Contents lists available at ScienceDirect

Computers & Operations Research

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

Computational enhancements to fluence map optimization for total marrow irradiation using IMRT

D.M. Aleman [a,*], V.V. Mišić [a], M.B. Sharpe [b]

[a] Department of Mechanical and Industrial Engineering, University of Toronto, 5 King's College Road, Toronto, Ont., Canada M5S 3G8
[b] Princess Margaret Hospital, Department of Radiation Oncology, University of Toronto, 610 University of Avenue, Toronto, Ont., Canada M5G 2M9

ARTICLE INFO

Available online 15 June 2011

Keywords:
Total marrow irradiation
Intensity modulated radiation therapy
Fluence map optimization
Parallel computing
Projected gradient
Line search
Warm start

ABSTRACT

The fluence map optimization (FMO) problem is a core problem in intensity modulated radiation therapy (IMRT) treatment planning. Although it has been studied extensively for site-specific treatment planning, few studies have examined efficient computational methods for solving it for intensity modulated total marrow irradiation (IM-TMI) planning; few studies have also looked at exploiting prior beamlet information to solve the FMO problem in a beam orientation optimization context. In this study, we consider different types of line search strategies and different types of warm-start techniques to improve the speed with which the FMO problem for IM-TMI is solved and the quality of the end solution. We also consider a parallelism-enhanced algorithm to solve the FMO problem for IM-TMI treatment planning with a large number of beams (36 equispaced beams at each of 11 isocenters, for a total of 396 beams). We show that the backtracking line search strategy with step reduction exhibits the best performance and that using either of the two types of warm-start techniques which we consider leads to significant improvements in both solution time and quality. We also provide results for the aforementioned 396-beam plan and show that 30-beam solutions obtained using beam orientation optimization attain a comparable level of quality as this larger solution.

1. Introduction

Bone marrow transplantation, or hematopoietic stem cell transplantation, is one method of treatment for a number of diseases of the blood and bone marrow. These include certain forms of cancer (such as leukemia and lymphoma) as well as other diseases (such as aplastic anemia and sickle cell disease). To prepare a patient for a bone marrow transplant, the patient's existing diseased bone marrow must be completely eliminated. This is typically done through a procedure known as total body irradiation (TBI). In TBI, the patient's entire body is irradiated with a single, wide-angle beam to a single prescribed level of dose (e.g., 12 Gy). This, however, leads to unnecessary irradiation of healthy tissue, which can lead to many complications and severely affect the patient's quality of life post-treatment.

An alternative to TBI is total marrow irradiation (TMI), which specifically targets the bone marrow of the patient and avoids healthy tissue. In our previous work [1], we examined the use of a high accuracy form of radiotherapy known as intensity modulated radiotherapy (IMRT) to perform TMI. In order to use IMRT for TMI, there are two basic problems that must be solved. The first problem, known as the *fluence map optimization* (FMO) problem, is concerned with how the beamlet intensities, or *fluences*, should be set for a given ensemble of beams to deliver a certain dose to the bone marrow while minimizing the dose delivered to healthy tissue.

The second problem is the *beam orientation optimization* (BOO) problem, which is concerned with determining how a set of beams should be oriented for the dose to be delivered optimally. The FMO problem plays a substantial role in our formulation of the BOO problem, because the optimal objective function value of the FMO problem for a set of beams quantifies the quality of that set of beams—that is, how capable the beams are of delivering the prescribed dose to the bone marrow while minimizing the dose delivered to healthy tissue. We have already studied the BOO problem for TMI previously in [1]; although we will not directly be dealing with the BOO problem, part of this study is concerned with fluence map optimization in the context of BOO.

The FMO problem is the primary problem we will be considering in this study; in particular, we will be studying techniques for improving both the speed with which the FMO problem is solved, and improving the quality of the final beamlet intensities returned. These objectives are motivated by clinical requirements, as there is typically a limited time frame available for treatment plan optimization. Projected gradient methods are commonly used to solve FMO

[*] Corresponding author. Tel.: +1 416 978 6780; fax: +1 416 978 7753.
E-mail addresses: aleman@mie.utoronto.ca (D.M. Aleman), velibor.misic@utoronto.ca (V.V. Mišić), michael.sharpe@rmp.uhn.on.ca (M.B. Sharpe).

0305-0548/$ - see front matter © 2011 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cor.2011.05.028

problems (see, e.g., [1,2]) due to their speed and ease of implementation. We consider three different classes of computational enhancements to improve both the effectiveness and speed of projected gradient methods applied to FMO (and therefore BOO):

1. We consider different line search strategies—that is, different methods of selecting the final step length in each iteration of the projected gradient algorithm. We test the performance of projected gradient with each of these line search strategies on a set of randomly generated 30-beam solutions.
2. We consider different warm-start approaches for speeding up the execution of projected gradient when it is used within a neighborhood search algorithm for BOO (the Add/Drop algorithm, previously studied for TMI in [1]). Each warm-start method involves using bixels for the current set of beams as the initial bixels of each of the neighboring sets of beams, whose optimal FMO objective values are unknown and are to be calculated. We test the performance of the Add/Drop algorithm with each type of warm-start approach on a set of randomly generated 30-beam solutions.
3. We consider a method for parallelizing FMO objective function and gradient evaluation for large-scale treatments. We provide computational results for the performance of projected gradient with this type of enhancement on a very large set of beams.

With regard to parallel and distributed computing in the context of IMRT, there have been few prior studies, and there has not yet been any previous research into using parallel computation for FMO for large-scale (in both the number of beams and the size of the geometry) IMRT treatment planning. In [3], the authors describe an implementation of projected gradient which uses graphics processing units (GPU) for parallel computation; our work differs from this work in that the dose calculation in our work is parallelized with respect to the beams and not with respect to the voxels. This is an important consideration because by splitting the calculations up into several smaller calculations corresponding to subsets of the beams, communication overhead can be reduced. In [4], distributed computing is used to simultaneously sample several potential beam sets for the purpose of BOO. In this paper, parallelization is studied in Section 5, where it is employed in the evaluation of the objective function of the FMO problem and its gradient in order to increase the speed with which the FMO problem is solved when the number of beams is large. Parallelization is also used in the implementations of the warm-start approaches in Section 4, where multiple FMO evaluations are performed in parallel in the context of BOO; this form of parallelization is not the same as the form used in Section 4 and is not the primary focus of this class of computational enhancements.

In addition to the computational gains of parallel computing, another motivation is to be able to solve the FMO problem with all possible beam orientations. Such a treatment plan will allow us to learn about the absolute quality of our solutions in [1]. Another reason for considering the FMO problem with a large number of beams is that the resulting beamlet intensities could be used as a potential starting point for an algorithm to perform intensity modulated arc therapy (IMAT) treatment planning, which may be a more appropriate method of performing TMI than IMRT.

The rest of this paper is organized as follows. In Section 2, we provide a brief overview of the BOO and FMO problems, and the Add/Drop and projected gradient algorithms for solving them, respectively. We describe the different types of line search methods in Section 3, the different types of warm-start techniques in Section 4 and a single parallelism-enhanced algorithm for objective function and gradient evaluation in the context of FMO in Section 5. In Section 6 we provide computational and treatment quality results for our methods. In Section 7, we provide some concluding remarks and some possible directions for future work.

2. BOO and FMO background

We employ the BOO and FMO approaches presented by [1], and provide a brief overview of each here. Although the BOO problem is not the focus of this study and we do not explicitly define any new algorithms to solve it, we will need to make reference to some variables from BOO (and Add/Drop) when we discuss warm-start methods in Section 4.

2.1. BOO problem

We use $\boldsymbol{\theta}$ to represent a single beam orientation and $\mathcal{B}$ to represent the set of all possible beam orientations for a single beam. We use Θ to represent a set of beams, $\Theta=(\boldsymbol{\theta}_1,\ldots,\boldsymbol{\theta}_n)\in\mathcal{B}^n$, where n is the number of beams. We use $\mathcal{F}(\Theta)$ to represent the quality of the beam set Θ (where smaller values correspond to better treatments), which is the optimal FMO objective function value for that set of beams. The BOO problem for n beams can then be stated as

$$\begin{aligned}&\text{minimize} \quad \mathcal{F}(\Theta)\\ &\quad\text{subject to} \quad \Theta\in\mathcal{B}^n.\end{aligned}$$

Due to the nonlinearity and nonconvexity of the function $\mathcal{F}$ with respect to Θ, the BOO problem is difficult to solve [5]. We therefore use the Add/Drop algorithm, a neighborhood search heuristic for BOO in TMI described in [1] (see Algorithm 1). We give a brief overview of the algorithm here.

Let D represent the set of degrees of freedom of each beam; in this study, $D=\{G,z\}$, where G corresponds to the gantry angle and z to the couch-z translation. In iteration i, we have a current iterate, $\Theta^{(i)}$. We establish a neighborhood $\mathcal{N}_{bd}(\Theta^{(i)})$ around $\Theta^{(i)}$, determined by beam $b\in\Theta^{(i)}$ and degree of freedom $d\in D$ (see [1] for full details regarding the construction of $\mathcal{N}_{bd}(\Theta^{(i)})$). We evaluate $\mathcal{F}$ for every neighbor in $\mathcal{N}_{bd}(\Theta^{(i)})$ and identify the solution $\overline{\Theta}\in\mathcal{N}_{bd}(\Theta^{(i)})$ which minimizes $\mathcal{F}$ on $\mathcal{N}_{bd}(\Theta^{(i)})$. If the new point improves the objective function, the algorithm moves to the new point. Otherwise, we consider a different pair $(b',d')\in\{1,\ldots,n\}\times D$ and repeat this process with $\mathcal{N}_{b'd'}(\Theta^{(i)})$ around $\Theta^{(i)}$. The algorithm terminates when every neighborhood of the current solution $\Theta^{(i)}$ has been examined without improvement.

Algorithm 1. Basic Add/Drop.

```
1:  Generate initial starting point $\Theta_0$.
2:  Set $\Theta^* := \Theta_0$ and $i := 0$.
3:  while Stopping criterion is not met do
4:    Select $d\in D$ and $b\in\{1,\ldots,|\Theta|\}$.
5:    Set $\overline{\Theta}\in \arg\min \mathcal{F}(\mathcal{N}_{bd}(\Theta_i)$.
6:    if $\mathcal{F}(\overline{\Theta})<\mathcal{F}(\Theta^*)$ then
7:      Set $\Theta^* := \overline{\Theta}$ and $\Theta_{i+1} := \overline{\Theta}$.
8:      Set $i := i+1$.
9:    end if
10:   if All points in $\bigcup_{b=1}^{k}\bigcup_{d\in D}\mathcal{N}_{bd}(\Theta_i)$ have been sampled
      without improvement then
11:     $\Theta^*$ is a local minimum; go to Step 16.
12:   else
13:     Go to Step 4.
14:   end if
15: end while
16: return $\Theta^*$.
```

2.2. FMO problem

Let B_Θ represent the set of beamlet indices corresponding to the set of beams Θ, and let x_i represent the intensity of beamlet $i\in B_\Theta$. Let S and T represent the set of critical structures and target structures, respectively. The number of voxels of structure $s \in S \cup T$ is given by v_s. Define z_{js} as the total dose received by voxel j in structure s, where D_{ijs} is the dose deposition coefficient of beamlet i in voxel j of structure s.

The objective function of the FMO problem is a weighted sum of penalty functions for each structure. We use $\overline{w}_s$ and $\underline{w}_s$ to represent the weights for overdose and underdose in voxels of structure s, with T_s defining the point at which under- or overdose is penalized. We use $\overline{p}_s$ and $\underline{p}_s$ to represent the powers for overdose and underdose in voxels of structure s. The objective function of the FMO problem is

$$F(x) = \sum_{s \in S \cup T} \sum_{j=1}^{v_s} F_{js}(z_{js}), \tag{1}$$

where z_{js} is given by

$$z_{js} = \sum_{i \in B_\Theta} D_{ijs} x_i \tag{2}$$

and the penalty function F_{js} is defined as

$$F_{js}(z_{js}) = \frac{1}{v_s}(\underline{w}_s[(T_s - z_{js})_+]^{\underline{p}_s} + \overline{w}_s[(z_{js} - T_s)_+]^{\overline{p}_s}), \tag{3}$$

where $(\cdot)_+ = \max\{0, \cdot\}$. The FMO problem can then be stated as

$$\begin{aligned} &\text{minimize} \quad F(x) \\ &\text{subject to} \quad x_i \geq 0 \quad \forall i \in B_\Theta. \end{aligned}$$

By choosing $\underline{w}_s, \overline{w}_s \geq 0$ and $\underline{p}_s, \overline{p}_s \geq 1$ for each structure $s \in S \cup T$, the FMO objective function F is a convex function, and the FMO problem is a convex optimization problem. To solve this problem, we employ the projected gradient algorithm (Algorithm 2), a general algorithm commonly used in large convex optimization problems. Given an initial solution, the algorithm iteratively moves an appropriate step length in the direction of the negative gradient. If the solution is infeasible at any time, the solution is projected onto the feasible set using the function π. The algorithm repeats until the relative improvement between consecutive iterations falls below a given threshold ε.

Algorithm 2. Generalized Projected Gradient.

```
Require Percentage change tolerance $\varepsilon$
1: Generate an initial solution $x^{(0)}$
2: Set $p = 1$.
3: while $p > \varepsilon$ do
4:     Generate a step length $\lambda$
5:     Set $x^{(i+1)} = \pi(x^{(i)} - \lambda \nabla F(x^{(i)}))$
6:     Set $p = (F(x^{(i)}) - F(x^{(i+1)}))/F(x^{(i)})$
7: end while
```

3. Line search strategies

In Algorithm 2, the procedure which is used to generate a step length λ in each iteration is known as a *line search*. The line search step of projected gradient methods is known to be computationally intensive, and thus we first examine several line search strategies as computational enhancements to FMO in the context of total marrow irradiation. In particular, we study the standard backtracking line search, a backtracking line search with step length reduction; a forward line search; a golden section line search; a dichotomous line search; and a quadratic interpolation line search.

The type of line search strategy is of direct interest to us because the line search strategy determines both the quality of the step (which affects the final FMO value obtained) and how many different step lengths are tested (and thus how many objective function evaluations occur) in each iteration of projected gradient. Although the projected gradient algorithm has been widely used in radiotherapy treatment planning (e.g., [3,6–9]), line search strategies for projected gradient in the context of IMRT have not been previously studied as extensively as in our study.

For our line search strategies, the step length λ must satisfy the *sufficient decrease* (also known as Armijo) condition,

$$F(\pi(x^{(k)} - \lambda \nabla F(x^{(k)}))) \leq F(x^{(k)}) - \frac{\sigma}{\lambda}\|x^{(k)} - \pi(x^{(k)} - \lambda \nabla F(x^{(k)}))\|^2, \tag{4}$$

where $\sigma \in (0,1)$ to ensure that the sequence of iterates obtained from Algorithm 2 converges to the global minimum of the FMO problem at a reasonable rate. Details on this condition in the context of projected gradient can be found in [10,11] for unconstrained problems.

3.1. Backtracking line search

The backtracking line search [10] determines a step length in each projected gradient iteration by starting with a step length R_0, checking whether the resulting solution $\tilde{x}$ satisfies the sufficient decrease, and scaling it by a factor β (where $\beta \in (0,1)$) until it satisfies the condition.

3.2. Reduced step line search

The reduced step line search is identical to the backtracking line search, with the slight difference that after m iterations of the outer loop, the initial step length is changed from R_0 to a new value $\overline{R}$. This algorithm grew out of observations that the backtracking projected gradient algorithm in general takes steps as large as R_0 in the early iterations and in later iterations, takes steps that are generally of length βR_0 or smaller. By reducing the initial step length after a few outer loop iterations, we eliminate step lengths that are not likely to satisfy the sufficient decrease condition and thus reduce the overall number of line search iterations (without severely affecting the quality of the end solution).

3.3. Forward line search

The forward line search is similar to the backtracking line search, with the exception that the step lengths move forward rather than backward. In the forward line search, we start from an initial step length R_0 and scale the step length λ up by a factor of γ (where $\gamma \in (1,\infty)$) to just before the point where λ no longer satisfies the sufficient decrease condition.

The reasoning behind scaling the step length up to just before the point where it no longer yields a sufficient decrease in F can be explained as follows. To ensure that the algorithm finds a satisfactory step length using a forward line search method, the initial step length should be sufficiently small so that the line search loop can examine a larger range of step lengths. However, small step lengths typically always meet the sufficient decrease condition, so if the initial length is very small, the algorithm will almost always accept the first step, leading to a very slow rate of change in the iterate x and consequently a slow change in the objective function value $F(x)$. By increasing the step length until it no longer yields a sufficient decrease in F, we are able to force the algorithm to take larger steps and increase the rate at which the sequence of iterates converges to the minimizer of F. (In the case that the first step length $\lambda = R/\|\nabla F(x^{(i)})\|$ does not satisfy the initial step length, it is also scaled down by γ. Although we do not check whether the new step length $\gamma \cdot \lambda$ satisfies the sufficient decrease condition, the initial step length of R can be chosen to be small enough that this is never an issue in practice.)

3.4. Golden section and dichotomous line searches

The golden section and dichotomous line search (similar to bisection) methods are well known methods for minimizing a univariate function when a minimum is known to exist in an interval $[a_0,b_0]$ (the region of uncertainty). For this application, the function that is to be minimized in each projected gradient iteration k is

$$\tilde{f}(\lambda)=F(\pi(x^{(k)}-\lambda\nabla F(x^{(k)}))),$$

and the interval of uncertainty is of the form $[0,R_0/\|\nabla F(x^{(k)})\|]$, where R_0 is an initial length that must be specified. Let m_k be the midpoint of the interval $[a_k,b_k]$.The dichotomous search operates by calculating $\tilde{f}(m_k-\delta)$ and $\tilde{f}(m_k+\delta)$, and comparing them to see whether the minimum lies in $[a_k,m_k+\delta]$ or $[m_k-\delta,b_k]$. Therefore, for the dichotomous search, we also need to specify δ, which is a distinguishability constant.

Our implementations of these methods follow closely the descriptions given in [12], with the following modification: when the objective function is evaluated at multiple step lengths, it is evaluated from largest length to smallest, and the first step length which passes the Armijo condition accepted.

3.5. Quadratic interpolation

The quadratic interpolation line search [11] is described as follows. Given the univariate function $\tilde{f}$,

$$\tilde{f}(\lambda)=F(\pi(x^{(k)}-\lambda\nabla F(x^{(k)}))),$$

we check an initial step length λ_0. If λ_0 satisfies the Armijo condition (4), then λ_0 is chosen as the step length. Otherwise, we form a quadratic approximation to $\tilde{f}$:

$$\tilde{f}_q(\lambda)=\left(\frac{\tilde{f}(\lambda_0)-\tilde{f}(0)+\tilde{f}'(0)\lambda_0}{\lambda_0^2}\right)\lambda^2+(\tilde{f}'(0))\lambda+\tilde{f}(0).$$

We then calculate the minimizer of this quadratic, λ_1:

$$\lambda_1=-\frac{\tilde{f}'(0)\lambda_0^2}{2(\tilde{f}(\lambda_0)-\tilde{f}(0)-\tilde{f}'(0))}.$$

If λ_1 satisfies the Armijo condition (4), we accept it. Otherwise, we repeat this process, with λ_1 taking the place of λ_0, until we find an acceptable step length.

4. Warm-start techniques

The Add/Drop algorithm, as explained in Section 2.1, is an iterative algorithm which improves a solution (a set of beams) by modifying a single beam in a single degree of freedom in each iteration. As a result, when we compare the set of beams $\Theta^{(i)}$ in iteration i to a neighboring set of beams $\tilde{\Theta}\in\mathcal{N}_{bd}(\Theta^{(i)})$ (where $(b,d)\in\{1,\dots,n\}\times\{G,z\}$), we find that the two solutions differ by only one beam, and the two non-matching beams differ in only one degree of freedom. Since the two solutions share all but one beam, the two solutions are very similar to one another, and our intuition suggests that the optimal beamlet intensities of the common beams of $\Theta^{(i)}$ and $\tilde{\Theta}$ should also be very close to one another.

This observation about how the beamlet intensities should change within Add/Drop gives rise to a computational enhancement to FMO evaluation in the context of Add/Drop: specifically, the idea of warm-starting the FMO evaluation of a neighboring set of beams $\tilde{\Theta}$ by using the bixels of the current set of beams $\Theta^{(i)}$. The benefit of this enhancement is that it reduces the time required by the Add/Drop algorithm to find high quality solutions. By warm-starting the FMO evaluation of a neighboring solution $\tilde{\Theta}$ – that is, using the bixels of the current solution $\Theta^{(i)}$ to determine the starting bixels of the neighboring solution – the time required for FMO evaluation can be greatly reduced. By reducing the time required for FMO evaluation, the Add/Drop algorithm can explore the solution space $\mathcal{B}^n$ more rapidly and locate good solutions earlier in the search process. This is important from a clinical standpoint because the amount of time available for beam orientation optimization is limited and it is not possible to run the Add/Drop algorithm until it naturally terminates by finding a local minimum.

We will study three different modes of FMO initialization: cold-start, warm-start using averaging and warm-start using least-squares. There have not been any prior studies into warm-start procedures in the context of BOO to facilitate FMO evaluations. In [13], the authors use warm-starting for the purpose of aperture weight optimization and not in the context of BOO. In [14], the authors also perform warm-starting in a different way, with a set of beams identified from an integer programming model being used as the starting set of beams for a heuristic algorithm.

In the warm-start approaches, we use $\Theta^{(i)}$ to represent the current set of beams; $\tilde{\Theta}$ to represent a neighboring solution of $\Theta^{(i)}$, whose FMO value we are interested in evaluating; $\boldsymbol{\theta}_k$ to represent the single beam in $\Theta^{(i)}$ which is altered; $\tilde{\boldsymbol{\theta}}_k$ to represent the beam in $\tilde{\Theta}$ which corresponds to $\boldsymbol{\theta}_k$ in $\Theta^{(i)}$; $B_{\tilde{\boldsymbol{\theta}}_j}$ to represent the set of indices of the beamlets which belong to beam $\tilde{\boldsymbol{\theta}}_j$ in $\tilde{\Theta}$; $B_{\boldsymbol{\theta}_j}$ to represent the set of bixel indices of beam $\boldsymbol{\theta}_j$ in $\Theta^{(i)}$; x_i to represent the optimal intensity value of beamlet $i\in B_{\Theta^{(i)}}$; and $\tilde{x}_i$ to represent the starting intensity value of beamlet $i\in B_{\tilde{\Theta}}$.

4.1. Cold-start

The most basic form of FMO initialization that we will consider is cold-started FMO evaluation. When an FMO evaluation is cold-started, the bixels of each beam are initialized to a pre-defined constant, and these bixels are fed into the projected gradient algorithm which calculates the FMO value for the beam. This mode of FMO evaluation does not make use of the bixels of the current solution: whenever an FMO is evaluated in this way, it is essentially calculated "from scratch".

Mathematically, if we let κ represent this pre-defined constant, we would set $\tilde{x}_i=\kappa$ for all $i\in B_{\tilde{\Theta}}$. We would then use the resulting set of bixels $\tilde{x}$ as our initial solution for FMO evaluation.

4.2. Warm-start using averaging

The first warm-start method we consider is warm-started FMO evaluation using averaging. Given a neighboring solution whose FMO we are trying to calculate, we initialize the bixels of its beams in the following way: for each beam that the neighboring solution $\tilde{\Theta}$ has in common with the current solution $\Theta^{(i)}$, set the initial bixels of the beam to the optimal bixel values of the same beam in $\Theta^{(i)}$. For beam number k where the neighboring solution $\tilde{\Theta}$ and the current solution $\Theta^{(i)}$ differ, calculate $\overline{x}$, the average of the bixels of the altered beam $\boldsymbol{\theta}_k$ in the current solution $\Theta^{(i)}$, and initialize all of the bixels of the corresponding beam $\tilde{\boldsymbol{\theta}}_k$ in $\tilde{\Theta}$ to $\overline{x}$.

As mentioned earlier, the neighboring solution and the current solution differ by only one degree of freedom of one beam, so we intuitively expect the optimal bixel values of the common beams of the neighboring solution and the current solution to be very close to one another. By allowing FMO evaluation to begin from a set of bixels that should be quite close to the actual optimal set of bixels, the projected gradient algorithm we employ to solve the FMO problem should be able to converge more quickly to the optimal set of bixels. This improvement in the speed of FMO evaluation should result in a higher quality solution being found by Add/Drop, as the algorithm will be able to go through a higher number of iterations.

The step which involves averaging the bixels of the beam that is altered in the current solution and uniformly setting the bixels of the corresponding beam in the neighboring solution to this average can be justified as follows. The sum of the bixels of the beam can be thought of as a measure of the total energy of the beam; the higher the sum of the bixels, the more radiation is being delivered, and the greater the effect the beam has on the patient. (If the sum of the bixels is zero, all of the bixels are zero, and the beam is essentially not delivering any radiation in the treatment.) By setting the bixels of the corresponding beam in the neighboring solution to the average of the altered beam in the current solution, the individual bixel sums of the two beams will be very close to one another, and so the effects of the two beams on the patient should be somewhat similar.

Mathematically, we set $\tilde{x}_i = x_i$ for every $i \in B_{\theta_j}$ and all $\boldsymbol{\theta}_j \in \Theta^{(i)} \backslash \{\boldsymbol{\theta}_k\}$. For those $i \in B_{\tilde{\theta}_k}$, we first calculate

$$\overline{x} = \frac{1}{|B_{\theta_k}|} \sum_{i' \in B_{\theta_k}} x_{i'} \tag{5}$$

and then set $\tilde{x}_i = \overline{x}$ for all $i \in B_{\tilde{\theta}_k}$. The vector of bixels $\tilde{x}$ is then our initial solution when the projected gradient algorithm is executed for $\tilde{\Theta}$.

4.3. Warm-start using least-squares

In the method of warm-starting FMO evaluation using averaging, the motivation behind averaging the bixels of the altered beam $\boldsymbol{\theta}_k$ in the current solution $\Theta^{(i)}$ and setting the bixels of the corresponding beam $\tilde{\boldsymbol{\theta}}_k$ in the neighboring solution $\tilde{\Theta}$ to this average was to attempt to get $\tilde{\boldsymbol{\theta}}_k$ to have a similar effect to $\boldsymbol{\theta}_k$. In this second warm-start method, we take this notion further by selecting the bixels of $\tilde{\boldsymbol{\theta}}_k$ so that the dose contributed by $\tilde{\boldsymbol{\theta}}_k$ is as close as possible to the contribution of $\boldsymbol{\theta}_k$ in a least-squares sense.

To make this notion rigorous, we define $z_{js}^{(k)}$ to be the dose contributed to voxel j in structure s by the bixels of $\boldsymbol{\theta}_k$; it is given by

$$z_{js}^{(k)} = \sum_{i \in B_{\theta_k}} D_{ijs} x_i. \tag{6}$$

To obtain the initial bixel values of $\tilde{\boldsymbol{\theta}}_k$, that is, the values of $\tilde{x}$, we solve the following optimization problem:

$$\begin{aligned} &\text{minimize} \quad \mathcal{Z} = \sum_{s \in S} \sum_{j=1}^{v_s} (\tilde{z}_{js}^{(k)} - z_{js}^{(k)})^2 \\ &\text{subject to} \quad \tilde{z}_{js}^{(k)} = \sum_{i \in B_{\tilde{\theta}_k}} D_{ijs} \tilde{x}_i \quad \forall s \in S,\ j \in \{1, \ldots, v_s\}, \\ &\qquad \tilde{x}_i \geq 0 \quad \forall i \in B_{\tilde{\theta}_k}. \end{aligned} \tag{Warm – LS}$$

For the other beams in $\tilde{\Theta}$, we set the bixels to the corresponding bixels in $\Theta^{(i)}$; i.e., $\tilde{x}_i = x_i$ for every $i \in B_{\theta_j}$ and all $\boldsymbol{\theta}_j \in \Theta^{(i)} \backslash \{\boldsymbol{\theta}_k\}$.

To see why initializing the bixels of $\tilde{\boldsymbol{\theta}}_k$ in this manner might be desirable, consider the following example. Suppose that the beamlets of $\tilde{\boldsymbol{\theta}}_k$ are able to attain a dose contribution that is very close to (or the same as) the dose contribution of the beamlets of the old beam $\boldsymbol{\theta}_k$. The two solutions $\tilde{\Theta}$ and $\Theta^{(i)}$ will then result in very similar doses in the patient. This in turn means that the objective function value associated with $\tilde{x}$ will be very similar to, if not the same as, the objective function value associated with x. If we begin evaluating the FMO value of $\tilde{\Theta}$ starting from $\tilde{x}$, the projected gradient phase will either be short if the starting solution is already sufficiently good, or will yield a significantly better solution.

On the other hand, if the beamlets of the $\tilde{\boldsymbol{\theta}}_k$ are unable to attain a contribution in dose that is at all similar to the beamlets of the old beam, then that could mean two things. One scenario is that it may still be possible for the beam to improve the solution (e.g., by delivering dose to target voxels that are already covered by other beams, but doing so at a lower penalty to the objective). In this case, the projected gradient algorithm still has to iterate as usual, and we do not gain any improvement in the time needed to calculate the FMO. In contrast, if the beam is "bad" (i.e., taking out the old beam prevents the solution from hitting a lot of target voxels and forces overdosing in critical structure voxels), then once again, the projected gradient algorithm still has to iterate normally.

5. Parallel computations for FMO evaluation

The process of fluence map optimization for TMI is a highly computationally intensive process because the objective function and gradient evaluations require the summation of the dose delivered from $\approx 75{,}000$ beamlets (for 30 beams) to $\approx 650{,}000$ voxels. These doses are then penalized accordingly.

Despite the number of computations, there is a need for the ability to generate IM-TMI plans using a large number of beams. One reason is that it allows us to validate our earlier plans: by solving the FMO problem with all possible beams, we can determine the best possible treatment plan for TMI to use as a benchmark. Another reason is that a transition to arc-based treatments will require dose calculations at a far larger number of beams than step-and-shoot treatments.

Parallelization of the objective function and gradient evaluation allows us to counteract the increase in computation time associated with large numbers of beams. Suppose that we wish to calculate the objective function for a set of bixels corresponding to n beams. Suppose also that Q processors are available to us, where Q divides evenly into n. Then to calculate the total dose delivered, we can divide up the set of n beams into Q subsets, each consisting of n/Q beams. We can then, in parallel, calculate the contribution of each subset to the total dose, collect the contributions and add them up to obtain the overall dose to each voxel. To calculate the actual objective function, we penalize the overall voxel dose in a serial manner. The gradient can be similarly calculated. The advantage of splitting up the beams into Q subsets is that each subset consists of n/Q beams, so we only need to transfer n/Q dose deposition matrices to each worker node. If we were to split the dose calculation by voxel, or subsets of voxels as in [3], we would need to transfer matrices for all n beams.

In addition to the notation defined earlier, let $\mathbf{z}$ represent the vector of z_{js} values for all $s \in S, j \in \{1, \ldots, v_s\}$; $\mathbf{D}^{(k)}$ represent the matrix of D_{ijs} values corresponding to a single beam and all voxels; $x^{(k)}$ represent the part of the x vector associated with beam $\boldsymbol{\theta}_k \in \Theta$; Q represent the total number of processors available for use; q refer to a particular processor in $\{1, \ldots, Q\}$, and to the subset of beams of Θ assigned to processor q; $\mathbf{z}^{(q)}$ represent the voxel dose contribution of the qth beam subset; $\mathbf{g}^{(q)}$ represent the piece of the gradient associated with the qth beam subset; $\mathbf{h}^{(k)}$ represent the piece of the gradient associated with beam $\boldsymbol{\theta}_k \in \Theta$; y_{js} represent an auxiliary scalar associated with voxel j in structure s; and $\mathbf{y}$ represent the column vector of y_{js} values for all $s \in S, j \in \{1, \ldots, v_s\}$.

Suppose that $\Theta = (\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \ldots, \boldsymbol{\theta}_n)$, and subset number q of Θ consists of beams $\boldsymbol{\theta}_{L_q}, \boldsymbol{\theta}_{L_q+1}, \ldots, \boldsymbol{\theta}_{U_q-1}, \boldsymbol{\theta}_{U_q}$. We can then define the voxel contribution of the qth beam subset as $\mathbf{z}^{(q)} = \sum_{k=L_q}^{U_q} \mathbf{D}^{(k)} x^{(k)}$. If the subsets numbered 1 to Q comprise the entire set of beams Θ, we can calculate the overall voxel dose as $\mathbf{z} = \sum_{q=1}^{Q} \mathbf{z}^{(q)}$, i.e., summing up the voxel dose contributions of each beam subset to obtain the overall voxel doses associated with Θ.

We can express the gradient calculation similarly. We let $\mathbf{g}^{(q)}$ be the part of ∇F corresponding to beam subset number q, so that we can express the gradient as $\nabla F = [\mathbf{g}^{(1)}\ \mathbf{g}^{(2)}\ \dots\ \mathbf{g}^{(Q)}]$. To define $\mathbf{g}^{(q)}$, we first let $\mathbf{y}$ be an auxiliary column vector, defined as $\mathbf{y} = [y_{js}]_{s \in S, j \in \{1,\dots,v_s\}}$, with

$$y_{js} = \frac{1}{v_s}\left[-\underline{w}_s \underline{p}_s (T_s - z_{js})_+^{\underline{p}_s - 1} + \overline{w}_s \overline{p}_s (z_{js} - T_s)_+^{\overline{p}_s - 1}\right], \qquad (7)$$

where $(\cdot)_+$ indicates $\max\{\cdot,0\}$.

If subset number q of Θ consists beams $\boldsymbol{\theta}_{L_q}, \boldsymbol{\theta}_{L_q+1}, \dots, \boldsymbol{\theta}_{U_q-1}, \boldsymbol{\theta}_{U_q}$, we can then define $\mathbf{g}^{(q)} = [\mathbf{h}^{(L_q)}\ \mathbf{h}^{(L_q+1)}\ \dots\ \mathbf{h}^{(U_q-1)}\ \mathbf{h}^{(U_q)}]$, where $\mathbf{h}^{(k)} = \mathbf{y}^T \mathbf{D}^{(k)}$.

We now formally define the algorithms used to execute these parallel operations. Algorithm 3 describes how to calculate the voxel dose of the qth beam subset ($\mathbf{z}^{(q)}$), which is then used by Algorithm 4 to calculate the complete voxel dose $\mathbf{z}$. Similarly, Algorithm 5 describes how to calculate the piece of the gradient associated with the qth beam subset ($\mathbf{g}^{(q)}$), which is used by Algorithm 6 to obtain the complete gradient ∇F.

Algorithm 3. Calculate voxel dose $\mathbf{z}^{(q)}$ of beam subset q on processor q.

```
Require Parameter $Q$ (total number of processors), parameter $n$ (total number of beams), processor number $q$
1:  Initialize $z_{js}^{(q)} = 0$ for all $s \in S, j \in \{1,\dots v_s\}$
2:  Set $L = n(q-1)/Q+1$, $U = nq/Q$
3:  for $k=L$ to $U$ do
4:    Load $\mathbf{D}^{(k)}$ from hard disk
5:    Set $\mathbf{z}^{(q)} = \mathbf{z}^{(q)} + \mathbf{D}^{(k)} x^{(k)}$
6:  end for
```

Algorithm 4. Calculate complete voxel dose $\mathbf{z}$ (parallel).

```
Require Parameter $Q$ (total number of processors), parameter $n$ (total number of beams)
1: Initialize $z_{js} = 0$ for all $s \in S, j \in \{1,\dots,v_s\}$
2: for $q = 1$ to $Q$ do
3:   Execute Algorithm 3 on processor $q$ with parameters $Q$, $n$
4: end for
5: Wait for Algorithm 3 to finish on processors $1,2,\dots,Q$
6: for $q = 1$ to $Q$ do
7:   Set $\mathbf{z} = \mathbf{z} + \mathbf{z}^{(q)}$
8: end for
```

Algorithm 5. Calculate partial gradient $\mathbf{g}^{(q)}$ associated with beam subset q on processor q.

```
Require Parameter $Q$ (total number of processors), parameter $n$ (total number of beams), parameter $\mathbf{z}$ (total voxel dose), processor number $q$
1:   Initialize $y_{js} = 0$ for all $s \in S, j \in \{1,\dots,v_s\}$
2:   for $k = 1$ to $n$ do
3:     for all $s \in S$ do
4:       for $j = 1$ to $v_s$ do
5:         if $z_{js} < T_s$ then
6:           Set $y_{js} = -\frac{1}{v_s}\underline{w}_s \underline{p}_s (T_s - z_{js})^{\underline{p}_s - 1}$
7:         else
8:           Set $y_{js} = \frac{1}{v_s}\overline{w}_s \overline{p}_s (z_{js} - T_s)^{\overline{p}_s - 1}$
9:         end if
10:      end for
11:    end for
12:  end for
13:  Set $L = n(q-1)/Q+1$, $U = nq/Q$
14:  Initialize $g_i^{(q)} = 0$ for all $i \in \bigcup_{k=L}^{U} B_{\boldsymbol{\theta}_k}$
15:  for $k=L$ to $U$ do
16:    Load $\mathbf{D}^{(k)}$ from hard disk
17:    Set $\mathbf{h}^{(k)} = \mathbf{y}^T \mathbf{D}^{(k)}$
18:  end for
19:  Set $\mathbf{g}^{(q)} = [\mathbf{h}^{(L)}\ \mathbf{h}^{(L+1)}\ \dots\ \mathbf{h}^{(U-1)}\ \mathbf{h}^{(U)}]$
```

Algorithm 6. Calculate complete gradient ∇F (parallel).

```
Require Parameter $Q$ (total number of processors), parameter $n$ (total number of beams)
1: for all $q = 1$ to $Q$
2:   Execute Algorithm 5 with parameters $q$, $n$ on processor $q$
3: end for
4: Wait for Algorithm 5 to finish on processors $1,2,\dots,Q$
5: Set $\nabla F = [\mathbf{g}^{(1)}\ \mathbf{g}^{(2)}\ \dots\ \mathbf{g}^{(Q-1)}\ \mathbf{g}^{(Q)}]$
```

6. Results

All of the algorithms presented were implemented in MATLAB and tested on a 64-bit, 32-node CentOS cluster, with each node having 8 GB of memory and eight AMD Opteron 2354 2.2 GHz processors. All parallelizations were performed using the MATLAB Parallel Computing Toolbox, with each task to be performed in parallel specified using the `createTask` function and all of the tasks assigned to a specific job created using the `createJob` function.

A single patient case provided by The Princess Margaret Hospital (Toronto, ON, Canada) under research ethical clearance was used to demonstrate the performance of the algorithms. Table 1 displays the number of voxels in each structure that was considered for this patient case. Only the bone marrow from the pelvis up was considered, as the legs do not contain any critical structures and can be irradiated separately. The total set of beams $\mathcal{B}$ consisted of 396 beams. These beams were obtained by allowing the gantry angle to range from 0° to 350° in 10° increments and consider 11 equispaced couch-z translations from just below the patient's pelvis to the top of the patient's head. The number of bixels in each beam ranged from 1759 to 3146, with a

Table 1
Number of voxels in each structure. The bone marrow is the planning target volume (PTV).

Structure	Number of voxels
Left lung	29,274
Right lung	40,248
Spinal cord	1856
Heart	18,746
Left kidney	5638
Right kidney	5263
Liver	74,986
Stomach	12,195
PTV	331,715
Parotid and submandibular glands	959
Esophagus	1138
Bowel	119,557
Bladder	4662
Oral cavity	4926
Left eye	130
Right eye	117
Total	651,410

mean of 2469 and a median of 2464. The only bixels allowed in each beam were those which were able to deliver positive dose to at least one target voxel; as the target is essentially the entire skeletal structure, each beam contains a large number of usable bixels.

6.1. Computational results

Each type of line search procedure was tested on 50 different randomly generated 30-beam solutions. The initial bixels of all of the beams were set to 0.3. The percentage change tolerance was 0.01 and σ was set to 0.0001. These values were chosen as our preliminary testing indicated that they provided the best performance for all of the line search procedures. For the reduced step projected gradient implementation, the step length is reduced after $m=3$ iterations to a new step length of $\overline{R}=12.5$. For the backtracking, reduced step, golden section, dichotomous and quadratic interpolation implementations, the initial step length was $R_0=50$. For the backtracking and reduced step implementations, the scale factor applied in each line search iteration was $\beta=0.25$. For the dichotomous search, the distinguishability constant δ was set to $0.5/\|\nabla F(x^{(i)})\|$ in each iteration. For the forward line search implementation, the initial step length was $R_0=3$ and the scale factor applied in each line search iteration was $\gamma=8$. These procedure-specific parameter values were selected as our preliminary testing indicated that they provided the best balance between computational time and FMO value for their respective procedures. No parallelization was employed in the implementations of these line search methods.

Table 2 shows statistics related to the number of projected gradient outer loop iterations, the total execution time and final objective function value for all six implementations over all 50 starting points. On average, the reduced step projected gradient yields the lowest final objective function values while the golden section projected gradient yields the highest values. The golden section and dichotomous strategies both have the highest execution times, likely because these algorithms decrease the step length relatively slowly compared to the other methods.

Each warm-start procedure was tested on the sequential cycling Add/Drop (SCAD) algorithm, described in [1]. In the resulting Add/Drop implementations, the neighborhood search was parallelized by assigning each FMO evaluation to a separate processor/node. We note that this form of parallelization is different from the type of parallelization described in Section 5; it was done to speed up the execution of the Add/Drop method, because the FMO calculations for solutions which neighbor the current solution are independent of each other and can all be performed at the same time. We emphasize that this problem-based parallelization is not the focus of this particular computational experiment. No parallelizations were applied at the FMO level.

The neighborhood sizes used for each warm-started Add/Drop implementation were the same as those used in [1]. For the cold-start mode of FMO initialization, κ was set to 0.3 (the same value used for the line search procedures). The optimization problem WARM-LS for the warm-start using least-squares mode was solved using the MATLAB command `lsqlin` using the default parameters. These three implementations were then executed on 50 randomly chosen 30-beam starting points and were allowed to run for 12 h, as this is typically how much time would be available in a clinical environment. Table 3 presents computational results. For each starting point, the projected gradient run times of all of the sets of beams sampled over the course of the Add/Drop execution were averaged to obtain an average projected gradient time; the average projected gradient time statistics are obtained from these (50) average projected gradient

Table 3
Add/Drop iteration number, projected gradient (P.G.) time, average total FMO time and final FMO value statistics for the three types of warm-start techniques.

Statistic	Cold-start	Warm-start (averaging)	Warm-start (least-squares)
Add/Drop iterations			
Mean	11.8	37.8	41.7
St. dev.	2.5	3.4	2.5
Minimum	7	28	36
Maximum	17	49	48
PG time (min)			
Mean	58.9	16.5	8.7
St. dev.	12.5	1.4	0.5
Minimum	37.5	12.4	7.4
Maximum	92.4	19.4	10.0
AT FMO time (min)			
Mean	58.9	16.5	13.7
St. dev.	12.5	1.4	0.6
Minimum	37.5	12.4	12.4
Maximum	92.4	19.4	15.0
$\mathcal{F}(\Theta^*)$			
Mean	13,571.9	11,139.8	10,799.7
St. dev.	2684.6	1001.1	971.2
Minimum	11,216.0	10,169.4	9992.9
Maximum	21,628.5	16,378.4	15,994.9

Table 2
Projected gradient (PG) iteration numbers, execution times and final objective function value statistics for the six types of line search techniques.

Statistic	Backtracking	Reduced step	Forward	Golden section	Dichotomous	Quad. interp.
PG iterations						
Mean	12.2	13.0	13.9	10.3	10.7	9.0
St. dev.	4.1	4.5	5.8	4.7	4.1	2.2
Minimum	3	3	2	2	2	5
Maximum	20	22	23	22	24	15
Time (min)						
Mean	54.4	48.5	69.1	55.8	61.1	40.1
St. dev.	19.1	15.8	26.3	29.4	28.9	11.4
Minimum	11.8	12.1	13.9	9.7	10.0	21.3
Maximum	96.5	83.0	115.6	131.5	164.4	72.6
$F(x^*)$						
Mean	17,328.0	17,310.0	17,765.5	20,875.0	19,670.5	18,789.2
St. dev.	5763.9	5414.4	6753.9	6363.2	5430.3	2792.5
Minimum	11,663.1	12,009.4	11,662.1	14,301.4	13,360.4	14,872.1
Maximum	35,339.0	35,339.0	42,207.6	42,220.2	36,904.4	25,445.6

times. Similarly, for each starting point, the total FMO time (which is the projected gradient run time plus any pre-projected gradient warm-start procedure) of all of the sets of beams sampled over the course of the Add/Drop execution were averaged to obtain an average total FMO time; the average total FMO time statistics are obtained from these (50) average total FMO times.

Table 3 shows that the two warm-start techniques lead to both significantly lower projected gradient (PG) run times, which is a contributing factor to the significantly lower average total FMO times. The averaging warm-start procedure requires a negligible amount of time (less than 0.25 s), while the least-squares warm-start procedure on average takes 5.0 min with a standard deviation of 0.4 min. This is reflected in the Average Total FMO Time section of Table 3, which shows that in terms of average total FMO time, the warm-start using least-squares method is only slightly better than the warm-start using averaging method, which explains the similarity in Add/Drop iterations and final optimal FMO values.

The parallelism-enhancements described in Section 5 were used in an implementation of projected gradient with a backtracking line search with initial step length $R_0 = 50$, scale factor of $\beta = 0.25$, Armijo condition constant $\sigma = 0.0001$ and percentage change tolerance $\varepsilon = 10^{-6}$. These parameters are the same as those used for the line search testing described earlier in this section, with the exception of ε, which was greatly reduced in order to ensure that the algorithm does not terminate too early. This implementation was used to solve the FMO problem using all 396 beam orientations in $\mathcal{B}$ to obtain the best treatment possible with our model. Each objective function and gradient evaluation was split among 11 nodes; this number was chosen as it divides evenly into 396 beams and from our preliminary testing, this was the minimum number of nodes required to ensure that the 396-beam solution would be very strong after 6 h of execution. (This is particularly important if the solution from this algorithm is to be used as a precursor to an algorithm for generating arc therapy treatments, as discussed at the beginning of Section 5.) Only one 2.2 GHz processor was used from each node and each node had 8GB of memory available. The total number of bixels in the plan was 977,601 while the total number of voxels was 651,410. The projected gradient algorithm was allowed to execute for just over 24 h.

Fig. 1 shows that the greatest amount of change in the objective function occurs within the first 5 h. Similarly, Fig. 1 also shows that the greatest amount percentage change in the objective function occurs within the first 5 h, indicating that the projected gradient algorithm can obtain a good solution with 396 beams in 5 h.

6.2. Treatment plan quality

In order to judge treatment quality beyond the simple value of $\mathcal{F}(\Theta)$, we use a dose volume histogram (DVH), a graphical representation of the dose delivered to each structure. A DVH is a graph for a particular structure which indicates, for each level of dose, what percentage of the structure's volume receives that amount of dose or higher. The TMI treatment requirements are that at least 95% of the bone marrow volume must receive at least 12 Gy and a maximum of 25 Gy, and at most 20% of the bone marrow can receive more than 20 Gy. Further, the majority of each organ volume should receive 8 Gy or less.

Fig. 2 shows the reduced step line search solution for a single starting point. The solution achieves fairly high target coverage (95% of the target volume receives at least 11.5 Gy) and meets the target overdose constraints. Most organs meet the treatment criterion. The most overdosed organs are the stomach, bowel, bladder and spinal cord; although it is difficult to control overdosing in the spinal cord [1], it may be possible to lower the overdosing in the stomach, bowel and bladder with more projected gradient iterations.

Fig. 3 shows DVHs of representative final beam solutions obtained using the cold-start and the warm-start using least-squares methods. The warm-start using averaging solution is not shown as it is very similar to the warm-start using least-squares solution. From these DVHs, we can see that the warm-start methods are able to achieve better plan quality than the cold-start method. In particular, we can see that the warm-start methods lead to lower median doses in the critical organs while achieving slightly higher dose in the bone marrow (indicated as "hemiPTV" in the DVHs). We can also see that the tails of a number of organ DVH curves drop off at a slower rate in the cold-start plan. Furthermore, we can see that a larger volume of the bone marrow receives more than 20 Gy in the cold-start plan than in both the warm-start plans, which is an important consideration as extensive overdose to the bone marrow can lead to fibrosis, which can detrimentally affect the success of the subsequent bone marrow transplant. Another important difference is that both of the warm-start plans achieve satisfactory target dose; the warm-start using least-squares plans ensures that 95% of the bone marrow receives at 12.7 Gy, while the cold-start plan can only ensure that 95% of the bone marrow receives at least 9.8 Gy.

Fig. 4 shows the 396-beam solution at 6 h, 12 h and shortly after 24 h of parallelism-enhanced projected gradient execution. From these DVHs, we gain a number of insights. First, the DVHs of the 396-beam solution are qualitatively similar to those of the least-squares warm-start 30-beam solution. In particular, of all of

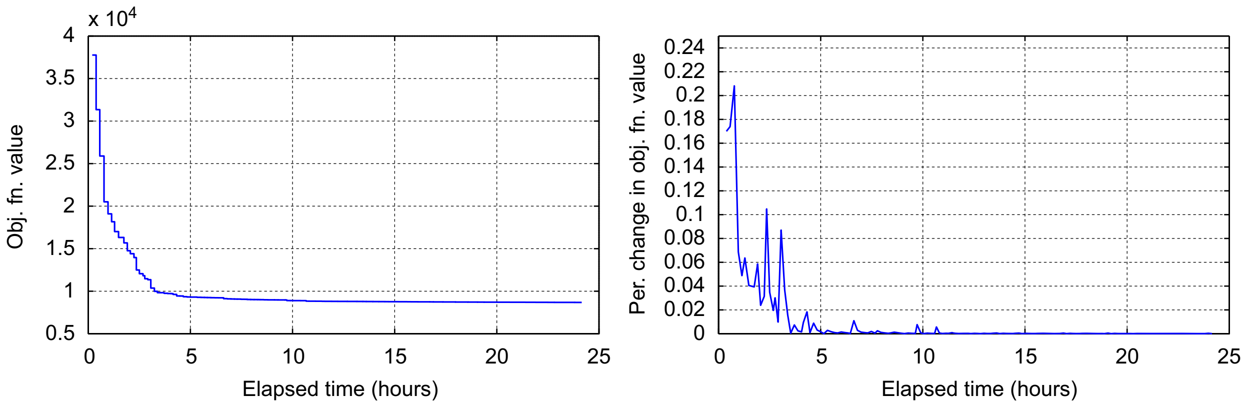

Fig. 1. Plots of objective function value versus time (left) and percentage change in objective function value versus time (right) for projected gradient using parallelized objective function and gradient evaluation for the 396 beam solution. Each objective function and gradient evaluation was split across 11 processors in the manner described in Section 5.

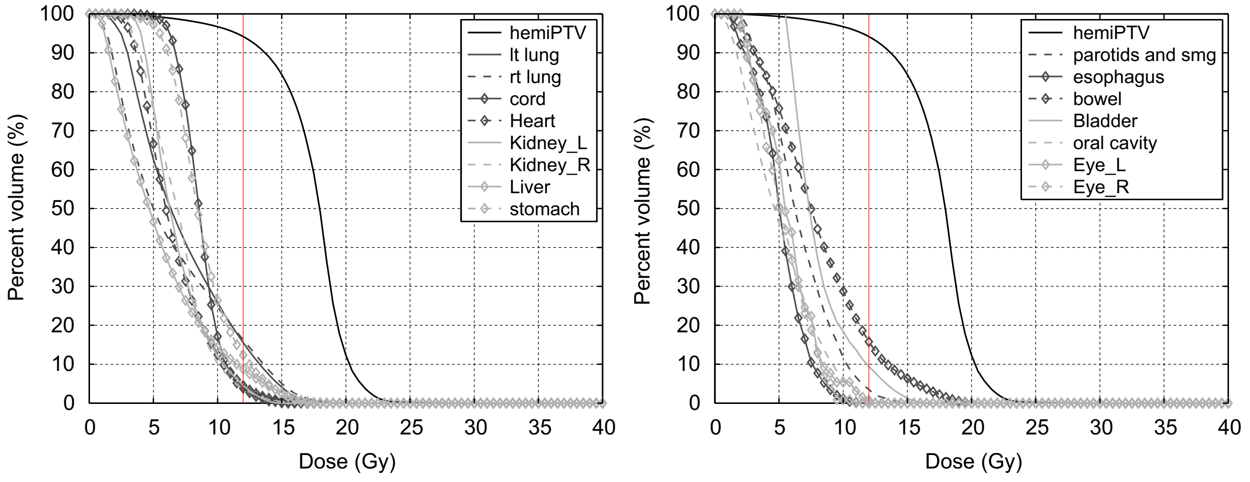

Fig. 2. DVHs of a 30-beam solution, with bixels optimized using the projected gradient algorithm with the reduced step backtracking line search strategy (described in Section 3).

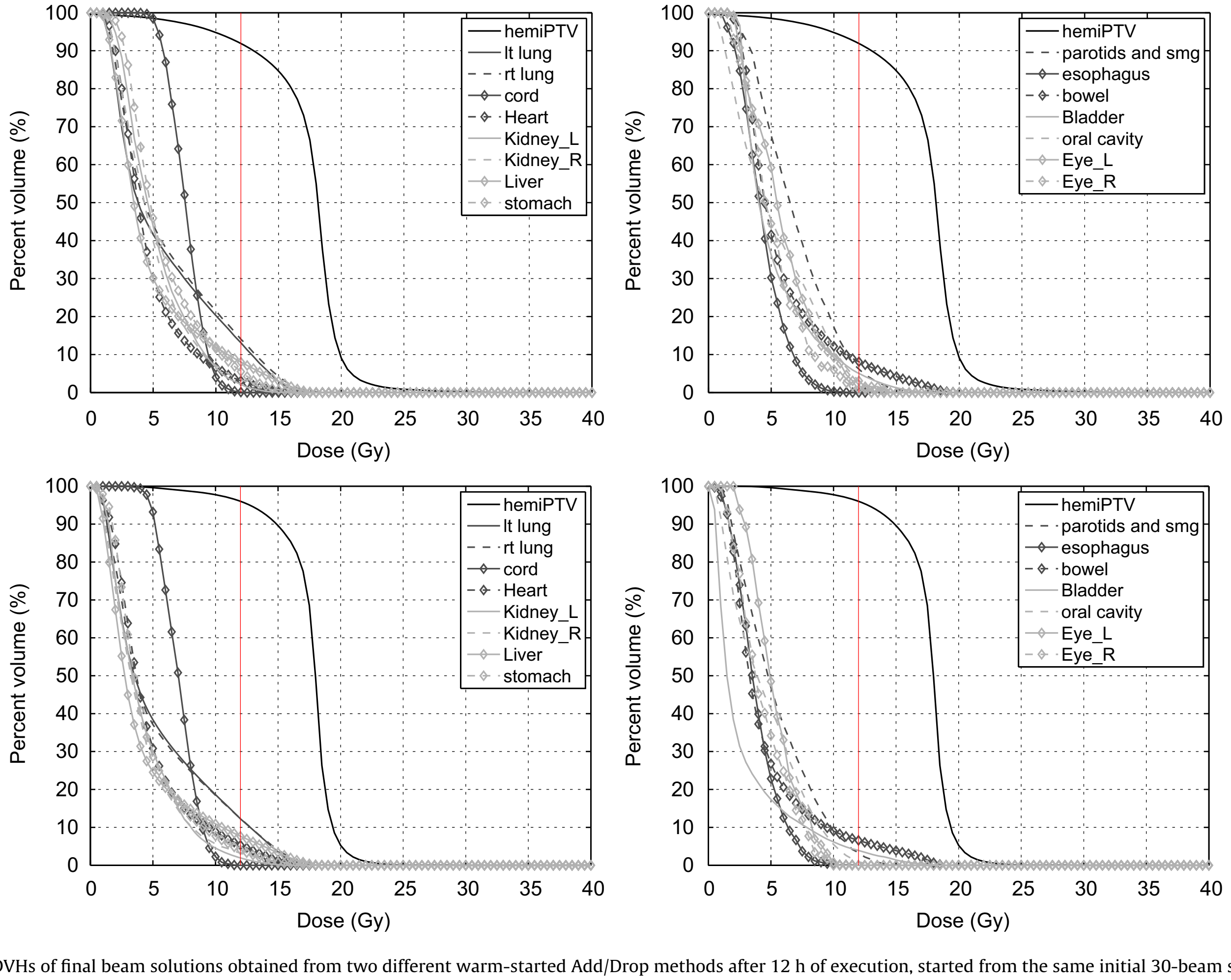

Fig. 3. DVHs of final beam solutions obtained from two different warm-started Add/Drop methods after 12 h of execution, started from the same initial 30-beam solution. Top: cold-start solution and bottom: warm-start using least-squares solution.

the critical organs, the spinal cord receives the most dose; after the spinal cord, the lungs receive the next highest amount of dose, with just under 20% of the volumes of the left and right lungs receiving more than 10 Gy. With regard to improvement, there is an increase in dose to the target—in the 30-beam plan, 95% of the target volume receives more than 12.7 Gy, while the same percentage of the target volume in the 396-beam plan receives more than 13.5 Gy. There are also slight improvements to the dose received by many critical organs. Comparing the 396-beam solutions at the 6-, 12- and 24-h marks, we can see that there are very small changes in the dose received by many organs and the bone marrow (95% of the target volume receives 13.1 Gy or more at 6 h, 13.4 Gy or more at 12 h and 13.5 Gy in the final plan). These small changes match the behaviour shown in the plots

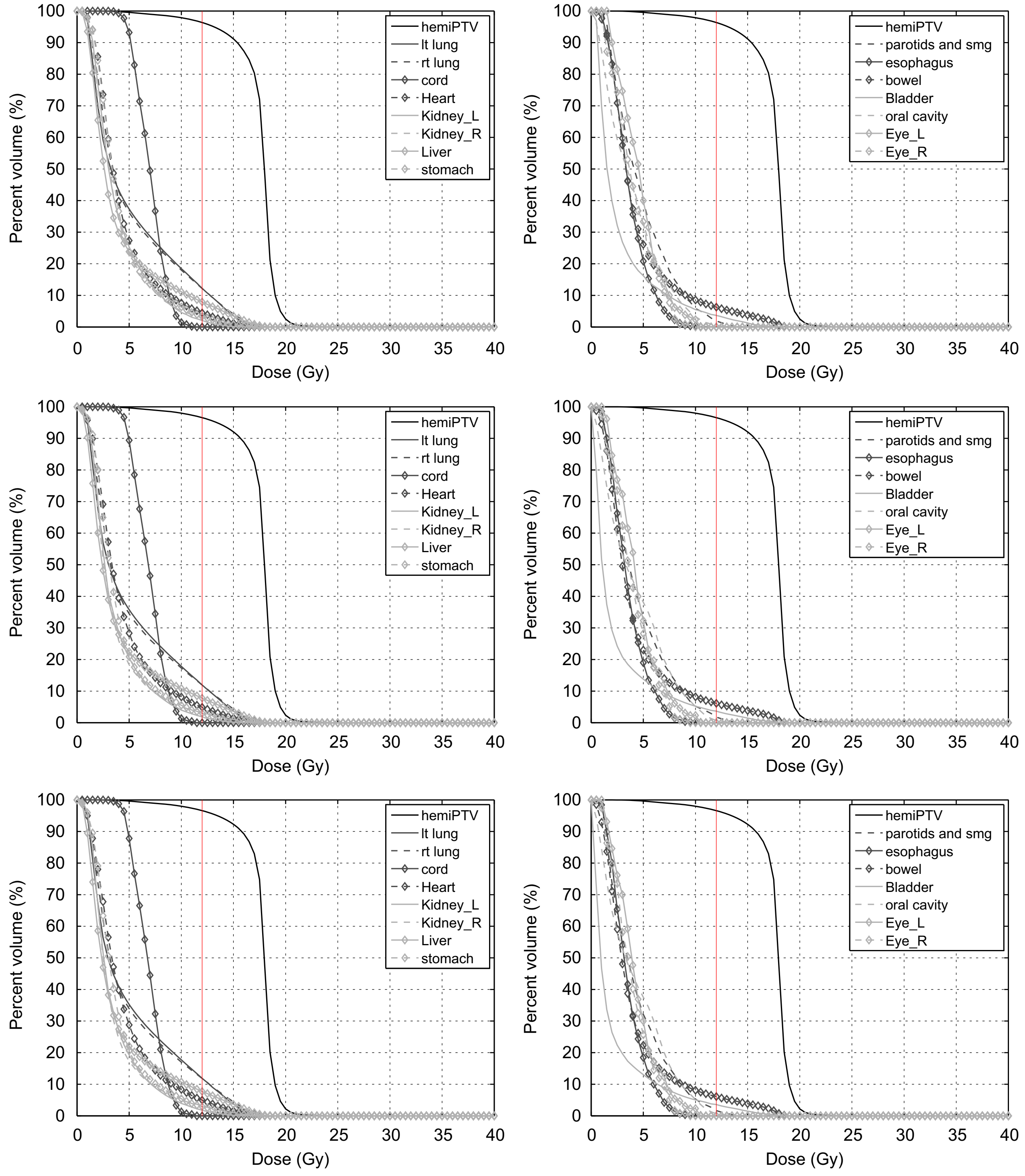

Fig. 4. DVHs of 396-beam solution after 6 h (top), 12 h (middle) and 24 h (bottom) of projected gradient execution time. Each objective function and gradient evaluation was split across 11 processors in the manner described in Section 5.

of the previous section, which show that most of the change in the objective function value occurs in the first 5 h, and that the rate of change in the objective function is greatly reduced after 6 h. It is interesting to observe that the 396-beam solution obtained after 6 h is just as good as the warm-start using least-squares 30-beam solution in Fig. 3 with respect to both target dose and organ sparing. This suggests that this type of parallelized projected gradient algorithm has strong potential to be used to generate fixed beam solutions that could then be used to develop arc therapy treatment plans in a clinically realistic timeframe.

7. Conclusion

Several methods for improving the computational tractability of TMI using IMRT have been presented. The reduced step back-tracking line search strategy provides the best results of the six line search methods, both in terms of computation time and final objective function value. However, the clinical improvement that this strategy exhibits over the ordinary backtracking line search strategy is only modest.

Both warm-start techniques led to a significant reduction in computation time over the standard cold-start method, and also led to significantly lower final FMO values and improved treatment plans. Both of these warm-start methods can be readily implemented in MATLAB and can be applied to FMO for other treatment planning problems, not just TMI. Also, these methods help to highlight the versatility of our FMO formulation and why it may be desirable to use the exact FMO value (as opposed to an approximation or a different figure of merit) as the quality of a set of beams.

Finally, from our parallelism-enhanced FMO results, we have discovered that it is possible to obtain strong 396-beam solutions with 11 processors after just 6 h of execution. Furthermore, we have also discovered that the 30-beam Add/Drop solutions compare quite favorably to the 396-beam solutions we have presented here, highlighting the power of the Add/Drop algorithm for IM-TMI treatment planning.

In future work, we could consider other kinds of warm-start procedures to speed up FMO evaluation in Add/Drop. In particular, we could consider warm-start schemes where instead of directly reusing the bixels of $\Theta \backslash \{\boldsymbol{\theta}_k\}$, we modify those bixels in response to the dose deposition coefficients of the new beam, $\tilde{\boldsymbol{\theta}}_k$. It would also be interesting to more closely examine the structure of our FMO problem and the structure of the D_{ijs} values to determine if it is possible to place some kind of guarantee on the quality of the initial set of bixels obtained by the warm-start using averaging and warm-start using least-squares techniques (e.g., if the D_{ijs} of $\boldsymbol{\theta}_k$ and $\tilde{\boldsymbol{\theta}}_k$ differ by some amount, then what kind of objective function value can we expect for the initial bixel values $\tilde{x}$?).

The implementation of projected gradient using the parallelized objective function and gradient algorithms has yielded some interesting results about where our previous 30-beam solutions stand. An intriguing observation that we made about this algorithm was that the solution obtained after 6 h generally compared quite favorably to the 30-beam solution obtained after 12 h. It would be interesting to explore whether these solutions could in some way be used to identify which beams in the entire collection are better than others and guide the design of plans using smaller numbers of beams.

Finally, as stated earlier, arc therapy is a promising alternative to conventional IMRT for TMI. The parallelism-enhanced projected gradient algorithm could constitute a first step in designing an arc therapy plan; the next step would involve designing an algorithm to determine an appropriate “route” to take through the beams, and to fill in the bixels along the path in some optimal way.

References

[1] Mišić VV, Aleman DM, Sharpe MB. Neighborhood search approaches to non-coplanar beam orientation optimization for total marrow irradiation using IMRT. European Journal of Operational Research 2010;205(3):522–7.
[2] Romeijn HE, Ahuja RK, Dempsey JF, Kumar A. A new linear programming approach to radiation therapy treatment planning problems. Operations Research 2006;54(2):201–16.
[3] Men C, Gu X, Choi D, Majumdar A, Zhen Z, Mueller K, et al. GPU-based ultrafast IMRT plan optimization. Physics in Medicine and Biology 2009;54: 6565–73.
[4] Nazareth DP, Brunner S, Jones MD, Malhotra HK, Bakhtiari M. Optimization of beam angles for intensity modulated radiation therapy treatment planning using genetic algorithm on a distributed computing platform. Journal of Medical Physics 2009;34(3):129–32.
[5] Aleman DM, Romeijn HE, Dempsey JF. A response surface approach to beam orientation optimization in intensity modulated radiation therapy treatment planning. INFORMS Journal on Computing: Computational Biology and Medical Applications 2009;21(1):62–76.
[6] Trofimov A, Rietzel E, Lu H-M, Martin B, Jiang S, Chen GTY, et al. Temporo-spatial IMRT optimization: concepts, implementation and initial results. Physics in Medicine and Biology 2005;50:2779–98.
[7] Thieke C, Bortfeld T, Niemierko A, Nill S. From physical dose constraints to equivalent uniform dose constraints in inverse radiotherapy planning. Medical Physics 2003;30(9):2332–9.
[8] Fox C, Romeijn HE, Lynch B, Men C, Aleman DM, Dempsey JF. Comparative analysis of 60 Co intensity-modulated radiation therapy. Physics in Medicine and Biology 2008;53:3175–88.
[9] Ólafsson A, Jeraj R, Wright SJ. Optimization of intensity-modulated radiation therapy with biological objectives. Physics in Medicine and Biology 2005;50: 5357–79.
[10] Bertsekas DP. On the Goldstein–Levitin–Polyak gradient projection method. IEEE Transactions on Automatic Control 1976;21(2):174–84.
[11] Nocedal J, Wright SJ. Numerical optimization. Springer Science; 2006.
[12] Bazaraa M, Sherali H, Shetty C. Nonlinear programming: theory and algorithms. Wiley-Interscience; 2006.
[13] Zhang H, Meyer R, Wu J, Naqvi S, Shi L, D'Souza W. A two-stage sequential linear programming approach to IMRT dose optimization. Physics in Medicine and Biology 2010;55:883–902.
[14] D'Souza WD, Zhang HH, Nazareth DP, Shi L, Meyer RR. A nested partitions framework for beam angle optimization in intensity-modulated radiation therapy. Physics in Medicine and Biology 2008;53:3293–307.