European Journal of Operational Research 000 (2018) 1–15

Contents lists available at ScienceDirect

European Journal of Operational Research

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

Innovative Applications of O.R.

Solution methods for the tray optimization problem

Twan Dollevoet[a,*], J. Theresia van Essen[b], Kristiaan M. Glorie[c]

[a] Erasmus School of Economics, Erasmus University Rotterdam, P.O. Box 1738, 3000DR Rotterdam, The Netherlands
[b] Delft Institute of Applied Mathematics, Delft University of Technology, P.O. Box 5031, 2600GA Delft, The Netherlands
[c] Erasmus Quantitative Intelligence, Erasmus University Rotterdam, P.O. Box 1738, 3000DR Rotterdam, The Netherlands

ARTICLE INFO

Article history:
Received 12 April 2017
Accepted 28 May 2018
Available online xxx

Keywords:
OR in health services
Sterile inventory
Integer Linear Programming
Row & column generation
Heuristics

ABSTRACT

In order to perform medical surgeries, hospitals keep large inventories of surgical instruments. These instruments need to be sterilized before each surgery. Typically the instruments are kept in trays. Multiple trays may be required for a single surgery, while a single tray may contain instruments that are required for multiple surgical procedures. The tray optimization problem (TOP) consists of three main decisions: (i) the assignment of instruments to trays, (ii) the assignment of trays to surgeries, and (iii) the number of trays to keep in inventory. The TOP decisions have to be made such that total operating costs are minimized and such that for every surgery sufficient instruments are available. This paper presents and evaluates several exact and heuristic solution methods for the TOP. We compare solution methods on computation time and solution quality. Moreover, we conduct simulations to evaluate the performance of the solutions in the long run. The novel methods that are provided are the first methods that are capable of solving instances of realistic size. The most promising method consists of a highly scalable advanced greedy algorithm. Our results indicate that the outcomes of this method are, on average, very close to the outcomes of the other methods investigated, while it may be easily applied by (large) hospitals. The findings are robust with respect to fluctuations in long term OR schedules.

1. Introduction

Throughout the developed world, health care costs have shown a tremendous increase over the last decades. Since 1970 the inflation adjusted government spending on health care has risen by nearly 5% per year (Hagist & Kotlikoff, 2005), currently averaging 9% of gross domestic product (OECD, 2014). At the same time, patient waiting lists have become longer and are nowadays a major problem (Worthington, 1991). Managerial efforts to control the steeply rising health care costs and long patient waiting lists have not only placed attention on main health care processes, but have focused especially on efficiency management of various, priorly neglected, side processes. One such side process which has received recent attention is hospital sterilization logistics.

Hospital sterilization logistics is a relatively large side process in the hospital sector. Hospitals in developed countries typically have invested millions of euros in sterile instruments used for surgeries and other procedures. Optimizing the logistic design of the sterilization processes can free substantial amounts of money and working capital. It is estimated that in a small country such as the Netherlands, the saving potential is well over 100 million euros (van de Klundert, Muls, & Schadd, 2008). Moreover, the optimization of the sterilization logistics can lead to increased throughput, allowing more patients to be helped in a fixed time frame. This can help reduce patient waiting lists and reduce opportunity cost of health care which is currently not provided.

In this paper, we focus on the inventory management of sterile instruments. These sterile instruments flow in a return cycle between the central sterilization department (CSD) and for instance the operating theater. The sterile instruments are mostly grouped in special trays. Such a tray can contain the items needed for a particular surgery, but it can also happen that the content of a tray is needed for several types of surgery, or that one type of surgery requires multiple trays of different types. Hospitals face three questions regarding the management of instrument trays: (i) how many trays of each type should be obtained, (ii) how should the tray types be composed, and (iii) how should the tray types be assigned to surgeries? The problem of optimizing the tray composition is called the *tray optimization problem* (TOP).

The first two questions of the TOP are interlinked: the number of required trays and instruments depends largely on the composition of the trays. Optimizing the composition of trays can lead to substantial cost savings and to increased availability of instruments, see van de Klundert et al. (2008). In this research,

* Corresponding author.
E-mail addresses: dollevoet@ese.eur.nl (T. Dollevoet), J.T.vanEssen@tudelft.nl (J.T. van Essen), glorie@ese.eur.nl (K.M. Glorie).

https://doi.org/10.1016/j.ejor.2018.05.051
0377-2217/

we investigate ways for solving the TOP to minimize cost whilst guaranteeing instrument availability.

Substantial savings can already be achieved by removing unused instruments from the existing trays. The contents of a tray develop historically as new surgeons place additional instruments on a tray while the instruments used by retired surgeons remain on the tray. In addition, surgeons may prefer different instruments for the same surgery which increases the inventory and sterilization costs. By reaching consensus on which instruments to use for each surgery type, substantial savings can be made.

As mentioned, even higher cost savings can be achieved by composing the trays from scratch by solving the TOP. Currently, only few papers consider this tray optimization problem. Van de Klundert et al. (2008) present an Integer Linear Program (ILP) to solve the TOP. However, the authors also prove that the TOP is NP-hard, and therefore, no solution is guaranteed within polynomial time.

Reymondon, Pellet, and Marcon (2008) introduce a Simulated Annealing (SA) approach to solve the TOP. As the computation time for the presented SA approach is still extremely long for realistic instance sizes, they also introduce a simpler heuristic. This heuristic starts with two extreme solutions, namely (i) one tray for each instrument, and (ii) one tray for each surgery type. For both extreme solutions, the cost per instrument is calculated. If it is cheaper to individually wrap an instrument, this particular instrument is removed from the trays in the second extreme solution. A next step would be to distribute the individually wrapped instruments over the created trays to further reduce costs. However, this step is not yet implemented.

The contribution of this paper is threefold. First, we give an overview of the existing models and solution approaches for the TOP. Second, we present three new solution methodologies. The first one is based on delayed row & column generation (Muter, Birbil, & Bülbül, 2013), the second is a greedy heuristic, and the third is a genetic algorithm. Third, we compare the new solution methodologies to the existing approaches on several problem instances derived from real-world data sets. We perform numerical experiments to assess both the quality and the computation time of the solution approaches.

A key aspect of the TOP is that the composition of the trays and the assignment of trays to surgeries is considered simultaneously. A similar structure is observed in many other applications as well. For example, consider a workforce consisting of employees with different skill sets. A number of tasks is given that all require a set of skills and have to be performed by a group of employees. In order to be able to manage the workforce smoothly, the employees are distributed over teams. Then, the teams are assigned to the tasks. In this assignment, it must be ensured that the employees in a team have the required skills for the tasks the team is assigned to. The approaches described in this paper can be applied to this problem as well.

This paper is structured as follows. In Section 2, we formally introduce the problem. In addition, we describe several variants for the objective function and capacity constraints. In Section 3, several exact and heuristic solution approaches are described. A simulation approach to compare the developed solution approaches in a realistic setting is introduced in Section 4. In Section 5, we describe and analyze the computational results for the introduced solution methods and their performance in the simulation study. Section 6 presents conclusions and gives recommendations for future research.

2. Problem formulation

The basis of the considered problem is creating instrument trays and assigning these instrument trays to surgeries. The instrument trays are created by assigning several instruments to a tray while respecting some capacity constraints which are described in Section 2.1. The created instrument trays are assigned to surgeries such that for each surgery the required instruments are available in the assigned instrument trays. The set of instruments is given by set I, the set of surgeries by set J, and the set of instrument trays by set K. Integer variable X_{ik} denotes the amount of instruments $i \in I$ assigned to instrument tray $k \in K$. Integer variable Y_{jk} indicates how many instrument trays of type $k \in K$ are assigned to surgery $j \in J$. For each surgery $j \in J$, the number of instruments of type $i \in I$ needed to perform this surgery is given by parameter d_{ij}. Then, the following constraint ensures that at least d_{ij} instruments of type $i \in I$ are available in the instrument trays $k \in K$ assigned to surgery $j \in J$.

$$\sum_{k \in K} X_{ik} Y_{jk} \geq d_{ij}, \quad \forall i \in I, j \in J. \tag{1}$$

Note that this constraint is non-linear, however, it can be easily linearized as shown in Section 3.

2.1. Capacity restrictions

As mentioned, the created instrument trays must respect some capacity constraints. Normally, a tray has a restriction on the weight and volume it can contain. To specify the restrictions on the total weight and volume a tray can contain, we introduce α_i and β_i as the volume and weight of instrument $i \in I$, respectively. The maximum allowed weight and volume on a tray is given by r_α and r_β, respectively. This leads to the following two constraints.

$$\sum_{i \in I} \alpha_i X_{ik} \leq r_\alpha, \quad \forall k \in K, \tag{2}$$

$$\sum_{i \in I} \beta_i X_{ik} \leq r_\beta, \quad \forall k \in K. \tag{3}$$

However, in practice, the weight and volume of the instruments may not be available. Therefore, an alternative is to limit the number of instruments that can be placed on a certain tray. This can be done with the following constraint, where r_γ represents the maximum number of instruments allowed on a tray.

$$\sum_{i \in I} X_{ik} \leq r_\gamma, \quad \forall k \in K. \tag{4}$$

These three constraints can also be summarized into the following constraint:

$$\sum_{i \in I} p_i^n X_{ik} \leq r^n, \quad \forall k \in K, n \in N, \tag{5}$$

where $N = \{1, 2, 3\}$ is the set of characteristics of the instruments and trays. In particular, $p_i^1 = \alpha_i$, $p_i^2 = \beta_i$, $p_i^3 = 1$, $r^1 = r_\alpha$, $r^2 = r_\beta$ and $r^3 = r_\gamma$.

Note that it is preferred to consider the weight and volume of the instruments when filling the instrument trays. This information should in general be known by the manufacturer, but in case this information is not available, Constraint (4) can be used.

2.2. Objective function

The preferred assignment made by variables X_{ik} and Y_{jk} is the one that minimizes the total incurred costs. These costs can be divided into several parts, namely:

- fixed periodic costs,
- sterilization costs,
- handling costs,
- costs for new tray type.

Please cite this article as: T. Dollevoet et al., Solution methods for the tray optimization problem, European Journal of Operational Research (2018), https://doi.org/10.1016/j.ejor.2018.05.051

2.2.1. Fixed periodic costs

The fixed periodic costs consist of several parts, namely acquiring or depreciation costs and storage costs. These fixed periodic costs can be specified for instrument $i \in I$ by a_i and for each tray by c_1. To determine the total fixed periodic costs, we need to know how many trays of type $k \in K$ should be acquired, which is denoted by variable N_k. Usually, a tray can only be used once a day due to the time needed to sterilize a tray and its instruments. This means that when multiple surgeries needing tray $k \in K$ are scheduled on the same day, more trays of this type are needed. To determine these costs correctly, we thus need the OR-schedule. The OR-schedule is described by parameter s_{jt} which indicates the number of times surgery $j \in J$ is performed on day $t \in T$. Here, set T gives the set of days in the considered planning horizon. Then, the total fixed periodic costs are given by

$$\sum_{k \in K} \sum_{i \in I} a_i X_{ik} N_k + c_1 \sum_{k \in K} N_k, \tag{6}$$

where the value of N_k is determined by

$$\sum_{j \in J} s_{jt} Y_{jk} \leq N_k, \quad \forall k \in K, t \in T. \tag{7}$$

Note that cost function (6) is non-linear, however, it can be easily linearized as shown in Section 3.

Unfortunately, the OR-schedule may change over time and may even not be available. Therefore, another way to determine the fixed periodic costs is to use parameter m_j which indicates an estimation of the maximum number of times surgery $j \in J$ is scheduled on the same day. Then, the variable N_k can be determined by

$$m_j Y_{jk} \leq N_k, \quad \forall k \in K, j \in J. \tag{8}$$

The downside of this approach is that we do not consider the case that two types of surgeries needing the same instrument tray are scheduled on the same day. Thus, this approach gives a lower bound on the realized fixed periodic costs. An upper bound on the fixed periodic costs is given when the value of N_k is determined by

$$\sum_{j \in J} m_j Y_{jk} \leq N_k, \quad \forall k \in K. \tag{9}$$

2.2.2. Sterilization costs

The sterilization costs are incurred whenever an instrument tray is sterilized after it has been used for surgery. In practice, all instruments placed on a tray used for surgery are sterilized, even though they might not have been used during surgery. Similarly as for the fixed periodic costs, the sterilization costs can be divided into sterilization cost b_i for instrument $i \in I$ and sterilization cost c_2 for an instrument tray. The number of times f_j a tray and its instruments are sterilized in a given planning horizon can be deduced from the OR-schedule s_{jt} and is given by $f_j = \sum_{t \in T} s_{jt}$ if an OR-schedule is available. When no OR-schedule is available, often an estimate of f_j can be made which can be used to determine the sterilization costs as follows:

$$\sum_{k \in K} \sum_{i \in I} \sum_{j \in J} b_i f_j X_{ik} Y_{jk} + c_2 \sum_{k \in K} \sum_{j \in J} f_j Y_{jk}. \tag{10}$$

2.2.3. Handling costs

The third mentioned costs are the handling costs. These are the costs for preparing instrument trays to be used during surgery. These costs are defined by the number of times f_j a certain instrument tray is used during the planning horizon. The total handling costs are given by the following formula, where c_3 gives the handling costs for a tray:

$$c_3 \sum_{k \in K} \sum_{j \in J} f_j Y_{jk}. \tag{11}$$

2.2.4. Costs for new tray type

The costs mentioned last are the costs for creating a new tray type. These costs c_4 are incurred because creating a new tray type increases the amount of administrative work and inventory control. These costs thus depend on the number of tray types that are created. To determine this number, we introduce binary variable Z_k which indicates whether tray type $k \in K$ is used, i.e., whether one or more instruments are assigned to this tray type. This leads to the following costs:

$$c_4 \sum_{k \in K} Z_k, \tag{12}$$

where the value of Z_k is determined by

$$N_k \leq M Z_k, \quad \forall k \in K, \tag{13}$$

with M a sufficiently large number.

If costs c_4 are not known, we can also put a limit r on the number of created tray types. This can be achieved with the following two constraints:

$$N_k \leq M Z_k, \quad \forall k \in K, \tag{14}$$

$$\sum_{k \in K} Z_k \leq r. \tag{15}$$

3. Solution methodology

In the previous section, we have introduced the TOP and specified the objective function and all the relevant constraints. We noted that Constraint (1) and the cost contributions in (6) and (10) are non-linear in the decision variables. In this section, we present several solution methods to solve the TOP. First, we present an exact formulation in which the quadratic terms are linearized. Then, we present four heuristic approaches. The first one is based on row & column generation. The second one is a greedy construction heuristic. The third heuristic applies simulated annealing and the fourth one a genetic algorithm.

3.1. Exact solution method

In this section, we formulate the problem defined in Section 2 as an Integer Linear Program (ILP). This ILP is based on the model presented by van de Klundert et al. (2008). Its main feature is a linearization of the terms $X_{ik}Y_{jk}$ and $X_{ik}N_k$. In order to linearize these terms, the variable X_{ik} is first written as a sum of binary variables $\bar{X}_{lk}$, where the index l ranges over a set of instruments *specimen* of type i:

$$X_{ik} = \sum_{l \in L_i} \bar{X}_{lk}. \tag{16}$$

Recall that the index $i \in I$ indicates the instrument *type*. This means that if instruments of a certain type $i \in I$ might appear more than once in a tray, there are *several* variables $\bar{X}_{lk}$ corresponding to instruments of type i. In order to determine the required size of the set L_i, define $D_i = \max_{j \in J} d_{ij}$ as the maximum number of instruments of type i needed for a surgery. Given that surgeries require at most D_i instruments of type i, each tray will contain at most D_i instruments of type i as well. Therefore, we need D_i elements in the set L_i. Note that the set L_i is pseudo-polynomial in size. Second, we introduce variables Q^1_{jkl} and Q^2_{kl} that represent the products $Y_{jk}\bar{X}_{lk}$ and $N_k\bar{X}_{lk}$, respectively. By definition, it then holds for all $i \in I$, $j \in J$ and $k \in K$ that

Please cite this article as: T. Dollevoet et al., Solution methods for the tray optimization problem, European Journal of Operational Research (2018), https://doi.org/10.1016/j.ejor.2018.05.051

$$\sum_{l\in L_i} Q^1_{jkl} = \sum_{l\in L_i} Y_{jk}\bar{X}_{lk} = Y_{jk}\sum_{l\in L_i}\bar{X}_{lk} = Y_{jk}X_{ik}.$$

Similarly, for all $i\in I$ and $k\in K$, we have

$$\sum_{l\in L_i} Q^2_{kl} = \sum_{l\in L_i} N_k\bar{X}_{lk} = N_k\sum_{l\in L_i}\bar{X}_{lk} = N_kX_{ik}.$$

Using the equations above, the TOP can be formulated as follows:

$$\min c_1\sum_{k\in K} N_k + (c_2+c_3)\sum_{k\in K}\sum_{j\in J} f_jY_{jk} + c_4\sum_{k\in K} Z_k + \sum_{k\in K}\sum_{i\in I}\sum_{l\in L_i}\sum_{j\in J} b_if_jQ^1_{jkl} + \sum_{k\in K}\sum_{i\in I}\sum_{l\in L_i} a_iQ^2_{kl} \tag{17}$$

such that

$$Q^1_{jkl} \le Y_{jk}, \quad \forall j\in J, k\in K, i\in I, l\in L_i, \tag{18}$$

$$Q^1_{jkl} \le M_1\bar{X}_{lk}, \quad \forall j\in J, k\in K, i\in I, l\in L_i, \tag{19}$$

$$Q^1_{jkl} \ge M_1(\bar{X}_{lk}-1) + Y_{jk}, \quad \forall j\in J, k\in K, i\in I, l\in L_i, \tag{20}$$

$$Q^2_{kl} \le N_k, \quad \forall k\in K, i\in I, l\in L_i, \tag{21}$$

$$Q^2_{kl} \le M_2\bar{X}_{lk}, \quad \forall k\in K, i\in I, l\in L_i, \tag{22}$$

$$Q^2_{kl} \ge M_2(\bar{X}_{lk}-1) + N_k, \quad \forall k\in K, i\in I, l\in L_i, \tag{23}$$

$$\sum_{k\in K}\sum_{l\in L_i} Q^1_{jkl} \ge d_{ij}, \quad \forall i\in I, j\in J, \tag{24}$$

$$\sum_{i\in I} p^n_i\sum_{l\in L_i}\bar{X}_{lk} \le r^n, \quad \forall k\in K, \forall n\in N, \tag{25}$$

$$\sum_{j\in J} s_{jt}Y_{jk} \le N_k, \quad \forall k\in K, t\in T, \tag{26}$$

$$N_k \le MZ_k, \quad \forall k\in K, \tag{27}$$

$$Q^1_{jkl}\in \mathbf{N}, \quad \forall j\in J, i\in I, l\in L_i, k\in K, \tag{28}$$

$$Q^2_{kl}\in \mathbf{N}, \quad \forall i\in I, l\in L_i, k\in K, \tag{29}$$

$$Y_{jk}\in \mathbf{N}, \quad \forall j\in J, k\in K, \tag{30}$$

$$N_k\in \mathbf{N}, \quad \forall k\in K, \tag{31}$$

$$\bar{X}_{kl}\in\{0,1\}, \quad \forall k\in K, i\in I, l\in L_i, \tag{32}$$

$$Z_k\in\{0,1\}, \quad \forall k\in K. \tag{33}$$

The objective function contains all terms presented in Sections 2.2. It has been linearized by replacing the quadratics terms, $\bar{X}_{lk}Y_{jk}$ and $\bar{X}_{lk}N_k$, by Q^1_{jkl} and Q^2_{kl}, respectively. Constraints (18)–(20) make sure that the variables Q^1_{jkl} obtain their correct value. Here, M_1 is an upper bound on the number of times a certain tray is used for the same surgery. Similarly, Constraints (21)–(23) make sure that the variables Q^2_{lk} obtain their correct value. The parameter M_2 is an upper bound on the number of trays of a certain type that are needed. Constraints (24) ensure that the required instruments are available for each surgery. Constraints (25) impose capacity restrictions for the trays. Constraints (26) compute the number of trays of each type that are needed. If the surgery schedule is not available, this constraint can be replaced by (8) or (9). Constraints (27) ensure that the number of different tray types is counted correctly. The other constraints give the ranges of the decision variables.

Note that the set K of instrument trays is assumed given. This means that the maximum number of trays to be formed is input to the model and must be specified before the model can be solved. In our computational experiments, the size of the set K is based on the solution of the greedy heuristic, which is described in Section 3.2.2.

Note that the number of constraints in this formulation is pseudo-polynomial and that it requires many decision variables. In particular, for every combination of $j\in J$, $k\in K$, $i\in I$ and $l\in L_i$, a decision variable Q^1_{jkl} and three constraints are introduced. Therefore, for real-life instances, this ILP might become huge. Furthermore, because of the big-M constraints that are needed for the linearization, the LP-bounds might be relatively weak. In order to improve the quality of the LP-relaxation, we can add the following valid inequalities to the formulation.

$$\sum_{k\in K} N_k \ge \sum_{j\in J} s_{jt}, \quad \forall t\in T, \tag{34}$$

$$\sum_{k\in K}\sum_{l\in L_i}\bar{X}_{lk} \ge 1, \quad \forall i\in I. \tag{35}$$

Constraints (34) ensure that the number of trays that are acquired is larger than the number of surgeries on each day. This holds because for all surgeries on a given day, at least one tray must be acquired. Constraints (35) make sure that at least one instrument of every type is present in the collection of trays.

Another concern with the formulation is the symmetry that is present. To give an example of this symmetry, consider two indices $k_1, k_2\in K$. If one would, for a given solution, replace the values of all decision variables with index k_1 for those with index k_2 and vice versa, the same solution would be obtained. A consequence of symmetries like these are a multitude of feasible solutions with exactly the same solution value. This might lengthen the computation time considerably.

By introducing symmetry-breaking constraints into the model, the number of solutions with exactly the same objective value can be reduced. As stated by Sherali and Smith (2001), such symmetry-breaking constraints might shorten the computation time. One way of breaking symmetry in the model is to ensure that the trays generated are ranked in a certain order. For example, one could make sure that the trays that are not used are those with higher indices k. In order to do so, denote $K=\{1,2,\ldots,|K|\}$. The following constraints can be added to the model.

$$Z_1 \ge Z_2 \ge \ldots \ge Z_{|K|}. \tag{36}$$

One other way to rank the trays would be to sort them by the number of instruments included in the trays. The number of instruments in a given tray $k\in K$ can be found by summing over the variables $\bar{X}_{lk}$. This leads to the following symmetry-breaking constraints:

$$\sum_{i\in I}\sum_{l\in L_i}\bar{X}_{l1} \ge \sum_{i\in I}\sum_{l\in L_i}\bar{X}_{l2} \ge \cdots \ge \sum_{i\in I}\sum_{l\in L_i}\bar{X}_{l|K|}. \tag{37}$$

Finally, we note that other linearizations than (16) are possible. Two alternative linearizations, based on the unary and binary

expansion of the integer variables to be multiplied, have been proposed by Harjunkoski, Pörn, Westerlund, and Skrifvars (1997). These linearizations can be applied to the TOP as well.

3.2. Heuristic solution methods

In the previous section, we formulated the TOP as an ILP. This ILP contains many decision variables and big-M constraints. As a consequence, it might be impossible to find a solution for larger, real-world TOP instances. In this section, we develop four heuristic approaches to find a feasible solution to the TOP.

3.2.1. Delayed row & column generation

In this section, we develop a row & column generation approach to solve the linear relaxation of the TOP. The theory on row & column generation has been introduced by Muter et al. (2013) and applied to time-constrained routing by Muter, Birbil, and Bülbül (2018); Muter, Birbil, Bülbül, and Şahin (2012).

The key aspect of this approach is that it considers a set of *given* instrument trays. As a consequence, for a tray $k \in K$, the number of instruments X_{ik} is a parameter instead of a variable. To emphasize this aspect, we denote for a given tray $k \in K$, the *parameter* x_{ik} as the number of instruments of type i in tray k.

We start with a given set of trays $\bar{K} \subseteq K$. Then, we alternately solve a master problem with this reduced set of trays and a pricing problem where we generate new trays that can be added to the set $\bar{K}$. When adding a newly generated tray k to the set $\bar{K}$, we also have to add constraints to the master problem. We have to take this into account when solving the pricing problem. The ILP formulation of the TOP, with a given set of instrument trays $\bar{K}$, reads as follows:

$$\min \sum_{k \in \bar{K}} c_4 Z_k + \sum_{k \in \bar{K}} c_k^1 N_k + \sum_{k \in \bar{K}} c_k^2 \sum_{j \in J} f_j Y_{jk} \tag{38}$$

such that

$$\sum_{k \in \bar{K}} Y_{jk} x_{ik} \geq d_{ij}, \quad \forall i \in I, j \in J, \tag{39}$$

$$\sum_{j \in J} s_{jt} Y_{jk} \leq N_k, \quad \forall k \in \bar{K}, t \in T, \tag{40}$$

$$N_k \leq M Z_k, \quad \forall k \in \bar{K}, \tag{41}$$

$$N_k \in \mathbf{N}, \quad \forall k \in \bar{K}, \tag{42}$$

$$Y_{jk} \in \mathbf{N}, \quad \forall k \in \bar{K}, j \in J, \tag{43}$$

$$Z_k \in \{0, 1\}, \quad \forall k \in \bar{K}. \tag{44}$$

Here, we define the cost parameters c_k^1 and c_k^2 as follows:

$$c_k^1 = c_1 + \sum_{i \in I} a_i x_{ik},$$
$$c_k^2 = c_2 + c_3 + \sum_{i \in I} b_i x_{ik}.$$

From now on, we refer to this problem of assigning a given set of trays to surgeries as the *tray assignment problem*. It is proven in Appendix B that this problem is NP-complete as well. Note that Constraints (39) are the only constraints coupling the different trays. If we would dualize these constraints into the objective function, the problem would decompose over the tray types. This observation is key for our row & column generation framework.

As is common for a column generation approach, we solve the LP-relaxation of the above ILP for a given set $\bar{K}$ of tray types. Given the dual solution to this problem, we generate new tray types in a pricing problem and add these to the set $\bar{K}$. We repeat this procedure until no tray types with negative reduced costs can be found. When solving the pricing problem, we generate complete tray types: we simultaneously add the variables Z_k, N_k, and Y_{jk} and Constraints (40) to the LP. When solving the pricing problem, we therefore have to anticipate the dual values of the constraints currently missing from the master problem. It is proven in Appendix A that the reduced costs of a given tray $k \in K$ are given by the following linear program:

$$z_k(\lambda) = \min \frac{c_4}{M} + c_1 + \sum_{i \in I} a_i x_{ik} + \sum_{j \in J} \left((c_2 + c_3) f_j + \sum_{i \in I} (b_i f_j - \lambda_{ij}) x_{ik} \right) Y_{jk}$$

such that

$$\sum_{j \in J} s_{jt} Y_{jk} \leq 1, \quad \forall t \in T,$$
$$Y_{jk} \geq 0, \quad \forall j \in J.$$

Here, the variables λ_{ij} are the dual multipliers corresponding to Constraints (39). If there does not exist a new tray $k \in K \setminus \bar{K}$ with negative reduced costs, we obtained the optimal solution to the LP-relaxation of the TOP. (The proof is given in the appendix.) Otherwise, if a tray $k \in K \setminus \bar{K}$ with negative reduced costs exists, we add it to the set $\bar{K}$ and repeat the procedure.

In order to determine whether $z_k(\lambda)$ is positive for all $k \in K \setminus \bar{K}$, we solve the following non-linear pricing problem. Note that we have omitted the index k from the X- and Y-variables for brevity purposes.

$$z_{pp}(\lambda) = \min \frac{c_4}{M} + c_1 + \sum_{i \in I} a_i X_i + \sum_{j \in J} \left((c_2 + c_3) f_j + \sum_{i \in I} (b_i f_j - \lambda_{ij}) X_i \right) Y_j$$

such that

$$\sum_{i \in I} p_i^n X_i \leq r_n, \quad \forall n \in N,$$
$$\sum_{j \in J} s_{jt} Y_j \leq 1, \quad \forall t \in T,$$
$$X_i \in \mathbf{N}, \quad \forall i \in I,$$
$$Y_j \geq 0, \quad \forall j \in J.$$

In order to solve this non-linear program, we first linearize it and then solve it as an ILP using a commercial solver. For completeness, the linearization is given in the appendix. The solution of this pricing problem is a new tray that should be added to the set $\bar{K}$. By iterating this process, the linear relaxation of the TOP can be solved. The next step is to find an integer solution. In order to do so, we apply a commercial solver to the ILP (38)–(44) using all trays that are generated when solving the linear relaxation.

3.2.2. Greedy heuristic

In Section 3.2.1, the quadratic terms in the model were dealt with by decomposing the TOP into a pricing problem of composing the trays and a master problem of assigning the trays to surgeries. For a given set of trays, the master problem is to decide which trays to use and which trays to assign to each surgery. We now describe an alternative approach to decide on the tray composition. In this approach, the composition of the trays is generated via a greedy heuristic.

For this greedy heuristic, we generate a set of trays and determine the optimal selection of the trays to be used and the assignment of trays to surgeries by (38)–(44).

Please cite this article as: T. Dollevoet et al., Solution methods for the tray optimization problem, European Journal of Operational Research (2018), https://doi.org/10.1016/j.ejor.2018.05.051

A tray is specified by how many instruments of each instrument type $i \in I$ are placed on this tray. The set of trays used in solving the ILP given by (38)–(44) is determined by the greedy heuristic. We include nine options for constructing trays which guarantee a feasible solution to our original problem.

1. Create trays for each surgery type $j \in J$. For each instrument $i \in I$, we add d_{ij} instruments to the tray until the tray is full, i.e., if the next instrument type would be added, Constraint (5) would be violated. In this case, we open a new tray and continue the process.
2. Sort the surgeries in decreasing order of $\Sigma_{i \in I} d_{ij}$. For each surgery $j \in J$, add an instrument type to the tray if this instrument is used for surgery $j \in J$ and if this instrument is not yet added to another tray in this step. For this instrument type $i \in I$, we add $\max_{j \in J} d_{ij}$ instruments to the tray. If Constraint (5) is violated, we open a new tray.
3. Sort the instrument types in decreasing order of the number of surgeries that use this instrument type. Then, fill the tray with $\max_{j \in J} d_{ij}$ instruments of this type. If Constraint (5) is violated, we open a new tray.
4. Same as 3. except that we add $\left\lceil \frac{\sum_{j \in J} d_{ij}}{|J|} \right\rceil$ instruments of type $i \in I$ to the tray.
5. Same as 3. except that we add one instrument of type $i \in I$ to the tray.
6. Create trays with instruments needed for all surgeries. For each instrument $i \in I$ that is used for all surgeries, add $\max_{j \in J} d_{ij}$ instruments of this type to the tray. If Constraint (5) is violated, we open a new tray.
7. Same as 6. except that we add $\left\lceil \frac{\sum_{j \in J} d_{ij}}{|J|} \right\rceil$ instruments of type $i \in I$ to the tray.
8. Same as 6. except that we add one instrument of type $i \in I$ to the tray.
9. Create trays with instruments needed for only one surgery. For each surgery $j \in J$, add d_{ij} instruments of type $i \in I$ to the tray if this instrument is only used for this surgery. If Constraint (5) is violated, we open a new tray.

After these instrument trays are generated, we solve the ILP given by (38)–(44) to determine which trays will be used and to determine the assignment of the chosen trays to the surgeries.

3.2.3. Simulated annealing

The approach described in this section is similar to the greedy heuristic described in Section 3.2.2, however, the initial set of trays found by the greedy heuristic is further improved by using simulated annealing as proposed by Reymondon et al. (2008). Simulated annealing has proven to be very effective for solving large-scale combinatorial optimization problems (see Kirkpatrick, Gelatt, & Vecchi, 1983). In this section, we develop a simulated annealing algorithm for the TOP.

Simulated annealing is a local search heuristic specially designed to escape from local minima. In every iteration, a feasible solution is slightly modified and evaluated. If the solution quality has improved, the new solution is accepted. In plain local search approaches, the new solution is rejected if the solution quality deteriorates. By doing so, the algorithm might get stuck in a local minimum. In simulated annealing, a solution with a worse quality is therefore accepted with a certain probability. This probability depends on the quality decrease and on a parameter called the temperature. By decreasing the temperature during the execution of the algorithm, the algorithm rejects more solutions with a worse quality in later iterations. In contrast, in early stages of the algorithm, when the temperature is higher, the method is able to escape from local minima. In what follows, we describe how to evaluate a given solution, which modifications of the current solution are allowed, and which cooling scheme we employ. The initial solution for SA is determined by the greedy heuristic.

In the simulated annealing algorithm, a solution s is represented by a set of trays. For a given set of trays, the used ILP solver is given a fixed amount of time to determine the assignment of trays to surgeries using the ILP given by (38)–(44). The value of the objective function given by (38) represents the solution value of s which is denoted by $f(s)$. Note that the costs of composing the trays are included in (38).

For a given solution s_1, we apply one of the following modifications with equal probability to obtain a neighbour solution s_2.

1. Exchange of instruments: randomly select two trays. Randomly select one instrument from the first tray and place it on the second tray. Then, randomly select one instrument from the second tray and place it on the first tray.
2. Adding instruments: randomly select a tray. Then, randomly select one instrument type and add a random number of instruments of this type to the selected tray.
3. Removing instruments: randomly select a tray. Then, randomly select one instrument type and remove a random number of instruments of this type from the selected tray.
4. Exchange for one instrument type: randomly select two trays. Then, randomly select one instrument type and exchange all instruments of this type between the two selected trays.
5. Exchange for multiple instrument types: randomly select two trays. Next, randomly select two different instrument types from the ordered set of instrument types, where the instruments are ordered by their index. Then, exchange all instruments of the instrument types that are in between the two selected instrument types between the two selected trays. More specifically, for each instrument type between the two selected instrument types, the total number of instruments of this type is exchanged between the two trays.

If the modification applied to s_1 leads to trays violating Constraint (5), we reset the solution to s_1 and apply another modification. This step is repeated until a new solution s_2 is obtained for which none of the trays violate (5).

After a new solution s_2 is obtained, we determine the objective function value $f(s_2)$. If $f(s_2) \leq f(s_1)$, we always accept s_2 as the current solution. Otherwise, if $f(s_2) > f(s_1)$, we accept the new solution with a probability

$$p(s_1, s_2) = \exp\left(\frac{f(s_1) - f(s_2)}{T}\right),$$

where T is the current temperature. The temperature T equals $\overline{T}$ at initialization and decreases during the execution of the algorithm. In our implementation, the temperature is multiplied by a factor $\alpha < 1$ every N iterations. The algorithm terminates if $T < \underline{T}$. The parameters $\overline{T}$, $\underline{T}$, α and N are control parameters of the algorithm and are specified in Section 5.

3.2.4. Genetic algorithm

In this section, we describe a Genetic Algorithm (GA) to improve the initial set of instrument trays found by the greedy heuristic described in Section 3.2.2. GA is a probabilistic search algorithm, which imitates the process of natural selection and evolution. The process starts with an initial population of solutions, where solutions are treated as individuals, also denoted by chromosomes. The fitness of each individual is based on the corresponding objective function value. Pairs of individuals of a given population are selected to act as parents and reproduce the next population of better individuals through a structured yet randomized information exchange, known as crossover. Diversity is added to the population by mutation. Unfit individuals in the

Please cite this article as: T. Dollevoet et al., Solution methods for the tray optimization problem, European Journal of Operational Research (2018), https://doi.org/10.1016/j.ejor.2018.05.051

population are replaced using the concept of survival of the fittest. This evaluation-selection-reproduction cycle is repeated until a satisfactory solution is found or other stopping criteria are met.

In our implementation of GA, an individual or chromosome represents a tray, where the tray is encoded by how many instruments of each instrument type $i \in I$ are placed on this tray. Our implementation of GA differs from the way we implemented SA as we consider the trays to be the individuals in our population instead of a feasible combination of trays. This is because generating a feasible combination of trays from two other feasible combinations of trays is not trivial. In contrast, generating a single new tray from two other feasible trays is much easier to implement. Thus, our approach for GA is slightly different from standard GA, as multiple individuals are needed to provide a solution to our original problem. As in the other described heuristic solution methods, our set $\bar{K}$ consists of the trays in our population, which means that x_{ik} is an input parameter instead of a variable. Given a certain population, the ILP given by (38)–(44) has to be solved to find a solution to our original problem.

As previously stated, the initial population is given by the set of instrument trays generated by the greedy heuristic. In the next sections, we describe how we determine the fitness value of each individual and our reproduction scheme.

Fitness function. As we need several trays to form a feasible solution, it is hard to define a fitness value for each individual. We have chosen to determine the fitness value by solving $|\bar{K}|+1$ ILPs. The used ILP solver is given a fixed amount of time to solve each of these $|\bar{K}|+1$ ILPs. The first ILP to be solved is the ILP given by (38)–(44). This ILP determines a solution for our given population represented by set $\bar{K}$. After this, we solve another ILP for each tray. If tray $k \in \bar{K}$ is chosen in the solution to (38)–(44), we solve the ILP again without tray $k \in \bar{K}$. If tray $k \in \bar{K}$ is not chosen in the solution for (38)–(44), we solve the ILP again with the additional constraint that tray $k \in \bar{K}$ should be chosen. Then, we define the fitness value based on the following four criteria:

1. Number of times T_k that tray $k \in \bar{K}$ is chosen in a solution of one of the $|\bar{K}|+1$ ILPs.
2. Total number of trays NT_k of type $k \in \bar{K}$ chosen in the solutions of the $|\bar{K}|+1$ ILPs.
3. The objective function if tray $k \in \bar{K}$ is used in the solution, O_k.
4. The objective function if tray $k \in \bar{K}$ is not used in the solution, NO_k.

Using this information, the fitness value of tray $k \in \bar{K}$ is given by

$$\left(\max_{k' \in \bar{K}} O_{k'} - O_k\right) + \left((NO_k - O_k) - \min_{k' \in \bar{K}}(NO_{k'} - O_{k'})\right) + (T_k + NT_k). \tag{45}$$

The first part of the fitness value describes the relative advantage of using tray $k \in \bar{K}$ in the solution compared to the other trays in $\bar{K}$. In the second part of the fitness value, we consider the difference in the objective function when tray $k \in \bar{K}$ is used or not. For each tray $k \in \bar{K}$, we compare this to the minimum difference over all trays, which again gives the relative advantage. The third part of the fitness value counts the number of times T_k that a tray of type $k \in \bar{K}$ is selected in the solution of one of the $|\bar{K}|+1$ ILPs and the total number of trays NT_k of type $k \in \bar{K}$ that are selected in the solutions of the $|\bar{K}|+1$ ILPs. Note that the scale of the values of the various terms differs, however, this results exactly in the scaling we want.

Reproduction. In each iteration, we replace either a fixed number of individuals by children or we replace all individuals that have not been selected in all but one ILP solved to determine the fitness function. Therefore, the size of the population does not change throughout the execution of the genetic algorithm. To reproduce children, we first have to select parents. In order to do this, we first translate the fitness value to a probability such that the probabilities of all individuals together sum up to one. To actually reproduce a child, we consider four different methods.

1. 1-point crossover: we randomly select two parents according to their selection probability. Then, we randomly select an instrument type from the ordered set of instrument types. For the reproduced child, the number of instruments of the types up till this instrument type are a copy of the first parent and the number of instruments from this type on are a copy of the second parent.
2. 2-point crossover: we randomly select two parents according to their selection probability. Then, we randomly select two instrument types from the ordered set of instrument types. Up till the first selected instrument type and from the second instrument type on, the child is a copy of the first parent, and from the first selected instrument type until the second selected instrument type, the child is a copy of the second parent.
3. Uniform crossover: we randomly select two parents according to their selection probability. Then, for each instrument type, we randomly choose one of the parents to determine how many instruments of that type will be placed on the tray for the child.
4. Cloning: we randomly select one individual to be cloned.

After the reproduction process, we apply mutation. This means that the number of instruments of type $i \in I$ placed on the tray can be modified. For each instrument type $i \in I$, we apply mutation with a fixed mutation probability. If we apply mutation for instrument type $i \in I$, the number of instruments of this type placed on the tray is randomly generated and lies between 0 and $\max_{j \in J} d_{ij}$.

After a child is reproduced, we have to check if this child fulfills the capacity constraints (5). If not, we repeat the reproduction process.

The process ends after a fixed number of iterations V or if after a fixed number of iterations W no improved solution is found with $W < V$.

4. Simulation design

To compare the performance of the algorithms, we use a realistic simulator based on actual data from Dutch hospitals. The simulator is described in detail in van der Kooij (2015). In this section, we briefly explain the main aspects of the simulation procedures and the data sets they are based on. We distinguish between instance generation, in which we use a simulator to generate instances to evaluate the performance at the planning stage, and dynamic simulations, in which we evaluate the performance over a longer time horizon. The former is useful for comparing computation times and optimality gaps of the various algorithms. The latter is useful for comparing long term costs.

4.1. Data

Our simulations are based on actual CSD data from two large Dutch hospitals. We indicate these data sets as hospital 1 (H1) and hospital 2 (H2). Table 1 gives an overview of the characteristics of each data set. In this table, 'OR schedule' refers to the availability of a daily schedule of all surgeries, 'Surgery frequencies' refers to the availability of surgery specific scheduling frequencies (the overall number of times surgeries are performed in the data set), 'instrument demand' refers to the instruments required per surgery, and 'current tray composition' refers to a description of the composition of the current trays and the quantities in which they are available. The other fields concern statistics regarding the number of surgeries in the data set, the number of instruments,

Please cite this article as: T. Dollevoet et al., Solution methods for the tray optimization problem, European Journal of Operational Research (2018), https://doi.org/10.1016/j.ejor.2018.05.051

Table 1
Characteristics of the two data sets.

	H1	H2
OR schedule	Yes	Yes
Surgery frequencies	Yes	Yes
Instrument demand	Yes	Yes
Current tray composition	Yes	–
# Surgeries	16	174
# Instruments	87	1125
# Days	365	337
Annual tray depreciation	475	500
Max # instruments per tray	60	65
Sterilization costs	1	1
Handling costs	20	20

the number of days in the OR schedule, the annual tray depreciation costs, the maximum number of instruments per tray, the sterilization costs, and the handling costs.

As we do not have data on the weight and volume of the instruments, each instrument is assigned a unit volume and unit weight.

4.2. Instance generation

Instance generation is needed to evaluate how well algorithms perform in the planning stage. The data sets themselves can be used directly as instances. However, we also want to randomly generate instances. Such random instance generation allows for a larger variety of instances and for instances of larger size.

An instance for the TOP consists of instrument demand, an OR schedule, cost parameters and maximum tray sizes. The cost parameters and maximum tray sizes will be directly inherited from the hospital data set the new instance is based on. Therefore, it remains to randomly generate instrument demand and an OR schedule.

In order to generate instrument demand and an OR schedule for instances of arbitrary size, we make two assumptions. The first assumption concerns the generation of instrument demand and entails that the number of different instrument types required per surgery does not change when the ratio between instrument types and surgery types changes. In other words, we assume an individual surgery always requires about the same number of different instrument types, irrespective of how many instrument types there are in total. The second assumption concerns the generation of an OR schedule and entails that the number of performed surgeries increases linearly with the number of surgery types.

In our instance generator, we randomly generate instances based on one of the original data sets. This means that each generated instance is derived from either the H1 or the H2 data set.

4.2.1. Generation of instrument demand

In order to randomly generate instrument demand, we generate instruments as well as surgeries that are not included in the original data sets. The idea behind generating new instruments is that each instrument is based on a random instrument from the original instance (hospital data set). In our instance generator, a newly generated instrument inherits the demand frequency (percentage of surgeries demanding it) and the distribution of the demanded amount per surgery of a randomly selected instrument in the original data set. For each surgery requiring the new instrument, we randomly generate the demanded amount from the empirical distribution of the parent instrument.

As with instrument generation, new surgeries are also based on random surgeries from the original instance. Each surgery requires a different set of instruments. Some surgeries may require many specialized tools, while others may only require basic tools. Based on data analysis, van der Kooij (2015) concludes that for H1 and H2, instruments can be categorized into three groups with a different demand frequency (number of surgeries for which the instrument is required): rarely used instruments, moderately used instruments and frequently used instruments. Table 2 shows the demand frequencies for the different groups. A surgery can be characterized by the number of instruments it needs from each of the three instrument groups. When generating a new surgery, a random 'parent' surgery is selected from the original instance and the new surgery inherits the number of instruments used from each instrument group.

Table 2
Number of surgeries requiring an instrument for different categories of instruments.

	1: Rarely used instruments	2: Moderately used instruments	3: Frequently used instruments
H1	0–5	6–8	8–16
H2	0–40	40–100	101–152

Formally, the process of instrument demand generation can be described as follows. Let I and I' denote the set of all original instrument types and the set of all new instrument types, respectively, and let n_I denote the number of new instruments that need to be generated. Similarly, let J and J' denote the set of all original and all new surgery types, respectively, and let n_J denote the number of surgeries that need to be generated. Let $I_g \subset I$ and $I'_g \subset I'$ be the collections of all instruments belonging to group $g \in G = \{1, 2, 3\}$. These groups represent the categories of (1) rarely used instruments, (2) moderately used instruments, and (3) frequently used instruments, respectively. Let g_i denote the group of instrument $i \in I$ and let $f_{j,g}$ and $f'_{j,g}$ be the number of instrument types from group $g \in G$ required by surgery $j \in J$ and $j \in J'$, respectively. Let p_i and p'_i denote the percentage of surgeries demanding instrument $i \in I$ and $i \in I'$, respectively, and let d_i and d'_i denote the empirical distribution of the demanded amount per surgery for instrument $i \in I$ and $i \in I'$, respectively. Finally, let $D = \{D_{ij}\}$ be the new instrument demand, with $D_{ij} \in \mathbf{N}$ the number of times instrument $i \in I'$ is needed for surgery $j \in J'$. The new instrument demand can then be created using Algorithm 1.

4.2.2. Generation of an OR schedule

The OR schedule is a matrix with the surgeries as rows and the days as columns. The elements of the matrix indicate the number of times that a particular surgery is performed on a particular day. When generating a new OR schedule based on an existing one, we want to preserve some characteristics of the original plan. There are two different characteristics we want to maintain. The first characteristic is the week pattern for individual surgeries and the second characteristic is whether the plan for a particular surgery can be modeled by a Poisson distribution.

If, for a particular surgery, a Poisson distribution provides a significant fit for the planning distribution, the parameter of the Poisson distribution has to be determined. In order to preserve the week pattern in the new surgery plan, seven different λ's are computed, one for every day of the week. As estimator for these parameters, the Maximum Likelihood Estimator (MLE) for the Poisson distribution, is used. This MLE is equal to the sample mean for the respective day of the week.

If, for a particular surgery, a Poisson distribution does not provide a significant fit for the planning distribution, the empirical planning distribution is used. In order to also preserve the week pattern in this case, the number of times a surgery is performed on a given weekday in the new surgery plan is set equal to a random value in the past on the same weekday.

Please cite this article as: T. Dollevoet et al., Solution methods for the tray optimization problem, European Journal of Operational Research (2018), https://doi.org/10.1016/j.ejor.2018.05.051

Algorithm 1 Generation of instrument demand.

procedure DEMAND–GENERATION
  set $D \leftarrow 0, I' \leftarrow 0, J' \leftarrow 0$
  for $k = 1 \dots n_I$ **do**
    draw a random instrument i from I
    set $I' \leftarrow I' \cup \{i\}$, $p'_i \leftarrow p_i$, $d'_i \leftarrow d_i$, $I'_{g_i} \leftarrow I'_{g_i} \cup \{i\}$
  end for
  for $k = 1 \dots n_J$ **do**
    draw a random surgery j from J
    set $J' \leftarrow J' \cup \{j\}$, $f'_{j,g} \leftarrow f_{j,g}$ $\forall g \in G$
  end for
  for each $j \in J'$ **do**
    for each $g \in G$ **do**
      set $S \leftarrow I'_g$
      for $k = 1 \dots f'_{j,g}$ **do**
        for each $i \in S$ **do**
          compute $\pi'_i = p'_i / (\sum_{i' \in S} p'_{i'})$
        end for
        draw a random instrument i from S (every instrument a has a probability π'_a of being selected)
        draw D_{ij} from d'_i
        set $S \leftarrow S \setminus \{i\}$
      end for
    end for
  end for
end procedure

4.3. Dynamic simulation

Dynamic simulations are intended to evaluate the long term costs associated with the tray compositions generated by the algorithms. As all algorithms optimize with respect to either surgery frequencies or a fixed OR schedule, they are essentially heuristics to the dynamic setting in which the surgery plan is subject to change. At the very best, the frequencies with respect to which is optimized can be considered as estimates of the actual future frequencies. Similarly, the fixed OR schedule with respect to which is optimized can only be considered to be a proxy of the OR schedule over the entire time horizon. Of particular importance to cost assessment is that optimal tray combinations may depend on patterns of typical day combinations of surgeries. These factors make it important to assess long term costs in addition to the costs determined in the static simulation procedures.

For each algorithm, we use the following procedure to assess long term costs. First, an algorithm is used to construct a tray composition based on an instance generated by the static simulation procedure described in Section 4.2. Next, a realization of the OR schedule is generated for a longer time period. This time period contains twenty times the number of days of the original OR schedule. The tray composition is then evaluated based on this realization of the OR schedule. Outcome measures of interest are the total sterilization and handling costs over the longer time period, and the percentage of surgeries for which not all instruments are available in time.

We consider four methods for simulating an OR schedule:

1. historical frequencies;
2. perturbed historical frequencies;
3. historical sampling;
4. perturbed historical sampling.

All methods rely on Monte Carlo simulation using 1000 iterations.

4.3.1. Historical frequencies

The first method, 'historical frequencies', randomly allocates surgeries to planning days based on relative historical frequencies of surgery types. The number of surgeries scheduled on a day is equal to the number of surgeries of a random day in the original OR schedule.

4.3.2. Perturbed historical frequencies

The second method, 'perturbed historical frequencies', is similar to the first method except that now the surgery type frequencies are ex-ante perturbed by a perturbation factor. We consider a perturbation factor of 10%. This means that if for a certain surgery type the historical frequency is, for example, 8%, then the perturbed frequency used to determine the actual planning is within the range 7.2–8.8%.

4.3.3. Historical sampling

The third method, 'historical sampling', uses sampling with replacement to assign to each planning day a surgery schedule identical to the schedule on a randomly selected historical planning day. This method preserves the day combinations of surgeries in the planning stage, and therefore, none of the solutions should have a canceled surgery.

4.3.4. Perturbed historical sampling

The last method, 'perturbed historical sampling', is similar to the previous method except that now there is a 10% perturbation for each assignment of a surgery to a planning day compared to the historical planning day. To be more specific, each surgery that is scheduled on a given historical planning day has a probability of 10% to be replaced by another surgery on the same day. The idea behind this method is to test what happens when small changes occur in the schedule.

5. Computational results

In this section, we evaluate the performance of the TOP solution methods presented in Section 3. First, we consider the performance of each solution method on randomly generated instances (Section 5.1). Next, we consider how the solutions perform over a simulated longer time horizon (Section 5.2). The latter is particularly relevant as the solution methods optimize with respect to a small planning horizon and, therefore, it is worthwhile to know how the solution performance metrics, e.g., costs, over the optimization horizon correspond to the performance metrics over a longer time horizon.

We specifically test the following TOP solution methods: Greedy (GR), Column Generation with Greedy initial solution (CGG), Column Generation with Greedy trays (CG), Simulated Annealing (SA), Genetic Algorithm (GA), Integer Linear Program with Greedy initial solution (ILPG), and Integer Linear Program without Greedy initial solution (ILP) (see Section 3 for a description of all methods). Note that both CG and CGG first consider the LP-relaxation of (38)–(44). Here, both solution methods start with the trays generated by GR. Then, in order to find an integer solution, an ILP including only the generated trays is solved. In this step, CGG starts from the assignments of trays that is obtained in GR whereas CG optimizes the tray assignment from scratch. Finally, ILP does not use any trays from GR at all to start with.

Some of the algorithms require parameters to be set. We list the values of the parameters used in our experiments for each algorithm below.

- For GR, the tray assignment problem is solved to optimality.
- For GA, 10 minutes of computation time is allowed each time the tray assignment problem is solved in order to find the

Please cite this article as: T. Dollevoet et al., Solution methods for the tray optimization problem, European Journal of Operational Research (2018), https://doi.org/10.1016/j.ejor.2018.05.051

Table 3
Average computation time (in seconds) and standard deviation over 75 randomly generated instances based on the H1 and H2 datasets.

	GR	CGG	CG	SA	GA	ILPG	ILP
Average H1	165	2842	2853	1907	807	3179	3107
Std H1	636	1101	1111	1426	1240	1149	1227
Average H2	4	2649	2646	1217	460	2773	2870
Std H2	5	1132	1145	1239	891	1487	1430

costs of a solution. Most instances take only a few seconds to solve, but for some of the instances, this implies that the costs of a solution are not proven to be minimal. The population size of GA is set to 40.

- For SA, also 10 minutes of computation time is allowed whenever the tray assignment problem is solved. The number of trays is set to 30.
- For ILP and ILPG, the maximum number of instrument trays that can be used is set two higher than the number of trays in the solution obtained by GR. We have implemented three approaches to linearize the integer program for the TOP. Here, we report only the results that are obtained by using (16). Using the other linearizations, we obtained similar results. We did not include any symmetry breaking constraints and added only the valid inequalities (35). With other settings we obtained similar, but slightly worse results.
- In CG and CGG, a maximum of 50 minutes of computation time is reserved for generating new trays, while 10 minutes are reserved for solving the tray assignment problem. In every column generation iteration, the pricing problem is truncated after 1000 seconds.

All tests are performed with an Intel Xeon E5-1620 v3 processor and 16 gigabytes of RAM. We used Cplex 12.6.0 to solve the ILPs.

5.1. Performance evaluation on randomly generated instances

In order to compare the performance of the TOP solution methods presented in Section 3, we randomly generate 150 instances using the instance generator described in Section 4.2. Half of the instances (75) are generated based on the H1 data set, and half (75) are generated based on the H2 data set (see Section 4.1 for a description of the data sets). The instances vary with respect to the number of instruments (25, 50, 75, or 100), the number of surgeries (10, 15, 20, 30, or 50) and the number of days in the planning horizon (10, 20, or 40). For each instance and method, we allow a maximum computation time of 3600 seconds in order to make a fair comparison between solution methods.

Table 3 displays the average computation time and the standard deviation of the computation time over all instances (nota bene: computation time is capped at 3600 seconds). The greedy algorithm has the overall shortest computation time, followed by the genetic algorithm and then simulated annealing. The substantial difference in computation time for the greedy algorithm between the H1 and H2 based instances is caused by five instances in the H1 set for which the ILP solver used by the greedy algorithm has difficulty solving the underlying ILP. For all solution methods, the standard deviation of the computation time is substantial. In order to better analyze the relationship between the size of the instances and the computation time, we next study the computation times in more detail.

Fig. 1 displays a plot of the average computation time versus instance size (the number of variables in the problem formulation of Section 2) for GR, CGG, SA, GA, and ILPG. We have excluded ILP and CG from these plots, because they are outperformed by their counterparts ILPG and CGG. The plots are based on both H1 and H2 instances. GR has a reasonably low computation time for most instances. For ILPG and CGG, the computation time quickly increases with the instance size. For SA and GA, the computation time shows a more moderate growth pattern.

Table 4 displays the average objective function value over all instances for which a feasible solution can be found, as well as the percentage of instances for which a feasible solution was found. With the exception of the ILP solution method, all methods find a feasible solution for all instances (nota bene: as the ILP solution method could not find a feasible solution for all instances, the average for the ILP solution method is computed over less instances than the average for the other methods). If we consider only the solution methods that could solve all instances, SA achieves the highest average objective function value and ILPG the lowest for the H1 based instances (the relative difference is 1.7%). For the H2 based instances, GR achieves the highest average objective function value and, again, ILPG the lowest.

The effect of starting with a greedy solution (compare columns CGG and CG, and ILPG and ILP) is clearly visible. Although SA and GA also start with a greedy solution, sometimes the number of trays used in the greedy solution exceeds the limit on the number of trays used for SA and GA, and therefore, the greedy solution can only be partially used. Also, for two instances, the 10 minutes time limit imposed for solving the ILP subproblems in SA and GA is too short for computing optimal tray assignments. This negatively impacts the performance of SA and GA. The average objective function value of GR is only 0.8% (H1) and 7.7% (H2) worse than the average objective function value of ILPG.

Table 5 shows statistics on the relative performance, in terms of performance ranking on the objective function value, of the various TOP solution methods. The rank of each solution method is defined as 1 plus the number of solution methods ranked above it. Consequently, ranks range from 1 to 7 (1 being best, 7 being worst) and ties are possible. The table shows for each base data set the average rank, the number of instances for which a TOP solution method ranked first (possibly tied), and the average and maximum ratio of the objective function value and the objective function value of the best ranked solution method. ILPG has the lowest average rank (and thus the best performance) for both H1 and H2 based instances. When ILPG is not the best performing solution method, the ratio to the best performing solution method is at most 1.03 for H1 based instances and 1.19 for H2 based instances. On the other hand, a simple technique such as GR returns solutions that are, on average, only 2% (H1) or 12% (H2) worse than those computed by the best solution method.

Table 6 displays the number of solutions for which optimality has been proven. Interestingly, for the H1 based instances, whenever ILP or ILPG could prove that a particular solution is optimal, that solution is also achieved by all of the other methods. For H2 based instances this is not the case. There, ILPG finds provably optimal solutions in 23 out of 75 of the instances, but the heuristic methods find provably optimal solutions in at most 7 out of 75 instances.

Fig. 2 displays the optimality gap of ILPG versus instance size (again measured by the number of variables in the problem formulation of Section 2). The gap clearly increases with instance size and the relationship appears to be concave. For instances with a size up to 250 variables, ILPG could always solve the problem to optimality. For instances with a size exceeding 1000 variables, ILPG was never able to prove optimality.

To conclude, the computational results clearly show that ILPG performs best when computation time is not an issue. Using this approach, a significant amount of money can be saved when compared to the other solution methods. However, when computation time is limited, GR is preferred. The computation time needed for GR is significantly less compared to the other solutions methods, whereas the results are similar (except for ILPG and ILP).

Please cite this article as: T. Dollevoet et al., Solution methods for the tray optimization problem, European Journal of Operational Research (2018), https://doi.org/10.1016/j.ejor.2018.05.051

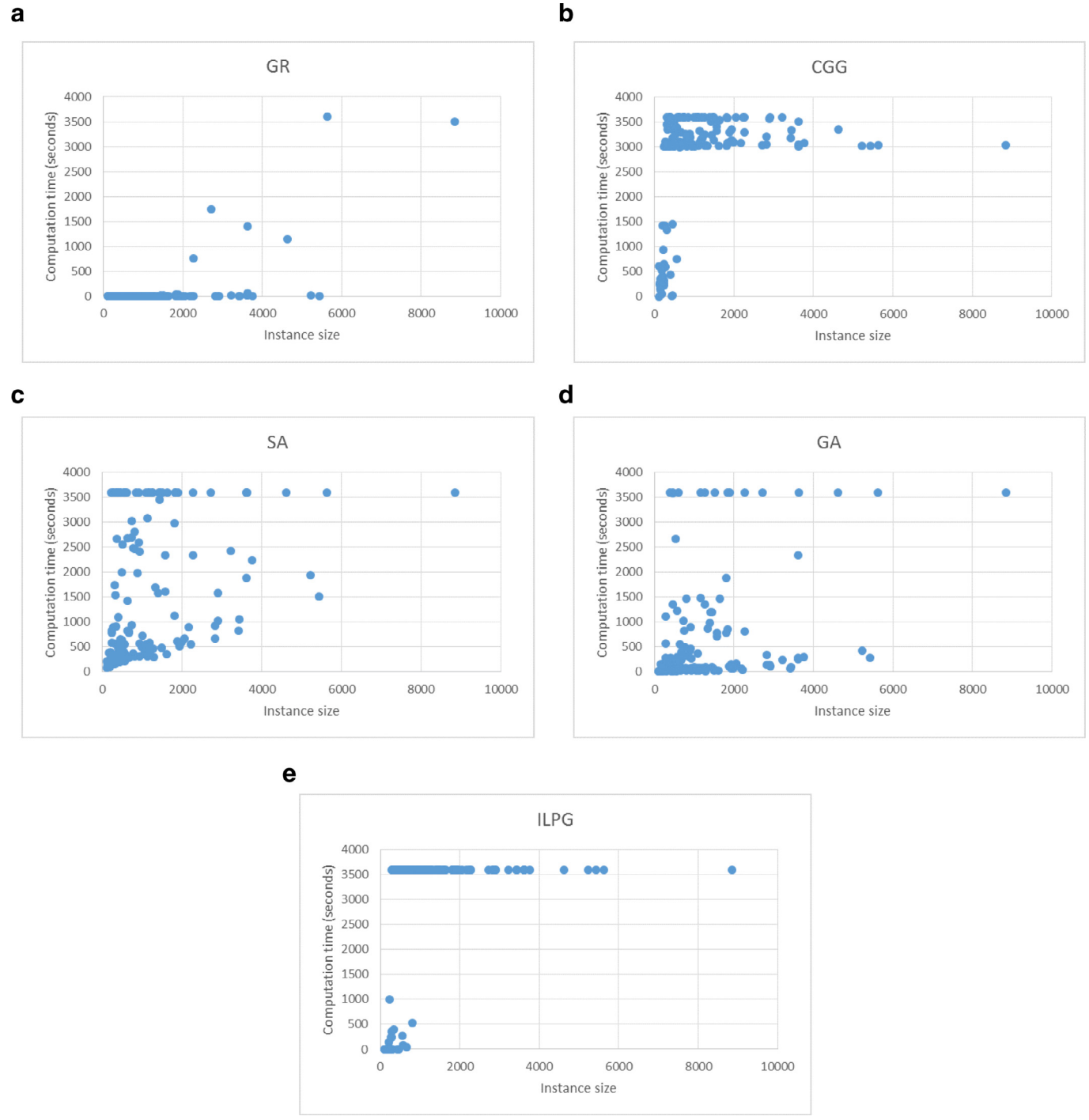

Fig. 1. Average computation time versus instance size (the number of variables in the problem formulation of Section 2) for (a) GR, (b) CGG, (c) SA, (d) GA, and (e) ILPG.

Table 4
Average objective function value (if a feasible solution is found), and percentage of instances for which a feasible solution is found, over 75 randomly generated instances.

	GR	CGG	CG	SA	GA	ILPG	ILP
Average H1	29139	29136	29258	29391	29239	28908	29886
% feasible H1	100.00	100.00	100.00	100.00	100.00	100.00	86.67
Average H2	10934	10883	10876	10854	10892	10156	19998
% feasible H2	100.00	100.00	100.00	100.00	100.00	100.00	98.67

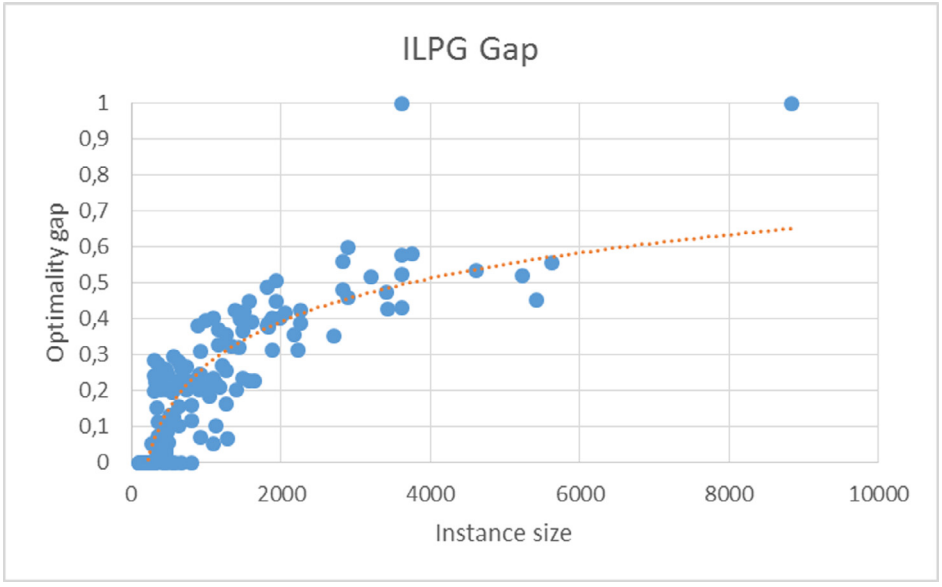

Fig. 2. Optimality gap versus instance size (the number of variables in the problem formulation of Section 2) for ILPG.

Table 5
Statistics on performance ranking over 75 randomly generated instances. Ranks are based on objective function value and range from 1 to 7 (1 is best, 7 is worst). In addition to average rank, the table displays the number of instances for which a TOP solution method ranked first (possibly tied), the average ratio to the best ranked solution, and the maximum ratio to the best ranked solution.

	GR	CGG	CG	SA	GA	ILPG	ILP
Average rank H1	2.45	2.31	2.57	2.29	2.35	1.16	4.32
# rank 1 H1	18	20	16	21	17	63	24
Avg ratio to best H1	1.02	1.02	1.02	1.02	1.02	1.00	1.10
Max ratio to best H1	1.19	1.19	1.19	1.19	1.19	1.03	2.33
Average rank H2	4.04	3.39	3.28	3.51	3.43	1.37	1.92
# rank 1 H2	6	7	7	7	10	54	46
Avg ratio to best H2	1.12	1.11	1.11	1.11	1.11	1.01	1.35
Max ratio to best H2	1.60	1.42	1.42	1.53	1.52	1.19	22.90

Table 6
Number of solutions for which optimality has been proven (out of 75 randomly generated instances).

	GR	CGG	CG	SA	GA	ILPG	ILP
H1	11	11	11	11	11	11	11
H2	5	6	6	5	7	23	23

5.2. Performance evaluation over a longer time horizon

In this subsection, we evaluate the performance of the solutions generated by the various TOP solution methods over a longer time horizon. We do this using the dynamic simulation procedure described in Section 4.3.

Table 7 displays cost and failure rate metrics of the solutions generated by the TOP solution methods over a longer time horizon. The upper part of the table shows the deviation (in percentage) of realized total costs from expected total costs. The expected total costs are calculated by multiplying the computed periodic costs of the static solution with the number of periods in the horizon. The lower part of the table shows the percentage of surgeries for which not all instruments are available in time (i.e., at the start of the surgery). Both averages and standard deviations are reported.

The performance of the TOP solution methods appears to be quite similar. This is likely because of the structure of the problem and the fact that the static solutions where relatively close (see Table 4). The static solutions, in general, appear to perform quite well in the dynamic setting. The long term costs are very close to what can be expected based on an extrapolation of the static solution. In fact, the costs are often slightly lower than the expectation, mainly because the percentage of surgeries for which not all instruments are available in time is likely to be higher than in the static solution (because the static solution is not necessarily feasible in the dynamic setting) and no costs are incurred for surgeries for which not all instruments are available in time. Of course, in reality there may be costs associated to this percentage of 'missed' surgeries, but as we do not have a measure of these costs we only report the percentage of missed surgeries.

Out of the four investigated dynamic simulation policies, the surgery schedule based on historical sampling is closest to the surgery schedule used in the static solution approach. Perturbed historical frequencies on the other hand is expected to generate a surgery schedule that is substantially more different. In general, the closer the long term surgery plan is to the surgery plan used in the static solution approach, the better the availability of instruments can be guaranteed.

6. Conclusions and recommendations

In this paper, we have developed several solution methods for the tray optimization problem (TOP). In the TOP, one simultaneously determines the composition of instrument trays, the assignment of these trays to surgeries, and the number of trays of a certain type to acquire. In particular, we have considered an exact ILP formulation, a row & column generation approach, a greedy heuristic, and some metaheuristics. In an extensive computational study, we have compared these solution methods on several instances that are inspired by practice. Furthermore, we have tested the performance of the solutions on a longer time horizon using simulation.

The greedy heuristic constructs several tray types and then solves the ILP model for the tray assignment problem optimally. Our computational results indicate that this method performs surprisingly well, especially considering the short computation times. The greedy heuristic gives solutions that are on average roughly 7% worse than the best solutions we found.

The exact ILP formulation, the row & column generation approach, and the metaheuristics can all make use of the solution obtained by the greedy heuristic. Among these solution methods, solving the ILP model gives the best performance. Next to that,

Please cite this article as: T. Dollevoet et al., Solution methods for the tray optimization problem, European Journal of Operational Research (2018), https://doi.org/10.1016/j.ejor.2018.05.051

Table 7
Long run performance metrics: average and standard deviation (in brackets).

Deviation of total costs from expectation based on static solution					
		Hist. freq.	Pert. hist. freq.	Hist. sampling	Pert. hist. sampling
H1	GR	−1.68% (1.81%)	−1.69% (1.81%)	0.00% (0.05%)	−1.04% (0.76%)
	CGG	−1.71% (1.83%)	−1.72% (1.83%)	−0.01% (0.06%)	−1.07% (0.78%)
	SA	−1.65% (1.76%)	−1.66% (1.76%)	0.00% (0.06%)	−0.97% (0.73%)
	GA	−1.83% (1.89%)	−1.84% (1.89%)	0.00% (0.05%)	−1.03% (0.77%)
	ILPG	−1.69% (1.79%)	−1.70% (1.79%)	0.00% (0.05%)	−1.22% (0.79%)
H2	GR	−3.60% (4.25%)	−3.61% (4.23%)	1.71% (2.56%)	1.18% (2.59%)
	CGG	−4.16% (5.31%)	−4.17% (5.32%)	1.20% (3.87%)	0.42% (3.99%)
	SA	−3.81% (4.40%)	−3.82% (4.38%)	1.74% (2.57%)	1.40% (2.42%)
	GA	−3.58% (4.27%)	−3.59% (4.26%)	1.68% (2.52%)	0.96% (2.49%)
	ILPG	−4.17% (4.64%)	−4.18% (4.64%)	1.42% (3.25%)	−0.04% (3.2%)
Percentage of surgeries for which instruments are not available in time					
		Hist. freq.	Pert. hist. freq.	Hist. sampling	Pert. hist. sampling
H1	GR	6.12% (4.00%)	6.13% (4.00%)	0.00% (0.00%)	1.73% (1.03%)
	CGG	6.12% (4.01%)	6.14% (4.01%)	0.00% (0.00%)	1.68% (0.99%)
	SA	6.04% (3.84%)	6.06% (3.84%)	0.00% (0.00%)	1.79% (1.10%)
	GA	6.35% (4.28%)	6.37% (4.29%)	0.00% (0.00%)	1.86% (1.20%)
	ILPG	6.17% (4.00%)	6.17% (4.00%)	0.00% (0.00%)	1.61% (0.96%)
H2	GR	7.97% (4.94%)	8.02% (4.94%)	0.00% (0.00%)	1.61% (1.15%)
	CGG	8.03% (5.19%)	8.08% (5.20%)	0.00% (0.00%)	1.52% (1.01%)
	SA	8.14% (5.16%)	8.19% (5.16%)	0.00% (0.00%)	2.21% (1.53%)
	GA	7.91% (4.95%)	7.96% (4.95%)	0.00% (0.00%)	1.74% (1.33%)
	ILPG	7.98% (5.52%)	8.02% (5.52%)	0.00% (0.00%)	1.42% (0.92%)

it is the only exact method we considered, which means that is able to prove optimality for some of the instances. However, this does not hold for the larger instances, where the improvement in comparison to the greedy heuristic is small.

Summarizing the above, the greedy heuristic is a good method if solutions have to be found quickly. If more time is available, and if the instances are small- to medium-sized, solving the ILP model can improve this greedy solution considerably. For the other methods, the improvements in comparison to the greedy approach are limited.

We have also tested the solutions in a dynamic setting. Here, we first obtained the tray composition and the number of trays to be acquired by solving a static instance. Then, we generated a surgery schedule for a longer time horizon and evaluated the sterilization costs and the number of times an instrument was not available. The results show that all solutions of the static problem perform well in the dynamic setting. First, the average costs are slightly lower than in the static simulation. Second, only for a small percentage of the surgeries an instrument is missing.

For our experiments, we have generated the instances based on data of two different hospitals and considered instances of various sizes. Nonetheless, some simplifying assumptions had to be made. For example, we have not considered the weight and size restrictions of the trays due to a lack of reliable data, but have assumed that every combination of a fixed number of instruments fits in a tray. Similarly, we have omitted instrument depreciation costs. Note, though, that our algorithms are capable to deal with these parameters if they would be available.

To summarize, we have presented and evaluated multiple solution methods for the TOP and are the first to provide TOP solution methods that can actually be used to solve instances of realistic size. The most promising method that we presented is, perhaps, the greedy method as it is highly scalable. Moreover, our results indicate that the outcomes of the greedy method are, on average, very close to the other methods investigated. We hope our findings may contribute to health care management practice by making it possible to achieve substantial cost savings on inventory of medical instruments.

Acknowledgments

We would like to thank the following BSc and MSc students for their contribution to this paper: Elske Florijn, Suzanne de Hoog, Bas Kamphorst and Sven van der Kooij.

This research is supported by the Dutch Technology Foundation STW, applied science division of NWO and the Technology Program of the Ministry of Economic Affairs.

Appendix A. Column & row generation: proofs and derivations

In this section, we derive an optimality criterion for the row & column generation approach presented in Section 3.2.1. Our analysis is based on a Lagrangian relaxation, in a similar fashion as in Muter et al. (2012). Consider therefore the linear relaxation of (38)–(44). In any optimal solution, it holds that $Z_k = \frac{1}{M} N_k$. We can therefore replace Z_k by $\frac{1}{M} N_k$ in the objective and discard constraints (41) and (44). The costs $c_4 \frac{1}{M} N_k$ can be absorbed in the term $c_k^1 N_k$. From now on, we assume that this has been done. For a given subset $\bar{K} \subseteq K$ of tray types, we then denote this linear relaxation and its objective by $z_{LP}(\bar{K})$. Now relax constraints (39) with Lagrangian multipliers $\lambda \geq 0$. We then get the following Lagrangian subproblem.

$$z(\lambda; \bar{K}) = \min \sum_{k \in \bar{K}} \left(c_k^1 N_k + c_k^2 \sum_{j \in J} f_j Y_{jk} \right) + \sum_{i \in I} \sum_{j \in J} \lambda_{ij} \left(d_{ij} - \sum_{k \in \bar{K}} x_{ik} Y_{jk} \right)$$

$$= \min \sum_{i \in I} \sum_{j \in J} \lambda_{ij} d_{ij} + \sum_{k \in \bar{K}} \left(c_k^1 N_k + \sum_{j \in J} \left(c_k^2 f_j - \sum_{i \in I} \lambda_{ij} x_{ik} \right) Y_{jk} \right)$$

such that

$$N_k \geq \sum_{j \in J} s_{jt} Y_{jk}, \quad \forall k \in \bar{K}, t \in T,$$

$$Y_{jk} \geq 0, \quad \forall k \in \bar{K}, j \in J,$$

$$N_k \geq 0, \quad \forall k \in \bar{K}.$$

We define a Lagrangian dual problem $z_{LR}(\bar{K}) = \max\{z(\lambda; \bar{K}) : \lambda \geq 0\}$. In order to solve the Lagrangian subproblem for given λ, we

can consider the trays $k \in \bar{K}$ individually, because all constraints that couple different trays have been dualized into the objective function. For given $k \in \bar{K}$, we have to solve

$$\bar{z}_k(\lambda) = \min c_k^1 N_k + \sum_{j \in J} \left(c_k^2 f_j - \sum_{i \in I} \lambda_{ij} x_{ik} \right) Y_{jk}$$

such that

$$N_k \geq \sum_{j \in J} s_{jt} Y_{jk}, \quad \forall t \in T, \tag{A.1}$$

$$Y_{jk} \geq 0, \quad \forall j \in J, \tag{A.2}$$

$$N_k \geq 0. \tag{A.3}$$

Substituting $N_k = 0$ and $Y_{jk} = 0$ for all j yields a feasible solution with an objective value of zero to the Lagrangian supbproblem for $k \in \bar{K}$, and we therefore conclude that $\bar{z}_k(\lambda) \leq 0$. Furthermore, the feasible region of the Lagrangian subproblem is a cone: If (N_k, Y_{jk}) is feasible, then (cN_k, cY_{jk}) is feasible, too, for all $c \geq 0$. We can therefore restrict attention to the case $N_k = 1$. Then, the problem reduces to

$$z_k(\lambda) = c_k^1 + \min \sum_{j \in J} \left(c_k^2 f_j - \sum_{i \in I} \lambda_{ij} x_{ik} \right) Y_{jk}$$

such that

$$\sum_{j \in J} s_{jt} Y_{jk} \leq 1, \quad \forall t \in T,$$

$$Y_{jk} \geq 0, \quad \forall j \in J.$$

Whenever this value $z_k(\lambda) < 0$, then the problem $\bar{z}_k(\lambda)$ is unbounded. If $z_k(\lambda) \geq 0$, $N_k = 0$ and $Y_{jk} = 0$ is an optimal solution. We are now ready to derive the reduced cost criterion.

Theorem. Let λ^* be the optimal dual multipliers for a given set $\bar{K} \subseteq K$ of trays and define $z_{LP}(\bar{K})$ as the objective value for the corresponding linear problem. If

$$z_k(\lambda^*) \geq 0$$

for all $k \in K \setminus \bar{K}$, then $z_{LP}(\bar{K})$ is also the optimal solution if all trays are considered. As a consequence, we can interpret $z_k(\lambda^*)$ as the reduced cost of tray $k \in K$.

Proof. Recall that $z_{LP}(\bar{K})$ is the optimal solution of the LP-relaxation of (38)–(44) when only the subset $\bar{K}$ of trays is considered. Assume that we have solved this linear program and obtained dual multipliers λ^* and μ^*. The variables λ^* are also optimal Lagrangian multipliers, if we consider the same subset of trays in the Lagrangian relaxation (Fisher, 2004). It holds that $z_{LP}(\bar{K}) = z(\lambda^*; \bar{K})$. As $\bar{z}_k(\lambda^*) \geq 0$ for all $k \in K \setminus \bar{K}$, the objective of the Lagrangian subproblem does not change if we include any of the trays $k \in K \setminus \bar{K}$. Hence, $z(\lambda^*; K) = z(\lambda^*; \bar{K})$. We then obtain

$$z_{LP}(K) \leq z_{LP}(\bar{K}) = z(\lambda^*; \bar{K}) = z(\lambda^*; K) \leq z_{LR}(K) = z_{LP}(K).$$

It follows that all inequalities in the formula above are actually equalities. Hence, $z(\lambda^*, \bar{K}) = z_{LR}(K)$. This shows that the columns in $\bar{K}$ solve the Lagrangian relaxation optimally. Furthermore, this shows that $z_{LP}(\bar{K})$ equals $z_{LP}(K)$. Thus, the linear problem is solved to optimality, too. This suggests to define $z_k(\lambda^*)$ as the reduced costs of tray $k \in K$. □

In order to linearize the pricing problem, we first introduce D_i binary decision variables $\bar{X}_{il}$ for every instrument type i. Here, D_i is the upper bound on the number of instruments of type i in a tray that is defined in Section 3.1. We then substitute $X_i = \sum_{l=1}^{D_i} \bar{X}_{il}$. It holds that

$$X_i Y_j = \left(\sum_{l=1}^{D_i} \bar{X}_{il} \right) Y_j = \sum_{l=1}^{D_i} \bar{X}_{il} Y_j = \sum_{l=1}^{D_i} Q_{ijl}.$$

We obtain the following Integer Linear Program for the pricing problem.

$$z_{PP}(\lambda) = \min \frac{c_4}{M} + c_1 + \sum_{i \in I} \sum_{l=1}^{D_i} a_i \bar{X}_{il} + \sum_{j \in J} \left((c_2 + c_3) f_j Y_j + \sum_{i \in I} \sum_{l=1}^{D_i} (b_i f_j - \lambda_{ij}) Q_{ijl} \right)$$

such that

$$\sum_{i \in I} p_i^n \sum_{l=1}^{D_i} \bar{X}_{il} \leq r^n, \quad \forall n \in N,$$

$$\sum_{j \in J} s_{jt} Y_j \leq 1, \quad \forall t \in T,$$

$$Q_{ijl} \leq \bar{X}_{il}, \quad \forall i \in I, j \in J, l \in \{1, \ldots, D_i\},$$

$$Q_{ijl} \leq Y_j, \quad \forall i \in I, j \in J, l \in \{1, \ldots, D_i\},$$

$$Q_{ijl} \geq Y_j + \bar{X}_{il} - 1, \quad \forall i \in I, j \in J, l \in \{1, \ldots, D_i\},$$

$$\bar{X}_{il} \in \{0, 1\}, \quad \forall i \in I, l \in \{1, \ldots, D_i\},$$

$$Y_j \in [0, 1], \quad \forall j \in J,$$

$$Q_{ijl} \in [0, 1], \quad \forall i \in I, j \in J, l \in \{1, \ldots, D_i\}.$$

We obtained a MILP formulation for the pricing problem that can be solved by a standard solver. Note that we have restricted Y_j to the bounded interval [0,1], which is allowed if $\Sigma_{t \in T} s_{jt} > 0$. If this is not the case, surgery j is not scheduled at all and can be removed from the problem.

Appendix B. Tray assignment problem is NP-hard

In this section, we will show that the tray assignment problem (TAP), as defined by (38)–(44) in Section 3.2.1, is NP-hard. We first show that the decision version of TAP is in NP. Then, in order to show NP-completeness, we reduce the (decision version of the) set-covering problem to this decision version.

In the decision version of the TAP, an additional value q is given. The decision version of TAP asks whether a feasible solution exists with an objective function value of at most q. For a given solution to the TAP, both its feasibility and its objective function value can be determined in polynomial time. Hence, the TAP is in NP.

Let now a set-covering instance be given. Such an instance is specified by a so-called universe $\mathcal{U} = \{1, \ldots, n\}$, a set $\mathcal{S}$ containing subsets of $\mathcal{U}$, and a value q'. A *cover* is defined as a subset $\mathcal{C} \subseteq \mathcal{S}$,whose union is $\mathcal{U}$. The question is whether a cover $\mathcal{C}$ of cardinality at most q' exists. For a given set-covering instance, we define an instance of the TAP by $I = \mathcal{U}$, $J = \{1\}$, $\bar{K} = \mathcal{S}$, $T = \{1\}$, and $q = q'$. (Thus, we consider only one surgery and the planning horizon consists of only one day.) We set $s_{11} = 1$, $d_{i1} = 1$ for all $i \in \mathcal{U}$, and define the cost coefficients as $c_1 = 1$, $c_2 = c_3 = c_4 = 0$, and $a_i = b_i = 0$ for all $i \in \mathcal{U}$. It follows that $c_k^1 = 1$ and $c_k^2 = 0$. Finally, for $i \in \mathcal{U}$ and $k \in \mathcal{S}$, we define $x_{ik} = 1$ if $i \in k$. The ILP for the tray assignment problem can then be written as

$$\min \sum_{k \in \mathcal{S}} N_k$$

such that

$$\sum_{k \in \mathcal{S}} Y_{1k} x_{ik} \geq 1, \quad \forall i \in \mathcal{U},$$
$$Y_{1k} \leq N_k, \quad \forall k \in \mathcal{S},$$
$$N_k \leq M Z_k, \quad \forall k \in \mathcal{S},$$
$$N_k \in \mathbf{N}, \quad \forall k \in \mathcal{S},$$
$$Y_{1k} \in \{0, 1\}, \quad \forall k \in \mathcal{S},$$
$$Z_k \in \{0, 1\}, \quad \forall k \in \mathcal{S}.$$

It follows easily from the above that $N_k = Y_{1k}$ for all $k \in \mathcal{S}$ in any optimal solution. Furthermore, setting $Z_k = 1$ for all $k \in \mathcal{S}$ changes neither the feasibility nor the objective function value of a solution. We can thus restrict the range of N_k to $\{0, 1\}$ and remove the Z_k variables for all $k \in \mathcal{S}$. This yields the following ILP:

$$\min \sum_{k \in \mathcal{S}} N_k$$

such that

$$\sum_{k \in \mathcal{S}} N_k x_{ik} \geq 1, \quad \forall i \in \mathcal{U},$$
$$N_k \in \{0, 1\}, \quad \forall k \in \mathcal{S}.$$

This is precisely the ILP formulation of the set-covering problem (Wolsey, 1998): A set $k \in \mathcal{S}$ is included in the cover if and only if $N_k = 1$. Given that $x_{ik} = 1$ if and only if $i \in k$ for all $i \in \mathcal{U}$ and $k \in \mathcal{S}$ ensures that the union of all selected subsets contains every $i \in \mathcal{U}$. It follows that a cover of cardinality at most q' for this set-covering instance exists if and only if this instance of the TAP has a solution with objective at most q. Given that the decision version of the set-covering problem is NP-complete (Karp, 1972), it follows that the decision version of the tray assignment problem is NP-complete as well. Therefore, we conclude that the TAP is NP-hard.

References

Fisher, M. L. (2004). The Lagrangian relaxation method for solving integer programming problems. *Management Science, 50*(suppl12), 1861–1871.
Hagist, C., & Kotlikoff, L. (2005). Who's going broke? Comparing growth in healthcare costs in ten OECD countries. NBER working paper no. 11833.
Harjunkoski, I., Pörn, R., Westerlund, T., & Skrifvars, H. (1997). Different strategies for solving bilinear integer non-linear programming problems with convex transformations. *Computers & Chemical Engineering, 21*, S487–S492. Supplement to Computers and Chemical Engineering.
Karp, R. M. (1972). Reducibility among combinatorial problems. In R. E. Miller, J. W. Thatcher, & J. D. Bohlinger (Eds.), *Proceedings of the symposium on the complexity of computer computations, March 20–22* (pp. 85–103). Boston, MA: Springer US. The IBM Thomas J. Watson Research Center, Yorktown Heights, New York, and Sponsored by The Office of Naval Research, Mathematics Program, IBM World Trade Corporation, and The IBM Research Mathematical Sciences Department.
Kirkpatrick, S., Gelatt, C. D., & Vecchi, M. P. (1983). Optimization by simulated annealing. *Science, 220*, 671–680.
Muter, I., Birbil, S. İ., & Bülbül, K. (2013). Simultaneous column-and-row generation for large-scale linear programs with column-dependent-rows. *Mathematical Programming, 142*(1–2), 1–36.
Muter, I., Birbil, S. İ., & Bülbül, K. (2018). Benders decomposition and column-and-row generation for solving large-scale linear programs with column-dependent-rows. *European Journal of Operational Research, 264*(1), 29–45.
Muter, I., Birbil, S. İ., Bülbül, K., & Şahin, G. (2012). A note on "A LP-based heuristic for a time-constrained routing problem". *European Journal of Operational Research, 221*(2), 306–307.
OECD (2014). OECD health data 2014. http://www.oecd.org/els/health-systems/oecd-health-statistics-2014-frequently-requested-data.htm.
Reymondon, F., Pellet, B., & Marcon, E. (2008). Optimization of hospital sterilization costs proposing new grouping choices of medical devices into packages. *International Journal of Production Economics, 112*(1), 326–335.
Sherali, H. D., & Smith, J. C. (2001). Improving discrete model representations via symmetry considerations. *Management Science, 47*(10), 1396–1407.
van de Klundert, J., Muls, P., & Schadd, M. (2008). Optimizing sterilization logistics in hospitals. *Health Care Management Science, 11*(1), 23–33.
van der Kooij, S. (2015). Creating a benchmark for the tray optimization problem. Research paper.
Wolsey, L. A. (1998). Integer programming. *Wiley-Interscience series in discrete mathematics and optimization*. John Wiley & Sons.
Worthington, D. (1991). Hospital waiting list management models. *The Journal of the Operational Research Society, 42*(10), 833–843.

Please cite this article as: T. Dollevoet et al., Solution methods for the tray optimization problem, European Journal of Operational Research (2018), https://doi.org/10.1016/j.ejor.2018.05.051