A new sequential extraction heuristic for optimising the delivery of cancer radiation treatment using multileaf collimators

Davaasteren Baatar, Natashia Boland, Robert Johnston
Department of Mathematics and Statistics, The University of Melbourne, Victoria 3010, Australia,
{d.baatar@ms.unimelb.edu.au, natashia@unimelb.edu.au, , rj@unimelb.edu.au}

Horst W. Hamacher
Department of Mathematics, University of Kaiserslautern, 67653 Kaiserslautern, Germany,
hamacher@mathematik.uni-kl.de

Finding a delivery plan for cancer radiation treatment using multileaf collimators operating in "step-and-shoot mode" can be formulated mathematically as a problem of decomposing an integer matrix into a weighted sum of binary matrices having the consecutive-ones property – and sometimes other properties related to the collimator technology. The efficiency of the delivery plan is measured by both the sum of the weights in the decomposition, known as the *total beam-on time*, and the number of different binary matrices appearing in it, referred to as the *cardinality*, the latter being closely related to the set-up time of the treatment. In practice, the total beam-on time is usually restricted to its minimum possible value, (which is easy to find), and a decomposition that minimises cardinality (subject to this restriction) is sought.

This decomposition problem is known to be NP-hard and the best available exact solution methods cannot solve, in reasonable time, problems with dimensions large enough to be of use in actual medical applications. In this paper we propose a new heuristic. To ensure that the heuristic is computationally efficient, we make use of exact bounds that apply to the decomposition, and prove that these bounds can be computed efficiently. We demonstrate that the heuristic performs very well numerically against the best previously published heuristic, (that of Kalinowski), reducing the average gap between the cardinality of the solution found and the optimal value by 37% on the largest problems tested (for which optimal solutions could be found). Importantly, this new heuristic performs well on those instances that are problematical for Kalinowski's heuristic. A "best-of" algorithm, combining heuristics, produces a decomposition with cardinality within one of the optimal in about 98.7% of instances tested (for which an optimal solution is available). It reduces the cardinality of solutions produced by about 5% on average. On instances for which optimal solutions can be found, it more than halves the optimality gap and finds an optimal solution in about 28% more cases than Kalinowski's heuristic.

Key words: matrix decomposition, multileaf collimator sequencing, cancer radiation therapy
History:

1. Introduction and Preliminary Results

In this paper we are interested in the problem of decomposing a given $M \times N$ matrix G of non-negative integers into a weighted sum of binary matrices, i.e. we seek $\mathcal{K} \subseteq \mathcal{X}$ and positive integers $\alpha_1, \ldots, \alpha_K$ such that

$$G = \sum_{k=1}^{K} \alpha_k Y^k \tag{1}$$

where $K = |\mathcal{K}|$,

$$\mathcal{K} = \{Y^k \ : \ k = 1, \ldots, K\}$$

and $\mathcal{X}$ is a set of all binary $M \times N$ matrices having the consecutive-ones property, (all ones in any row of the matrix must be consecutive), and perhaps some additional properties. This problem is of interest largely because of its application to the treatment of cancer with intensity-modulated radiation therapy, (IMRT), when the treatment is delivered using a multileaf collimator operating in "step-and-shoot" mode. In this context, the matrix G is called the *intensity matrix*, and represents a part of the treatment plan, in particular, it is the modulated radiation field that should be delivered to the patient from a particular angle (see Hamacher and Küfer (2002) for a recent discussion of IMRT treatment planning).

The multileaf collimator delivers a rectangular field of radiation, that can be discretized into bixels. Each entry in the matrix G represents the extent of radiation that should be delivered in the corresponding bixel. The radiation source in the collimator, however, delivers only a uniform field of radiation. To modulate the field, and apply different radiation extents to different bixels, the collimator has lead "leaves", that are aligned with the rows of the corresponding matrix on both the left and right sides, and that can slide across the field, to block the radiation in the bixel they cover. By positioning the leaves across the field, the radiation can be shaped. As radiation is "additive" in the body, any desired treatment can be delivered by shaping the field, applying radiation for a period of time, re-shaping the field, again applying radiation for perhaps a different length of time, and so on.

The shaped field can be represented as a binary matrix, in which the zeroes represent the bixels covered by the collimator leaves, and the ones represent the bixels that are exposed to radiation. Such a matrix is here called a "shape matrix". The period of time that radiation is applied to the shaped field is represented by the weighting of the corresponding matrix. We can thus view the decomposition in (1) as representing a delivery plan in which the field is shaped K times, the kth shape formed by the collimator leaves is indicated by Y^k, and radiation is applied to that shape for α_k units of time. $\mathcal{X}$ represents the set of all possible shape matrices (shaped fields) that could be delivered by the collimator.

From the above discussion, it is readily seen that shape matrices must have the consecutive-ones property: the consecutive ones in each row represent the bixels exposed between the corresponding left and right leaf pair of the multileaf collimator. Shape matrices are often required to have other properties that arise from the multileaf collimator technology. Most commonly, they are required to have the *interleaf constraint*, in which the left leaf in one row cannot cover any columns covered by the right leaf in any adjacent rows, and vice versa. But other constraints, such as arise from *tongue-and-groove* effects, may also apply (see, for example, Deng et al. (2001) for a discussion of such constraints). Most of the results in this paper would extend to other constraints, but in our models, algorithms and test problems, we require only the interleaf constraints to apply.

Two measures of the matrix decomposition in (1) are of interest in IMRT: the *total beam-on time*, B, given by $B = \sum_{k=1}^{K} \alpha_k$, and the *cardinality*, given by K. Both contribute to the total time required to deliver the treatment, which is often modelled as the beam-on time plus the time required to set up each new field shape. The recent survey paper of Ehrgott et al. (2007), which provides an excellent description of treatment delivery problems and associated solution techniques, discusses treatment time in more detail. This paper also confirms the observation that minimizing the beam-on time is usually perceived by practitioners to be of primary importance. Fortunately, the problem of minimizing beam-on time is known to be solvable in polynomial time, both with and without interleaf constraints Baatar et al. (2005); Boland et al. (2003); Kalinowski (2005a); Kamath et al. (2003). For a given intensity matrix G we write $B^*(G)$ to denote the total minimum beam-on time, and accept that this can be efficiently calculated.

Minimizing K, the cardinality, is a much harder problem. Methods that minimize beam-on time rarely give solutions with cardinality close to the minimum possible (see, for example, Boland et al. (2003); Kalinowski (2005a); Kamath et al. (2003)). Even if shape matrices are only required to have the consecutive-ones property, and only consist of a single row, it has been proved that minimising the cardinality of the decomposition is still strongly NP-hard Baatar et al. (2005).

Nevertheless, in the last decade, many (largely heuristic) decomposition algorithms have been proposed Ahuja and Hamacher (2004); Baatar et al. (2007, 2005); Baatar and Hamacher (2003); Boland et al. (2003); Kalinowski (2005a); Kamath et al. (2003); Langer et al. (2001); Siochi (1999); Xia and Verhey (1999). Most seek to minimize either the beam-on time or the cardinality, at the expense of the other objective, or to find a decomposition which offers the best compromise between the two. Several of these Baatar et al. (2005); Baatar (2005); Kalinowski (2005b); Langer et al. (2001) seek lexicographical minimization of the two objectives, in which beam-on time is minimized first. Since minimum beam-on time can be calculated efficiently, this problem, which we refer to as *lex_min*(B, K), can be stated as follows:

$$\begin{array}{lll}
\min & K & \\
\text{s.t.} & \displaystyle\sum_{k=1}^{K} \alpha_k Y^k = G & \\
 & \displaystyle\sum_{k=1}^{K} \alpha_k = B^*(G) & \\
 & \alpha_k \geq 0, \quad \text{integer}, & \forall k = 1, \ldots, K \\
 & Y^k \in \mathcal{X}, & \forall k = 1, \ldots, K.
\end{array}$$

It is this problem that is the focus of our paper.

It is not hard to prove from existing results that *lex_min*(B, K) is strongly NP-hard.

Proposition 1. *lex_min*(B, K) *is strongly NP-hard.*

Proof. Proposition 3 in Baatar and Hamacher (2003) states that if the given matrix G consists of a single row, there must exist a decomposition which simultaneously minimizes both total beam-on time B and cardinality K. Now if we solve *lex_min*(B, K) for a single-row matrix, we must find such a solution. Thus solving *lex_min*(B, K) also solves the minimum cardinality problem. But Baatar et al. prove in Baatar et al. (2005) that even for single-row matrices, minimizing cardinality is strongly NP-hard. □

Unfortunately it is not always possible to minimize both beam-on time and cardinality simultaneously; for matrices with more than one row, an optimal solution to *lex_min*(B, K) might not minimize K, as the following example demonstrates.

Example 1. Consider a matrix G and the following decomposition of it

$$\begin{pmatrix} 5 & 10 & 6 \\ 4 & 1 & 1 \\ 7 & 0 & 0 \end{pmatrix} = \begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \end{pmatrix} + \begin{pmatrix} 1 & 1 & 1 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \end{pmatrix} + 3\begin{pmatrix} 1 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} + 5\begin{pmatrix} 0 & 1 & 1 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{pmatrix}.$$

This decomposition has beam-on time $1+1+3+5=10$, which is easily seen to be minimal, since $G_{12}=10$. Thus $B^*(G)=10$. Moreover, it is an optimal solution to $lex\_min(B,K)$. This can be observed from the second row of the matrix G. The second row has maximum entry 4 which is less than $B^*(G)$, so there must be at least one matrix in the decomposition with no ones in the second row. Furthermore, the second row has two different integers in it, so there must be at least two different matrices in the decomposition which do have ones in the second row. So the cardinality must be at least three. Now the solution above has cardinality four, so if it is not optimal for $lex\_min(B,K)$, there must be a solution with beam-on time 10 and cardinality of three. Closer inspection of the second row reveals that, in this case, only two sets of weights are possible in the decomposition: either $\alpha_1=1$, $\alpha_2=3$, and $\alpha_3=6$, or $\alpha_1=1$, $\alpha_2=4$, and $\alpha_3=5$. The first case is impossible, as $G_{11}=5$ cannot be written as the sum of any combination of 1, 3 and 6. The second case is also impossible, as $G_{31}=7$ cannot be written as the sum of any combination of 1, 4 and 5. Thus, the decomposition given, with beam-on time 10 and cardinality 4 solves $lex\_min(B,K)$. However, the minimal cardinality over any decomposition of G is in fact $K=3$, which is clear from the following decomposition, which has a beam-on time of 11:

$$\begin{pmatrix} 5 & 10 & 6 \\ 4 & 1 & 1 \\ 7 & 0 & 0 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \end{pmatrix} + 4\begin{pmatrix} 1 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} + 6\begin{pmatrix} 0 & 1 & 1 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{pmatrix}.$$

Although it is in general not possible to minimize B and K simultaneously, it is reported (see, for example, Baatar et al. (2005); Baatar (2005); Kalinowski (2005b); Langer et al. (2001); Boland et al. (2007)) that lexicographical minimization of the two objectives, i.e. solving $lex\_min(B,K)$, typically yields superior solutions. Thus we choose to focus on the $lex\_min(B,K)$ problem.

Whilst there are exact methods available for $lex\_min(B,K)$ in the case *without* interleaf constraints, (see, for example, Kalinowski (2005b); Baatar et al. (2007)), it is not obvious how to extend them to cope with interleaf constraints. Furthermore, these methods do not perform well on problems with dimensions large enough to be of use in the medical application. To be specific, with current technology, they are not able to solve to optimality problems with, say 20×20 matrices having maximum entry of 20 or more.

Previously reported heuristics Baatar et al. (2005); Baatar and Hamacher (2003); Kalinowski (2005a); Langer et al. (2001) for $lex\_min(B, K)$ typically produce solutions with smaller cardinality than those found with alternative objectives (and of course have minimal beam-on time). All of these successful heuristics use some form of greedy approach, in which shape matrices are sequentially "extracted" from G, each with the maximum possible beam-on time coefficient, while still maintaining the property that the decomposition will have minimum beam-on time.

In this paper we introduce a new heuristic algorithm for $lex\_min(B, K)$ that is similar in spirit to the sequential extraction approaches. However instead of extracting a *single* shape matrix from G with the maximum coefficient possible, we simultaneously extract a *set* of shape matrices at each step of our approach. Our goal is to extract, simultaneously, as many shape matrices as possible that can receive the maximum coefficient. Our extraction step involves solving a difficult optimization problem, that we are able to solve efficiently by using a formulation based on ideas presented in Baatar et al. (2005), and by further strengthening the formulation with the use of bounds that we can derive on some of the variables. We are able to prove that we can calculate the best possible such bounds in polynomial time: in Baatar (2005) this result is proved with a very lengthy argument employing a recursive combinatorial procedure; here we simply show that it can be formulated as a linear programme having the integrality property. Our computational results show that the extraction step is not only computationally efficient, but scales reasonably well as problem dimensions grow; it remains practical even for larger problem instances.

We compare the performance of our heuristic to that of the best previously published heuristic, Kalinowski (2005a), and show that our approach nicely complements it; our heuristic performs well on instances that are problematic for that of Kalinowski, and vice versa. As a consequence, our heuristic, used in tandem with Kalinowski's heuristic in a "best-of" algorithm, demonstrates significant improvements.

The paper is structured as follows. In the next section, we give an overview of our algorithm, which we call the *sequential maximizing extraction heuristic.* In Section 3, we show how we tackle the central extraction step, and give results that allow us to strengthen the formulation of the extraction optimization problem. We also demonstrate the computational effectiveness of this strengthening. Finally, in Section 4, we present the results of numerical experiments with our heuristic, analysing its performance and comparing it with optimal solutions from the exact constraint programming method of Baatar et al. (2007) on smaller problems, and with the heuristic of Kalinowski (2005a) on larger problems.

2. The Sequential Maximizing Extraction Heuristic

It is essential to note throughout this section that we always assume that for any non-negative integer matrix G, its minimal beam-on time $B^*(G)$ and the corresponding shape matrix decomposition can be easily calculated. As we discussed in the previous section, this is indeed the case for decomposition into consecutive-ones matrices both with and without the interleaf constraints.

Our algorithm begins by calculating $B^*(G)$, and by finding a value $c_{max}(G)$, where c_{max} is a function that for any integer matrix G returns $c_{max}(G)$, an upper bound on any coefficient appearing in any decomposition of G that achieves its minimal beam-on time, $B^*(G)$. We initialise the parameter $c_1 = c_{max}(G)$ and set $G^1 = G$.

Starting with $q = 1$, at iteration q of the algorithm we seek a matrix E and a matrix F so that $c_q E + F = G^q$. At all iterations, we set c_q equal to $c_{max}(G^q)$. The idea is that we will try to find a (minimal beam-on time) decomposition of G^q so that as many shape matrices as possible get the largest possible coefficient, c_q. The sum of these shape matrices will be stored in E. Once we have found E and F, we update $G^{q+1} = G^q - c_q E (= F)$, i.e. we "extract" E (with multiplicity c_q).

We expect that if we can choose a decomposition that extracts as many of these shape matrices as possible, then this will "reduce" G rapidly, and so lead to a decomposition with a small number of shape matrices. Thus we will try to maximize the number of shape matrices that are summed to form E. However we clearly don't want E to be represented by a decomposition that has an "unnecessarily" large number of shape matrices, (what we mean by this will become clearer as we progress), since each of these will appear in the decomposition of G. To be precise, let $\mathbf{D}_G$ denote a decomposition of matrix G into a positively integer weighted sum of shape matrices, i.e. matrices in $\mathcal{X}$. Also let $\mathcal{K}(\mathbf{D}_G)$ denote its cardinality, and $\mathcal{B}(\mathbf{D}_G)$ denote its beam-on time. We now show that for the matrix E and its decomposition that we seek to extract, the beam-on time and cardinality will be identical.

Proposition 2. *Let G be a non-negative integer matrix, and c be an upper bound on any coefficient appearing in any shape matrix decomposition of G that achieves its minimal beam-on time, $B^*(G)$. Now suppose non-negative integer matrices E and F satisfy $cE + F = G$*

and have some shape matrix decompositions $\mathbf{D}_E$ *and* $\mathbf{D}_F$ *respectively satisfying* $c\mathcal{B}(\mathbf{D}_E) + \mathcal{B}(\mathbf{D}_F) = B^*(G)$. *Then it must be that all shape matrix coefficients in* $\mathbf{D}_E$ *are 1, and so* $\mathcal{K}(\mathbf{D}_E) = \mathcal{B}(\mathbf{D}_E)$.

Proof. A shape matrix decomposition of G can be formed by taking all shape matrices in $\mathbf{D}_F$ with their respective coefficients, together with all shape matrices in $\mathbf{D}_E$, with coefficients c times their respective coefficients. Since $cE + F = G$ this is obviously a valid shape matrix decomposition of G; call it $\mathbf{D}_G$. Since $c\mathcal{B}(\mathbf{D}_E) + \mathcal{B}(\mathbf{D}_F) = B^*(G)$ it also obvious that $\mathcal{B}(\mathbf{D}_G) = B^*(G)$. Now any matrix in $\mathbf{D}_E$ with coefficient d, say, greater than 1 would give a matrix in $\mathbf{D}_G$ with coefficient $cd > c$, contradicting the definition of c. The result follows. □

As we discussed earlier, although we want to extract as many shape matrices as possible at the maximum coefficient, there would clearly be no point in considering a decomposition for E which used more than its minimum number of shape matrices. As Proposition 2 shows, the number of shape matrices in the decomposition of E must be exactly the beam-on time. This motivates the following *extraction optimization problem*, which we denote by EOP(G,c):

$$\begin{aligned} \max_{E,F,\mathbf{D}_F} \quad & B^*(E) \\ \text{s.t.} \quad & cE + F = G \\ & cB^*(E) + \mathcal{B}(\mathbf{D}_F) = B^*(G). \end{aligned}$$

This problem appears to be rather hard. Fortunately, we can show it is equivalent to a somewhat simpler problem, and we go on to give a formulation for the simpler problem that is highly tractable in practice. In the simpler problem, we can replace $B^*(E)$ with $\mathcal{B}(\mathbf{D}_E)$, where now we may also optimize over the choice of decomposition of E, $\mathbf{D}_E$. Despite appearing more complex, this is actually simpler to solve, as we subsequently show.

Proposition 3. *The extraction optimization problem EOP(G,c) is equivalent to the* simple extraction optimization problem, *which we denote by SEOP(G,c), given by*

$$\begin{aligned} \max_{E,\mathbf{D}_E,F,\mathbf{D}_F} \quad & \mathcal{B}(\mathbf{D}_E) \\ \text{s.t.} \quad & cE + F = G && (2) \\ & c\mathcal{B}(\mathbf{D}_E) + \mathcal{B}(\mathbf{D}_F) = B^*(G). && (3) \end{aligned}$$

Proof. We will show that for any optimal solution $(E, \mathbf{D}_E, F, \mathbf{D}_F)$ of the simple extraction problem, it must be that $\mathcal{B}(\mathbf{D}_E) = B^*(E)$. Since the simple extraction problem is obviously a relaxation of the original extraction problem, the result follows. So consider an optimal solution $(E, \mathbf{D}_E, F, \mathbf{D}_F)$ of the simple extraction problem and suppose that $\mathcal{B}(\mathbf{D}_E) \neq B^*(E)$, i.e. $\mathcal{B}(\mathbf{D}_E) > B^*(E)$. Then there must exist another decomposition of E, $\mathbf{D}^*_E$, say, which has minimal beam-on time, i.e. with $\mathcal{B}(\mathbf{D}^*_E) = B^*(E)$. But by (2), and since $\mathbf{D}^*_E$ is a shape matrix decomposition of E, we can form a shape matrix decomposition of G by taking all shape matrices in $\mathbf{D}_F$ with their respective coefficients, together with all shape matrices in $\mathbf{D}^*_E$, with coefficients c times their respective coefficients; call this $\mathbf{D}_G$. So by (3)

$$\mathcal{B}(\mathbf{D}_G) = c\mathcal{B}(\mathbf{D}^*_E) + \mathcal{B}(\mathbf{D}_F) = cB^*(E) + \mathcal{B}(\mathbf{D}_F) < c\mathcal{B}(\mathbf{D}_E) + \mathcal{B}(\mathbf{D}_F) = B^*(G),$$

which contradicts the minimality of $B^*(G)$. □

We are now in a position to more formally define our algorithm.

`The Sequential Maximizing Extraction Heuristic (SMEH)`

```
Set $q := 1$, $G^1 := G$, and $c_1 := c_{max}(G)$
**while** $c_q \geq 2$ **do**
    Solve SEOP($G^q$,$c_q$) to obtain solution $E^q$, $F^q$
    **if** $E_q = 0$ **then** set $c_{q+1} := c_q - 1$
    **else**
        Set $G^{q+1} := F^q$ $(= G^q - c_q E^q)$
        Set $c_{q+1} := \min\{c_{max}(G^{q+1}), c_q - 1\}$
    **endif**
    Set $q := q + 1$
**endwhile**
```

Let Q denote the value of q at the conclusion of this algorithm. Then our solution (shape matrix decomposition of G) is found by taking all shape matrices in the minimal beam-on time decomposition of E^q, with coefficient c^q, for each $q = 1, \ldots, Q-1$, together with the minimal beam-on time decomposition of G^Q (in which all coefficients must be 1).

We note that the decision to decrease c_q by 1 in the case that $E^q = 0$, and to ensure $c_{q+1} \leq c_q - 1$ otherwise, is easily justified from the structure of the SOEP(G^q,c_q): in any optimal solution of this problem $\mathbf{D}_F$ cannot include any matrix coefficient of value c_q, since otherwise the corresponding matrix could be subtracted from F and added to E, thereby increasing $\mathcal{B}(\mathbf{D}_E)$ and improving the supposedly optimal objective value. (Note $\mathcal{B}(\mathbf{D}_E)$ must increase, since for it to stay the same would mean that the matrix with coefficient c_q in $\mathbf{D}_F$ already appeared in $\mathbf{D}_E$, in which case that matrix must have coefficient $2c_q$ in the corresponding decomposition of G^q, contradicting maximality of c^q.)

We defer discussion of the c_{max} function to the following section, as it depends on decomposition properties discussed therein. For now, we simply note that if G is the zero matrix, then we set $c_{max}(G) = 0$. This ensures that we can "jump out" of the algorithm while loop as is appropriate in this case.

3. Solving the Simple Extraction Optimization Problem

In this section, we will show how to formulate the SEOP as an integer linear program, using a convenient characterization of matrix decompositions, developed in Baatar et al. (2005); Baatar (2005), that we will call the *L-R representation*. We then show how lower and upper bounds on the variables could be used to strengthen the formulation, and to provide c_{max}. We give "obvious" bounds, and then go on to show that bounds that are in some sense the "best possible" can be derived by solving auxilliary integer programs which have the integrality property. Finally we discuss the addition of a "density" term to the SEOP objective. This term seeks to expand the "total quantity" (in a sense that will be made clear later) extracted from the given matrix, but not at the expense of the number of shape matrices extracted. We present the computational effect of the formulation strengthening, the improved bounds, and the density term, on the overall algorithm.

3.1 An Integer Linear Programming Formulation of the SEOP

A convenient characterization of integer matrix decompositions satisfying the interleaf constraint is as follows.

Theorem 1. *(Theorem 3.1 in Baatar et al. (2005)) An $M \times N$ non-negative integer matrix G has a decomposition with beam-on time β if and only if there exist non-negative $M \times (N+1)$*

matrices L and R such that

$$\begin{aligned}
L - R &= \tilde{G} && && (4)\\
\sum_{k=1}^{n} L_{m-1,k} &\geq \sum_{k=1}^{n} R_{m,k} && \forall m = 2,\ldots,M,\ \forall n = 1,\ldots,N+1 && (5)\\
\sum_{k=1}^{n} L_{m,k} &\geq \sum_{k=1}^{n} R_{m-1,k} && \forall m = 2,\ldots,M,\ \forall n = 1,\ldots,N+1 && (6)\\
\sum_{n=1}^{N+1} L_{p,n} &= \sum_{n=1}^{N+1} R_{m,n} = \beta && \forall p, m = 1,\ldots,M, && (7)
\end{aligned}$$

where the notation $\tilde{G}$ is defined for any matrix G as

$$\tilde{G}_{m,n} = \begin{cases} G_{m,1}, & n = 1,\\ G_{m,n} - G_{m,n-1}, & n = 2,\ldots,N,\\ -G_{m,N}, & n = N+1, \end{cases} \qquad (8)$$

for each m=1,...,M, n=1,...,N+1.

We call a pair of matrices (L, R) satisfying the conditions of Theorem 1 an *L-R representation*: they represent a family of decompositions of G having beam-on time $\beta = \sum_{n=1}^{N+1} L_{1,n}$. Now to relate G directly to L and R, we use $\tilde{G} = L - R$ and the definition of $\tilde{G}$ to deduce that

$$G_{m,n} = \sum_{j=1}^{n}(L_{m,j} - R_{m,j}), \qquad \forall m = 1,\ldots,M,\ n = 1,\ldots,N. \qquad (9)$$

From the above it is clear that $L_{m,n}$ represents the total beam-on time applied to shape matrices having left leaf in row m in position n, (i.e. covering columns $1, 2, \ldots, n-1$), and that, similarly, $R_{m,n}$ represents the total beam-on time applied to shape matrices having right leaf in row m in position n (i.e. covering columns $n, n+1, \ldots, N+1$). We will make use of this understanding later, for example, in determining c_{max}.

We now apply this characterization to help us formulate the SEOP(G,c). We represent the decomposition $\mathbf{D}_E$ by the pair (E^L, E^R) and the decomposition $\mathbf{D}_F$ by the pair (F^L, F^R). These four matrices are the variables in our model. Then we can express the beam-on times of the decompositions as

$$\mathcal{B}(\mathbf{D}_E) = \sum_{n=1}^{N+1} E^L_{1,n}$$

and

$$\mathcal{B}(\mathbf{D}_F) = \sum_{n=1}^{N+1} F^L_{1,n}$$

and so model (3) as

$$c\sum_{n=1}^{N+1} E^L_{1,n} + \sum_{n=1}^{N+1} F^L_{1,n} = B^*(G). \tag{10}$$

Of course, the L-R representations of E and F also have to satisfy the conditions of Theorem 1. We ask first that the equivalents of (5), (6) and (7) apply to the L-R representation of E:

$$\begin{aligned}
\sum_{k=1}^{n} E^L_{m-1,k} &\geq \sum_{k=1}^{n} E^R_{m,k} && \forall m = 2,\dots,M,\ \forall n = 1,\dots,N+1 && (11)\\
\sum_{k=1}^{n} E^L_{m,k} &\geq \sum_{k=1}^{n} E^R_{m-1,k} && \forall m = 2,\dots,M,\ \forall n = 1,\dots,N+1 && (12)\\
\sum_{n=1}^{N+1} E^L_{m,n} &= \sum_{n=1}^{N+1} E^L_{1,n} && \forall m = 2,\dots,M, && (13)\\
\sum_{n=1}^{N+1} E^L_{m,n} &= \sum_{n=1}^{N+1} E^R_{m,n} && \forall m = 1,\dots,M. && (14)
\end{aligned}$$

Here we have expressed the row sum equality part of (7) with the equivalent combination of (13) and (14). Similarly, for the L-R representation of F we ask that

$$\begin{aligned}
\sum_{k=1}^{n} F^L_{m-1,k} &\geq \sum_{k=1}^{n} F^R_{m,k} && \forall m = 2,\dots,M,\ \forall n = 1,\dots,N+1 && (15)\\
\sum_{k=1}^{n} F^L_{m,k} &\geq \sum_{k=1}^{n} F^R_{m-1,k} && \forall m = 2,\dots,M,\ \forall n = 1,\dots,N+1 && (16)\\
\sum_{n=1}^{N+1} F^L_{m,n} &= \sum_{n=1}^{N+1} F^L_{1,n} && \forall m = 2,\dots,M, && (17)\\
\sum_{n=1}^{N+1} F^L_{m,n} &= \sum_{n=1}^{N+1} F^R_{m,n} && \forall m = 1,\dots,M. && (18)
\end{aligned}$$

Now to recover E and F from their L-R representations, we apply the relationship give in (9):

$$E_{m,n} = \sum_{j=1}^{n}(E^L_{m,j} - E^R_{m,j}), \qquad \forall m = 1,\dots,M,\ n = 1,\dots,N,$$

and similarly

$$F_{m,n} = \sum_{j=1}^{n}(F^L_{m,j} - E^R_{m,j}), \qquad \forall m = 1,\dots,M,\ n = 1,\dots,N.$$

Now we have no need to explicitly include E and F in the integer programming model, but we do need to ensure that they are non-negative matrices. Thus we require the additional constraints

$$\begin{aligned} \sum_{j=1}^{n}(E^L_{m,j} - E^R_{m,j}) &\geq 0, \qquad \forall m = 1, \ldots, M,\ n = 1, \ldots, N, \qquad (19)\\ \sum_{j=1}^{n}(F^L_{m,j} - F^R_{m,j}) &\geq 0, \qquad \forall m = 1, \ldots, M,\ n = 1, \ldots, N. \qquad (20)\end{aligned}$$

Finally, we of course need the fundamental relationship (2) to hold. Substituting in the expressions for E and F in terms of their L-R representations, (2) becomes

$$c\sum_{j=1}^{n}(E^L_{m,j} - E^R_{m,j}) + \sum_{j=1}^{n}(F^L_{m,j} - F^R_{m,j}) = G_{m,n}, \qquad \forall m = 1, \ldots, M,\ n = 1, \ldots, N. \qquad (21)$$

We can now summarize the integer linear programming model of SEOP(G,c) as SEO-IP(G,c):

$$\begin{array}{ll} \max & \sum_{n=1}^{N+1} E^L_{1,n} \\ \text{s.t.} & (10), (11), (12), (13), (14), (15), (16), (17), (18), (19), (20), (21) \\ & E^L, E^R, F^L, F^R \in \mathbb{Z}_+^{M\times(N+1)} \end{array}$$

Table 1, in the column labelled "SEOP-IP", presents typical average computation times for the SME heuristic, using the above model to solve SEOP in the inner loop. These use the most obvious value for c_{max}, the maximum value in the intensity matrix. In later sections, we show how values for c_{max} can be tightened, to give the improvements shown in the second column of results in Table 1. We discuss these further in Section 3.4. Here we focus on initial improvements to the basic model above.

This model can be further improved with an additional constraint. Since $cE + F = G$, it must be that $E_{m,n} \leq \lfloor G_{m,n}/c \rfloor$ for all $m = 1, \ldots, M$ and $n = 1, \ldots, N + 1$. Recalling how E can be recovered from E^L, E^R, we deduce the additional constraint

$$\sum_{k=1}^{n}(E^L_{m,k} - E^R_{m,k}) \leq \lfloor G_{m,n}/c \rfloor, \qquad \forall m = 1, \ldots, M,\ n = 1, \ldots, N. \qquad (22)$$

The third column of results in Table 1, (labelled "+ Cuts"), shows the performance benefit that results from this additional constraint; it reduces the computation time by about 55%. Note that the second, third and fourth column of results all use the same method of calculating c_{max}, so these show the relative benefits of improvements to the model by way of the above cuts, and the additional variable bounds discussed further in Section 3.3.

Table 1: Computation time improvements

		Simple L-R Bounds			Best L-R Bounds
ϱ	SEO-IP c_{max} from (29)	c_{max} from (30)	+ Cuts	+ Var Bounds (27) and (28)	
3	18.23	17.69	7.11	5.86	3.5
4	22.75	20.81	9.83	6.82	4.58
5	33.44	32.11	13.97	9.59	6.05
6	38.01	38.00	16.12	10.78	7.67
7	46.44	46.47	20.35	13.36	9.26
8	53.5	52.20	23.69	15.52	10.46
9	60.01	59.68	27.17	16.98	11.30
10	66.42	64.19	30.39	18.62	12.88
Total CPU	338.8	331.15	148.63	97.53	65.70

Note. Average computation times (in seconds) taken over 30 instances, each with matrix G of dimension 10x10 with entries randomly generated in the range $0, 1, \ldots, \varrho$, where each value in the range is equally likely, for running SMEH, with SEOP solved via the SEO-IP model, with and without improvements due to additional constraints and bounds tightening. For the column labelled "SEO-IP", the basic form of the model SEO-IP(G,c) is used, where $c = c_{max}(G)$ is calculated according to (29). In the column labelled "c_{max} from (30)", the simple bounds $\bar{L}$ and $\bar{R}$ calculated from (25) and (26) are used in (30) to calculated c_{max}. Results in the column labelled "+ Cuts" show the effect of adding constraints (22). Results in columns labelled "+ Var Bounds" show the effect of tightening bounds on the variables according to (27) and (28), using the simple bounds on $\bar{L}$ and $\bar{R}$ from (25) and (26), and using the best bounds defined in Section 3.2, in the two columns respectively.

In the next sections, we show how we can strengthen this formulation further, and how our c_{max} function is derived

3.2 Bounds on L-R Representations

Before discussing how we further strengthen the SEO-IP(G,c) formulation, we make a few observations that will be helpful.

Firstly, we note that if (E^L, E^R), (F^L, F^R) are feasible for the SEO-IP(G,c), then not only is (E^L, E^R) an L-R representation of E and (F^L, F^R) an L-R representation of F having $cE + F = G$, but $(cE^L + F^L, cE^R + F^R)$ is an L-R representation of G having beam-on time $B^*(G)$. We will use this fact, together with further properties of L-R representations, to deduce bounds on the variables in the SEO-IP(G,c) model.

Previous work (see Baatar et al. (2005); Baatar (2005) and in particular Theorem 2 of Baatar and Hamacher (2003)) has established that if the pair (L, R) is an L-R representation of G, then there exists a non-negative matrix W such that

$$L = \tilde{G}^+ + W \qquad \text{and} \tag{23}$$

$$R = \tilde{G}^- + W. \tag{24}$$

where $\tilde{G}^+$ and $\tilde{G}^-$ are defined by

$$\begin{aligned} \tilde{G}^+_{m,n} &= \max\{0, \tilde{G}_{m,n}\}, \qquad \text{and,} \\ \tilde{G}^-_{m,n} &= \max\{0, -\tilde{G}_{m,n}\}, \qquad\qquad \forall m = 1, \ldots, M,\ n = 1, \ldots, N+1. \end{aligned}$$

We will use this observation to deduce upper and lower bounds on the entries in L and R, for any L-R representation of G having beam-on time $B^*(G)$. We seek matrices $\underline{L}$, $\underline{R}$, $\bar{L}$, and $\bar{R}$ so that $\underline{L} \le L \le \bar{L}$ and $\underline{R} \le R \le \bar{R}$ for such an L-R representation.

We first discuss some obvious choices for the bounds, then discuss how tighter bounds can be deduced. We will discuss both lower and upper bounds, since both are readily derived from the same analysis, but in practice, we only make use of the upper bounds. We conclude by showing how these bounds can be used to strengthen the SEO-IP(G,c) formulation.

It is obvious from (23) and (24), and since W is required to be non-negative, that we may take $\underline{L} = \tilde{G}^+$ and $\underline{R} = \tilde{G}^-$. Furthermore, we observe that for each row $m = 1, \ldots, M$, by (7) and since (L, R) is an L-R representation of G having beam-on time $B^*(G)$, we have that the row sum $\sum_{j=1}^{N+1} L_{m,j} = B^*(G)$. But by (23), $\sum_{j=1}^{N+1} L_{m,j} = \sum_{j=1}^{N+1} \tilde{G}^+_{m,j} + \sum_{j=1}^{N+1} W_{m,j}$. Thus

$$\sum_{j=1}^{N+1} W_{m,j} = B^*(G) - \sum_{j=1}^{N+1} \tilde{G}^+_{m,j}.$$

We deduce that since W is non-negative, each element $W_{m,n}$ in row m satisfies $W_{m,n} \le B^*(G) - \sum_{j=1}^{N+1} \tilde{G}^+_{m,j}$. It follows from (23) that $L_{m,n} \le \tilde{G}^+_{m,n} + B^*(G) - \sum_{j=1}^{N+1} \tilde{G}^+_{m,j}$, and so we can define $\bar{L}$ by

$$\bar{L}_{m,n} = \tilde{G}^+_{m,n} + B^*(G) - \sum_{j=1}^{N+1} \tilde{G}^+_{m,j}, \qquad \forall m = 1, \ldots, M,\ n = 1, \ldots, N+1. \tag{25}$$

Similarly, we may take

$$\bar{R}_{m,n} = \tilde{G}^-_{m,n} + B^*(G) - \sum_{j=1}^{N+1} \tilde{G}^-_{m,j}, \qquad \forall m = 1, \ldots, M,\ n = 1, \ldots, N+1. \tag{26}$$

We call the above bounds the *simple bounds* and note that they actually derive from bounds on W: we have from the argument above that for each $m = 1, \ldots, M$ and $n = 1, \ldots, N+1$,

$$0 \le W_{m,n} \le B^*(G) - \sum_{j=1}^{N+1} \tilde{G}^+_{m,j} = B^*(G) - \sum_{j=1}^{N+1} \tilde{G}^-_{m,j},$$

by the definition of $\tilde{G}$. Using $\underline{W}$ and $\bar{W}$ to store the left- and right-hand side of these inequalities respective, we see that the simple bounds are obtained from (23) and (24) via

$$\begin{aligned} \tilde{G}^+ + \underline{W} \le L \le \tilde{G}^+ + \bar{W}, \qquad & \text{and} \\ \tilde{G}^- + \underline{W} \le R \le \tilde{G}^- + \bar{W}. & \end{aligned}$$

If we can get tighter lower and upper bounds on W, we can thus get tighter bounds on L and R. To do this, we propose to minimize and maximize each individual value of $W_{m,n}$, subject to the requirement that W correspond to an L-R representation of G having beam-on time $B^*(G)$. This will give us bounds that are in some sense "best possible". From Theorem 1, and the results above, it is not hard to see that this leads to the integer linear programming model

$$\begin{array}{ll} \max & W_{m,n} \\ \text{s.t.} & \sum_{k=1}^{N+1} W_{j,k} = B^*(G) - \sum_{k=1}^{N+1} \tilde{G}^+_{j,k}, \ \forall j = 1, \ldots, M \\ & \sum_{k=1}^{h} \tilde{G}^+_{j-1,k} + \sum_{k=1}^{h} W_{j-1,k} \ge \sum_{k=1}^{h} \tilde{G}^-_{j,k} + \sum_{k=1}^{h} W_{j,k}, \ \forall j = 2, \ldots, M, \ h = 1, \ldots, N+1, \\ & \sum_{k=1}^{h} \tilde{G}^+_{j,k} + \sum_{k=1}^{h} W_{j,k} \ge \sum_{k=1}^{h} \tilde{G}^-_{j-1,k} + \sum_{k=1}^{h} W_{j-1,k}, \ \forall j = 2, \ldots, M, \ h = 1, \ldots, N+1, \\ & W \in \mathbb{Z}_+^{M \times (N+1)}, \end{array}$$

to obtain the upper bound on $W_{m,n}$; of course the objective should switch to minimization when seeking the lower bound. When we take $\underline{W}$ and $\bar{W}$ to be the lower and upper bounds obtained by solving these optimization problems we call these the *best bounds*, as they represent in some sense the best possible bounds on W.

Now integer programs having this structure are proved in (see Baatar, 2005, chap. 3.2) to have the integrality property, and thus can be solved efficiently. In fact, their solutions can also be calculated via a direct combinatorial approach, using multi-objective optimization arguments similar to those discussed in Baatar et al. (2005) (see also Baatar (2005)). Justifying the combinatorial approach requires a lengthy and rather technical argument, so we omit it here, but note that we do use the approach in our implementation.

3.3 Using Bounds to Strengthen the Integer Programming Formulation

We now discuss how bounds on the L-R representations of G having beam-on time $B^*(G)$ can be used to strengthen the SEO-IP(G,c), and also to calculate $c_{max}(G)$.

Recall that if (E^L, E^R), (F^L, F^R) is feasible for the SEO-IP(G,c), then $(cE^L + F^L, cE^R + F^R)$ is an L-R representation of G having beam-on time $B^*(G)$. Thus if $\bar{L}$ and $\bar{R}$ are upper bounds on any L-R representation of G having beam-on time $B^*(G)$, then it must be that $cE^L + F^L \leq \bar{L}$ and $cE^R + F^R \leq \bar{R}$. From this it is not hard to see that we can add the bounds constraints

$$E^L_{m,n} \leq \lfloor \bar{L}/c \rfloor, \qquad \forall m = 1, \ldots, M,\ n = 1, \ldots, N+1, \tag{27}$$

and

$$E^R_{m,n} \leq \lfloor \bar{R}/c \rfloor, \qquad \forall m = 1, \ldots, M,\ n = 1, \ldots, N+1. \tag{28}$$

Table 1 shows that tightening the bounds on the E^L and E^R variables according to (27) and (28), with $\bar{L}$ and $\bar{R}$ taken to be the simple bounds defined in Section 3.2, reduces the computation time by about 34.4%. Taking $\bar{L}$ and $\bar{R}$ to be the best bounds reduces it still further, by another 32.6%. The effect of all the improvements to the SEOP-IP formulation – the additional cuts and bound tightening with best bounds – reduces the computation time by around a factor of 5, with the combined average computation times reduced from over 330 to under 66 seconds.

3.4 Using Bounds to Calculate $c_{max}(G)$

For a given matrix G, we must calculate $c_{max}(G)$, an upper bound on the maximum beam-on time that could be applied to any shape matrix in any decomposition of G having beam-on time $B^*(G)$. A naive choice would be to simply take the maximum intensity of any entry in G, i.e.

$$c_{max}(G) = \max_{\substack{m=1,\ldots,M \\ n=1,\ldots,N}} G_{m,n}. \tag{29}$$

Using the bounds we derived in the previous section, we can do better than that. Firstly recall our understanding of the L and R matrices in an L-R representation: their entries represent the total beam-on time applied to shape matrices with left and right leaves respectively in the corresponding position. Now any shape matrix Y has the property that for each row $m = 1, \dots, M$ there exists a pair $l_m, r_m \in \{1, \dots, N+1\}$ with $l_m \le r_m$ so that $Y_{m,n} = 1$ for all $n = l_m, \dots, r_m - 1$, and $Y_{m,n} = 0$ otherwise (l_m is the left leaf position, r_m the right). Thus any shape matrix Y in a decomposition represented by L and R must have beam-on time α satisfying the following for each row $m = 1, \dots, M$:

1. $\alpha \le L_{m,l_m}$,
2. $\alpha \le R_{m,r_m}$, and
3. $\alpha \le G_{m,n}$ for all $n = l_m, \dots, r_m - 1$.

As a consequence, we can say that the beam-on time α applied to any shape matrix in a decomposition of G belonging to the family represented by the pair (L, R) must have

$$\alpha \le \min_{\{m=1,\dots,M\}} \max_{\{l,r=1,\dots,N+1,\ l\le r\}} \min\{L_{m,l}, R_{m,r}, G_{m,l}, G_{m,l+1}, \dots, G_{m,r-1}\}.$$

For $\bar{L}$ and $\bar{R}$ upper bounds on any such L and R respectively, we can clearly take

$$c_{max}(G) = \min_{\{m=1,\dots,M\}} \max_{\{l,r=1,\dots,N+1,\ l\le r\}} \min\{\bar{L}_{m,l}, \bar{R}_{m,r}, G_{m,l}, G_{m,l+1}, \dots, G_{m,r-1}\}. \quad (30)$$

The benefits to SMEH of using this formula for calculating c_{max} can be seen in the change between the first and second columns of results in Table 1.

Clearly improving the bounds $\bar{L}$ and $\bar{R}$ can give better bounds for $c_{max}(G)$, potentially avoiding wasted iterations of the algorithm, as well as tightening the constraints (27) and (28), and so potentially reducing solution time in the SEOP subproblem. The net benefit of the best bounds is seen in the last column in Table 1.

3.5 The Density Objective Term

As mentioned earlier, we consider the addition of a "density" term to the SEOP objective, that seeks to expand the "total quantity" extracted from the given matrix. The issue is that we may maximize the number of shape matrices extracted at the maximum level, but still "leave behind" a matrix that needs more shape matrices to decompose than is necessary. For example, $c_{max}(G) = 3$ for

$$G = \begin{pmatrix} 2 & 5 & 4 & 0 \\ 1 & 3 & 2 & 0 \\ 6 & 3 & 2 & 2 \\ 3 & 0 & 6 & 3 \end{pmatrix}$$

and both

$$E = \begin{pmatrix} 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 2 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 \end{pmatrix} \quad \text{and} \quad \hat{E} = \begin{pmatrix} 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 \end{pmatrix}$$

solve SEOP(G,3), i.e. either can play the role of the matrix extracted at beam-on time 3, and furthermore, both decompose optimally into two shape matrices. Note that E is "more dense" than $\hat{E}$ in the sense that E extracts a greater "total quantity" of required radiation from G, with a total of $8c_{max}$, (the sum of the entries in E is 8), versus $6c_{max}$ for $\hat{E}$ (the sum of the entries in $\hat{E}$ is 8). Now consider the matrices "left behind" after the extraction of E and $\hat{E}$ respectively. In the first case, $c_{max}(G - 3E) = 2$, and we can decompose $G - 3E$ using two more patterns, one at beam-on time 2 and the other at beam-on time 1:

$$G - 3E = \begin{pmatrix} 2 & 2 & 1 & 0 \\ 1 & 3 & 2 & 0 \\ 0 & 0 & 2 & 2 \\ 0 & 0 & 3 & 0 \end{pmatrix} = 2 \begin{pmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} + \begin{pmatrix} 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}.$$

But in the second case,

$$G - 3\hat{E} = \begin{pmatrix} 2 & 2 & 1 & 0 \\ 1 & 0 & 2 & 0 \\ 3 & 3 & 2 & 2 \\ 0 & 0 & 3 & 3 \end{pmatrix}$$

and $c_{max}(G - 3\hat{E}) = 1$. To see this, observe that otherwise, to preserve minimum beam-on time of the decomposition, it must be that $G - 3\hat{E}$ decomposes into two matrices, one with beam-on time 2, the other with beam-on time 1, call them F^2 and F^1 respectively. Now any occurrence of 2 in $G - 3\hat{E}$ must correspond to an entry of 1 in F^2 and zero in F^1, and any occurrence of 1 in $G - 3\hat{E}$ must correspond to an entry of zero in F^2 and 1 in F^1. Thus it must be that the first two rows of F^1 are:

$$\begin{pmatrix} 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix}.$$

But this violates the interleaf motion constraints; the left leaf in the first row overhangs the right leaf in the second row. So this is impossible. Thus it must be that $c_{max}(G - 3\hat{E}) = 1$, and we require (at least) three different shape matrices to decompose $G - 3\hat{E}$. Thus in this example, choosing to extract the shape matrix with greater density, E, reduces by one the number of shape matrices needed to decompose the residual matrix, $G - 3E$.

This leads us to hope that choosing a denser solution of SEOP(G,c) may in some cases produce a better decomposition. We thus extend our model by introducing an additional density term, defined to be

$$\sum_{m=1}^{M}\sum_{n=1}^{N} E_{m,n} = \sum_{m=1}^{M}\sum_{n=1}^{N}\left(\sum_{k=1}^{n}(E^L_{m,k} - E^R_{m,k})\right)$$

where the right-hand side is obtained by substituting for each $E_{m,n}$ its expression in terms of E^L and E^R as discussed previously.

We then solve the SEOP(G,c) with the following objective function:

$$\sum_{n=1}^{N+1} E^L_{1,n} + \lambda\sum_{m=1}^{M}\sum_{n=1}^{N}\left(\sum_{k=1}^{n}(E^L_{m,k} - E^R_{m,k})\right)$$

where $\lambda > 0$ is a parameter that represents the preference for the density term over the decomposition cardinality of the extracted matrix. If λ is chosen small enough, in practice the model will lexicographically maximize the total beam-on time term first, and then the density term, i.e. choose from amongst the optimal solutions of SEOP the one with greatest density.

We tested the use of the density term numerically, using a variety of values for λ. The results are given in Table 2. Total number of shape matrices produced by the SEM Heuristic, taken over 1000 instances, each with intensity matrix of dimension 10x10 with entries randomly generated in the range $0, 1, \ldots, \varrho$, where each value in the range is equally likely, for $\lambda = 0$, 0.000001, and 1. The least entry in each row is indicated in bold font which shows that the solutions generated by the Sequential Maximizing Extraction Heuristic with $\lambda = 0$ and $\lambda = 1$ are better than $\lambda = 0.000001$.

Thus, in this paper we will focus on Sequential Maximizing Extraction Heuristics with $\lambda = 0$ and $\lambda = 1$; we refer to them as $SEMH_0$ and $SEMH_1$, respectively.

In subsequent results, we see that $SEMH_0$ and $SEMH_1$ have somewhat different properties. As Table 6 shows, $SEMH_0$ produces better quality solutions for smaller values of maximum intensity level, ϱ, and the switch-over point increases as the dimension increases, until for 20×20 problems, $SEMH_0$ is better on average for all ϱ up to the highest value tested ($\varrho = 16$). In terms of run-time, however, $SEMH_1$ more reliable than $SEMH_0$. As Table 3 shows, a much larger proportion of problems require excessively long run-times for $SEMH_0$ than $SEMH_1$. Furthermore, our initial experiments with the 72 of the 14,000 instances of dimension 20×20 that could not be solved within 1800 seconds by $SEMH_0$ shows

Table 2: SEM Heuristics with $\lambda = 0$, 0.000001, and 1

	λ		
ϱ	0	0.000001	1
3	**8158**	8172	8297
4	**9393**	9431	9408
5	10426	10485	**10374**
6	11163	11279	**11095**
7	11849	11930	**11785**
8	12465	12573	**12322**
9	12970	13193	**12885**
10	13504	13731	**13403**
11	13982	14146	**13818**
12	14392	14651	**14269**
13	14793	15027	**14620**
14	15021	15378	**14841**
15	15427	15741	**15265**
16	15741	16073	**15534**

that many of them require hundreds of thousands of seconds, whereas the longest run-time for any 20×20 instance for $SEMH_1$ was under 70,000 seconds (see Table 4).

Because of the excessively long run-times experienced with $SEMH_0$ on some (admittedly very rare) instances, we curtail the heuristic, as follows. A test on elapsed time is added to the test to enter the inner loop of the SEM Heuristic, so that the loop is only entered if $c_q \geq 2$ *and* elapsed time is below some preset limit. Now the same post-processing step is applied, calculating the minimum beam-on time of the residual matrix, G^q. Clearly, in the worst case, each unit of beam-on time will be delivered with a unique shape matrix, so a feasible solution is given by the number of shape matrices found by the heuristic up to the time it exits the loop, plus the minimum beam-on time of the residual. This is the value of the solution returned by the curtailed heuristic. In all experiments with $SEMH_0$ presented in this paper, we use this curtailed version, with the time limit set to 1800 seconds.

4. Computational Performance

We implemented our algorithm using CPLEX 9.0 embedded in C++ on PC, Dell Latitude D800, Pentium(R) M Processor 2.00 GHz, 2.00 GB RAM. In what follows, we test the algorithm computationally on both randomly generated and clinical data. We compare its performance as a heuristic against optimal values, where possible, and against what we believe is the current state-of-the-art heuristic, confirmed in results given in the recent survey of Ehrgott et al. (2007) to be that of Kalinowski (2005a). We observe that it is often the case that when one variant of the SEMH does badly, Kalinowski's algorithm performs well, and vice versa. This leads us to propose a "best of" heuristic; we demonstrate that this produces substantial improvements over Kalinowski's algorithm alone.

Tests are done on randomly generated 10x10, 15x15 and 20x20 matrices with entries selected from $0, 1, \ldots, \varrho$. Each element in this range is equally likely, and we test $\varrho = 3, \ldots, 16$. We refer to ϱ as the *intensity level* of the matrix. For each matrix size and for each intensity level, ϱ, the algorithms are tested on 1000 matrices, i.e. for each matrix size we tested 14000 instances.

We also test on clinical data. We have 42 problem instances. After stripping off surrounding zero rows and columns, the instances have from 9 to 23 rows, and from 8 to 32 columns. The maximum intensity levels in the matrices range from 9 up to 40.

4.1 A Comparison of the SEM Heuristic with Kalinowski's Algorithm

Table 6 gives detailed results for all algorithms on randomly generated data. From the columns labelled "$SEMH_0$", "$SEMH_1$", and "K", we can compare the performance of $SEMH_0$ and $SEMH_1$ against the algorithm of Kalinowski (2005a), respectively. In terms of computation time, the latter is exceedingly fast. However both $SEMH_0$ and $SEMH_1$ complete in reasonable times, even for large problems. In terms of the quality of solutions produced, both $SEMH_0$ and $SEMH_1$ give better values, on average, for all matrix sizes and intensity levels, than Kalinowski's algorithm. $SEMH_1$ dominates for smaller sizes and higher intensity levels; as size grows, $SEMH_0$ takes over as the dominant algorithm of the three.

We provide more detail on the relative computation time performance for the algorithms in Tables 3 and 4. All three algorithms typically complete very rapidly, but show "heavy tail" behaviour on randomly generated problems, i.e. there are significant proportions of the instances having run-times orders of magnitude apart. Table 3 shows the number of problems unsolved by each algorithm after 50 and 1800 seconds of computing time for $SEMH_0$, $SEMH_1$, and Kalinowski's algorithm respectively, for 20×20 matrices. The number of instances not solved by either of the SEM heuristic is shown in the column labeled as "Both SEM Heuristics" and every instance tested was solved by one of the three algorithms within 50 seconds. Clearly Kalinowski's algorithm is the most robust, in this sense, and $SEMH_1$ is the more robust of the two SEM heuristics.

Table 4 shows average and worst-case run-times for $SEMH_1$ and Kalinowski's algorithm. As was mentioned earlier, the "pathological" instances for $SEMH_0$, (the 72 instances un-

Table 3: Number of instances unsolved after 50 and 1800 seconds for 20 × 20 matrices

	50 second time limit				1800 second time limit			
ρ	$SEMH_0$	$SEMH_1$	K	Both SEM heuristics	$SEMH_0$	$SEMH_1$	K	Both SEM heretic
3	42	28	2	7	11	2	0	1
4	27	25	1	1	5	2	0	0
5	43	18	2	2	7	0	2	0
6	26	22	1	0	5	1	0	0
7	29	20	0	3	7	0	0	0
8	42	14	4	3	7	0	1	0
9	24	15	7	0	2	1	1	0
10	22	16	3	1	6	1	0	0
11	33	16	12	1	6	0	1	0
12	21	14	1	3	3	0	1	0
13	21	14	7	1	3	0	1	0
14	16	10	7	0	2	0	2	0
15	21	11	6	0	4	1	0	0
16	23	10	9	0	4	0	3	0
Total	390	233	62	22	72	8	12	1

solved after 1800 seconds), can take hundreds of thousands of seconds to solve, so we did not solve all of them, instead curtailing $SEMH_0$ to stop at 1800 seconds. So we do not know maximum run times for $SEMH_0$, and average run times would not be directly comparable to the other two algorithms, which were allowed to run without a time limit. Thus we do not include results for $SEMH_0$ in this table. Although Kalinowski's algorithm is generally much faster than $SEMH_1$ on average, the worst CPU time for $SEMH_1$ was much better than that of Kalinowski's algorithm. In about 98% of cases the largest computation for $SEMH_1$ was below 50 seconds and for these instances the average computation time was below 15 seconds, which is acceptable for practical application. Both show extremely long run times in "pathological" instances, but $SEMH_1$ is somewhat more robust, with largest times in the tens of thousands of seconds, rather than the hundreds of thousands that were exhibited by Kalinowski's heuristic. Interestingly, for many instances on which one algorithm is very slow, the other is fast, a point we will return to later.

In order to demonstrate the relative behaviour of the algorithms across instances, we show the frequency of objective differences between the $SEMH_1$ and Kalinowski's algorithms in Table 5. Table 5 summarizes, for each matrix dimension tested, and for each value of Δ shown, the number of instances in which the decomposition cardinality given by $SEMH_1$ exceeded that given by Kalinowski's algorithm, by a margin of Δ. For example, in the 20×20 case, in one instance $SEMH_1$ was worse than Kalinowski's algorithm by 7 matrices, in seven instances $SEMH_1$ was better than Kalinowski's algorithm by 7 matrices, and in 3288 instances they gave the same value. $SEMH_1$ and Kalinowski's algorithms give the

Table 4: CPU time performance for $SEMH_1$ and Kalinowski's algorithm, over 20×20 matrices

	$SEMH_1$				Kalinowski's Algorithm			
ϱ	Avg. CPU time (s)	% instances $\leq$ 50s	Avg. CPU time (s) if $\leq$ 50s	Max CPU time (s)	Avg. CPU time (s)	% instances $\leq$ 3s	Avg. CPU time (s) if $\leq$ 3s	Max CPU time (s)
3	16.7	97.20%	7.6	3327.0	0.7	99.90%	0.7	407.4
4	40.2	97.50%	7.5	24507.2	0.7	99.90%	0.7	510.4
5	10.2	98.20%	8.0	402.3	41.1	99.20%	0.3	24819.5
6	19.1	97.90%	8.9	4710.2	0.6	99.70%	0.3	142.8
7	12.7	98.00%	9.8	741.1	0.5	99.00%	0.4	29.4
8	13.1	98.70%	10.8	535.0	3.0	98.80%	0.5	1952.5
9	27.3	98.60%	11.4	13695.6	357.4	98.70%	0.5	354088.0
10	18.9	98.50%	11.9	5620.6	2.0	98.50%	0.6	1107.3
11	15.6	98.50%	12.5	977.8	7.3	98.00%	0.7	2901.6
12	14.8	98.70%	12.7	841.7	53.4	98.30%	0.7	52514.8
13	15.5	98.80%	13.6	472.3	6.1	98.50%	0.8	3859.6
14	16.0	98.60%	14.3	593.5	210.8	98.20%	0.9	203760.0
15	85.7	98.90%	14.6	69320.0	3.5	98.10%	0.9	1318.7
16	16.5	99.00%	15.4	290.1	55.3	97.90%	1.0	36618.2

Table 5: Frequency of the decomposition cardinality differences

	$SEMH_0$			$SEMH_1$		
Δ	10×10	15×15	20×20	10×10	15×15	20×20
7						1
6		3			1	9
5		24	4	2	14	28
4	10	100	31	24	64	137
3	64	406	147	107	262	359
2	502	1255	568	609	925	953
1	2375	2699	1566	2345	2144	1956
0	6300	4810	3501	5545	3979	3288
-1	3799	2989	4226	3652	3731	3627
-2	802	1230	2649	1279	1913	2129
-3	135	384	971	361	712	1005
-4	12	84	253	65	193	372
-5	1	14	68	11	54	111
-6		1	14		6	18
-7		1	2		2	7
net # of instances	-1798	-3687	-5867	-2281	-3201	-3826
net # of matrices	-2250	-5727	-10600	-3618	-5673	-7412

same decomposition cardinality in only about 25% of instances overall.

These results suggest that the algorithms may be "complementary", in the sense that in a fair proportion of instances, one algorithm is producing a significant smaller objective value than the other. They are also clearly complementary in terms of computation time, as Table 3 shows: the number of instances unsolved by both SEM heuristics after a given time limit is much lower than the number unsolved by either taken separately, and *zero* instances remained unsolved by all three algorithms after the smaller time limit reported (40 seconds). These observations motivate the idea of a "best of" algorithm.

4.2 The Best-Of Heuristic

Since all three of $SEMH_0$, $SEMH_1$ and Kalinowski's algorithm solve in reasonable computing time on the vast majority of instances, we propose a "best-of" heuristic, which runs all three algorithms and returns the best solution found over the three. We call this the *Best-Of* heuristic.

The results are shown in the Table 6 in the columns labelled "Best-Of". The Best-Of algorithm of course provides the best solution in all cases. Importantly, it reduces the total number of shape matrices found by 3%, 3% and 5%, over $SEMH_0$, $SEMH_1$ and Kalinowski's algorithm, respectively. The significance of these reductions is best appreciated when seen in comparison with optimal solution values, as discussed later.

This provides clear confirmation that the heuristics are complementary in terms of solution quality. As mentioned earlier, they also tend to be complementary in terms of computation time, which leads to the following modification.

4.3 The Time-Restricted Best-Of Heuristic

Many instances for which $SEMH_0$ or $SEMH_1$ need longer computation time can be solved by Kalinowski's algorithm in a very short time, and vice versa. For example, compare the CPU times, reported in seconds, consumed by the algorithms for the following instances of 20x20 matrices with intensity levels 9 and 14.

instances	$SEMH_0$	$SEMH_1$	Kalinowski's
m20_ 9_ 267	8.14	13695.61	0.54
m20_ 9_ 695	14.41	12.73	354088.23
m20_ 14_ 215	16.09	13.71	203760.36
m20_ 3_ 18	122803.00	14.35	0.17

We propose an time-restricted best-of heuristic, in which a time limit is set, and each of $SEMH_0$, $SEMH_1$ and Kalinowski's algorithm are run until either they finish, or the time limit is exceeded. The best solution found, taken over all methods that finished within the time limit, is reported. We call this the *Restricted Best-Of* heuristic.

The columns labelled "Restricted Best-Of" in Table 6 show the results of this heuristic, where the time limit is set to the value recorded in the column labelled "Limit (s)". With these limits, in every instance, at least one of $SEMH_0$, $SEMH_1$ or Kalinowski's algorithm found a solution. As can be seen from the third-last column, restricting the time resulted in only very small increases above the Best-Of values. For example, for 10×10 matrices, with a time limit of 1 second, the total number of shape matrices in the decomposition found increased by only one shape matrix over all 1000 randomly generated instances for each value of $\varrho \leq 15$. We investigate the relative contributions of each of $SEMH_0$, $SEMH_1$ and

Table 6: Number of shape matrices and CPU times averaged over 1000 randomly generated instances

		$SEMH_0$		$SEMH_1$		K		Best-Of		Restricted Best-Of		
	ϱ	NS	CPU	NS	CPU	NS	CPU	NS	CPU	NS	CPU	Limit
		(%)	(s)	(%)	(s)	(%)	(s)		(s)	(%)	(s)	(s)
10×10	3	0.44	0.16	2.15	0.15	3.39	0.01	**8122**	0.31	0.01	0.31	1
	4	1.83	0.21	1.99	0.20	5.26	0.01	**9224**	0.42	0.00	0.42	1
	5	3.22	0.26	2.70	0.24	5.68	0.02	**10101**	0.52	0.00	0.52	1
	6	3.95	0.32	3.32	0.29	6.21	0.02	**10739**	0.62	0.00	0.62	1
	7	3.92	0.37	3.36	0.34	5.58	0.02	**11402**	0.73	0.00	0.73	1
	8	4.55	0.41	3.36	0.38	6.17	0.02	**11922**	0.81	0.00	0.81	1
	9	4.47	0.46	3.79	0.41	5.98	0.02	**12415**	0.90	0.00	0.90	1
	10	4.71	0.51	3.92	0.45	5.98	0.03	**12897**	0.99	0.00	0.99	1
	11	5.06	0.55	3.83	0.50	5.89	0.03	**13308**	1.08	0.00	1.08	1
	12	4.81	0.59	3.92	0.53	5.80	0.03	**13731**	1.15	0.01	1.15	1
	13	5.12	0.64	3.89	0.57	6.06	0.03	**14072**	1.24	0.01	1.24	1
	14	5.47	0.68	4.21	0.60	6.24	0.03	**14242**	1.31	0.01	1.31	1
	15	5.50	0.72	4.39	0.63	5.56	0.04	**14623**	1.39	0.00	1.39	1
	16	5.54	0.76	4.15	0.67	5.36	0.04	**14915**	1.46	0.02	1.46	1
15×15	3	0.49	10.66	1.53	1.49	5.03	0.05	**11933**	12.19	0.09	2.81	3
	4	1.87	1.59	2.16	1.89	6.00	0.07	**13707**	3.55	0.01	3.45	10
	5	2.52	2.63	2.82	2.14	6.08	0.07	**15051**	4.84	0.03	4.15	10
	6	2.94	4.15	3.14	2.39	6.04	0.09	**16308**	6.62	0.01	4.67	10
	7	3.23	6.03	3.08	2.93	6.05	0.10	**17253**	9.06	0.01	5.17	10
	8	3.15	4.69	2.99	3.18	6.04	0.31	**18087**	8.17	0.01	5.81	10
	9	3.46	3.05	3.29	3.17	5.96	0.15	**18856**	6.36	0.01	6.23	10
	10	3.57	3.17	3.60	3.70	5.72	0.15	**19590**	7.01	0.01	6.74	15
	11	3.56	3.34	3.79	3.63	5.92	0.24	**20195**	7.21	0.00	7.14	15
	12	3.93	3.60	3.72	3.92	5.40	0.18	**20812**	7.70	0.00	7.70	15
	13	4.22	3.82	3.83	4.06	5.26	0.19	**21365**	8.07	0.00	8.07	15
	14	3.76	4.09	3.73	4.31	5.07	0.21	**21914**	8.60	0.00	8.52	20
	15	4.07	4.28	3.90	4.48	4.90	0.22	**22389**	8.97	0.00	8.95	20
	16	3.92	4.62	3.96	4.82	4.75	0.28	**22861**	9.72	0.00	9.48	20
20×20	3	0.73	36.78	1.03	16.65	6.45	0.71	**15711**	54.14	0.31	14.83	20
	4	1.46	21.13	2.27	40.23	6.49	0.74	**18105**	62.10	0.18	15.01	20
	5	2.12	29.32	2.56	10.23	5.99	41.09	**20139**	80.65	0.30	16.90	20
	6	2.16	23.24	2.96	19.07	5.73	0.57	**21687**	42.88	0.11	20.00	40
	7	2.08	28.81	3.24	12.65	6.42	0.49	**22992**	41.96	0.10	21.70	40
	8	2.42	32.61	3.03	13.13	6.03	3.02	**24265**	48.77	0.19	24.20	40
	9	2.56	22.33	3.12	27.26	5.80	357.36	**25344**	406.95	0.07	24.74	40
	10	2.37	26.30	3.47	18.91	5.56	1.97	**26315**	47.18	0.04	25.98	40
	11	2.46	30.12	3.58	15.61	5.09	7.35	**27345**	53.08	0.10	27.78	40
	12	2.61	24.07	3.50	14.78	5.01	53.41	**28123**	92.26	0.05	27.71	40
	13	2.45	27.65	3.58	15.46	4.68	6.15	**28966**	49.25	0.07	29.36	40
	14	2.67	20.17	3.82	15.99	4.53	7.06	**29764**	43.22	0.07	30.85	40
	15	2.58	26.75	3.76	85.67	4.56	3.50	**30357**	115.92	0.06	31.80	40
	16	2.72	26.92	3.69	16.45	4.35	55.27	**31045**	98.64	0.05	33.44	40

Note. The column labelled "Best-Of NS" reports the total number of shape matrices generated by the Best-Of heuristic. Columns labelled "NS (%)" report the excess shape matrices over the Best-Of value, averaged over the 1000 instances, as a percentage.

Kalinowski's algorithm to the Best-Of results in Table 7, which reports the percentage of instances for which each of the three heuristics gave the best-of value. Note that in many instances more than one of the algorithms gave the best-of value, so we do not expect these to sum to one across a row. Clearly $SEMH_0$ dominates, and the SEM heuristics increasingly dominate Kalinowski's algorithm for larger problems.

Table 7: Percentage of the 1000 randomly generated instances for which each of $SEMH_0$, $SEMH_1$ and Kalinowski's algorithm produced the best value in the Best-Of and Restricted Best-Of heuristics

	ϱ	Best-Of			Restricted Best-Of			
		$SEMH_0$	$SEMH_1$	K	$SEMH_0$	$SEMH_1$	K	limit
10×10	3	96.7	84.2	74.1	96.6	84.2	74.2	1
	4	84.8	83.2	56.9	84.8	83.1	56.9	1
	5	71.7	76.1	53.0	71.7	76.1	53.0	1
	6	65.5	69.6	48.4	65.5	69.6	48.4	1
	7	64.4	68.8	50.1	64.4	68.8	50.1	1
	8	59.0	68.3	47.2	59.0	68.2	47.2	1
	9	58.3	63.5	47.3	58.3	63.5	47.3	1
	10	54.8	61.9	46.0	54.6	61.9	46.0	1
	11	52.2	61.5	45.3	52.2	61.4	45.3	1
	12	52.9	60.3	44.4	53.0	60.2	44.4	1
	13	50.7	61.1	44.5	50.6	61.0	44.5	1
	14	49.1	59.5	44.8	48.7	59.6	45.0	1
	15	46.3	56.4	48.0	46.0	56.3	48.0	1
	16	47.6	57.6	48.0	46.7	57.7	48.2	1
15×15	3	95.0	82.7	48.3	90.2	79.3	49.3	3
	4	77.8	74.1	37.2	77.8	73.4	37.3	10
	5	68.6	66.7	35.9	68.2	66.4	36.0	10
	6	63.5	61.6	34.4	63.1	61.6	34.5	10
	7	60.5	61.5	33.1	60.1	61.5	33.2	10
	8	59.5	61.9	33.2	59.2	61.6	33.3	10
	9	57.2	57.6	35.3	57.0	57.5	35.4	10
	10	53.9	54.3	36.6	54.0	54.1	36.7	15
	11	55.2	54.1	33.3	55.1	54.1	33.2	15
	12	48.8	53.8	38.5	48.8	53.8	38.5	15
	13	46.6	52.9	39.3	46.5	52.9	39.3	15
	14	49.3	54.6	37.5	49.2	54.6	37.6	20
	15	49.8	51.1	41.2	49.7	51.1	41.2	20
	16	48.1	49.8	40.4	48.0	49.7	40.4	20
20×20	3	91.0	84.5	25.7	85.1	78.8	28.4	20
	4	78.3	68.1	21.6	75.0	63.3	23.5	20
	5	66.8	61.9	26.4	61.2	60.5	29.5	20
	6	65.5	56.6	28.6	63.8	55.4	29.6	40
	7	67.5	52.1	23.2	65.8	51.5	24.5	40
	8	62.0	55.2	25.1	59.3	54.4	26.8	40
	9	59.5	53.4	27.3	57.9	53.2	28.1	40
	10	62.4	49.2	28.0	61.4	48.6	28.4	40
	11	60.3	47.8	30.4	58.7	47.6	30.7	40
	12	56.2	48.2	32.2	55.2	47.2	32.9	40
	13	57.4	47.8	33.0	56.5	47.1	33.8	40
	14	54.5	44.8	34.8	54.2	44.5	35.3	40
	15	55.6	43.6	35.2	54.6	43.3	36.0	40
	16	55.0	43.5	34.7	53.7	43.5	35.2	40

4.4 Comparison to Optimal Values

Although there is currently no exact algorithm that can solve the $lex\_min(B, K)$ problem for instances with dimensions large enough to be of use in practical applications, exact algorithms do exist that can obtain optimal values for problems of smaller dimension, or lower intensity level. We adapted the integer programming model presented by Baatar et al. (2007) to take account of interleaf constraints, and used it to solve randomly generated matrices for each dimension from 4×4 up to 10×10, and for each intensity level $\varrho = 3, \ldots, 10$. In these experiments, only 30 instance were tested for each size and intensity level combination, rather than 1000, because of the much longer run times needed to solve the integer program. In Table 8 we report the number of instances out the 30 that can be solved exactly within a time limit of one hour, for each combination.

Table 8: Number of instances solved exactly within a time limit of one hour

Matrix	Intensity level ϱ								Total
size	3	4	5	6	7	8	9	10	
4×4	30	30	30	30	30	30	30	30	240
5×5	30	30	30	30	30	30	30	30	240
6×6	30	30	30	30	30	30	30	30	240
7×7	30	30	30	30	30	28	26	21	225
8×8	30	30	30	30	27	25	23	8	203
9×9	30	29	28	29	21	14	4	1	156
10×10	30	29	27	15	7	1			109

In Table 9, we show the summary performance of $SEMH_0$, $SEMH_1$, Kalinowski's algorithm, and the Best-Of heuristic, against the optimal values. Each entry in Table 9 shows the number of instances in which the heuristic produced a solution with cardinality in excess of the optimal by a margin of Δ. For example, on the 203 instances of size 8×8 which could

Table 9: Frequency of decomposition cardinality excess over the optimal value produced by each heuristic

$SEMH_0$

Matrix	Δ				
size	0	1	2	3	4
4×4	205	34	1		
5×5	184	52	4		
6×6	152	79	9		
7×7	139	69	14	1	2
8×8	119	64	16	4	
9×9	91	51	14		
10×10	75	32	2		
total	**965**	**381**	**60**	**5**	**2**

K

Matrix	Δ				
size	0	1	2	3	4
4×4	209	30	1		
5×5	183	52	5		
6×6	142	85	13		
7×7	121	89	12	1	2
8×8	96	87	16	4	
9×9	73	63	20		
10×10	61	41	5	2	
total	**885**	**447**	**72**	**7**	**2**

$SEMH_1$

Matrix	Δ				
size	0	1	2	3	4
4×4	203	35	2		
5×5	174	60	6		
6×6	154	70	15	1	
7×7	143	60	18	3	1
8×8	114	69	17	3	
9×9	86	46	22	2	
10×10	75	26	7	1	
total	**949**	**366**	**87**	**10**	**1**

Best-Of

Matrix	Δ				
size	0	1	2	3	4
4×4	219	20	1		
5×5	207	32	1		
6×6	180	58	2		
7×7	168	51	4	1	1
8×8	150	50	3		
9×9	115	36	5		
10×10	93	16			
total	**1132**	**263**	**16**	**1**	**1**

be solved to optimality within an hour, $SEMH_0$ found the optimal value in 119 instances, was off by one in 64 instances, off by two in 16 instances, and by three in the remaining 4 instances. $SEMH_0$, $SEMH_1$ and Kalinowski's algorithm were able to find solutions within one of optimal in about 95.2%, 93% and 94.3% of instances overall. In about 68.3%, 67.2% and 62.6% of instances, respectively, they produced optimal solutions. The Best-Of heuristic produced a solution within one of optimal in about 98.7% of instances, and produced optimal solutions in about 80.1% of instances, which is a much higher rate than that of any of the individual heuristics. The average gap and the worst gap of the solutions are shown in Table 10. The Best-Of heuristic closes a significant part of the optimality gap, performing much better on average than any of the individual heuristics alone. The worst-case behaviour still shows instances with large gaps, but these are still far less for the Best-Of heuristic, particularly so on larger instances. On average, the Best-Of heuristic closes more than 50% of the optimality gap shown by Kalinowski's algorithm.

Table 10: Average and worst-case optimality gaps for each size of problem

	Gap Average (%)				Gap Worst (%)			
	$SEMH_0$	$SEMH_1$	K	Best-Of	$SEMH_0$	$SEMH_1$	K	Best-Of
4×4	3.2	3.6	2.9	1.9	40.0	40.0	40.0	40.0
5×5	4.5	5.3	4.7	2.5	40.0	40.0	40.0	40.0
6×6	5.9	6.2	6.9	3.7	28.6	42.9	40.0	28.6
7×7	6.5	6.4	7.5	3.9	100.0	75.0	100.0	75.0
8×8	6.2	6.5	7.7	3.3	37.5	37.5	37.5	25.0
9×9	5.5	6.8	7.4	3.3	25.0	33.3	25.0	25.0
10×10	3.4	4.1	5.4	1.6	20.0	30.0	30.0	12.5

4.5 Results on Clinical Data

We also compared the performance of the heuristics on clinical data. Table 11 shows the results. For each of the 42 instances, the column labelled ϱ shows the highest value occurring in the intensity matrix. For each instance, any zero rows at the top or bottom, and any zero columns on either side of the intensity matrix were removed. The dimensions of the resulting matrix are shown in the columns labelled "# rows" and "# cols". The intensity profiles given in the matrices typically show several intensity "peaks", with intensity values changing smoothly between peaks and troughs. On this clinical data, $SEMH_1$ was relatively less successful, finding the best solution of the three in only half of the 42 instances. $SEMH_0$ was best on 33 instances (almost 78.6%), whereas Kalinowski's algorithm was best on only 26 instances (just under 62%). The Best-Of heuristic was clearly successful, needing at most 65 seconds to run, (and generally much less), and producing solutions that reduced the total decomposition cardinality by nearly 3.7%.

Table 11: Decomposition cardinality and CPU times for clinical data

Data	#	#			$SEMH_0$		$SEMH_1$		K		Best-Of	
set	rows	cols.	ϱ	B^*	NS	CPU	NS	CPU	NS	CPU	NS	CPU
1	9	9	10	17	**9**	0.1	**9**	0.1	**9**	0.0	9	0.25
2	9	9	10	20	10	0.1	**9**	0.1	10	0.0	9	0.28
3	9	10	10	19	**10**	0.2	11	0.2	11	0.0	10	0.39
4	9	10	35	35	11	0.3	11	0.3	**10**	0.0	10	0.63
5	9	10	40	69	**14**	0.9	15	0.8	**14**	0.0	14	1.71
6	9	12	29	46	13	0.6	13	0.8	**12**	0.0	12	1.45
7	9	12	31	45	15	0.7	15	0.7	**14**	0.0	14	1.41
8	9	13	29	45	**14**	1.3	15	1.4	15	0.0	14	2.75
9	10	9	10	18	10	0.2	**9**	0.2	**9**	0.0	9	0.44
10	10	9	10	18	**8**	0.1	9	0.2	9	0.0	8	0.34
11	10	9	10	16	**8**	0.1	**8**	0.1	**8**	0.0	8	0.23
12	10	10	10	16	**8**	0.2	**8**	0.1	**8**	0.0	8	0.31
13	10	15	26	54	**16**	1.1	17	1.3	**16**	0.0	16	2.39
14	11	8	21	21	**8**	0.3	9	0.3	9	0.0	8	0.55
15	11	9	14	23	**10**	0.3	**10**	0.3	**10**	0.0	10	0.68
16	11	9	16	22	**8**	0.2	**8**	0.3	**8**	0.0	8	0.46
17	11	11	22	33	**13**	0.5	**13**	0.5	14	0.0	13	1.03
18	11	12	16	19	10	0.3	**9**	0.3	10	0.0	9	0.62
19	11	12	19	34	14	0.7	14	0.7	**13**	0.0	13	1.43
20	11	12	26	49	**14**	1.1	**14**	1.0	16	0.0	14	2.13
21	11	14	22	38	**14**	1.0	15	1.2	15	0.1	14	2.26
22	14	10	10	23	**11**	0.7	12	1.6	13	0.0	11	2.38
23	14	10	10	30	**14**	0.8	**14**	1.0	15	0.0	14	1.85
24	14	10	10	22	**10**	0.6	12	0.5	**10**	0.0	10	1.13
25	14	10	10	23	**11**	0.6	**11**	0.6	12	0.0	11	1.22
26	14	10	10	24	**11**	0.4	**11**	0.4	**11**	0.0	11	0.85
27	15	10	10	22	**11**	0.5	**11**	0.5	**11**	0.0	11	0.96
28	15	28	9	12	**9**	2.6	**9**	2.1	**9**	0.3	9	4.95
29	16	27	10	12	**9**	1.7	**9**	1.6	**9**	0.2	9	3.53
30	16	28	10	13	**10**	1.7	11	3.9	**10**	0.3	10	5.84
31	16	28	10	11	**10**	1.7	**10**	1.8	**10**	0.3	10	3.78
32	16	29	10	13	**9**	1.6	**9**	1.4	**9**	0.2	9	3.13
33	16	30	10	15	**11**	3.2	**11**	33.7	**11**	0.4	11	37.26
34	20	23	10	11	**6**	2.5	**6**	62.1	7	0.1	6	64.71
35	20	25	9	17	**11**	4.9	12	5.4	12	0.2	11	10.57
36	22	15	26	42	**16**	5.8	18	6.4	**16**	0.2	16	12.41
37	22	18	31	41	**18**	6.9	21	9.7	**18**	0.2	18	16.82
38	22	21	31	58	**21**	14.2	23	15.1	**21**	0.3	21	29.66
39	22	22	22	58	**19**	14.7	20	15.7	21	0.4	19	30.8
40	22	23	24	34	18	10.9	18	11.3	**17**	0.3	17	22.48
41	23	16	33	48	**17**	6.3	**17**	6.5	**17**	0.2	17	12.95
42	23	17	27	46	**17**	8.1	**17**	7.7	**17**	0.2	17	15.97
Total					**506**		523		516		**498**	

5. Conclusions and Future Research

Although integer programming is still not able to produce provably minimum cardinality solutions for practical medical applications in IMRT delivery planning, using integer programming within a sequential extraction heuristic is both fast and effective. In sequential extraction, the "optimal" extraction problem appears to be relatively easy, and the speed of solution can be greatly accelerated with the use of additional constraints and tighter variable bounds. Furthermore, such a sequential extraction approach results in a heuristic which is complementary with the previous best-known heuristic, that of Kalinowski (2005a), in terms of both solution quality and time. Thus "best-of" heuristics prove to be highly effective: they are fast, more robust, and more than halve the optimality gap left by Kalinowski's method.

Whilst the new algorithms we propose here perform very well on average, even the best performs poorly on some instances. Thus future research should focus on robustness, and seek to improve worst-case performance.

Acknowledgement

We thank Dr. Thomas Kalinowski for kindly providing us with the code of his algorithm that made the comparisons of the algorithms possible.

References

Ahuja, R.K., H.W. Hamacher. 2004. Linear time network flow algorithm to minimize beam-on-time for unconstrained multileaf collimator problems in cancer radiation therapy. *Networks* **45** 36–41.

Baatar, D. 2005. Matrix decomposition with time and cardinality objectives: Theory, algorithms and application to multileaf collimator sequencing. Ph.D. thesis, University of Kaiserslautern, Germany.

Baatar, D., N. Boland, S. Brand, P.J. Stuckey. 2007. New constraint programming and integer programming approaches to minimum cardinality decomposition of integer matrices into matrices with the consecutive-ones property. *Proceedings of the Fourth International Conference on Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems*. [In Press], Brussels, Belgium, 1–15.

Baatar, D., H. W. Hamacher, M. Ehrgott, G. J. Woeginger. 2005. Decomposition of integer matrices and multileaf collimator sequencing. *Discrete Applied Mathematics* **152** 6–34.

Baatar, D., H.W. Hamacher. 2003. New LP model for multileaf collimators in radiation therapy planning. *Proceedings of the Operations Research Peripatetic Postgraduate Programme Conference ORP^3*. Lambrecht, Germany, 11–29.

Boland, N., H.W. Hamacher, F. Lenzen. 2003. Minimizing beam-on time in cancer radiation treatment using multileaf collimators. *Networks* **43** 226–240.

Boland, N., L. Jennings, G.H. Wake. 2007. Mixed integer programming approaches to exact minimization of total treatment time in cancer radiotherapy using multileaf collimators. Under revision at *Computers and Operations Research.*

Deng, J., T. Pawlicki, Y. Chen, J. Li, S. Jiang, C-M. Ma. 2001. The MLC tongue-and-groove effect on imrt dose distributions. *Physics in Medicine and Biology* **46** 1039–1060.

Ehrgott, M., H.W. Hamacher, M. Nußbaum. 2007. Decomposition of matrices and static multileaf collimators: a survey. C.J.S. Alves, P.M. Pardalos, L.N. Vicente, eds., *Optimization in Medicine*. Springer-Verlag, Berlin, 27–48.

Hamacher, H.W., K.-H. Küfer. 2002. Inverse radiation therapy planing – a multiple objective optimization approach. *Discrete Applied Mathematics* **118** 145–161.

Kalinowski, T. 2005a. A duality based algorithm for multileaf collimator field segmentation with interleaf collision constraint. *Discrete Applied Mathematics* **152** 52–88.

Kalinowski, T. 2005b. Optimal multileaf collimator field segmentation. Ph.D. thesis, University of Rostock, Germany.

Kamath, S., S. Sahni, J. Li, J. Palta, S. Ranka. 2003. Leaf sequencing algorithms for segmented multileaf collimator. *Physics in Medicine and Biology* **48** 307–324.

Langer, M., V. Thai, L. Papiez. 2001. Improved leaf sequensing reduces segments of monitor units needed to deliver IMRT using MLC. *Medical Physics* **28** 2450–2458.

Siochi, R. A. 1999. Minimizing static intensity modulation delivery time using an intensity solid paradigm. *International Journal of Radiation Oncology, Biology, Physics* **43** 671–680.

Xia, P., L. Verhey. 1999. Multileaf collimator leaf sequencing algorithm for intensity modulated beams with multiple segments. *Medical Physics* **25** 1424–1434.