Solving the Brachytherapy Seed Localization Problem Using Geometric and Linear Programming Techniques

Vikas Singh1  Lopamudra Mukherjee1
Jinhui Xu1  Kenneth R. Hoffmann2
1Department of Computer Science and Engineering
State University of New York at Buffalo
{vsingh, lm37, jinhui}@cse.buffalo.edu 
2Department of Neurosurgery
State University of New York at Buffalo
kh9@buffalo.edu

ABSTRACT
We propose a technique to solve the Brachytherapy Seed Localization problem in prostate brachytherapy. Our algorithm is based on novel geometric approaches to exploit the special structure of the problem combined with a number of key observations which help us formulate it as an Integer Program. We solve the equivalent linear program and present a variant of randomized rounding to yield an integral solution to our problem. The algorithm is efficient and performs extremely well in practice. We discuss in detail the underlying theory and performance evaluations based on our implementation.

Categories and Subject Descriptors: F.2.2 [Analysis of Algorithms and Problem Complexity]: Nonnumerical Algorithms and Problems - Geometrical problems and computations; J.2 [Life and Medical Science]: Miscellaneous.

General Terms: Algorithms.
Keywords: Brachytherapy Seed Localization; Geometric Optimization; Linear Programming.

1. INTRODUCTION
Prostate cancer is the most common form of cancer in men, accounting for 33,000 deaths each year in the United States alone (http://hsu.com/main/prostate_cancer.htm). One of the most common methods of treatment employed is low dose rate permanent seed brachytherapy [1], in which brachytherapy seeds are first implanted inside the soft tissue (tumor) during a surgical procedure. An optimal spatial arrangement of the seeds will yield the best possible radiation dose distribution for the tumor. Clinicians then use C-arm fluoroscopic cameras to image the 3D seed configuration on multiple 2D imaging planes. These 2D projection images of the seed configuration are carefully analyzed by professionals to calculate the dosimetric deviations from the optimal implant plan. The

1This work was supported in part by NIH Grant HL 52567 and Toshiba Medical Systems Corporation.

success of a brachytherapy procedure depends critically on being able to tailor the treatment based on the patient’s anatomy. Researchers have reported that more than 60% of practitioners use some form of qualitative implant analysis in the operating room [2]. The analysis relies on, among other factors, the estimation of the 3D seed positions, based on 2D information available from the images. For this reason, several research efforts have focussed on reconstructing a 3D configuration of the implanted seeds. It is expected that accurate reconstruction and seed localization techniques will mitigate the inaccuracies due to human estimation.

Reconstruction of an unknown 3D object from two or more 2D projection views is a widely studied problem in medical imaging. In general, information about the geometry relating the coordinate systems of two or more views and known corresponding points are used to reconstruct the 3D object. For our problem, in order to determine the 3D seed-configuration, the correspondence of the 2D seeds needs to be determined from the X-ray images2. Observe that if the relative geometry between the various coordinate systems is known accurately, corresponding seed points in the images can be determined -- the projection lines (a 3D line from the 2D image point to the X-ray source of its respective coordinate system) of two corresponding points will intersect, yielding the representative 3D point [5]. Most commercial imaging systems, however, suffer from certain electro-mechanical limitations. In case of a small configuration of brachytherapy seeds, the relative error introduced due to a small geometry error could be significant. Secondly, the acquired images have some distortion or segmentation errors (together referred to as image error). Such errors make the task of the determination of corresponding seeds in the images rather challenging. As can be expected, a reconstruction procedure with incorrect correspondence information will be inaccurate and hence unreliable for clinical purposes. In this paper, we propose efficient techniques for solving this brachytherapy seed localization problem (BSL, for short).

Due to its importance in clinical diagnosis and treatment, the BSL problem has been studied by a number of researchers in the recent past. One of the first efforts was made by Siddon and Chin [6] to mathematically formulate the problem. A cost matrix data

---

1corresponding points are projections of the same 3D point on different imaging planes.

2In the Biplane Imaging Geometry Determination problem, the user indicates corresponding points based on landmarks in the images, the geometry is then calibrated based on the known correspondences [3, 4]. In our problem, however, the geometry cannot be refixed prior to the determination of the correspondences because the correspondences are not known a priori.
structure was constructed followed by an exhaustive brute force search to find the lowest cost solution. Subsequently, researchers have proposed several heuristics to speed-up the search process, for example, a greedy choice using several different cost metrics [7], performing a brute force search on a subset [8, 9], and using epipolar constraints to impose some forms of partial ordering [10]. Other AI heuristics like simulated annealing (see [11], [12]) have also been used with some success. Recently, Jain et. al. [2] proposed an interesting approach which first constructs a multi-partite graph and then obtains a solution by solving several instances of bipartite graphs. In the presence of errors, however, local choices in each bipartite graph may no longer be the best (or even near-best) choice, when considered globally. In short, albeit significant progress in the last few years, the problem remains far from solved.

Part of the reason is that the BSL problem is closely related to several notorious NP-hard problems (e.g., Maximum Clique problem and k-Partite Matching problem), possibly indicating that obtaining an optimal solution in polynomial time is unlikely. Heuristics may perform well on some data sets but may yield poor results on other instances of the problem. It seems that to come up with better solutions for this problem, a rigorous theoretical analysis needs to be undertaken (see article by Yu [13]). However, a quest for stronger theoretical bounds could easily take the approach away from the realm of practical feasibility. For instance, approximation algorithms for the k-partite matching problem yield theoretically good approximation ratios of $O\left(\frac{k}{\ln k}\right)$ [14], but are hardly suitable for most practical applications. A key challenge lies in avoiding these pitfalls.

We tackle the BSL problem in a two phase manner: (1) We first match 2D points (correspondences) in the k imaging planes, $P_1, P_2, ..., P_k$, and then (2) reconstruct the 3D points based on the obtained matching information. The main focus of this paper is to address (1). We will also discuss one possible way to reconstruct the points of (2), using results of (1).

2. MAIN IDEAS

2.1 Formulation as a k-partite graph matching problem

The most natural way to think of the 2D point matching (i.e., determining corresponding point-tuples) is to view it as a minimum weight k-partite graph matching problem in a to-be-constructed graph $G$. A graph is multipartite if the set of vertices in the graph can be divided into non-empty subsets, which we call ‘parts’, such that no two vertices in the same part have an edge connecting them. Furthermore, a complete multipartite graph is such that any two vertices that are not in the same part have an edge connecting them. The problem of minimum weight matching in a multipartite graph is to find the edge set of smallest weight which ‘covers’ every vertex from a part exactly once. We consider each image plane $P_i$ as a ‘part’ in the graph $G$; the vertices in that part are representative of the 2D seed points on that imaging plane. Let the $i^{th}$ 2D projection point on the $j^{th}$ imaging plane be represented as a vertex, $v_{ij}$. In all, we have $k$ ‘parts’ in the graph $G$, with each part containing $n$ vertices. Since there may exist an edge in $G$ joining any pair of vertices not in the same part, $G$ could contain a total of $\binom{k}{2}n(k-1)$ edges. We desire a set of $n$ matchings, $M = \{M_1, M_2, ..., M_k\}$, such that each $M_r : r \in \{1, 2, ..., n\}$ is a $k$-tuple of the form $(v_{1j}, v_{2j}, ..., v_{kj})$, where $i$, $j$, and $q$ could be one of the $2D$ projection points in $P_1, P_2$, and $P_k$ respectively. These $k$-tuples should have no vertices in common with each other, ensuring that each 2D point participates in one and only one matching. This construction helps in formulating the BSL problem as a $k$-partite graph matching problem in a graph with a total of $kn$ vertices (see Figs. 1(a) and (b)). It is useful to point out that our algorithm is independent of the layout of the constructed graph. A random layout of the vertices on a 2D plane will be as effective (see Fig.1 (c)).

Once all 2D projection points on $P_1, P_2, ..., P_k$ (the sets of imaging planes) have been represented in the constructed graph $G = \{V,E\}$, we assign weights to the edges joining the vertices. Intuitively, the weight of an edge needs to reflect the likelihood that the two end vertices of the edge are a possible match. That is, if two points are constituents of a possible match, the edge in $G$ joining the corresponding vertices should have a small weight. To this end, let us consider two ‘projection lines’ [3], $l_1$ and $l_2$, for two known corresponding points on imaging planes, $P_1$ and $P_2$. A projection line is the 3D line that passes through the 2D image point (considered as a 3D point in its coordinate system) and the x-ray source of that coordinate system. If the situation were ideal (i.e., error free), the lines, $l_1$ and $l_2$, would intersect, and the intersection point would yield the actual 3D point. Clearly, the minimum distance between $l_1$ and $l_2$ would be 0, since they intersect. This leads us to believe that the distance between the projection lines of the two points would be useful as the weight of the edge joining them.

![Figure 1](image)

2.2 Preprocessing step

In this section, we discuss our preprocessing technique which exploits the implicit geometric information in the spatial orientation of the set of projection lines. Notice that in the previous graph construction step, we may add an edge between any pair of vertices that are not in the same part. However, this construction loses a great deal of geometric information inherent in the 2D images. For example, consider three imaging planes ($k = 3$), where three projection lines are oriented as shown in Fig. 2(a). It is obvious that the lines are mutually pair-wise close (the associated edges in graph $G$ may have a near-zero weight), but well-separated in space and enclose a triangle of a relatively large area. It is unlikely that
these three lines would correspond to the same 3D point. Hence, it becomes important to (a) encode this information in our algorithm and (b) remove the edges which are unlikely to be part of a match. The preprocessing phase serves these two purposes.

Let the total error introduced in the imaging system be upper bounded by a constant, \( \sigma \) (which can be determined from the precision of the imaging system). The ‘maximal’ error of the reconstructed 3D points, in the presence of an error, \( \sigma \), can be upper bounded by \( \delta = f(\sigma) \), where \( f(\cdot) \) is a function determined by the rotation, \( R \), and translation, \( t \) relating the imaging systems. Also for two projection lines corresponding to the same 3D point, their shortest 3D distance would be upper bounded by \( \delta \). In other words, in the presence of an error \( \sigma \), the 3D projection lines, move apart by a distance no larger than \( \delta \). This observation forms the basis for our preprocessing step: a ball of radius \( 2\delta \), sliding (sweeping) along each projection line starting from the origin or X-ray source and stopping at its corresponding 2D point, would hit (intersect) each projection line that is a likely candidate for a match. It may also hit other projection lines which are not matches; redundancy, however, can be tackled later. For the ball sliding along a projection line, say \( i \), the set of all other projection lines it hits, \( LH_i \), will be no larger than \( (k-1)\mu \). Clearly, the correct (unknown) matches for \( i \), are included in \( LH_i \). For our preprocessing, we first compute a sequence of \( LH_i \)s for each projection line, as the ball sweeps. This process is similar to a plane-sweep \([15]\); we sweep a ball from the x-ray source to its corresponding 2D image point. When some projection line enters (intersects) the ball along its sweep-path, we treat it as an event. The “ball-sweep-status” is updated to reflect this event. Similarly, when any projection line currently on the ball-sweep-status leaves the ball (ball moves away far enough), the ball-sweep-status is again appropriately updated. Once the sweep process for a particular projection line is completed (ball has reached the 2D image point), the sweep is evaluated. At any ‘event’ in the sweep, if the ball-sweep-status had more than \( k \) projection lines, a ball of radius \( 2\delta \) is placed in that spatial event position. In other words, when \( k \) unique projection lines intersect the ball at an event, it denotes a possible positioning for a 3D seed point. The ball-sweep process is repeated for each projection line and redundant ball positions are removed. The process yields a set of all possible ball positions \((\leq (n-1)^2(k(k-1))\), a set of less than \( (k-1)n \) \( LH_i \)s for each \( i \); the unknown 3D points are located inside \( n \) of these balls (see Fig. 2).

We create a comprehensive set of balls, \( B = B_1 \cup B_2 \cup \ldots \cup B_C = LH_1 \cup LH_2 \cup \ldots \cup LH_C \) where \( C \leq \frac{(n-1)^2(k(k-1))}{\mu} \). Each ball \( B_i \in B \) defines a possible location of 3D seed points. If a projection line hits a ball, \( B_i \), the corresponding vertex ‘belongs’ to the ball \( B_i \). If two vertices belong to the same ball, \( B_i \), their edge also belongs to \( B_i \). Next, we reduce the number of edges in the constructed graph \( G \) by using the preprocessing information.

For a pair of vertices not belonging to any single ball, their edge is removed from \( G \) (compare Fig. 1 (a) with (b)). We also create a binary matrix, \( B(E, C) \), where \( C = |C| \) is the total number of balls, \( B(i, c) = 1 \) if the edge \( e_i \) is in ball \( B_c : c \in [1, C], 0 \) otherwise.

2.3 Finding \( n \) complete mutually disjoint subgraphs

The objective now is to find \( n \) complete mutually disjoint subgraphs (CMDS, for short) of \( G = (V, E) \) such that (1) each has exactly \( k \) nodes, one from each imaging plane, and (2) each CMDS belongs entirely to one of the \( B \) balls. In other words, each CMDS should form a clique structure, with each CMDS having one and only one vertex from each imaging plane. Ideally, it is also desired that the sum of weights of all CMDS, given by summing over all its constituent edges, is minimized globally. It is apparent that each CMDS can be used to reconstruct the corresponding 3D point. In addition, the cliques are required to be disjoint because each 3D seed point has a unique projection. Since each CMDS (clique) will together denote one such 3D point, vertices are allowed to participate in only one CMDS, justifying the disjointness condition.

Notice that imposing the clique structure requirement on each CMDS and globally minimizing the CMDS (clique) weights are both necessary for a good solution. It makes intuitive sense that any set of 2D projection points constituting a \( k \)-tuple should be ‘close’ to each other in some sense. The main difficulty of this approach lies in how to efficiently find the \( n \) disjoint cliques. It is known that even the problem of finding one minimum weight maximum clique\(^3\) in a graph (i.e., the Maximum-Clique problem) is NP-hard. For our problem, several others factors like the geometric information, etc., further complicate the problem. To overcome these difficulties, our strategy is to exploit our observations about the special structure of the problem and design a practical combinatorial approach. More specifically, we use the constructed graph \( G \) (see Fig. 1 (c)) to formulate an Integer Program (IP, for short). Each of our observations are transformed into equivalent constraints in the formulation. In the subsequent sections, we discuss our IP formulation, followed by the relaxed LP version and finally our technique for rounding the LP solution to derive an integral solution for our problem.

2.4 Integer Programming Formulation

To facilitate the presentation, we will start with a brief description of the variables. We will then introduce the IP and discuss each constraint in detail.

\(^3\)A clique is a graph, such that it has an edge between every pair of vertices.
$G = (V,E)$ is the graph discussed in §2.1 where $V$ and $E$ are the vertex set and the edge set respectively ($|V|$ and $|E|$ denote the number of vertices and edges in $G$, respectively). Each edge $e_i \in E$ has an associated weight (shortest distance between their respective 3D projection lines) represented by $w_i$. The number of imaging planes is denoted by $k$. $n$ is the total number of 3D points in the seed configuration casting a unique projection on each imaging plane. A vertex is referred to as $v_{ir}$, $(i \in [1,n], r \in [1,k])$, where $r$ is the index of the plane the vertex belongs to (e.g., $P_r$) and $l$ is the index of the point in $P_r$. Note that the set of all $e_i \in E$ incident on $v_{ir}$ is represented as $E_v : E_v \subset E$. We also use the 2D array $B(i,c) : e_i \in E, c \in [1,C]$ (initialized in §2.2) that keeps track of whether or not an edge belongs to a ball.

We now discuss the IP variables. One important IP variable is $X_{ij} : i \in [1,|E|], j \in [1,n]$. For each edge $e_i \in E$, the corresponding $X_{ij} : j \in [1,n]$ is a linear array of $n$ binary values. If all entries of this array are 0, it means that this edge is not ‘chosen’ in the solution. If the $j^{th}$ entry of the array is 1, the edge $e_i$ is chosen as part of the $j^{th}$ CMDS in the IP solution. The variable, $Y_{arc}$, imposes the clique condition on the vertices chosen as part of a CMDS. If the $\mu^{th}$ vertex (point) in image plane $P_x,s \in [1,k]$, has a chosen edge (is mapped) to vertex $v$ in image plane $P_r,q \in [1,k]$, then the value of $\sum_{j=1}^{n} Y_{arc}$ is 1. The variable, $s_{jc}$, is needed to impose the geometric constraints. A CMDS is valid if all its constituent vertices are inside a ball, $B_c$. In other words, all $k$ projection lines should intersect $B_c$ (be mutually close). $s_{jc}$ is 1 if component $j$ belongs entirely to some ball, $B_c$, otherwise it is 0. We now introduce our integer programming formulation. The equations are explained at the end of the formulation.

\[
\begin{align*}
\text{minimize} & \quad \sum_{i=1}^{n} X_{ij} \\
\text{subject to} & \quad \sum_{j=1}^{n} X_{ij} \leq 1, \forall e_i \in E \quad (2) \\
& \quad \sum_{j=1}^{n} X_{ij} - Y_{pqr} \leq 0, \forall e_i \in E : p \in P, q \in P_r \quad (3) \\
& \quad \sum_{j=1}^{n} X_{ij} = \frac{k(k-1)}{2}, \forall j = [1,n] \quad (4) \\
& \quad \sum_{j=1}^{n} X_{ij} B(i,c) - \frac{k(k-1)}{2} s_{jc} \leq \frac{k(k-1)}{2} - 1, \forall j = [1,n], \forall c = [1,C] \quad (5) \\
& \quad \sum_{j=1}^{n} X_{ij} = k-1, \forall \gamma_{ij} \in V \quad (6) \\
& \quad \sum_{j=1}^{n} X_{1j} = 0 \quad (8) \\
& \quad \sum_{j=1}^{n} X_{2j} = 0 \quad (9) \\
& \quad \vdots \\
& \quad \sum_{j=1}^{n} X_{nj} = 0 \quad (10)
\end{align*}
\]

For any triplet of vertices $v_{pq}, v_{rx},$ and $v_{mn}, q \neq r \neq n$,

\[
Y_{pqr} + Y_{rnx} + Y_{mnq} \neq 2; \quad (12)
\]

\[
X_{ij} = \{0,1\}, Y_{pqr} = \{0,1\}, s_{jc} = \{0,1\} \quad (13)
\]

Eqn. 1 is the objective function we would like to minimize. Equations 2 - 13 represent constraints that must be satisfied for a good solution to our problem. The first constraint we impose is that an edge in the final solution can belong to at most one CMDS, i.e., the sum over all $X_{ij}$ variables for an edge $e_i$ should be less than or equal to 1 (Eqn. 2). Next, we define the relationship between the $X_{ij}$ variable corresponding to an edge $e_i$ and the $Y$ variables corresponding to the end vertices of $e_i$ (Eqn. 3), which was somewhat implicit until now. We know that if an edge $e_i$ is chosen (i.e., $\sum_{j=1}^{n} X_{ij} = 1$), the respective $Y$ variable corresponding to the end vertices will be set to one, otherwise it will be zero. In either case, the difference between them should be zero (Eqn. 3). The next constraint (Eqn. 4) simply ensures that each subgraph or CMDS will have exactly $\frac{k(k-1)}{2}$ edges (as each CMDS is a complete graph with $k$ vertices).

Equations 5 - 6 are used to ensure that a CMDS belongs to one ball, as a whole. Based on our analysis, we want that there should be at least one ball for each CMDS, such that all constituent edges of that CMDS belong to that ball. The array, $B(i,c)$, initialized in the preprocessing phase, is useful here. Also recall that the binary variable $s_{jc}$ is 1 or 0 depending on whether or not the CMDS $j$ belongs to the ball $B_c$. We already imposed the constraint (see Eqn. 4) that each CMDS $j$ has $\frac{k(k-1)}{2}$ edges. Now, if all edges belonging to CMDS $j$ belong to ball $B_c$, then whenever $X_{ij} = 1$, $B(i,c)$ will be 1. Therefore, $\sum_{j=1}^{n} X_{ij} B(i,c)$ (keeping $j$ and $c$ fixed) will also be 1, implying that $\sum_{j=1}^{n} X_{ij} B(i,c) = \frac{k(k-1)}{2}$ if $CMDS_j \in B_c$, and $\sum_{j=1}^{n} X_{ij} B(i,c) \leq \frac{k(k-1)}{2}$ otherwise. Notice that we have imposed a lower bound of 0 and an upper bound of $\frac{k(k-1)}{2} - 1$ on Eqn. 5. Whenever $CMDS_j \notin B_c$, $s_{jc}$ has to be 0 for the equation to be satisfied. If $s_{jc} = 1$, the value evaluates to false, violating the lower bound of the equation. Similarly, if $s_{jc} = 0$ when $CMDS_j \in B_c$, the upper bound of the equation is violated as $\sum_{j=1}^{n} X_{ij} B(i,c) = \frac{k(k-1)}{2}$. Therefore, $s_{jc}$ has to be 1 when $CMDS_j \in B_c$. Finally, in Eqn. 6, we specify that each CMDS $j$ must belong completely to at least one ball by imposing a lower bound of 1 on the sum of $s_{jc}$ over all $c \in C$. These constraints together ensure our geometric requirements of the solution.

Eqn. 7 imposes the constraint that there should be exactly $k-1$ edges incident on any vertex in the final solution. We also need to verify that these edges belong to the same CMDS. This restriction is imposed using the sequence of conditional constraints (Eqs. 8 - 10). The final constraint in this group requires that the vertex should be connected to one (and only one) vertex from each of the other planes (Eqn. 11). We use Eqn. 12 for imposing the clique condition. Observe that if two vertices on different planes are individually connected to a common vertex, these two vertices should be connected among themselves (Eqn. 12). In other words, for any ‘triplet’ of vertices, the sum of the $Y$ variables for that triplet should never be equal to 2. To address the ‘$\neq 2$’ sign in Eqn. 12, we interpret it as follows:

\[
Y_{pqr} + Y_{rnx} + Y_{mnq} \leq 1 \Rightarrow Y_{pqr} + Y_{rnx} + Y_{mnq} \geq 3 \quad (14)
\]

Since the conditional constraints like “or” cannot be directly represented in the form of a linear equation, they are converted to ordinary constraints by introducing additional binary variables, $y_{mj} : m = [1,kn], j = [1,n]$ for each pair of equation (Eqs. 8 - 10) as follows. For each vertex $v_{ir} : i \in [1,n], r \in [1,k]$.
formulation of $n$ k-cliques should not be violated. This is done by adding edges to the final solution (rounding), in decreasing order of their associated fractional values, subject to some simple check routine.

To order the edges, we proceed as follows. For each edge $e_i$ with end vertices $v_{pq}$ and $v_{rs}$, a value which equals the sum of the maximum $X_{ij}$ and $Y_{pqrs}$ (i.e., $\max_{P_i} X_{ij} + Y_{pqrs}$) is calculated. The edges are sorted in descending order of these calculated values. The edges are then evaluated in order (and rounded to 1) provided that the formation of cliques is not violated; the system should still be in a safe state after rounding. Feasibility of rounding an edge $(e_i = (v_{pq}, v_{rs}))$ is ensured by checking the following:

- If $v_{pq}$ is connected to some $v_{rs} \in P_r$, then it cannot connect to any other member in plane, $P_r$. On the other hand, if two vertices have a common ‘connectee’, then they have to be connected to ensure formation of a clique.

Observe that the higher the calculated value for an edge, the more probable it is to be included in the final solution. In essence, we ‘emulate’ the concept of probabilistic rounding, though we do not strictly round the edges according to their probabilities alone. In almost all cases, this strategy directly yields a feasible solution. However, this approach may not give us a feasible solution directly in extremely pathological cases; for example, if a wrong edge is chosen to be a part of a ‘wrong’ CMDS early on, it prevents the end vertices of that edge from belonging to their respective ‘correct’ cliques, at a later step. Here, we may have cliques of size less than $k$. Such situations can be handled using only local modifications, if required. Details about the subroutines are omitted due to lack of space.

### 2.5 Linear Program

At this stage, the integer program discussed above can be solved directly using one of the many standard operations research techniques like branch and bound, branch and cut, or using any commercially available IP solver. The problem is that, in general, integer programming problems are undecidable in the worst case. In most practical situations, where the variables are bounded, these problems fall in the class of NP-hard problems.

0-1 integer programming, of the form we have here, is a special case of IP where variables are required to be binary (0 or 1). This special case is also NP-hard as is the decision version of the problem, first discussed in Karp’s seminal paper on 21 NP-complete problems. To avoid these pitfalls, we employ a methodology standard in the field of combinatorics. We convert the integer program to its corresponding linear program by relaxing the integrality constraints and try solving that optimally instead. This means that variables are no longer strictly binary, they can be fractional values in the range of $[0, 1]$. The solution to the linear program will be an assignment of values in the range of $[0, 1]$ for the variables, which minimizes the objective function optimally while satisfying the constraints. The formulation of the linear program is identical to the IP (details are omitted due to lack of space). The objective function is represented as Eqn. 1. The constraints are the same as Equations 2 - 7, 11 and the pair of equations in 15 and 17. The primary difference is that now the variables can have values between 0 and 1:

$$X_{ij} \in [0, 1], Y_{pqrs} \in [0, 1], x_{ij} \in [0, 1], y_{ij} \in [0, 1], z_i \in [0, 1]$$ (19)

### 2.6 Rounding Approach

The linear program discussed in the previous section is solved using the well known interior point method or simplex method, a standard tool for solving linear programs. We then perform rounding on the fractional LP solution to obtain an integral (0-1) solution for our problem. In general, rounding can be done deterministically or using randomized rounding techniques (probabilistic). Notice that we need to ensure the formation of $n$ k-cliques in the solution of our problem. A randomized rounding approach may not be able to satisfy this condition in all cases. Our strategy is to use a technique, conceptually similar to randomized rounding, but at each rounding step, ensure that the ‘system’ is still in a ‘safe state’ - the

$$\sum_{i=1}^{k-1} X_{ij} - \frac{k(k-1)}{2} y_{mj} \leq 0;$$

$$\sum_{i=1}^{k-1} X_{ij} - (k-1)^2 (1 - y_{mj}) \leq k - 1; \forall j = 1 : n$$ (15)

$$y_{mj} = \{0, 1\} : m = k \times r + l, j = [1, n]$$ (16)

In the present formulation, the equation $\sum_{i=1}^{k-1} X_{ij} = 0$ is satisfied when $y_{mj} = 1$, and the equation $\sum_{i=1}^{k-1} X_{ij} > k - 1$ is satisfied when $y_{mj} = 0$. Hence, these two equations are conditionally satisfied even though the two constraints in Eqn. 15 are always satisfied. Using a similar approach, the conditional constraint in Eqn. 14 is converted to an ordinary constraint by introducing another variable $z_i$ (one for each triplet in $G$).

$$Y_{pqrs} + Y_{srmn} + Y_{pqns} - 2z_i \leq 1;$$

$$Y_{pqrs} + Y_{srmn} + Y_{pqns} + 3(1 - z_i) \geq 3$$ (17)

$$z_i = \{0, 1\}$$ (18)

### 3. EVALUATION AND RESULTS

The evaluation of the proposed technique was performed by implementing the algorithm in C++ using CGAL[16], LEDA[17], and GLPK[18] on a machine running GNU/Linux. To simulate the seed implant, an input configuration of 30 seed points was generated using a normal distribution inside a 3D box with sides 2.5 cm. These 3D points were then projected onto 3 imaging planes (k = 3), yielding three sets of 2D points on each of the three 2D imaging planes. The source to image distance (SID) was approximately 120 cm. The magnification (SID/source-to-object distance) was assumed to be 1.2 and the pixel size was 0.02 cm. The input image error introduced in the 2D image points in the three (k = 3) image planes was selected from a normal distribution having full width at half maximum (sigma-value) of $\{0cm, 0.01cm, 0.02cm, 0.03cm, 0.04cm, 0.05cm\}$ (up to 2 pixels). The input geometry introduced in the angles of the geometries was selected from a normal distribution with sigma values $\{0^{\circ}, 0.25^{\circ}, 0.5^{\circ}, 0.75^{\circ}, 1^{\circ}, 1.25^{\circ}, 1.5^{\circ}\}$. The initial values for the two primary angles (RAO, CC) of the geometry acquired from the imaging system was assumed to be approximately $\{0^{\circ}, 0^{\circ}\}$, $\{0^{\circ}, 90^{\circ}\}$, and $\{0^{\circ}, 115^{\circ}\}$ for the three imaging planes respectively. It was manually verified that 2D seed projections on the imaging plane were unique (non-overlapping). For each level of input error, 60 realizations were performed, and the mean of the results was analyzed.

In Figs. 4(a) and (b), we plot the percentage of correct matchings (cliques) determined as a function of introduced image error and geometry error, respectively. For low introduced image error (in Fig. 4(a)), the technique determines an almost perfect set of
mappings. For high image error (up to 0.05 cm or ≥ 2 pixels), the algorithm consistently determines about 95% of the matches correctly. These results are especially encouraging because the image error introduced in the 2D points for the simulation studies was not correlated. For instance, in our simulations, two neighboring 3D points in the seed-configuration will cast close 2D projections on the imaging planes. If the introduced image error is greater than the L2 distance between those two 2D point projections, it is likely that in some cases, their mutual ordering will be reversed (after error introduction). A small percentage of the incorrect matches reported in the plots above can be attributed to such degeneracies. In a real imaging setup, however, distortion errors caused due to the magnetic field of the earth affect the image globally. It can be expected that the algorithm will perform better in the presence of such distortion errors. In Fig. 4(b), we illustrate the performance of the algorithm as the introduced geometry error is increased. We notice that the algorithm performs extremely well, even for introduced geometry error of up to 1.5°, and determines the correct mapping in about 98% cases (w.r.t. number of seeds). Once the correct mappings were determined by the algorithm, the corresponding point k-tuples (cliques of size k) were used to reconstruct the set of 3D points. To do this, members of the k-tuple were considered pairwise and the shortest 3D segment joining the two projection lines was determined. Assuming k = 3, we get 3 such ‘shortest’ segments. The centroid of the triangle defined by the midpoints of the 3 segments was considered as the 3D point for that k-tuple. The reconstructed points were then compared to original points, and the 3D RMS errors were calculated (see Figs. 4(c) and (d)). We notice that the errors in x, y, and z values and 3D RMS errors increase almost linearly with introduced image error (Fig. 4(c)), as expected. In Fig. 4(d), we plot the errors as a function of introduced geometry error. It should be pointed out that the the 3D RMS errors will reduce further if the geometry is refined before 3D reconstruction using techniques such as [3, 4] after the correspondences have been determined.

4. CONCLUSIONS
We have proposed a novel combinatorial algorithm for solving an important problem in prostate brachytherapy. The formalization uses paradigms from computational geometry and operations research, and enables transforming the BSL problem into an interesting optimization problem. In order to solve this optimization problem efficiently, we formulate it as an IP and then solve the equivalent linear program. We propose techniques for rounding the LP solution to yield an integral solution to our problem. We discuss both the theoretical as well as the practical aspects of the formulation and present simulation results based on our implementation. We observe that the algorithm works very well in practice (running time was under 5 secs. on a 2.4 GHz machine for up to 50 seed-points) and yields over 96% accuracy in matching corresponding 2D seed points. Furthermore, the algorithm scales well for arbitrarily high number of imaging planes (e.g., rotational angiography) with respect to the running time. It is non-iterative and is sufficiently generic for applications in several different imaging modalities.

5. REFERENCES