}$ based on the surgical length of patients can be modeled as a bin-packing problem (Day et al., 2012; Denton et al., 2010; Houdenhoven et al., 2007; Houdenhoven et al., 2008; Roland et al., 2010; Vijayakumar et al., 2013), which is an equally popular practice both in private (Denton et al., 2010) and public (Vijayakumar et al., 2013) hospitals, and can be formulated as an integer or constraint program. Unlike the packing SP of Fazel-Zarandi and Beck (2012), which is solved via constraint programming, our bin-packing problem with symmetry breaking constraints among ORs is solved via an IP model, which is based on the MIP model presented by Denton et al. (2010), with the exception that we do not allow overtime. The SP minimizes the number of ORs to open, and is formulated as follows:

$$\text{minimize} \quad \bar{Y}^{(i)}_{hd} = \sum_{r \in \mathcal{R}_h} y_r \tag{SP}$$

$$\text{subject to} \quad \sum_{r \in \mathcal{R}_h} x_{pr} = 1 \qquad \forall p \in \hat{\mathcal{P}}^{(i)}_{hd} \tag{14}$$

$$\sum_{p \in \hat{\mathcal{P}}^{(i)}_{hd}} T_p x_{pr} \le B_{hd} y_r \qquad \forall r \in \mathcal{R}_h \tag{15}$$

$$x_{pr} \le y_r \qquad \forall p \in \hat{\mathcal{P}}^{(i)}_{hd};\ r \in \mathcal{R}_h \tag{16}$$

$$y_r \le y_{r-1} \qquad \forall r \in \mathcal{R}_h \setminus \{1\} \tag{17}$$

$$x_{pr}, y_r \in \{0, 1\} \qquad \forall p \in \hat{\mathcal{P}}^{(i)}_{hd};\ r \in \mathcal{R}_h. \tag{18}$$

Constraint (14) ensures that each patient is assigned to exactly one OR. Constraint (15) ensures that no open OR is over-capacitated. Constraint (15) enforces that a patient can be assigned to only open ORs and is accompanied by Constraint (16) for a stronger SP LP relaxation (Denton et al., 2010). Constraint (17) breaks the symmetry among ORs in each hospital-day. The LBBD converges to optimality if $\hat{Y}^{(i)}_{hd}$ (lower bound) = $\bar{Y}^{(i)}_{hd}$ (upper bound); otherwise, logic-based Benders' cuts are required to make the MP and SP solutions converge.

4.3. First-fit decreasing heuristic algorithm

Packing SPs may be hard to solve, depending on $|\hat{\mathcal{P}}^{(i)}_{hd}|$. Therefore, prior to solving possibly hard SPs, we solve them via the first-fit decreasing (FFD) heuristic algorithm to find a feasible solution $(\bar{F}^{(i)}_{hd})$ for each SP. The FFD heuristic is used because it is faster than both constraint and integer programming packing SPs and often finds an optimal solution equal to $\hat{Y}^{(i)}_{hd}$ (Fazel-Zarandi & Beck, 2012). The relationship $\hat{Y}^{(i)}_{hd} \le \bar{Y}^{(i)}_{hd} \le \bar{F}^{(i)}_{hd}$ holds among the solutions of the MP, SP, and FFD. Therefore, the advantage of solving the FFD heuristic algorithm is that we detect optimal SPs without solving them if $\hat{Y}^{(i)}_{hd} = \bar{F}^{(i)}_{hd}$. Other possible relationships between $\hat{Y}^{(i)}_{hd}$ and $\bar{F}^{(i)}_{hd}$ can be treated differently by our LBBDs, which we address through Benders' cuts. Unlike (Fazel-Zarandi & Beck, 2012) which may solve an SP multiple times to find a feasible solution for their CP problem, we solve our SPs once, given the $\bar{F}^{(i)}_{hd}$.

4.4. Benders' cuts

For any SP that has a mismatch between $\hat{Y}^{(i)}_{hd}$ and $\bar{Y}^{(i)}_{hd}$, a Benders' cut is added to the MP to refine its patient allocation decision. A strong Benders' cut communicates all possible ways (remedial strategies) to the MP to remedy the existing mismatch between $\hat{Y}^{(i)}_{hd}$ and $\bar{Y}^{(i)}_{hd}$, but may be computationally expensive to obtain if SPs are hard to solve. Alternatively, a weak cut can communicate only partial information to the MP, but is achievable at a lower computational burden.

Various remedial strategies via different Benders' cuts can be incorporated to guide the master search towards global optimality if $\hat{Y}^{(i)}_{hd} \ne \bar{F}^{(i)}_{hd}$. A feasible MP solution may yield one of the following three possible scenarios:

1. Infeasible SP if $\hat{\mathcal{P}}^{(i)}_{hd}$ is not packable within $|\mathcal{R}_h|$, implying $\bar{Y}^{(i)}_{hd} > |\mathcal{R}_h|$.
2. Optimal MP and optimal SPs, but $\hat{Y}^{(i)}_{hd} \ne \bar{Y}^{(i)}_{hd}$.
3. Global optimal solution if $\hat{Y}^{(i)}_{hd} = \bar{Y}^{(i)}_{hd}$.

The set of infeasible SPs (Scenario 1) at iteration i of the LBBD is denoted $\bar{\mathcal{U}}^{(i)}$, and the set of SPs belonging to Scenario 2 is denoted $\bar{\mathcal{J}}^{(i)}$. We develop different variants of LBBD based on how $\bar{F}^{(i)}_{hd}$ is used, what types of Benders' cut are incorporated into the MP, and whether a cut propagation mechanism is used.

4.4.1. LBBD1

LBBD1 incorporates two types of Benders' cuts to guide the MP towards optimality by using $\bar{F}^{(i)}_{hd}$ to detect optimal SPs without solving them. If $\bar{F}^{(i)}_{hd} \ne \hat{Y}^{(i)}_{hd}$, we solve the SPs with $\min\{\bar{F}^{(i)}_{hd}, |\mathcal{R}_h|\}$ as the upper bound on the number of ORs that can be opened. The adjusted upper bound is useful only if $\bar{F}_{hd}^{(i)} < |\mathcal{R}_h|$ and is redundant otherwise since $\bar{Y}_{hd}^{(i)} \leq |\mathcal{R}_h|$. If the SP is infeasible, the following Benders' feasibility cut is added to the MP, forcing it to remove at least one patient from $\hat{\mathcal{P}}_{hd}^{(i)}$:

$$\sum_{p\in\hat{\mathcal{P}}_{hd}^{(i)}} (1 - x_{hdp}) \geq 1 \qquad \forall (h, d) \in \bar{\mathcal{U}}^{(i)}. \tag{19}$$

The set of feasibility cuts developed from $\bar{\mathcal{U}}^{(i)}$ is denoted $\mathcal{F}$. This cut is inspired by the Benders' cut proposed by Harjunkoski and Grossmann (2002), who generate a Benders' cut for each overloaded resource (machine), while we generate a Benders' cut for each hospital-day (site) and not for each OR. This cut is a valid cut (Theorem 1), defined as a logical expression that has two properties (Chu & Xia, 2004): (1) the cut must eliminate the current MP solution if it is not globally feasible, and (2) the cut must not remove any globally feasible solutions.

Theorem 1. *The Benders' feasibility cut (Inequality (19)) is valid.*

Proof. We first show that Inequality (19) cuts off the current infeasible MP solution from the feasible set but keeps all the other feasible solutions. Let $\hat{\mathcal{P}}_{hd}^{(i)}$ be the current set of patients, resulting in an infeasible MP solution (x_{hdp}). By definition, we have

$$\hat{\mathcal{P}}_{hd}^{(i)} = \left\{p \in \mathcal{P}, h \in \mathcal{H}, d \in \mathcal{D} \mid x_{hdp} = 1 \text{ at iteration } i \text{ of MP}\right\} \neq \emptyset$$

Therefore, the current MP solution with $\hat{\mathcal{P}}_{hd}^{(i)}$ does not satisfy Inequality (19) since

$$1 - x_{hdp} = 0, \qquad \forall p \in \hat{\mathcal{P}}_{hd}^{(i)} \quad \Rightarrow \quad \sum_{p\in\hat{\mathcal{P}}_{hd}^{(i)}} (1 - x_{hdp}) = 0 < 1.$$

$\forall (h, d) \in \bar{\mathcal{U}}^{(i)}$

We next show that Inequality (19) does not remove any globally feasible solution. Consider a new set of patients, $\tilde{\mathcal{P}}_{hd}$, leading to a feasible or infeasible future solution. We show that $\tilde{\mathcal{P}}_{hd}$ satisfies Inequality (19) if it is feasible. To this end, we provide intuitive explanations regarding how the validity of our Benders' feasible cut can be generally proven using the relational possibilities between $\hat{\mathcal{P}}_{hd}^{(i)}$ and $\tilde{\mathcal{P}}_{hd}$. There exist four relational possibilities between $\hat{\mathcal{P}}_{hd}^{(i)}$ and $\tilde{\mathcal{P}}_{hd}$:

- $\tilde{\mathcal{P}}_{hd}$ adds some new patients to $\hat{\mathcal{P}}_{hd}^{(i)}$ (i.e., $\hat{\mathcal{P}}_{hd}^{(i)} \subset \tilde{\mathcal{P}}_{hd}$ and $\hat{\mathcal{P}}_{hd}^{(i)}\backslash\tilde{\mathcal{P}}_{hd} = \emptyset$), yielding

$$\sum_{p\in\hat{\mathcal{P}}_{hd}^{(i)}} (1 - x_{hdp}) = \overbrace{\sum_{p\in\hat{\mathcal{P}}_{hd}^{(i)}\cap\tilde{\mathcal{P}}_{hd}} (1 - x_{hdp})}^{0} + \overbrace{\sum_{p\in\hat{\mathcal{P}}_{hd}^{(i)}\backslash\tilde{\mathcal{P}}_{hd}} (1 - x_{hdp})}^{0}$$
$$= 0 < 1.$$

  Therefore, $\tilde{\mathcal{P}}_{hd}$ is infeasible, which is intuitive because $\tilde{\mathcal{P}}_{hd}$ has more patients than $\hat{\mathcal{P}}_{hd}^{(i)}$, while $\hat{\mathcal{P}}_{hd}^{(i)}$ is not already feasibly packable.
- $\tilde{\mathcal{P}}_{hd}$ subtracts some patients from $\hat{\mathcal{P}}_{hd}^{(i)}$ (i.e., $\hat{\mathcal{P}}_{hd}^{(i)} \supset \tilde{\mathcal{P}}_{hd}$ and $\hat{\mathcal{P}}_{hd}^{(i)}\backslash\tilde{\mathcal{P}}_{hd} \neq \emptyset$), yielding

$$\sum_{p\in\hat{\mathcal{P}}_{hd}^{(i)}} (1 - x_{hdp}) = \overbrace{\sum_{p\in\hat{\mathcal{P}}_{hd}^{(i)}\cap\tilde{\mathcal{P}}_{hd}} (1 - x_{hdp})}^{0} + \overbrace{\sum_{p\in\hat{\mathcal{P}}_{hd}^{(i)}\backslash\tilde{\mathcal{P}}_{hd}} (1 - x_{hdp})}^{\geq 1}$$
$$\Rightarrow \sum_{p\in\hat{\mathcal{P}}_{hd}^{(i)}\backslash\tilde{\mathcal{P}}_{hd}} (1 - x_{hdp}) \geq 1.$$

  Therefore, $\tilde{\mathcal{P}}_{hd}$ satisfies Inequality (19).
- $\tilde{\mathcal{P}}_{hd}$ shares some patients with $\hat{\mathcal{P}}_{hd}^{(i)}$ (i.e., $\hat{\mathcal{P}}_{hd}^{(i)} \cap \tilde{\mathcal{P}}_{hd} \neq \emptyset$), yielding

$$\sum_{p\in\hat{\mathcal{P}}_{hd}^{(i)}} (1 - x_{hdp}) = \overbrace{\sum_{p\in\hat{\mathcal{P}}_{hd}^{(i)}\cap\tilde{\mathcal{P}}_{hd}} (1 - x_{hdp})}^{0} + \overbrace{\sum_{p\in\hat{\mathcal{P}}_{hd}^{(i)}\backslash\tilde{\mathcal{P}}_{hd}} (1 - x_{hdp})}^{\geq 1}$$
$$\Rightarrow \sum_{p\in\hat{\mathcal{P}}_{hd}^{(i)}\backslash\tilde{\mathcal{P}}_{hd}} (1 - x_{hdp}) \geq 1.$$

  Therefore, $\tilde{\mathcal{P}}_{hd}$ satisfies Inequality (19).
- $\tilde{\mathcal{P}}_{hd}$ shares no patient with $\hat{\mathcal{P}}_{hd}^{(i)}$ (i.e., $\hat{\mathcal{P}}_{hd}^{(i)} \cap \tilde{\mathcal{P}}_{hd} = \emptyset$), yielding

$$\sum_{p\in\hat{\mathcal{P}}_{hd}^{(i)}} (1 - x_{hdp}) = \overbrace{\sum_{p\in\hat{\mathcal{P}}_{hd}^{(i)}\cap\tilde{\mathcal{P}}_{hd}} (1 - x_{hdp})}^{0} + \overbrace{\sum_{p\in\hat{\mathcal{P}}_{hd}^{(i)}\backslash\tilde{\mathcal{P}}_{hd}} (1 - x_{hdp})}^{\geq 1}$$
$$\Rightarrow \sum_{p\in\hat{\mathcal{P}}_{hd}^{(i)}\backslash\tilde{\mathcal{P}}_{hd}} (1 - x_{hdp}) \geq 1.$$

  Therefore, $\tilde{\mathcal{P}}_{hd}$ satisfies Inequality (19). □

This cut is generated with minimal computational effort because as soon as we realize an SP is infeasible, we incorporate this cut into the MP. This cut communicates no information to the MP regarding the adjustment of $\hat{Y}_{hd}^{(i)}$ due to $|\hat{\mathcal{P}}_{hd}^{(i)}|$. If Scenario 2 occurs for some SPs at the same iteration of the MP, we solve our SPs with $\min\{\bar{F}_{hd}^{(i)}, |\mathcal{R}_h|\}$. If $\bar{Y}_{hd}^{(i)} = \hat{Y}_{hd}^{(i)}$, we are at optimality, otherwise, we generate the following Benders' optimality cut from $\bar{\mathcal{J}}^{(i)}$:

$$y_{hd} \geq \bar{Y}_{hd}^{(i)} - \sum_{p\in\hat{\mathcal{P}}_{hd}^{(i)}} (1 - x_{hdp}) \qquad \forall (h, d) \in \bar{\mathcal{J}}^{(i)}. \tag{20}$$

The set of optimality cuts developed from $\bar{\mathcal{J}}^{(i)}$ is denoted $\mathcal{O}$. A different variant of this cut was initially developed for scheduling problems with makespan minimization (Hooker, 2007), which was subsequently modified and proven valid for packing problems (Fazel-Zarandi & Beck, 2012). The Benders' optimality cut is a bi-purpose cut which communicates to the MP two remedial strategies for eliminating the current infeasible MP solution and driving it towards optimality in future iterations: (1) open at least one more OR and/or (2) subtract at least one patient from $\hat{\mathcal{P}}_{hd}^{(i)}$. This cut is a valid cut (Theorem 2).

Theorem 2. *The Benders' optimality cut (Inequality (20)) is valid.*

Proof. In order to prove the validity of our Benders' cut, we first show that Inequality (20) eliminates the current MP sub-optimal solution (Property 1). Let $\hat{\mathcal{P}}_{hd}^{(i)}$ and $\hat{\mathcal{R}}_{hd}^{(i)}$ be the set of assigned patients and ORs to hospital h on day d, respectively, resulting in a sub-optimal MP solution ($\hat{x}_{hdp}^{(i)}$ and $\hat{y}_{hd}^{(i)}$). By definition, these binary variables take value 1 in the current iteration of the MP. Our Benders' optimality cut is

$$y_{hd} \geq \bar{Y}_{hd}^{(i)} - \sum_{p\in\hat{\mathcal{P}}_{hd}^{(i)}} (1 - x_{hdp}) \qquad \forall (h, d) \in \bar{\mathcal{J}}^{(i)},$$

which can be reformulated as

$$y_{hd} - \bar{Y}_{hd}^{(i)} + \left(\left|\hat{\mathcal{P}}_{hd}^{(i)}\right| - \sum_{p\in\hat{\mathcal{P}}_{hd}^{(i)}} x_{hdp}\right) \geq 0 \qquad \forall (h, d) \in \bar{\mathcal{J}}^{(i)}.$$

We show that the current sub-optimal MP solution ($\hat{x}_{phd}^{(i)}$ and $\hat{y}_{hd}^{(i)}$) does not satisfy Inequality (20) by instantiating the MP variables with the current sub-optimal MP solution. The following equation shows that the sub-optimal MP solution does not satisfy Inequality

(20), and therefore

$$\overbrace{\hat{y}^{(i)}_{hd} - \bar{Y}^{(i)}_{hd}}^{<0} + \overbrace{\left(\left|\hat{\mathcal{P}}^{(i)}_{hd}\right| - \sum_{p\in\hat{\mathcal{P}}^{(i)}_{hd}} \hat{x}^{(i)}_{hdp} \right)}^{0} < 0 \qquad \forall (h,d) \in \bar{\mathcal{J}}^{(i)}.$$

The current MP sub-optimal solution is cut off by our Benders optimality cut, because the combination of patients and the selection of ORs has not changed and $\bar{Y}^{(i)}_{hd} > \hat{y}^{(i)}_{hd}$.

We next show that our Benders' cut does not remove any globally feasible solutions in future iterations (Property 2). Consider a hypothetical future solution ($\tilde{x}_{hdp}$ and $\tilde{y}_{hd}$) consisting of a new set of patients ($\tilde{\mathcal{P}}_{hd}$) and ORs ($\tilde{\mathcal{R}}_{hd}$), leading to a feasible or infeasible future solution. We show that our Benders' optimality cut does not remove the new solution if it is feasible. By keeping the same selection of patients, our optimality cut is reduced to

$$y_{hd} - \bar{Y}^{(i)}_{hd} \geq 0 \qquad \forall (h,d) \in \bar{\mathcal{J}}^{(i)}. \tag{21}$$

There are two relational possibilities between $\tilde{y}_{hd}$ and $\bar{Y}^{(i)}_{hd}$: (1) $\tilde{y}_{hd} < \bar{Y}^{(i)}_{hd}$ (i.e., infeasible MP solution), in which Inequality (21) cuts off $\tilde{y}_{hd}$; and (2) $\tilde{y}_{hd} \geq \bar{Y}^{(i)}_{hd}$ (i.e., feasible MP solution) and $\tilde{y}_{hd}$ satisfies Inequality (21). Note that $\tilde{y}_{hd} \geq \bar{Y}^{(i)}_{hd}$ results in a potential feasible solution even if more patients are selected. By allowing more patients to be selected with an infeasible solution ($\tilde{y}_{hd} < \bar{Y}^{(i)}_{hd}, \hat{\mathcal{P}}^{(i)}_{hd} \subseteq \tilde{\mathcal{P}}_{hd}$), we can show that our Benders' optimality cut removes the solution using the relational possibilities between $\hat{\mathcal{P}}^{(i)}_{hd}$ and $\tilde{\mathcal{P}}_{hd}$ and $\hat{\mathcal{R}}^{(i)}_{hd}$ and $\tilde{\mathcal{R}}_{hd}$:

$$\overbrace{y_{hd} - \bar{Y}^{(i)}_{hd}}^{<0} + \overbrace{\left(\left|\hat{\mathcal{P}}^{(i)}_{hd}\right| - \left(\overbrace{\sum_{p\in\hat{\mathcal{P}}^{(i)}_{hd}\cap\tilde{\mathcal{P}}_{hd}} x_{hdp}}^{|\hat{\mathcal{P}}^{(i)}_{hd}|} + \overbrace{\sum_{p\in\hat{\mathcal{P}}^{(i)}_{hd}\setminus\tilde{\mathcal{P}}_{hd}} x_{hdp}}^{\emptyset} \right) \right)}^{0}$$
$$< 0 \qquad \forall (h,d) \in \bar{\mathcal{J}}^{(i)}. \tag{22}$$

Note that if the directionality of any of the subsets in this scenario is reversed, then the corresponding term in Inequality (22) will assume a value ≥ 1, and thus no feasible solution is removed. □

4.4.2. LBBD2

Algorithmically, LBBD2 resembles LBBD1 with the exception that the Benders' feasibility cut (19) is replaced with

$$y_{hd} \geq (|\mathcal{R}_h| + 1) - \sum_{p\in\hat{\mathcal{P}}^{(i)}_{hd}} (1 - x_{hdp}) \qquad \forall (h,d) \in \bar{\mathcal{U}}^{(i)}. \tag{23}$$

This cut communicates more information to the MP compared to Benders' feasibility cut (19) when the SP is infeasible. If the SP is infeasible, we set $\bar{Y}^{(i)}_{hd}$ to $|\mathcal{R}_h|$ for Benders' cut (23). Based on the structure of the new Benders' feasibility cut, we ensure that at least one patient is subtracted from $\hat{\mathcal{P}}^{(i)}_{hd}$ if $\bar{Y}^{(i)}_{hd} = |\mathcal{R}_h|$. The strength of this cut becomes more evident when $\hat{Y}^{(i)}_{hd} < |\mathcal{R}_h|$ and the SP is infeasible. To enable the MP to break the SP infeasibility, the Benders' cut adds constraints to the MP to eliminate infeasible SP solutions from the MP feasible space. We refer to the ways that the added Benders' cuts can break the MP infeasibility as "remedial strategies". In this case, remedial strategies are to (1) remove at least two patients from $\hat{\mathcal{P}}^{(i)}_{hd}$ or (2) remove at least one patient from $\hat{\mathcal{P}}^{(i)}_{hd}$ and/or open at least one more OR. The computational effort to derive this cut is equivalent to that of the Benders' feasibility cut in LBBD1. This cut is theoretically stronger because it exploits the relationship between $\hat{\mathcal{P}}^{(i)}_{hd}$, $\hat{Y}^{(i)}_{hd}$, and $\bar{Y}^{(i)}_{hd}$.

4.4.3. LBBD3

Unlike LBBD1 and LBBD2 that are able to communicate with the MP in the face of infeasible SPs, LBBD3 cannot communicate with the MP unless the SP is feasible and optimally solvable, whether through the FFD heuristic (when $\hat{Y}^{(i)}_{hd} = \bar{F}^{(i)}_{hd}$, since $\hat{Y}^{(i)}_{hd} \leq \bar{Y}^{(i)}_{hd} \leq \bar{F}^{(i)}_{hd}$) or by actually solving SPs (when $\hat{Y}^{(i)}_{hd} \neq \bar{F}^{(i)}_{hd}$). We incorporate the following Benders' optimality cut into the MP when $\bar{Y}^{(i)}_{hd} \neq \hat{Y}^{(i)}_{hd}$:

$$y_{hd} \geq \bar{Y}^{(i)}_{hd} - \sum_{p\in\hat{\mathcal{P}}^{(i)}_{hd}} (1 - x_{hdp}) \qquad \forall (h,d) \in \bar{\mathcal{J}}^{(i)}. \tag{24}$$

Note that this inequality is the same optimality cut used in LBBD1 (Inequality (20)). This cut is solely developed for $\bar{\mathcal{J}}^{(i)}$ because the FFD solution ($\bar{F}^{(i)}_{hd}$) provides a feasible upper bound to each SP. LBBD3 is likely to experience a delayed convergence to optimality if SPs are difficult to solve. Thus, if all SP optimal solutions are not obtainable within the time limit, we cannot guarantee a globally feasible/optimal solution. Like LBBD3, LBBD1 and LBBD2 may also experience a delayed convergence due to the incorporation of weaker Benders' cuts. The advantage of using $\bar{F}^{(i)}_{hd}$ for feasible SPs is the decrease in the number of variables and constraints because for each feasible SP, $\bar{F}^{(i)}_{hd} \leq \bar{Y}^{(i)}_{hd}$. The disadvantage of using $\bar{F}^{(i)}_{hd}$ for infeasible SPs is the size dimensionality increase from adjusting $|\mathcal{R}_h|$ to $\bar{F}^{(i)}_{hd}$ because $\bar{F}^{(i)}_{hd} > |\mathcal{R}_h|$. Theoretically, we require substantial computational effort to derive this cut as the optimal solution of the SP is required. LBBD3 is expected to outperform other LBBDs if the SPs are easy to solve.

LBBD3 is to some extent a similar, but rectified, version of the LBBD developed in Fazel-Zarandi and Beck (2012), in the sense that if $\hat{Y}^{(i)}_{hd} \neq \bar{F}^{(i)}_{hd}$, we use $\bar{F}^{(i)}_{hd}$ as an upper bound on the minimum number of ORs required. When $\hat{Y}^{(i)}_{hd} \neq \bar{F}^{(i)}_{hd}$, the constraint programming SP in Fazel-Zarandi and Beck (2012) is solved via a series of satisfaction problems until a feasible solution is found. The issue with the LBBD in Fazel-Zarandi and Beck (2012) is that when the CP does not provide a feasible solution, the authors set the minimum number of trucks (ORs) in each SP equal to the maximum number of trucks ($|\mathcal{R}_h|$) that can be assigned to that facility (hospital-day) (see Fazel-Zarandi and Beck, 2012, p. 7). This strategy may direct their LBBD towards accepting an infeasible solution as the optimal solution when the MP optimal solution is also equal to $|\mathcal{R}_h|$.

Setting the solutions of infeasible SPs equal to $|\mathcal{R}_h|$ causes the LBBD to accept all infeasible SPs as optimal because $\bar{Y}^{(i)}_{hd} = \hat{Y}^{(i)}_{hd} = |\mathcal{R}_h|$ and to stop the search if no sub-optimal SP exists. Therefore, the LBBD developed in Fazel-Zarandi and Beck (2012) produces an accurate optimal solution if and only if the optimal solutions of all SPs are available through CP optimizer. Otherwise, their LBBD cannot handle infeasible SPs due to the lack of a Benders' feasibility cut. One may avoid accepting an infeasible solution as an optimal solution if the MP solution results in at least one sub-optimal SP in addition to other infeasible SPs.

4.5. Cut propagation for LBBDs

While it is customary in LBBD to only generate one cut from each infeasible SP, the nature of DORS allows us to generate multiple Benders' feasibility cuts from a single infeasible SP. If a combination of patients with known surgical times are not feasibly packable within the OR time of a certain hospital-day, it cannot be packed into other hospital-days with less or the same OR time. Hence, the feasibility cut is generated for all SPs whose OR availability times are less than or equal to that of the infeasible SP. Therefore, up to $|\mathcal{H}||\mathcal{D}|$ Benders' cuts can possibly be generated from a single infeasible SP. Similarly to feasibility cuts, more than one optimality cut can be developed for other hospital-days. We call this process Benders' cut propagation or simply propagation. Propagation will remove many integer solutions of the MP that

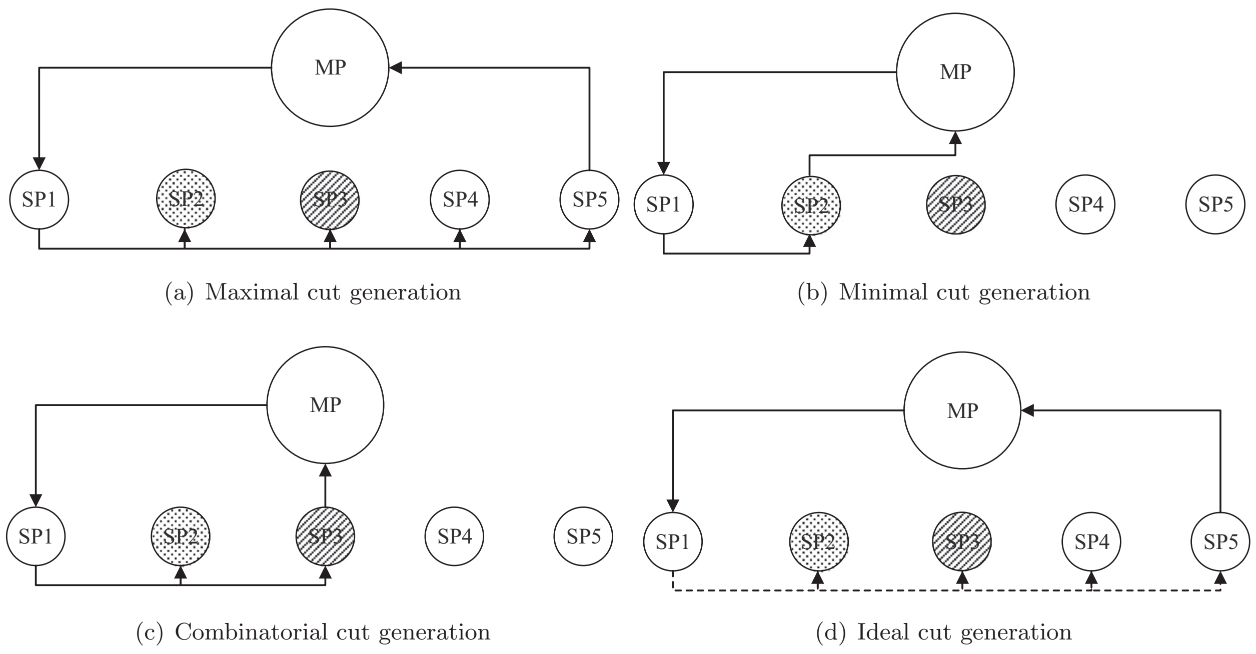

MP
SP1
SP2
SP3
SP4
SP5
(a) Maximal cut generation
(b) Minimal cut generation
(c) Combinatorial cut generation
(d) Ideal cut generation

Fig. 3. LBBD implementation techniques. Unshaded SPs are optimally solved, dotted SPs are sub-optimally solved, and striped SPs are infeasible. Dashed lines indicate that SP feasibility is checked before an SP is solved.

are infeasible from the solutions of other SPs. LBBD1, LBBD2, and LBBD3 with propagation are called LBBD1$_P$, LBBD2$_P$, and LBBD3$_P$, respectively.

4.6. LBBD implementation techniques

We discuss four LBBD implementations (Fig. 3) based on when to generate Benders' cuts and how to solve the SPs. The computational complexity of the MP and SP determines which of these implementations should be adopted for solving the optimization problem.

The maximal LBBD implementation (Fig. 3(a)) solves all SPs sequentially and finally communicates all possible cuts (optimality and/or feasibility) to the MP (Fazel-Zarandi & Beck, 2012). This implementation is assumed to be appropriate when the MP is computationally more burdensome than the SPs. The minimal LBBD (Fig. 3(b)) implementation sequentially solves SPs until it encounters an infeasible or a sub-optimal SP. Then, a cut from that SP is generated and added to the MP. We hypothesize that the minimal LBBD implementation is appropriate when the MP variable domain is small, SPs are hard to solve, and increased communication between the MP and SP guarantees faster convergence.

The combinatorial LBBD implementation (Fig. 3(c)) is an intersection of the maximal and minimal implementations. SP exploration is stopped when the LBBD encounters an infeasible SP, at which point the feasibility cut and all optimality cuts from explored sub-optimal SPs are incorporated into the MP. Combinatorial implementation is assumed to be appropriate when SPs are easy and providing information on infeasible SPs may help the MP converge faster. Therefore, the model can incorporate more information into the MP from sub-optimal SPs solved prior to the infeasible SP.

The ideal implementation (Fig. 3(d)) checks the feasibility of all SPs by setting the objective function coefficients of each SP to a dummy constant zero (Harjunkoski & Grossmann, 2002). Unlike the existing ideal implementation (Harjunkoski & Grossmann, 2002) that uses only feasibility cuts, we use both feasibility and optimality cuts. If any SPs are infeasible, we do not solve any SPs and instead re-solve the MP with new feasibility cuts from the infeasible SPs. Once all SPs are feasible, all SPs are solved sequentially, and all possible optimality cuts are passed to the MP. We anticipate that the ideal implementation will be best when the SPs are hard. Note that since LBBD3 and LBBD3$_P$ are always feasible due to the FFD heuristic upper bound feasible solution, they cannot benefit from combinatorial and ideal implementations due to the lack of a Benders' feasibility cut.

If all SPs are solvable in just few seconds, as in Fazel-Zarandi and Beck (2012), all LBBD implementation variants will probably yield the same performance and the most determinant factor for LBBD performance will be the cut strength and cut propagation mechanism. Unlike the easy packing SPs in Fazel-Zarandi and Beck (2012), the DORS SP difficulty will give the LBBD implementation a significant impact on performance.

5. Computational results

The test cases were run on a Windows XP computer with an Intel(R) Core(TM)2 Duo 3.00 gigahertz CPU with 8 gigabytes RAM. Gurobi Optimizer (Gurobi, Inc.) v5.63 is used for optimization. In order to evaluate the performances of our LBBDs, we generate two sets of data, each with three hospitals, 20–160 patients, and a one-week planning horizon, where there are five days per week. One dataset has three ORs per hospital (yielding easy SPs but a hard MP) and the other has five ORs (yielding hard SPs but an easy MP). Five trials with different random surgical times are generated for each test case. We compare IP+Gurobi optimal solutions to the LBBD approaches, with a maximum run time of 7200 seconds. We define the best-performing approach as the one with the best ability to find optimal solutions (robustness) within the maximum time limit; ties are broken by CPU time.

The probability distribution to generate random surgical times follows the truncated normal distribution with a mean of 160 minutes, a standard deviation of 40 minutes, and a lower bound of 45 and an upper bound of 480 minutes. The average number of surgical cases was obtained from 7500 elective and emergency patients from 2011 to 2013 in the General Surgery Departments of the University Health Network (UHN) hospitals (Toronto, Ontario, Canada), amounting to an average of 72 operations per week.

Table 2
Parameter values.

κ_1	50 dollars
κ_2	5 dollars
Γ	500 (6–10% of all patients identified as mandatory)
ρ_p	Uniform distribution [1, 5], where 1 is least urgent 5 is the most urgent
B_{hd}	Uniform distribution [420, 480] minutes in 15-minutes intervals
α_p	Uniform distribution [60, 120] days
F_{hd}	Uniform distribution [4000, 6000]
G_{hd}	Uniform distribution [1500, 2500]

Similarly to Canadian Triage and Acuity Scale (CTAS) that uses a 1-to-5 scale to rank the acuity of emergency patients, we use a 1-to-5 scale for ranking the health status score of elective patients, though in our model lower values indicate lower acuity to be consistent with the minimization objective. The DORS parameter values are shown in Table 2.

Table 3 shows that the average LBBD CPU times are more robust than IP+Gurobi and are at least one order of magnitude, and usually two orders of magnitude, faster. Interestingly, the robustness of LBBDs increases as we increase the number of ORs per hospital-day, whereas the IP+Gurobi robustness decreases.

Single-cut LBBDs which incorporate only optimality cuts (LBBD3 and LBBD3$_{\mathrm{P}}$) are consistently worse than bi-cut LBBDs with strong feasibility cuts (LBBD2 and LBBD2$_{\mathrm{P}}$) in terms of robustness on both hard (Table 4) and easy (Table 5) SP datasets. Strong feasibility cuts (LBBD2 and LBBD2$_{\mathrm{P}}$) consistently outperform weak feasibility cuts (LBBD1 and LBBD1$_{\mathrm{P}}$) in both robustness and CPU time on both datasets. On grand average, our cut propagation mechanism improved both robustness and CPU time in 67% and 33% of trials on hard and easy SP datasets, respectively. In terms of robustness across all LBBDs, the implementations ranked best to worst are minimal, maximal, combinatorial, and ideal with 21–27 and 31–39 unsolved trials for hard and easy SP datasets, respectively.

Since LBBD3 and LBBD3$_{\mathrm{P}}$ use the heuristic $\bar{F}_{hd}^{(i)}$ as a feasible upper bound, their SPs are always feasible, and due to their algorithmically inflexible structure, they yield very close robustness and CPU time performances under maximal, combinatorial, and ideal implementations for both datasets. The detailed LBBD statistics regarding number of iterations, feasibility, and optimality cuts are given in Appendix A.

5.1. Hard SP datasets

The best LBBD for solving DORS with hard SPs is LBBD1$_{\mathrm{P}}$ with the maximal implementation (Table 4). LBBD1$_{\mathrm{P}}$ is the most robust LBBD across all implementations on hard SP datasets with an average of two unsolved trials per implementation, whereas LBBD3 is the least robust LBBD across all implementations with an average of seven unsolved trials per implementation. All bi-cut LBBDs (LBBD1, LBBD1$_{\mathrm{P}}$, LBBD2, and LBBD2$_{\mathrm{P}}$) outperform single-cut LBBDs (LBBD3 and LBBD3$_{\mathrm{P}}$) in terms of robustness both within each implementation and across all implementations on hard SP datasets. For hard SP datasets, LBBDs with weak Benders' feasibility cuts (LBBD1 and LBBD1$_{\mathrm{P}}$) spend a significant portion of CPU time solving the MP, while other LBBDs spend a high portion of CPU time solving the SPs (Fig. 4).

5.2. Easy SP datasets

The best LBBD for solving DORS with easy SPs is LBBD2$_{\mathrm{P}}$ with the minimal implementation (Table 5). LBBD2$_{\mathrm{P}}$ is the most robust LBBD across all implementations on easy SP datasets, with an average of four unsolved trials per implementation, whereas LBBD1 is the least robust LBBD across all implementations with an average of 8.25 unsolved trials per implementation. Propagating LBBDs outperform non-propagating LBBDs in terms of robustness and CPU time on 67% and 33% of trials on easy SP datasets. Strong feasibility cuts (LBBD2 and LBBD2$_{\mathrm{P}}$) outperform weak feasibility cuts (LBBD1 and LBBD1$_{\mathrm{P}}$). Single-cut LBBDs (LBBD3 and LBBD3$_{\mathrm{P}}$) outperform bi-cut LBBDs with weak feasibility cuts (LBBD1, LBBD1$_{\mathrm{P}}$) in terms of robustness both within each implementation and across all implementations (Table 5). An interesting observation is the lower CPU time per iteration of LBBD2$_{\mathrm{P}}$ compared to LBBD1$_{\mathrm{P}}$, which is likely attributable to the propagation of stronger feasibility cuts, resulting in LBBD2$_{\mathrm{P}}$'s faster convergence even in the instances for which the LBBD2$_{\mathrm{P}}$ requires more iterations (Table 9 in Appendix A). The presence of asymmetric OR availability times in the network makes the performance of LBBD2$_{\mathrm{P}}$ less predictable in terms of number of iterations because we are unaware of the path that Gurobi takes after cuts are added to the MP. We anticipate that in a symmetric DORS environment, LBBD2$_{\mathrm{P}}$ will outperform LBBD1$_{\mathrm{P}}$ with both fewer iterations and less CPU time.

Although the minimal implementation has the fewest unsolved trials (31), LBBD1 and LBBD1$_{\mathrm{P}}$ with the minimal implementation could not solve test cases of 140 patients, whereas the same LBBDs could solve at least two of five trials of 140 patients with the maximal implementation (Table 5). Neither the LBBDs nor IP+Gurobi could solve trials of 160 patients within the time limit due to MP computational difficulty. On grand average, LBBD2 and LBBD3 are tied for best robustness, but LBBD3 is 32.96% faster. As expected for easy SP datasets, all LBBDs spend a significant portion

Table 3
Average CPU time (seconds) of LBBD implementations and IP+Gurobi over five trials (bold is the best performance in each test case; superscripts are the average number of unsolved trials per instance; the average CPU time of each implementation is taken over five trials for each of the six LBBDs).

	$\lvert\mathcal{P}\rvert$	Maximal	Minimal	Combinatorial	Ideal	LBBD grand average	IP+Gurobi
Five ORs	20	0.39	0.39	0.37	0.39	**0.39**	13.64
	40	7.04	0.93	6.20	7.08	**5.31**	68.57$^{(1.00)}$
	60	7.27	4.03	8.06	7.38	**6.69**	305.72$^{(3.00)}$
	80	21.07	183.55	35.07	15.54	**63.81**	2233.70$^{(3.00)}$
	100	109.43	103.79	140.83	129.63	**120.92**	******$^{(5.00)}$
	120	307.17$^{(0.67)}$	397.15$^{(0.50)}$	362.76$^{(0.50)}$	364.60$^{(0.67)}$	**357.92**$^{(0.59)}$	******$^{(5.00)}$
	140	1367.39$^{(1.67)}$	1227.58$^{(1.17)}$	926.42$^{(1.83)}$	1132.72$^{(1.67)}$	**1163.53**$^{(1.59)}$	******$^{(5.00)}$
	160	1263.14$^{(1.50)}$	1953.05$^{(1.83)}$	1744.88$^{(1.83)}$	1496.17$^{(2.17)}$	**1614.31**$^{(1.83)}$	******$^{(5.00)}$
Three ORs	20	0.49	0.49	0.52	0.45	**0.49**	14.40
	40	8.79	7.08	9.05	9.14	**8.52**	5500.00$^{(3.00)}$
	60	11.17	11.16	11.10	13.98	**11.85**	2904.60$^{(1.00)}$
	80	133.38	299.93	285.60	161.05	**1219.99**	5718.90$^{(4.00)}$
	100	554.65$^{(0.83)}$	1451.79$^{(0.33)}$	1210.53$^{(0.50)}$	976.55$^{(0.83)}$	**1048.38**$^{(0.62)}$	6500.90$^{(4.00)}$
	120	711.76$^{(2.00)}$	1589.68$^{(1.17)}$	1380.80$^{(1.83)}$	1755.94$^{(2.00)}$	**1359.55**$^{(1.75)}$	5934.00$^{(4.00)}$
	140	929.89$^{(3.17)}$	1282.93$^{(3.67)}$	731.80$^{(3.83)}$	1970.31$^{(3.67)}$	**1228.73**$^{(3.59)}$	4099.70$^{(4.00)}$

Table 4
Five ORs: average CPU time (seconds) of LBBDs over five trials (bold is the best performance in each test case; superscripts are the numbers of unsolved trials; highlight indicates improved performance with cut propagation).

	$\|\mathcal{P}\|$	LBBD1	LBBD1$_P$	LBBD2	LBBD2$_P$	LBBD3	LBBD3$_P$
Maximal	20	0.40	**0.38**	**0.38**	**0.38**	0.40	0.39
	40	0.92	0.97	**0.87**	8.49	**0.87**	30.13
	60	5.12	13.10	4.84	**3.30**	4.45	12.85
	80	62.28	18.11	**3.07**	5.59	13.34	24.00
	100	126.74	95.26	**36.73**	44.31	187.17	166.36
	120	309.55	810.01	**113.27**	176.36$^{(1)}$	119.94$^{(1)}$	313.88$^{(2)}$
	140	1770.20$^{(2)}$	**2060.60**	883.87$^{(2)}$	391.04$^{(1)}$	2229.10$^{(3)}$	869.50$^{(2)}$
	160	1164.30$^{(1)}$	1422.80$^{(1)}$	1629.10$^{(1)}$	**372.70**$^{(1)}$	2339.70$^{(3)}$	650.22$^{(2)}$
Average		429.94$^{(3)}$	**552.65**$^{(1)}$	334.02$^{(3)}$	125.27$^{(3)}$	611.87$^{(7)}$	258.42$^{(6)}$
Minimal	20	0.39	**0.38**	**0.38**	**0.38**	0.40	0.40
	40	0.87	**0.61**	0.85	1.05	0.86	1.31
	60	7.08	**1.66**	3.15	2.57	6.84	2.88
	80	4.21	**1.43**	94.08	33.76	726.50	241.34
	100	144.75	65.67	45.00	**39.55**	247.58	80.21
	120	653.11$^{(1)}$	**259.64**	557.61	421.86	126.07$^{(1)}$	364.62$^{(1)}$
	140	2336.90$^{(1)}$	1325.70$^{(1)}$	632.97$^{(1)}$	**400.28**$^{(1)}$	1149.20$^{(2)}$	1520.40$^{(1)}$
	160	**3548.90**	1944.20$^{(1)}$	1373.70$^{(1)}$	275.40$^{(2)}$	3907.30$^{(4)}$	665.82$^{(3)}$
Average		837.03$^{(2)}$	449.91$^{(2)}$	**338.47**$^{(2)}$	146.74$^{(3)}$	770.59$^{(7)}$	359.62$^{(5)}$
Combinatorial	20	0.37	0.37	0.37	**0.36**	0.38	0.37
	40	0.87	0.94	0.86	3.61	**0.85**	30.04
	60	7.41	17.95	3.17	**2.52**	4.43	12.88
	80	**4.04**	41.61	94.13	33.55	13.14	23.96
	100	133.50	272.98	45.05	**39.30**	187.47	166.70
	120	534.61	897.12	200.85	**110.67**	119.67$^{(1)}$	313.63$^{(2)}$
	140	655.00$^{(2)}$	755.93$^{(1)}$	**627.27**$^{(1)}$	430.06$^{(2)}$	2222.10$^{(3)}$	868.18$^{(2)}$
	160	3506.00$^{(1)}$	**2184.10**$^{(1)}$	1468.80$^{(2)}$	274.13$^{(2)}$	2402.60$^{(3)}$	633.66$^{(2)}$
Average		605.23$^{(3)}$	**521.38**$^{(2)}$	269.08$^{(3)}$	111.84$^{(4)}$	618.83$^{(7)}$	256.18$^{(6)}$
Ideal	20	0.40	**0.38**	0.39	0.41	0.40	**0.38**
	40	0.92	0.98	**0.89**	8.64	0.91	30.16
	60	5.16	13.24	4.99	**3.43**	4.51	12.93
	80	**4.30**	37.90	6.92	6.40	13.48	24.26
	100	270.79	**82.22**	31.98	38.24	187.59	166.96
	120	382.33$^{(1)}$	**303.33**	528.36	541.15	119.50$^{(1)}$	312.92$^{(2)}$
	140	**517.50**$^{(1)}$	1443.70$^{(1)}$	605.73$^{(2)}$	1111.10$^{(1)}$	2247.40$^{(3)}$	870.90$^{(2)}$
	160	2788.9$^{(2)}$	2487.90$^{(2)}$	**377.90**$^{(1)}$	253.91$^{(3)}$	2404.50$^{(3)}$	663.91$^{(2)}$
Average		496.29$^{(4)}$	546.33$^{(3)}$	**196.65**$^{(3)}$	245.41$^{(4)}$	622.29$^{(7)}$	260.30$^{(6)}$
Grand average		592.12$^{(3.00)}$	**516.08**$^{(2.00)}$	888.61$^{(2.75)}$	158.54$^{(3.5)}$	655.90$^{(7.00)}$	283.63$^{(5.75)}$

of CPU time solving the MP (above 95% on average), while they spend a very small portion of CPU time solving the SPs (Fig. 5).

5.3. Value of cut strength

While strong feasibility cuts (LBBD2 and LBBD2$_P$) consistently outperform weak feasibility cuts (LBBD1 and LBBD1$_P$) on both hard and easy SP datasets, we can quantify the value of cut strength by examining the maximal implementation on easy SP datasets. The stronger feasibility cut in LBBD2 reduces the number of iterations, feasibility cuts, and optimality cuts over LBBD1 by an average of 31.94%, 37.39%, and 30.77%, respectively, causing LBBD2 to be 4 and 1.44 times faster than LBBD1 in solving MP and SPs on easy SP datasets, respectively, leading to faster overall convergence (Table 6).

5.4. Value of bi-cut versus single-cut LBBDs

Bi-cut LBBDs on hard SP datasets have significantly varying number of iterations, feasibility cuts, and optimality cuts across all implementations (Table 8 in Appendix A); therefore, they do not lend themselves well to the same analysis used to quantify cut strength. Instead, we explicitly show the impact of incorporating both feasibility and optimality cuts (LBBD2) compared to only optimality cuts (LBBD3) on easy SP datasets. The rationale for choosing these two algorithms is that they have the same number of unsolved trials, iterations, and Benders' cuts, and nearly identical MP CPU time.

Table 6 shows that if no heuristic feasible upper bound is considered for SPs, the MP solution results in infeasible SPs in 99.63% ($|\mathcal{F}|_{\text{LBBD2}}/|\mathcal{O}|_{\text{LBBD3}}$) of trials for the maximal implementation. By a simple SP CPU time comparison between LBBD2 and LBBD3, we can see that LBBD3's average SP CPU time (22.07 seconds) is 1.63x higher than LBBD2's average SP CPU time (13.52 seconds). Equality in the number of cuts also shows that our Benders' feasibility cut (Inequality 23) is as strong as our Benders' optimality cut (Inequality (20)).

5.5. Surgical suite utilization

Table 7 shows the utilization of ORs and surgical suites with five, three, and two ORs. OR and surgical suite utilization is computed by the division of the mean number of opened ORs and surgical suites over total number of available ORs and surgical suites ($|\mathcal{H}| \times |\mathcal{D}| = 15$) in the network. While accommodating all patients in the current planning horizon in all but one scenario, network OR utilization is less than 70% for all five-OR scenarios, and for all but the three highest patient demands in the three-OR scenario.

Table 5
Three ORs: Average CPU time (seconds) of LBBDs over five trials (bold is the best performance in each test case; superscripts are the numbers of unsolved trials; asterisks are scenarios with no solved trials; highlight indicates improved performance with cut propagation).

	$\lvert\mathcal{P}\rvert$	LBBD1	LBBD1$_P$	LBBD2	LBBD2$_P$	LBBD3	LBBD3$_P$
Maximal	20	0.64	**0.45**	**0.45**	0.47	0.48	0.46
	40	**2.56**	6.84	10.59	10.73	10.82	11.17
	60	19.18	11.39	9.45	**8.94**	9.10	8.98
	80	183.22	297.55	**53.82**	103.97	55.05	106.65
	100	625.15$^{(2)}$	1581.00$^{(1)}$	**385.59**	135.47$^{(1)}$	402.16	138.54$^{(1)}$
	120	662.36$^{(2)}$	673.15$^{(2)}$	**291.47**$^{(2)}$	1175.5$^{(2)}$	295.80$^{(2)}$	1172.30$^{(2)}$
	140	2546.40$^{(3)}$	2276.30$^{(3)}$	305.51$^{(3)}$	51.95$^{(3)}$	341.57$^{(3)}$	57.60$^{(4)}$
Average		577.07$^{(7)}$	692.38$^{(6)}$	**150.98**$^{(5)}$	212.43$^{(6)}$	159.28$^{(5)}$	213.67$^{(7)}$
Minimal	20	0.65	0.46	**0.44**	0.45	0.45	0.46
	40	7.89	13.99	6.72	**3.49**	6.82	3.55
	60	12.69	16.87	9.41	**9.22**	9.45	9.32
	80	340.15	985.98	**52.44**	183.01	53.16	184.86
	100	2160.60$^{(1)}$	2440.60$^{(1)}$	**756.49**	1300.10	766.33	1286.6
	120	3668.30$^{(3)}$	3033.50$^{(2)}$	1033.10$^{(1)}$	**377.26**	1048.80$^{(1)}$	376.91
	140	******$^{(5)}$	******$^{(5)}$	**1086.40**$^{(3)}$	1476.20$^{(3)}$	1090.90$^{(3)}$	1479.10$^{(3)}$
Average		1031.71$^{(9)}$	1081.90$^{(8)}$	**420.71**$^{(4)}$	478.53$^{(3)}$	425.13$^{(4)}$	477.26$^{(3)}$
Combinatorial	20	0.60	0.52	0.51	0.50	0.50	**0.48**
	40	7.90	14.41	7.17	**3.71**	10.24	10.84
	60	12.96	16.97	9.48	9.23	9.06	**8.92**
	80	340.22	977.06	53.79	185.93	**52.47**	104.12
	100	2200.30$^{(1)}$	2480.00$^{(1)}$	768.24	1289.60	**388.91**	136.11$^{(1)}$
	120	3916.60$^{(3)}$	1280.10$^{(3)}$	950.97$^{(1)}$	**637.10**	300.12$^{(2)}$	1199.90$^{(2)}$
	140	******$^{(5)}$	******$^{(5)}$	1091.40$^{(3)}$	1488.80$^{(3)}$	297.87$^{(3)}$	49.12$^{(4)}$
Average		1079.76$^{(9)}$	794.84$^{(9)}$	411.65$^{(4)}$	**516.41**$^{(3)}$	151.314$^{(5)}$	215.64$^{(7)}$
Ideal	20	0.56	0.45	**0.42**	0.43	0.43	**0.42**
	40	**2.97**	14.56	10.99	6.52	9.98	9.84
	60	31.13	16.78	9.41	8.83	8.96	8.72
	80	243.60	469.33	53.97	**50.55**	51.93	96.91
	100	3147.40$^{(2)}$	1847.10$^{(1)}$	106.90$^{(1)}$	239.69	**386.11**	132.12$^{(1)}$
	120	4191.20$^{(2)}$	2853.50$^{(3)}$	960.99$^{(2)}$	**837.62**$^{(1)}$	309.02$^{(2)}$	1383.30$^{(2)}$
	140	7158.40$^{(4)}$	******$^{(5)}$	1277.30$^{(3)}$	1046.10$^{(3)}$	320.84$^{(3)}$	48.90$^{(4)}$
Average		2110.77$^{(8)}$	866.95$^{(9)}$	345.71$^{(6)}$	**312.82**$^{(4)}$	155.32$^{(5)}$	240.03$^{(7)}$
Grand average		1199.83$^{(8.25)}$	859.02$^{(8.00)}$	332.26$^{(4.75)}$	**380.05**$^{(4.00)}$	222.76$^{(4.75)}$	286.65$^{(6.00)}$

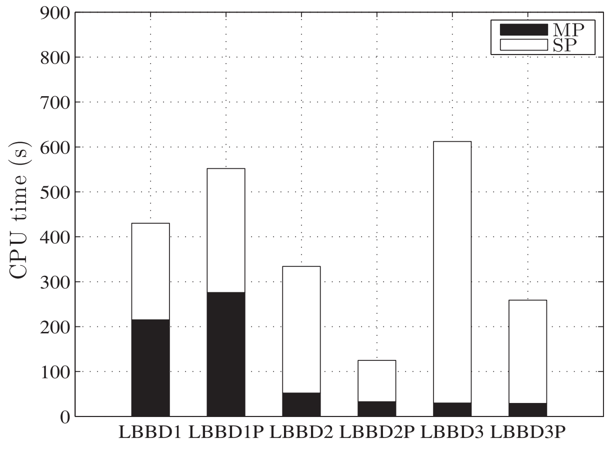
Network surgical suite utilization is similarly below 70% until 100 patients and three ORs. Within individual hospitals, the utilization of opened ORs is high, ranging from 81.4% to 96.7%. To show how the IP selectively chooses patients for the current planning horizon, we run more constrained scenarios with two ORs per hospital-day and 100–160 patients, in which 2.8% to 36.0% of patients are scheduled for the next planning horizon. All test cases with two ORs and less than 100 patients scheduled all patients.

6. Discussion

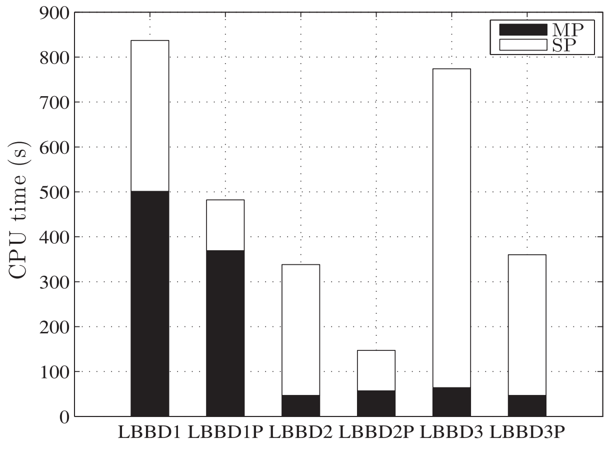
While we present an exact approach to solving large-scale distributed scheduling problem, there are a wide variety of existing heuristics (Behnamian & Fatemi Ghomi, 2013; Naderi & Ruiz, 2010; 2014) and sub-optimal column generation approaches (Fei et al., 2009; Lamiri et al., 2008) that have been applied to scheduling problems of various size (200–500 "jobs", 10–20 "machines", and 5–7 "locations") to obtain sub-optimal solutions in 10–70 minutes. Notably, Fei et al. (2009) and Lamiri et al. (2008) applied their column generation heuristics to operating room scheduling problems consisting of assigning a maximum of 160 and 210 patients to 6 and 12 ORs per day over a five-day planning horizon, respectively. The problems considered in Fei et al. (2009) are substantially smaller in size-dimensionality than our DORS problem due to our consideration of multiple hospitals, while the problems considered in Lamiri et al. (2008) are almost as large as our DORS problem.

Hooker (2007) optimally solved small- to medium-sized instances (up to 45 tasks and eight facilities) of a minimum makespan problem via LBBD in less than 96 minutes, but did not experiment with larger scenarios. Fazel-Zarandi and Beck (2012), whose work is most closely related to ours, applied an LBBD to the dataset of Albareda-Sambola et al. (2009), with a maximum of 40 clients, 20 facilities, and 10 allowable trucks per facility, and found that their LBBD, which took at most 6 hours to run, was at least two orders of magnitude faster than the IP+CPLEX approach taken by Albareda-Sambola et al. (2009). For our most similar problem instance (40 patients, 15 surgical suites (facilities), and five ORS (trucks) per surgical suite), our LBBDs took only 5.31 seconds on average. Further, we additionally solved instances as large as 160 patients, 15 surgical suites, and five ORs per surgical suite in under 27 minutes on average.

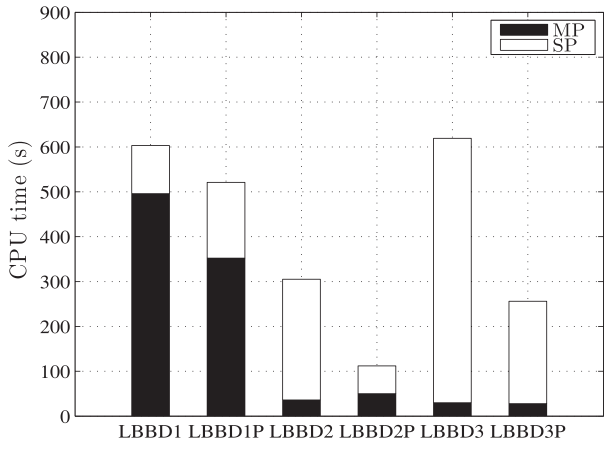
Conceptually, LBBD3 with the maximal implementation is to a large extent similar to the existing LBBD developed for location-allocation problems (Fazel-Zarandi & Beck, 2012), though we use mathematical programming instead of constraint programming to solve our SPs and we solve each SP only once at each iteration of the LBBD. Our results for LBBD3 are consistent with the existing claims that this type of LBBD works best when SPs constitute only 1% of the total CPU time (Fazel-Zarandi & Beck, 2012). However, for hard SP datasets, LBBD3 was the absolute worst-performer of all LBBDs across all implementations, demonstrating that the LBBD algorithmic structure has to change according to the complexity of the MP and SPs.



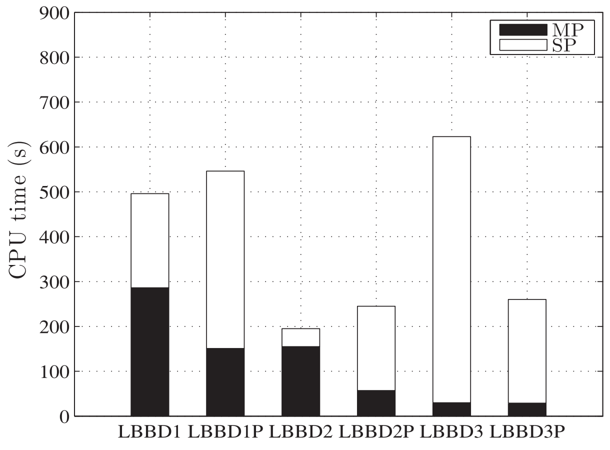
(a) Maximal cut generation (23 unsolved trials)



(b) Minimal cut generation (21 unsolved trials)



(c) Combinatorial cut generation (25 unsolved trials)



(d) Ideal cut generation (27 unsolved trials)

Fig. 4. Five ORs: breakdown of average total CPU time between MP and SP over solved trials.

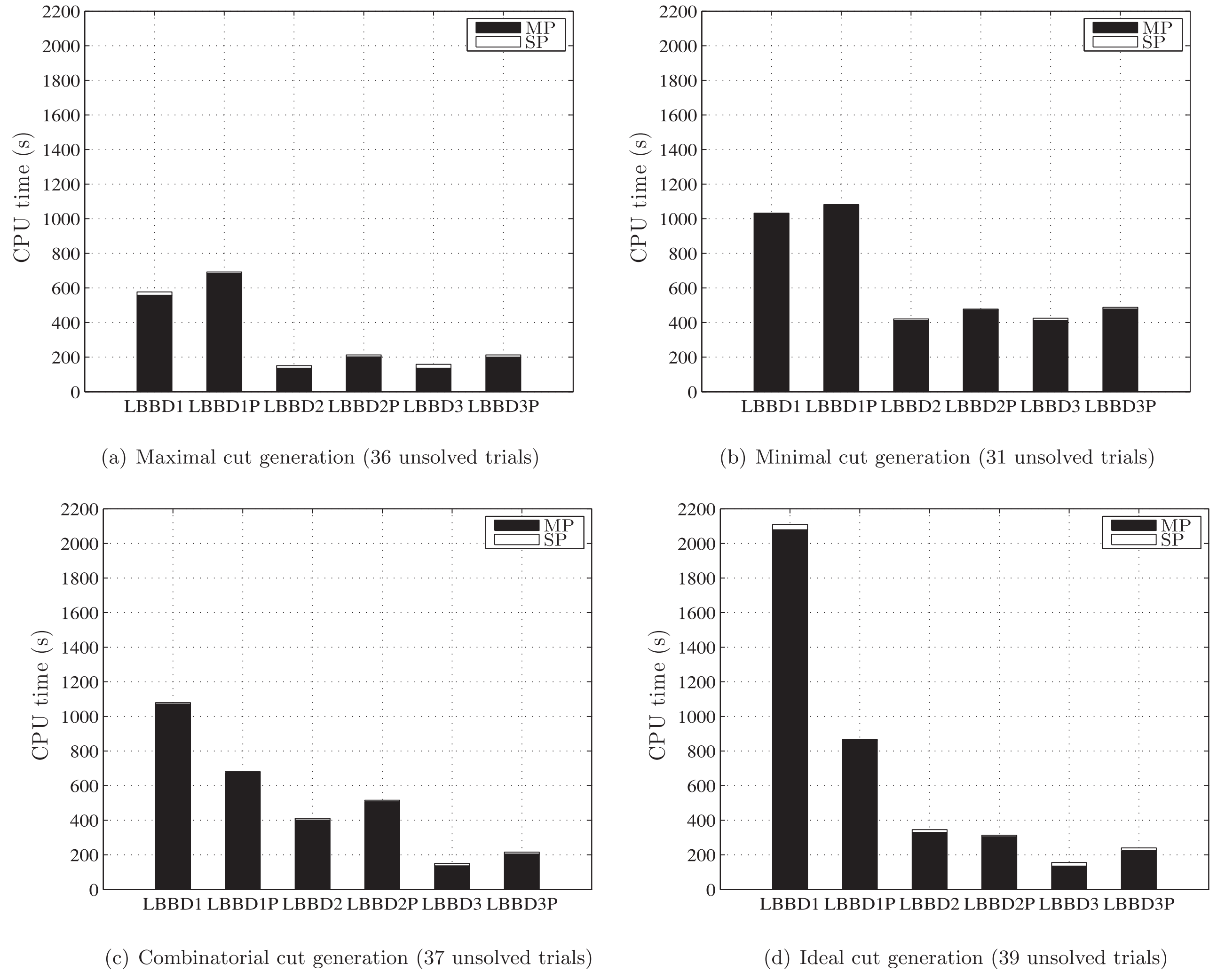
6.1. LBBD framework applications

The applications of our LBBDs are widespread. Generally speaking, many problems with a permanent (strategic) or temporary (tactical and operational) location-allocation problems with packing SPs can benefit from our LBBDs. Our developed LBBDs in its current form or with simple modifications can be applied to, for example, (a) transportation problems, requiring the selection of customers to serve based on tardiness costs and the determination of the number of trucks given the selected customers; (b) cross-dock location and truck allocation problems; (c) cellular manufacturing systems, requiring the determination of which cell to use given the pool of orders and the allocation of workers to each cell; (d) fleet assignment problems to allocate heterogeneous vehicles to scheduled routes, based on equipment capabilities and availabilities, operational costs, and potential revenues; and (e) strategic wind farm location and turbine allocation problems.

6.2. Structural sensitivity to DORS assumptions

We assumed that all ORs are functionally identical; therefore, the optimal allocation of patient to hospital-days can be conducted in a flexible fashion. We can partially relax this assumption and posit that ORs' functional identicality is local to each hospital-day, which limits the allocation of each patient to a subset of hospitals, reducing the number of binary allocation variables in the MP significantly, which will improve CPU time and worsen the objective function value. However, if we relax local functional identicality of ORs within each hospital, our LBBDs will no longer be valid as the packing SPs require equal availability time for ORs. Varying numbers of ORs per hospital only affect the cut propagation mechanism, significantly reducing the number of Benders' cuts that can be propagated, because the infeasibility of a hospital-day with n ORs cannot be generalized to a hospital-day with $> n$ ORs. The influence of this change on the CPU time is unknown and requires further investigation.

We also assumed that nurse and anesthesiologist assignments to ORs have been performed a priori. Under a block scheduling system, each block of OR is sufficiently staffed with a predetermined number of resources including nurses and an anesthesiologist. Therefore, determining the number of ORs is equivalent to determining the number of nurses due to pre-determined nurse-to-OR ratios. However, in some hospitals, anesthesiologists are shared among ORs, in which case another level of nested optimization (packing problems for anesthesiologists in each hospital-day) is needed. Our LBBDs remain valid for this problem with the addition of MP constraints on the lower bounds for both ORs and anesthesiologists, though our LBBDs require that the anesthesiologists have equal availability within each hospital.



(a) Maximal cut generation (36 unsolved trials)

(b) Minimal cut generation (31 unsolved trials)

(c) Combinatorial cut generation (37 unsolved trials)

(d) Ideal cut generation (39 unsolved trials)

Fig. 5. Three ORs: breakdown of average total CPU time between MP and SP over solved trials.

We additionally postulated that each OR block is assigned to only one surgeon in a day (therefore, patient-to-OR assignment is equivalent to patient-to-surgeon assignment). This assumption can be relaxed and solved as an additional nested optimization problem via our LBBDs if surgeon availability times are equal within each hospital. The structure of our LBBDs also allows the alternation of surgeons among hospitals if all surgeons have the same availability time in each hospital-day and no surgeon can go to more than one hospital in each day. Our Benders' feasibility and optimality cuts are exactly applicable even if the availability times of surgeons within one hospital-day is not equal to the availability times of its ORs. Then, our Benders' cuts are generated for infeasible or sub-optimal surgeons' schedules.

Relaxing the assumption of identical surgeons skills can be done by defining a subset of surgeons by whom a patient can be operated on. This change can also be accommodated by our LBBDs if surgeons with identical surgical skills with equal availability times are assigned to a certain hospital-day. Similarly, multiple surgical specialties can be incorporated by limiting patient-to-surgeon allocation in the same way as non-identical surgeons, and ORs belonging to each specialty are treated as non-identical ORs. In this scenario, one surgical specialty may be allocated ORs from other specialties if its surgical load exceeds its OR capacity. MP constraints to incorporate constraints regarding the number of ORs and their availability times must be added. However, the inclusion of new surgical specialties poses the challenge of size-dimensionality of the model due to the increased computational complexity of the SPs, and hence requires tailor-made decomposition techniques for both the MP and SPs. Alternatively, the addition of multiple surgical specialties can be considered a general assignment problem with location-allocation considerations, which can be solved using branch-and-price approaches (e.g., Ghoniem, Flamand, & Haouari, 2016). The deterministic assumption could be relaxed using two-stage stochastic optimization approaches, which has previously been done via L-shaped heuristic methods (Batun, Denton, Huschka, & Schaefer, 2011).

6.3. High OR utilization of UHN

At first sight, the opened ORs utilization seem impractically high. However, this high utilization is expected since it is a designed goal of the deterministic IP formulation which also results in entire ORs not being opened. As previously stated, these unopened ORs can be re-allocated to other services or to emergency patients, but they can also be used to accommodate overflow given the inherent uncertainty in surgical durations. However, in the UHN data, 58% of the realized surgical times were less than the booked times, so surgical duration stochasticity mostly results in OR under-utilization than over-utilization, which will mitigate the effect of the high utilization planned by DORS. There are two ways

Table 6
Three ORs: CPU time breakdown, number of iterations ($|\mathcal{I}|$), feasibility cuts ($|\mathcal{F}|$), and optimality cuts ($|\mathcal{O}|$) for the maximal implementation.

	$\|\mathcal{P}\|$	CPU (seconds)	MP (seconds)	SP (seconds)	$\|\mathcal{I}\|$	$\|\mathcal{F}\|$	$\|\mathcal{O}\|$	$\|\mathcal{F}\| + \|\mathcal{O}\|$
LBBD1	20	0.64	0.60	0.04	1.8	0.6	0.2	
	40	2.56	2.45	0.11	2.0	1.6	0.2	
	60	19.18	11.39	7.79	3.2	3.4	0.8	
	80	183.22	181.24	1.98	16.0	39.6	0.8	
	100	625.15	615.62	9.53	58.7	160.3	1.0	
	120	662.36$^{(2)}$	612.03	50.33	231.00	1034.30	3.67	
	140	2546.40$^{(3)}$	2480.00	66.40	254.00	1249.00	1.50	
Average		577.07$^{(5)}$	557.62	19.45	80.96	355.54	1.17	356.71
LBBD2	20	0.45	0.42	0.03	1.40	0.20	0.20	
	40	10.59	8.53	2.06	21.20	30.40	0.20	
	60	9.45	9.30	0.15	1.80	1.20	0.60	
	80	53.82	47.30	6.52	34.60	90.40	0.00	
	100	385.59	341.84	43.75	169.40	677.80	0.00	
	120	291.47$^{(2)}$	283.62	7.85	42.33	136.00	1.67	
	140	305.51$^{(3)}$	271.18	34.33	115.00	622.00	3.00	
Average		150.98$^{(5)}$	137.46	13.52	55.10	222.57	0.81	223.38
LBBD3	20	0.48	0.42	0.06	1.40	0.00	0.40	
	40	10.82	8.27	2.55	21.20	0.00	30.60	
	60	9.10	8.90	0.20	1.80	0.00	1.80	
	80	55.05	45.60	9.45	34.60	0.00	90.40	
	100	402.16	340.26	61.90	169.40	0.00	677.80	
	120	295.80$^{(2)}$	284.20	11.61	42.33	0.00	137.67	
	140	341.57$^{(3)}$	272.82	68.75	115.00	0.00	625.00	
Average		159.28$^{(5)}$	137.21	22.07	55.10	0.00	223.38	223.38

Table 7
OR and surgical suite utilization.

	$\|\mathcal{P}\|$	Opened ORs mean [min, max]	Opened OR utilization mean [min, max]	Network OR utilization (%)	Opened surgical suites mean [min, max]	Network surgical suite utilization (%)	Patients scheduled (%)
Five ORs	20	6.6 [6,7]	81.4 [71.3,91.3]	8.8	2.2 [2,3]	14.7	100
	40	10.2 [9,13]	87.4 [67.2,98.1]	13.6	3.2 [3,4]	21.3	100
	60	19.4 [18,20]	81.3 [51.0,98.3]	25.9	5.4 [4,6]	36.0	100
	80	26.0 [24,32]	89.1 [58.6,99.1]	34.7	6.0 [6,6]	40.0	100
	100	30.6 [25,32]	93.0 [66.7,99.6]	40.8	7.6 [7,8]	50.7	100
	120	36.8 [36,37]	90.8 [54.6,99.5]	49.1	8.8 [8,9]	58.7	100
	140	45.0 [42,48]	92.1 [55.8,99.7]	60.0	10.2 [9,11]	68.0	100
	160	46.2 [42,50]	92.5 [63.4,99.9]	61.6	10.4 [10,11]	69.3	100
Three ORs	20	6.6 [6,7]	83.0 [66.5,94.5]	14.7	2.6 [2,4]	17.3	100
	40	13.8 [13,15]	89.2 [67.9,99.0]	30.7	6.6 [5,9]	44.0	100
	60	19.0 [18,20]	89.0 [61.4,98.8]	42.2	7.6 [7,8]	50.6	100
	80	25.8 [25,28]	82.8 [29.2,99.7]	57.3	9.6 [9,10]	64.0	100
	100	30.0 [29,31]	94.4 [68.8,99.5]	66.7	10.8 [10,12]	72.0	100
	120	37.6 [36,38]	93.0 [54.7,99.0]	83.6	13.2 [12,14]	88.0	100
	140	41.3 [40,42]	95.0 [70.4 99.5]	91.8	14.7 [14,15]	97.8	100
	160	45.0 [45,45]	96.7 [92.4 99.4]	100.0	15.0 [15,15]	100.0	94.9
Two ORs	100	30.0 [30,30]	96.8 [94.9, 99.6]	100	15.0 [15,15]	100	97.2
	120	30.0 [30,30]	96.5 [95.3, 99.8]	100	15.0 [15,15]	100	82.5
	140	30.0 [30,30]	96.4 [92.0, 99.5]	100	15.0 [15,15]	100	71.4
	160	30.0 [30,30]	96.1 [91.9, 99.1]	100	15.0 [15,15]	100	64.0

to remedy this issue: (i) use the average of historical realized times instead of surgeon estimates, and (ii) decrease the booked times by, say, 20%. However, for other hospitals where the realized time of surgeries is usually higher than the booked times, a stochastic or robust variant of DORS is recommended.

7. Conclusion

We studied the problem of DORS in a network of collaborating hospitals to allocate patients to hospital-OR-days, while accounting for the health status score and wait time of each patient. We developed an IP and three LBBD approaches with four implementations, and a cut propagation technique applied to each for the DORS problem. These approaches were two orders of magnitude faster than IP+Gurobi, more successful at finding optimal solutions, and able to solve larger problems.

Since this research is the first attempt in OR scheduling literature to schedule patients collaboratively across a multi-hospital group, various future topics can be explored, including balanced workload, multiple surgical specialties overload, and stochastic and robust variations. The replacement of the current mathematical programming-based SPs with constraint programming SPs and the use of other cut strengthening techniques (Hooker, 2007) may be useful. Additionally, this research lends itself well to the game theory, where each hospital seeks to maximize its own utilization independent of the network.

Appendix A

Table 8
Five ORs: number of iterations ($|\mathcal{I}|$), feasibility cuts ($|\mathcal{F}|$), and optimality cuts ($|\mathcal{O}|$) for all implementations.

	$|\mathcal{P}|$	LBBD1 $|\mathcal{I}|, |\mathcal{F}|, |\mathcal{O}|$	LBBD1$_\text{P}$ $|\mathcal{I}|, |\mathcal{F}|, |\mathcal{O}|$	LBBD2 $|\mathcal{I}|, |\mathcal{F}|, |\mathcal{O}|$	LBBD2$_\text{P}$ $|\mathcal{I}|, |\mathcal{F}|, |\mathcal{O}|$	LBBD3 $|\mathcal{I}|, |\mathcal{F}|, |\mathcal{O}|$	LBBD3$_\text{P}$ $|\mathcal{I}|, |\mathcal{F}|, |\mathcal{O}|$
Maximal	20	1.2, 0, 0.2	1.2, 0, 0.2	1.2, 0, 0.2	1.2, 0, 0.2	1.2, 0, 0.2	1.2, 0, 0.2
	40	1.2, 0, 0.2	1.4, 0.2, 0.4	1.2, 0, 0.2	2.8, 2.2, 0.4	1.2, 0, 0.2	2.8, 0, 2.6
	60	3, 2, 0.6	8, 10.4, 1.4	2, 1.6, 0.2	3.4, 3, 0.6	2, 0, 1.8	3, 0, 3.2
	80	8, 13.6, 1.2	3.2, 3.4, 0	2.6, 2.6, 0	3.4, 4, 0	2.6, 0, 2.6	3.4, 0, 4
	100	4.4, 4.6, 0.6	9.6, 13.8, 0.8	5.6, 8, 0.2	6.6, 7.4, 0.2	5.6, 0, 8.2	6.6, 0, 7.6
	120	12.8, 21, 1.2	37.6, 80.2, 1.4	8.4, 14.8, 0.6	8.5, 14.5, 0.8	5.80, 0, 8.80	8, 0, 12.3
	140	73.2, 235.2, 2.4	64.8, 201.4, 2.6	85.3, 251.7, 2.7	44.25, 120, 1.8	14.5, 0, 26.5	24, 0, 58
	160	13.5, 30, 2	37.5, 103.8, 1.8	87.3, 235, 2.5	27, 67.5, 2.3	39.5, 0, 100	31.7, 0, 81.3
Minimal	20	1.2, 0, 0.2	1.2, 0, 0.2	1.2, 0, 0.2	1.2, 0, 0.2	1.2, 0, 0.2	1.2, 0, 0.2
	40	1.2, 0, 0.2	1.6, 0.2, 0.4	1.2, 0, 0.2	1.6, 0.2, 0.4	1.2, 0, 0.2	1.2, 0, 0.2
	60	4.8, 2.4, 1.4	6.4, 4.8, 0.6	3.4, 2.2, 0.2	2.4, 1.2, 0.2	3.4, 0, 2.4	2.4, 0, 1.4
	80	1.8, 0.8, 0	4.4, 3.4, 0	19.6, 18.6, 0	6.2, 5.2, 0	19.6, 0, 18.6	6.2, 0, 5.2
	100	5, 4, 0	13, 11, 1	6, 5, 0	7.2, 6.2, 0	6, 0, 5	7.2, 0, 6.2
	120	15.5, 14, 0.5	10.4, 8.4, 1	52.6, 50.8, 0.8	49.6, 47.8, 0.8	8.5, 0, 7.5	8.3, 0, 7.3
	140	27.8, 25.8, 1	138.3, 133.8, 3.5	36.8, 35.75, 0	90.3, 88, 1.25	37.7, 0, 36.7	90.3, 0, 89.3
	160	59, 57.3, 0.8	38.75, 36, 1.8	79.8, 77.5, 1.3	42.3, 40.3, 1	160, 0, 159	30, 0, 29
Combinatorial	20	1.2, 0, 0.2	1.2, 0, 0.2	1.2, 0, 0.2	1.2, 0, 0.2	1.2, 0, 0.2	1.2, 0, 0.2
	40	1.2, 0, 0.2	1.4, 0.2, 0.4	1.2, 0, 0.2	2.2, 1, 0.4	1.2, 0, 0.2	2.8, 0, 2.6
	60	2.8, 1.4, 0.6	6.4, 4.8, 0.6	3.4, 2.2, 0.2	2.4, 1.2, 0.2	2, 0, 1.8	3, 0, 3.2
	80	1.8, 0.8, 0	4.4, 3.4, 0	19.6, 18.6, 0	6.2, 5.2, 0	2.6, 0, 2.6	3.4, 0, 4
	100	5, 4, 0	12.4, 10.8, 0.8	6, 5, 0	7.2, 6.2, 0	5.6, 0, 8.2	6.6, 0, 7.6
	120	15, 13.8, 0	53.2, 51.8, 1.2	13.4, 12.4, 0.8	12.4, 11.4, 0.6	5.8, 0, 8.8	8, 0, 12.3
	140	23.7, 22.7, 0.7	68.5, 67.3, 3.8	36.8, 35.8, 0	90, 89, 0.3	14.5, 0, 26.5	24, 0, 58
	160	86, 84.8, 0.8	45.5, 43.8, 1.75	53, 51.7, 0.3	42.3, 40.3, 1	39.5, 0, 100	31.7, 0, 81.3
Ideal	20	1.2, 0, 0.2	1.2, 0, 0.2	1.2, 0, 0.2	1.2, 0, 0.2	1.2, 0, 0.2	1.2, 0, 0.2
	40	1.2, 0, 0.2	1.4, 0.2, 0.4	1.2, 0, 0.2	2.8, 2.2, 0.4	1.2, 0, 0.2	2.8, 0, 2.6
	60	3, 2, 0.6	8, 10.4, 1.4	2, 1.6, 0.2	3.4, 3, 0.6	2, 0, 1.8	3, 0, 3.2
	80	1.8, 0.8, 0	4.4, 4, 0	3.2, 3, 0	3.8, 4.6, 0	2.6, 0, 2.6	3.4, 0, 4
	100	4.6, 4, 0.6	8.8, 10.6, 0.8	3.8, 3.8, 0.2	6.2, 6.4, 0.2	5.6, 0, 8.2	6.6, 0, 7.6
	120	16, 22, 1.25	15.4, 25.2, 1	16.4, 24.2, 0.6	30.4, 46.6, 0.6	5.8, 0, 8.8	8, 0, 12.3
	140	19.8, 34.5, 1.3	108, 247.8, 2.8	48, 96.7, 0.7	73.3, 153.3, 0.5	14.5, 0, 58	14.5, 0,58
	160	15.3, 26.3, 1.3	17.7, 32.7, 1.7	55.3, 109, 0.8	27.5, 54, 0.5	39.5, 0, 100	31.7, 0, 81.3

Table 9
Three ORs: number of iterations ($|\mathcal{I}|$), feasibility cuts ($|\mathcal{F}|$), and optimality cuts ($|\mathcal{O}|$) for all implementations.

	$|\mathcal{P}|$	LBBD1 $|\mathcal{I}|, |\mathcal{F}|, |\mathcal{O}|$	LBBD1$_\text{P}$ $|\mathcal{I}|, |\mathcal{F}|, |\mathcal{O}|$	LBBD2 $|\mathcal{I}|, |\mathcal{F}|, |\mathcal{O}|$	LBBD2$_\text{P}$ $|\mathcal{I}|, |\mathcal{F}|, |\mathcal{O}|$	LBBD3 $|\mathcal{I}|, |\mathcal{F}|, |\mathcal{O}|$	LBBD3$_\text{P}$ $|\mathcal{I}|, |\mathcal{F}|, |\mathcal{O}|$
Maximal	20	1.8, 0.6, 0.2	1.4, 0.2, 0.2	1.4, 0.2, 0.2	1.4, 0.2, 0.2	1.4, 0.0, 0.4	1.4, 0.0, 0.4
	40	2.0, 1.6, 0.2	8.6, 12.8, 0.4	21.2, 30.4, 0.2	17.4, 23.0, 0.2	21.2, 0.0, 30.6	17.4, 0.0, 23.2
	60	3.2, 3.4, 0.8	2.0, 1.4, 0.8	1.8, 1.2, 0.6	1.6, 1.0, 0.6	1.8,0.0, 1.8	1.6, 0.0, 1.6
	80	16.0, 39.6, 0.8	28.0, 67.2, 0.8	34.6, 90.4, 0	49.8, 136, 0	34.6, 0.0, 90.4	49.8, 0.0, 136.0
	100	58.7, 160.3, 0.8	13.0, 32.8, 0.8	169.4, 677.8, 0	29.5, 89.5, 0.5	169.4, 0.0, 677.8	29.5, 0.0, 90.0
	120	231.0, 1034.0, 3.67	69.7, 265.0, 3.33	42.33, 136.0, 1.7	191.7, 777.7, 2.33	42.3, 0.0, 137.7	191.7, 0.0, 780.0
	140	254.0, 1249.0, 1.5	93.5, 392.0, 5.0	115.0, 622.0, 3.0	30.0, 79.0, 1.0	115.0, 0.0, 625.0	30.0, 0.0, 80.0
Minimal	20	1.8, 0.6, 0.2	1.4, 0.2, 0.2	1.4, 0.2, 0.2	1.4, 0.2, 0.2	1.4, 0.0, 0.4	1.4, 0.0, 0.4
	40	4.0, 3.0, 0.0	15.8, 14.6, 0.2	15.6, 14.6, 0.0	6.4, 5.4, 0.0	15.6, 0.0, 14.6	6.4, 0.0, 5.4
	60	3.0, 1.6, 0.4	3.6, 2.2, 0.4	2.6, 1.4, 0.2	2.4, 1.2, 0.2	2.6, 0.0, 1.6,	2.4, 0.0, 1.4
	80	20.4, 19.2, 0.2	26.8, 25.4, 0.4	41.4, 40.4, 0.0	115.8, 114.8, 0.0	41.4, 0.0, 40.4	115.8, ,0.0, 114.8
	100	43.75, 42.0, 0.75	68.5, 66.75, 0.75	503.0, 502, 0.0	302.2, 301.0, 0.2	503, 0.0, 502	302.2, 0.0, 301.2
	120	383.5, 376.0, 6.5	257.3, 251.3, 5.0	615.0, 612.0, 2.0	130.8, 128.8, 1.0	615.0, 0, 614.0	130.8, 0.0, 129.8
	140	******	******	68.0, 67.0, 0.0	219.0, 217.0, 1.0	68.0, 0.0, 67.0	219.0, 0.0, 218.0
Combinatorial	20	1.8, 0.6, 0.2	1.4, 0.2, 0.2	1.4, 0.2, 0.2	1.4, 0.2, 0.2	1.4, 0.0, 0.4	1.4, 0.0, 0.4
	40	4.0, 3.0, 0.0	15.8, 14.6, 0.2	2.6, 1.6, 0.4	2.2, 1.2, 0.4	1.8, 0.0, 1.8	1.6, 0.0, 1.6
	60	3.4, 2.2, 0.6	3.4, 2.2, 0.6	2.6, 1.6, 0.4	2.2, 1.2, 0.4	1.8, 0.0, 1.8	1.6, 0.0, 1.6
	60	19.6, 18.6, 0.2	26.8, 25.8, 0.6	41.4, 40.4, 0	115.8, 114.8, 0.0	34.6, 0.0, 90.4	115.8, 0.0, 114.8
	100	84.8, 83.8, 1.0	86.5, 85.3, 1.0	503.0, 502.0, 0.0	302.4, 301.4, 0.2	169.4, 0.0, 677.8	29.5, 0.0, 90.0
	120	534.0, 533.0, 7.0	163.5, 162.5, 5.5	468.8, 467.8, 1.0	195.8, 194.8, 1.4	42.33, 0,.0, 137.7	191.7, 0.0, 780.0
	140	*******	******	68.0, 67.0, 0.0	220.5, 219.0, 1.0	115.0, 0.0, 625.0	30, 0.0, 80.0
Ideal	20	1.8, 0.6, 0.2	1.4, 0.2, 0.2	1.4, 0.2, 0.2	1.4, 0.2, 0.2	1.4, 0.0, 0.4	1.4, 0.0, 0.4
	40	4.0, 3.0, 0.0	15.8, 14.6, 0.2	15.6, 14.6, 0.0	6.4, 5.4, 0.0	15.6, 0.0, 14.6	6.4, 0.0, 5.4
	60	3.0, 1.6, 0.4	3.6, 2.2, 0.4	2.6, 1.4, 0.2	2.4, 1.2, 0.2	2.6, 0.0, 1.6	2.4, 0.0, 1.4
	80	23.8, 40.2, 0.6	97.2, 207.2, 1.0	42.6, 76.4, 0.0	26.0, 48.0, 0.0	42.6, 0.0, 76.4	26.0, 0.0, 48.0
	100	27.3, 51.33, 1.0	72.0, 156.5, 1.0	13.8, 26.8, 0.0	82.0, 197.0, 0.0	13.8, 0.0, 26.8	82.0, 0.0, 197.0
	120	201.0, 494.3, 2.0	5.0, 9.0, 0.5	435.3, 917.7, 6.7	187.8, 418.8, 1.5	42.3, 0.0, 137.7	191.7, 0.0, 780.0
	140	1087.0, 2834.0, 3.0	******	173.0, 465.0, 0.5	32.0, 84.0, 0.0	115.0, 0.0, 625.0	30.0, 0.0, 80.0

References

Albareda-Sambola, M., Fernández, E., & Laporte, G. (2009). The capacity and distance constrained plant location problem. *Computers & Operations Research, 36*(2), 597–611. Scheduling for Modern Manufacturing, Logistics, and Supply Chains. http://dx.doi.org/10.1016/j.cor.2007.10.017.

Astaraky, D., & Patrick, P. (2015). A simulation based approximate dynamic programming approach to multi-class, multi-resource surgical scheduling. *European Journal of Operational Research, 245*(1), 309–319. http://dx.doi.org/10.1016/j.ejor.2015.02.032.

Augusto, V., Xie, X., & Perdomo, V. (2008). Operating theatre scheduling using Lagrangian relaxation. *European Journal of Industrial Engineering, 2*(2), 172–189. http://dx.doi.org/10.1016/j.cie.2009.04.019.

Augusto, V., Xie, X., & Perdomo, V. (2010). Operating theatre scheduling with patient recovery in both operating rooms and recovery beds. *Computers & Industrial Engineering, 58*(2), 231–238. http://dx.doi.org/10.1016/j.cie.2009.04.019.

Batun, S., Denton, B. T., Huschka, T. R., & Schaefer, A. J. (2011). Operating room pooling and parallel surgery processing under uncertainty. *INFORMS Journal on Computing, 23*(2), 220–237.

Beck, J. C. (2010). Checking-up on branch-and-check. In D. Cohen (Ed.), *Lecture Notes in Computer Science: Vol. 6308. Principles and practice of constraint programming – CP 2010* (pp. 84–98). Berlin, Heidelberg: Springer. doi:10.1007/978-3-642-15396-9_10.

Behnamian, J., & Fatemi Ghomi, S. M. T. (2013). The heterogeneous multi-factory production network scheduling with adaptive communication policy and parallel machine. *Information Sciences, 219*, 181–196.

Benders, J. F. (1962). Partitioning procedures for solving mixed-variables programming problems. *Numerische Mathematik, 4*, 238–252.

Bockmayr, A., & Pisaruk, N. (2006). Detecting infeasibility and generating cuts for mixed integer programming using constraint programming. *Computers & Operations Research, 33*(10), 2777–2786. Part Special Issue: Constraint Programming. http://dx.doi.org/10.1016/j.cor.2005.01.010

Cardoen, B., Demeulemeester, E., & Belien, J. (2009a). Optimizing a multiple objective surgical case sequencing problem. *International Journal of Production Economics, 119*(2), 354–366.

Cardoen, B., Demeulemeester, E., & Belien, J. (2009b). Sequencing surgical cases in a day-care environment: An exact branch-and-price approach. *Computers & Operations Research, 36*(9), 2660–2669.

Cardoen, B., Demeulemeester, E., & Belien, J. (2010). Operating room planning and scheduling: A literature review. *European Journal of Operational Research, 201*(3), 921–932.

Chu, Y., & Xia, Q. (2004). Generating Benders cuts for a general class of integer programming problems. In J.-C. Régin, & M. Rueher (Eds.), *Lecture Notes in Computer Science: Vol. 3011. Integration of ai and or techniques in constraint programming for combinatorial optimization problems* (pp. 127–141). Berlin, Heidelberg: Springer.

Cire, A., Coban, E., & Hooker, J. (2013). Mixed integer programming vs. logic-based Benders decomposition for planning and scheduling. In C. Gomes, & M. Sellmann (Eds.), *Lecture Notes in Computer Science: Vol. 7874. Integration of AI and OR techniques in constraint programming for combinatorial optimization problems* (pp. 325–331). Berlin, Heidelberg: Springer. doi:10.1007/978-3-642-38171-3_22.

Cire, A. A., & Hooker, J. N. (2012). A heuristic logic-based Benders method for the home health care problem. *Working paper*. http://repository.cmu.edu/cgi/viewcontent.cgi?article=2379&context-tepper.

Day, R., Garfinkel, R., & Thompson, S. (2012). Integrated block sharing: A winwin strategy for hospitals and surgeons. *Manufacturing & Service Operations Management, 14*(4), 567–583.

Denton, B. T., Miller, A. J., Balasubramanian, H. J., & Huschka, T. R. (2010). Optimal allocation of surgery blocks to operating rooms under uncertainty. *Operations Research, 58*, 802–816.

Fazel-Zarandi, M. M., & Beck, J. C. (2012). Using logic-based Benders decomposition to solve the capacity- and distance-constrained plant location problem. *INFORMS Journal on Computing, 24*(3), 387–398.

Fei, H., Chu, C., & Meskens, N. (2009). Solving a tactical operating room planning problem by a column-generation-based heuristic procedure with four criteria. *Annals of Operations Research, 166*(1), 91–108.

Fei, H., Meskens, N., & Chu, C. (2010). A planning and scheduling problem for an operating theatre using an open scheduling strategy. *Computers & Industrial Engineering, 58*(2), 221–230.

Ghoniem, A., Flamand, T., & Haouari, M. (2016). Exact solution methods for a generalized assignment problem with location/allocation considerations. *INFORMS Journal on Computing, 28*(3), 589–602.

Guerriero, F., & Guido, R. (2011). Operational research in the management of the operating theatre: A survey. *Health Care Management Science, 14*(1), 89–114.

Harjunkoski, I., & Grossmann, I. E. (2002). Decomposition techniques for multistage scheduling problems using mixed-integer and constraint programming methods. *Computers & Chemical Engineering, 26*(11), 1533–1552.

Hashemi Doulabi, S. H., Rousseau, L. M., & Pesant, G. (2016). A constraint-programming-based branch-and-price-and-cut approach for operating room planning and scheduling. *INFORMS Journal on Computing, 28*(3), 432–448.

Heinz, S., & Beck, J. C. (2012). Reconsidering mixed integer programming and mip-based hybrids for scheduling. In N. Beldiceanu, N. Jussien, & É. Pinson (Eds.), *Lecture Notes in Computer Science: Vol. 7298. Integration of AI and OR techniques in constraint programming for combinatorial optimization problems* (pp. 211–227). Berlin, Heidelberg: Springer. doi:10.1007/978-3-642-29828-8_14.

Herring, W. L., & Herrmann, J. W. (2012). The single-day surgery scheduling problem: Sequential decision-making and threshold-based heuristics. *OR Spectrum, 34*(2), 429–459. doi:10.1007/s00291-011-0270-3.

Hooker, J. N. (2007). Planning and scheduling by logic-based Benders decomposition. *Operations Research., 55*(3), 588–602.

Hooker, J. N., & Ottosson, G. (2003). Logic-based Benders decomposition. *Mathematical Programming, 96*(1), 33–60.

Houdenhoven, M. V., van Oostrum, J., Hans, E., Wullink, G., & Kazemier, G. (2007). Improving operating room efficiency by applying bin-packing and portfolio techniques to surgical case scheduling. *Anesthesia & Analgesia, 105*(3), 707–714.

Houdenhoven, M. V., van Oostrum, J., Wullink, G., Hans, E., Hurink, J., Bakker, J., & Kazemier, G. (2008). Fewer intensive care unit refusals and a higher capacity utilization by using a cyclic surgical case schedule. *Journal of Critical Care, 23*(2), 222–226.

Jebali, A., Alouane, A. B. H., & Ladet, P. (2006). Operating rooms scheduling. *International Journal of Production Economics, 99*, 52–62.

Lamiri, M., Xie, X., & Zhang, S. (2008). Column generation approach to operating theater planning with elective and emergency patients. *IIE Transactions, 40*(9), 838–852. doi:10.1080/07408170802165831.

Lin, J., Muthuraman, K., & Lawley, M. (2011). Optimal and approximate algorithms for sequential clinical scheduling with no-shows. *IIE Transactions on Healthcare Systems Engineering, 1*(1), 20–36. doi:10.1080/19488300.2010.549927.

Liu, Y., Chu, C., & Wang, K. (2011). A new heuristic algorithm for the operating room scheduling problem. *Computers & Industrial Engineering, 61*(3), 865–871. http://dx.doi.org/10.1016/j.cie.2011.05.020.

Marcon, E., Kharraja, S., & Simonnet, G. (2003). The operating theatre planning by the follow-up of the risk of no realization. *International Journal of Production Economics, 85*(1), 83–90.

Marques, I., Captivo, M. E., & Margarida, V. P. (2012). An integer programming approach to elective surgery scheduling. *OR Spectrum, 34*(2), 407–427. doi:10.1007/s00291-011-0279-7.

Naderi, B., & Azab, A. (2014). Modeling and heuristics for scheduling of distributed job shops. *Expert Systems with Applications, 41*(17), 7754–7763.

Naderi, B., & Ruiz, R. (2010). The distributed permutation flowshop scheduling problem. *Computers & Operations Research, 37*(4), 754–768.

Naderi, B., & Ruiz, R. (2014). A scatter search algorithm for the distributed permutation flowshop scheduling problem. *European Journal of Operational Research, 239*(2), 323–334.

Pisinger, D., & Sigurd, M. (2007). Using decomposition techniques and constraint programming for solving the two-dimensional bin-packing problem. *INFORMS Journal on Computing, 19*(1), 36–51.

Range, T. M., Lusby, R. M., & Larsen, J. (2014). A column generation approach for solving the patient admission scheduling problem. *European Journal of Operational Research, 235*(1), 252–264.

Rizk, C., & Arnaout, J. P. (2012). ACO for the surgical cases assignment problem. *Journal of Medical Systems, 36*(3), 1891–1899. doi:10.1007/s10916-010-9648-z.

Roland, B., Martinelly, C. D., Riane, F., & Pochet, Y. (2010). Scheduling an operating theatre under human resource constraints. *Computers & Industrial Engineering, 58*(2), 212–220.

Sadykov, R. (2008). A branch-and-check algorithm for minimizing the weighted number of late jobs on a single machine with release dates. *European Journal of Operational Research, 189*(3), 1284–1304. http://dx.doi.org/10.1016/j.ejor.2006.06.078.

Sadykov, R., & Wolsey, L. A. (2006). Integer programming and constraint programming in solving a multimachine assignment scheduling problem with deadlines and release dates. *INFORMS Journal on Computing, 18*(2), 209–217. doi:10.1287/ijoc.1040.0110.

Santibanez, P., Begen, M., & Atkins, D. (2007). Surgical block scheduling in a system of hospitals: an application to resource and wait list management in a British Columbia Health Authority. *Health Care Management Science, 10*(3), 269–282.

Silva, T. A. O., de Souza, M. C., Saldanha, R. R., & Burke, E. K. (2015). Surgical scheduling with simultaneous employment of specialised human resources. *European Journal of Operational Research, 245*(3), 719–730.

Tanfani, E., & Testi, A. (2010). A pre-assignment heuristic algorithm for the Master Surgical Schedule Problem (MSSP). *Annals of Operations Research, 178*(1), 105–119.

Thorsteinsson, E. S. (2001). Branch-and-check: A hybrid framework integrating mixed integer programming and constraint logic programming. In T. Walsh (Ed.), *Lecture Notes in Computer Science: Vol. 2239. Principles and practice of constraint programming – CP 2001* (pp. 16–30). Berlin, Heidelberg: Springer. doi:10.1007/3-540-45578-7_2.

Tran, T. T., Araujo, A., & Beck, J. C. (2016). Decomposition methods for the parallel machine scheduling problem with setups. *INFORMS Journal on Computing, 28*(1), 83–95.

Tran, T. T., & Beck, J. C. (2012). Logic-based Benders decomposition for alternative resource scheduling with sequence dependent setups. In *Proceedings of European conference on artificial intelligence, ECAI'12* (pp. 774–779).

Vijayakumar, B., Parikh, P. J., Scott, R., Barnes, A., & Gallimore, J. (2013). A dual bin-packing approach to scheduling surgical cases at a publicly-funded hospital. *European Journal of Operational Research, 224*(3), 583–591.

Wang, T., Meskens, N., & Duvivier, D. (2015). Scheduling operating theatres: Mixed integer programming vs. constraint programming. *European Journal of Operational Research, 247*(2), 401–413.