Electrical Impedance Tomography Reconstruction Through Simulated Annealing using a New Outside-in Heuristic and GPU Parallelization

R. S. Tavares  
E-mail: reseta@uol.com.br

T. C. Martins  
E-mail: thiago@usp.br

M. S. G. Tsuzuki  
Escola Politécnica da Universidade de São Paulo, São Paulo, Brazil.  
Mechatronics and Mechanical Systems Engineering Department  
Computational Geometry Laboratory  
E-mail: mtsuzuki@usp.br

Abstract. Electrical Impedance Tomography (EIT) is an imaging technique that attempts to reconstruct the conductivity distribution inside an object from electrical currents and potentials applied and measured at its surface. The EIT reconstruction problem is approached as an optimization problem, where the difference between the simulated and measured distributions must be minimized. This optimization problem can be solved using Simulated Annealing (SA), but at a high computational cost. To reduce the computational load, it is possible to use an incomplete evaluation of the objective function. This algorithm showed to present an outside-in behavior, determining the impedance of the external elements first, similar to a layer striping algorithm. A new outside-in heuristic to make use of this property is proposed. It also presents the impact of using GPU for parallelizing matrix-vector multiplication and triangular solvers. Results with experimental data are presented. The outside-in heuristic showed to be faster when compared to the conventional SA algorithm.

1. Introduction  
Electrical impedance tomography (EIT) is an imaging modality that estimates the electrical conductivity distribution within the body when a low amplitude current pattern is applied to a body surface and the potential at determined points of that surface is measured through electrodes or, alternatively, when a potential is applied and the current flowing through the surface is measured [1].

The two main forms of EIT are dynamic imaging and static imaging yielding differential and absolute images respectively. The images produced by differential imaging represent the conductivity changes of a region between two time intervals [2]. Imaging physiological
function within the body largely relies on this technique. This work is focused on the reconstruction of static conductivity images, which requires more advanced numerical algorithms and, consequently, heavier computational load.

The spatial resolution of EIT is not comparable to other medical imaging techniques such as magnetic resonance, computerized tomography or ultrasonic imaging. It is also sensitive to electrode positioning and the boundary geometry. However, EIT presents some advantages over these techniques, such as being harmless to the patient, low cost and portable. Additionally, EIT has faster time-response characteristics, which enables it to monitor cyclic changes in the living tissues better than conventional imaging modalities.

EIT has a wide range of medical applications, detection of acute cerebral stroke [3], breast cancer [4], monitor cardiac activity [5] and monitor lung aeration imposed by mechanical ventilation in critically ill patients [1, 6]. The last application is the main interest of our group, where EIT can be used to implement a protective lung strategy that can result in an improved lung function and survival [7].

In this work, it is shown that the reconstruction method proposed by Martins et al. [16] happens from the outside towards the interior of the domain. It is reasonable to presume that the objective function is more sensible to modifications on external cells, which can be explained by the fact that more information is present on the boundary. Additionally, in this work it is proposed an heuristic that uses this fact and accelerated the image reconstruction. The recent emergence of multiple-core architectures, mainly GPUs, as platforms that offer a large increase in computational power over traditional architectures motivates their use in scientific computation. Particularly, the high cost of reconstructing static EIT images through SA can be diminished by the use of parallel algorithms.

This work is structured as follows. Section 2 explains the problem formulation: direct problem, inverse problem and how to solve the inverse problem as an optimization problem. Section 3 explains how to apply SA to EIT, and briefly explains the SA with objective function incomplete evaluation [8]. The proposed outside-in heuristic is explained in section 4. Section 5 presents GPU parallelization algorithms for the sparse matrix-vector multiplication and triangular solvers. Section 6 shows some results and, finally, the conclusions are in section 7.

2. Formulation

The typical forward problem in EIT is, given the conductivity distribution $\sigma$ and the current $J$ injected through boundary electrodes, find the potential distribution $\phi$ within $\Omega$ and in particular the resulting potentials at the measurement electrodes $\phi_m$. The frequencies used in EIT are low enough so that the quasi-static approximation hold, and thus we can ignore capacitive and inductive effects. Under such quasi-static conditions, the solution of the forward problem is rather simple as it only requires solving the Laplace equation

$$\nabla (\sigma \nabla \phi) = 0$$  \hspace{1cm} (1)

At the boundary, currents are injected through electrodes; thus the current density through the $l$th electrode surface $J_l$ is given by

$$\sigma \frac{\partial \phi}{\partial \hat{n}} = J_l$$  \hspace{1cm} (2)

where $\hat{n}$ is the external normal versor and zero elsewhere.

2.1. Current Patterns

Data is collected by injecting current with a single source and measuring voltage. Current is injected sequentially to the body using a pair of electrodes. There are several ways in which the pair of electrodes is switched and the voltage measurements are collected in the literature.
Brown and Seagar [9] suggested a method whereby electrical currents are sequentially applied to the body using a pair of adjacent electrodes, and voltages between adjacent non current-carrying electrodes are measured. This procedure is repeated, applying current between each pair of adjacent electrodes to obtain a voltage data set. Hua et al. [10] suggested the cross method, where current is injected between a pair of electrodes which are separated from each other, obtaining a more uniform current distribution. In this method, two adjacent electrodes for current and voltage reference are chosen. Hua et al. [10] also introduced the opposite method whereby current is injected through two diametrically opposed electrodes and uses the electrode adjacent to the current injecting electrode as the voltage reference. Fig. 1 shows the current pattern used in this work.

2.2. Finite Element Model

The inverse problem is formulated as given the injected currents $J$ and the potentials at measurement electrodes $\phi_m$, find the electrical conductivity distribution $\sigma$ within $\Omega$. In practice, only a finite number of potential measurements is made through the electrodes, so the Dirichlet boundary condition is incomplete [11]. For an irregular domain and isotropic media, analytical solution to the Laplace equation (1) with boundary condition (2) are unknown; thus, the partial differential equations were approximated by the finite element method (FEM), the domain is discretized with triangular linear elements with constant conductivity and both problems, forward and inverse, are solved numerically. The virtual potential principle associated with the Laplace equation provides the local element matrices.

When the local element matrices are stated in terms of global coordinates of the mesh, the global conductivity matrix [1], which includes electrode contact impedance effects, is obtained; then the following relation holds

$$K \cdot \Phi = C$$

where $K(\sigma) \in \mathbb{R}^{s \times s}$ is the conductivity matrix calculated at a given particular distribution $\sigma_p$, $\Phi$ is a matrix containing nodal potentials corresponding to each applied current pattern, and $C$ represents $p$ linearly independent current patterns.

It is known that for a symmetric positive definite matrix $K$, a factorization $K = L \cdot L^T$ exists and it is called Cholesky decomposition. The Cholesky decomposition is mainly used for the numerical solution of linear equations $K \cdot \Phi = C$. If $K$ is symmetric and positive definite, then $K \cdot \Phi = C$ can be solved by first computing the Cholesky decomposition $K = L \cdot L^T$, then solving $L \cdot y = C$ for $y$, and finally solving $L^T \cdot \Phi = y$ for $\Phi$ [12, section 4.2.4]. Solving the forward problem, therefore, is done through Cholesky decomposition preconditioning and solving two triangular systems.

Figure 1: Current pattern with jump three.
2.3. The Inverse Problem as an Optimization Problem

In the inverse problem, the objective is to find the conductivity distribution $K$ given applied current patterns $C$ and measured potentials $\Phi$. Since there are known methods for efficiently solving the forward problem (such as FEM), one possible approach to the inverse problem is to look at it as an optimization problem, where the optimization variables are a parametrization of the conductivity inside the domain and the optimization function is some measure of how the solution of the forward problem applied to the conductivity distribution produced by the optimization variables matches the measured data.

One possible objective function $E(\sigma)$ is the Euclidean distance between the measured electric potentials and the calculated potentials for all the applied current patterns for a given conductivity distribution. Let $\phi^m_i$ be the measured potential vector at the electrodes and $\phi^c_i(\sigma)$ the corresponding calculated electrode potentials obtained from the solution of the forward problem for a conductivity distribution $\sigma$ at the $i$-th current pattern:

$$E(\sigma) = \sqrt{\sum |\phi^m_i - \phi^c_i(\sigma)|^2} \quad (4)$$

An example of such approach is in [13], where the objective function (4) is minimized by Sequential Linear Programming yielding estimations of the conductivity distribution. Mello et al. [13] pointed that it is difficult to solve this problem by methods based on gradients of the objective function due to the fact that the problem is often ill-posed. Numerical errors in the calculation of the objective function are greatly amplified in its derivatives. That is why the interest on the Simulated Annealing (SA) applied to EIT is increasing, as it requires no evaluation of objective function derivatives.

3. Applying Simulated Annealing to EIT

As seen in section 2.3, the EIT inverse problem can be formulated as an optimization problem, and as such, can be approached with SA. Herrera et al. [14] minimized objective function (4) with SA and by doing so, managed to reconstruct very accurate conductivity distributions of the body, but at a very high computational cost. This is unsurprising, as each step of the SA involves the solution of a full FEM problem in order to evaluate the objective function. Martins et al. [8] proposed a SA with incomplete evaluation of the objective function that is briefly explained in the following.

3.1. Incomplete Evaluation of the Objective Function [8]

The evaluation of the objective function is responsible for the bulk of the SA computational cost. Therefore, methods to reduce its cost are of extreme interest, and a possible manner is to employ an incomplete evaluation of the objective function, as introduced by Lakshmanan and Derin [15]. Martins et al. [16] presented a study of how the SA process behaves in the presence of a partial evaluation of the objective function. The exact distribution of the objective function is presumed unknown, but boundaries for its values are estimated. As shown by Martins et al. [16], there is a relationship between those boundaries and the probability of deviation of the algorithm from an exact SA. Using those relationships and imposing upper limits for those probabilities, a stopping criteria for the interactive calculation of the objective function can be obtained.

It can be shown that, by imposing $P_{err}$ as an upper limit on the probability of the process taking a “wrong” decision (rejecting a solution when it should accept it or accepting when it should reject), we can create conditions for the boundaries:

$$e^{-\Delta E_{max}/kT} \geq \begin{cases} 1 - P_{err} & \text{if } \Delta \tilde{E} \leq 0, \\ e^{-\Delta \tilde{E}/kT} & \text{if } \Delta \tilde{E} > 0 \end{cases}$$

(5)
Figure 2: Convergence behavior for the “Checkerboard” problem [16].

\[ e^{-\frac{\Delta E_{\text{min}}}{\delta t}} \leq \min(1, P_{\text{err}} + e^{-\frac{\Delta E}{\delta t}}) \] (6)

Where \( \Delta E_{\text{max}} \) and \( \Delta E_{\text{min}} \) are the partial solutions of two separated sets of FEM problems and \( \Delta E \) is variation of the partial evaluations of the objective function \( \Delta E = \hat{E}^j - \hat{E}^{j+1} \).

The conditions given by (5) and (6) do not translate directly into stopping criteria for the Conjugate Gradients (CG) algorithm. As such, every solution must have its objective function evaluation process stored so it may be possible to continue it from where it has stopped. In fact, it is often computationally more efficient to improve the boundaries on the evaluation of the previous SA solution than the current one. Since CG has an asymptotically geometric convergence and presuming that both the previous SA solution and the current one have similar convergence rates (a reasonable assumption considering that the \( K \) matrix does vary very little from one iteration to another and a pre-conditioner equalizes even more the convergence rates), a simple heuristic for picking which solution to evaluate further is to just pick the one with looser boundaries.

Figure 2 shows the convergence behavior of the SA with objective function incomplete evaluation. The impedance distribution is reconstructed from the outside towards the interior of the domain. It is reasonable to presume that the objective function is more sensible to modifications on external cells, which can be explained by the fact that more information is present on the boundary. The following section explains the proposed heuristic that makes use of this property.

4. Proposed Outside-In Heuristic

Considering the mesh presented in Fig. 3, all elements can be divided in circular layers according to each element’s distance to the center of the mesh. Somersalo et al. [17] was the first to propose an algorithm based on the outside-in property. They assumed first that a current density with very rapid spatial variation is applied to the surface of the body. Since most of the corresponding current does not penetrate very deeply into the body but rather is affected mainly by the conductivity near the boundary, the corresponding voltage measurement can be used to estimate the conductivity in a thin layer at the boundary. Once the conductivity in this thin layer is approximately known, the outcome of the same kind of experiment is computed if the known boundary layer were stripped away. The conductivity on this next layer can be computed, strip it away, and so on through the body. In this way an estimate of the conductivity of the
body is found, layer by layer. However, as boundary elements are dependent on even innermost elements, it is necessary to start over from the external layers when changes were applied to inner elements. The heuristic proposed in this work uses a similar concept with the SA with objective function incomplete evaluation.

Martins et al. [8] used a SA with adaptive neighborhood proposed by Martins and Tsuzuki [18, 19]. Such a SA randomly selects one parameter and creates a modification based on the parameter crystallization factor. The crystallization factor modifies the probability distribution of the next candidate in way to increase the number of accepted solutions and keep the SA exploratory capacity.

In this work, the SA selects an element for modification in two steps. One must consider that the mesh elements are classified in layers. Initially, the SA selects one layer and, then, it selects one element from the previously selected layer. All elements from the selected layer have equal probability of being selected. However, the probability of each layer to be selected is different and controlled by the outside-in heuristic.

In its initial configuration, the SA selects with higher probability the external layer. As the temperature decreases, the external layer probability of being selected decreases and the internal layers probability increases. It is necessary that such process happens sequentially from external to internal layers, one after the other. Such heuristic is implemented as follows: if a candidate is accepted, its layer’s probability is increased; if a candidate is rejected, its layer’s probability is reduced and the immediately inner layer’s probability is increased.

In Fig. 4, it is shown a graph of each layer probability through the SA processing. Elements were divided in five layers. As it can be seen, the probability of each layer, representing its probability of having an element selected as a solution candidate, switches from the outermost layer to the innermost layer, passing through all intermediary layers. Therefore, along the cooling procedure, every layer becomes the most dominant one for a moment, and have the most elements selected.
5. **GPU Parallelization**

GPUs nowadays are specialized processing units with huge computational capability, and highly parallelized. Their role is mainly to stock big and complex sets of triangles in an ideal three dimensional environment, to assemble several of them as the CPU commands. In the last years GPUs grew capable of doing the most amazing graphic effects, like blur for foggy environment, transparencies for glasses, windows and liquids, reflections as well as partial reflections, for water effects. As the possibilities increased, instead of producing different hardware for each effect, the graphic units manufacturers started conceiving programmable graphic units, which could run arbitrary user defined programs in order to manipulate images in some of the key steps of the graphic rendering [20].

Current CPUs have between 4 and 8 available cores, against GPUs’ hundreds of cores. Such capability led to researchers employing GPUs to solve problems with high computational costs, such as medical imaging, electromagnetism and linear systems solvers. NVIDIA introduced an API, called CUDA, for general purpose computing on GPU (GPGPU), allowing researchers to use high level programming languages for implementing their applications to run on highly parallelized GPUs. The EIT reconstruction algorithm originally uses a serial, albeit efficient, library for matrix operations called Eigen. Such library will be compared to CUDA’s in one of the most used matrix operations.

A graphic processing unit implements what is called the SIMT (Single Instruction Multiple Thread) architecture, that allows to have a conditional block based on a non constant value. This was impossible with the SIMD (Single Instruction Multiple Data) that is the architecture implemented by an ideal vector processor. Conditional (if statements) can greatly decrease performance inside an SM, as each branch conditional must be evaluated.

5.1. **Sparse Matrix Data Structures and Sparse Matrix-Vector Multiplication**

Typical sparse matrix formats incur storage and instruction overheads per non-zero element, since information is needed to keep track of which non-zero values have been stored. One aim in selecting a data structure is to minimize these overheads. A matrix is computationally represented by one dimensional array. When matrix $A_{m \times n}$ is dense, a common scheme stores element $(i, j)$ at position $v[i + j \cdot m]$. This mapping allows random access to any matrix element.

Sparse matrix vector multiplication is one of the most important kernels in scientific and engineering applications. The challenge in developing high performance implementations of such kernels is choosing the data structure and code that best exploits the matrix structural properties.

It will be considered the sparse matrix vector multiplication $y \leftarrow y + A x$, where $A$ is an $m \times n$ sparse matrix with $k$ non-zero elements, and $x$ and $y$ are dense vectors. The matrix vector multiplication can be defined as

$$\forall a_{i,j} \neq 0 : y_i \leftarrow y_i + a_{i,j} \cdot x_j$$

The conventional implementation of matrix vector multiplication is shown in Alg. 1.

Algorithm 1 Dense matrix vector multiplication.

```plaintext
for i = 0 to m - 1 do
    for j = 0 to n - 1 do
        y[i] ← y[i] + v[i + m \cdot j] \cdot x[j]
    end for
end for
```

In the case of sparse $A$, a sparse format is coordinate format, where $A$ is stored using three arrays, $v$, $r_i$ and $c_i$, each of length $k$. Array element $v[l]$ holds the value of the matrix element
The elements of each row are shaded using the same color. Each row is stored as a sparse vector, and all rows (i.e., all sparse vectors) are stored contiguously in $v$ and $ci$. The $p$ array indicates where each sparse vector begins in $v$ and $ci$.

at row $ri[l]$ and column $ci[l]$. Using this format, the matrix vector multiplication is shown in Alg. 2.

**Algorithm 2** Matrix vector multiplication in coordinate format.

```python
for l = 0 to k - 1 do
    $y[ri[l]] \leftarrow y[ri[l]] + v[l] \cdot x[ci[l]]$
end for
```

Another class of format is the compressed stripe storage that includes the compressed sparse row (CSR) format and compressed sparse column (CSC) format [21]. CSR is a collection of sparse vectors, allowing random access to entire rows and efficient enumeration of non-zeros within each row. CSR is illustrated in Fig. 5. The idea is to store each row (shown as elements having the same color) as a sparse vector. A single value array $v$ stores all sparse row vector values in order, and a corresponding array of integers $ci$ stores the column indexes. Each element $p$ of a third array stores the offset within $v$ and $ci$ of row $i$. The array $p$ has $m + 1$ elements, where the last element is equal to the number of non-zeros. This data structure allows random access to any row, and efficient enumeration of the elements of a given row. An implementation of sparse matrix-vector multiply using this format is shown in Fig. 3.

**Algorithm 3** CSR matrix vector multiplication.

```python
for i = 0 to m - 1 do
    for l = p[i] to p[i+1] - 1 do
        $y[i] \leftarrow y[i] + v[l] \cdot x[ci[l]]$
    end for
end for
```

In the matrix vector multiplication, the parallelism available across rows makes it a natural choice to distribute computations corresponding to a row to a thread block. The code shown in Alg. 4 allocates one thread to perform the computations corresponding to one row and a thread block to handle a set of rows. It corresponds to a mapping in which a one dimensional grid of thread blocks and a one dimensional block of threads are used to compute sparse matrix vector multiplication. This code is very similar to the one presented in Alg. 3.

1 the non-zero elements of a sparse vector are stored contiguously in an array. An additional integer array stores the corresponding integer index for each non-zero value.
Algorithm 4 Sparse matrix vector multiplication implemented with CUDA.

```c
int line = blockDim.x * blockIdx.x + threadIdx.x;
if (line < dim)
  double sum = y[line];
  sum = 0;
  int start = p[line];
  int end = p[line + 1];
  for i = start to end - 1 do
    sum += x[ci[i]] * v[i];
  end for
  y[line] = sum;
end if
```

5.2. Graph Colloring, Mesh Numeration and Triangular Solvers

In order to calculate the value for $x[r]$, all the values of $x[i]$, $0 \leq i < r$, above it must have been solved. This is called dependency for $x[r]$. Since $L$ is a sparse matrix, $x[r]$ only depends on the columns in row $r$ in which $A[i, r]$ is non-zero. The performance of the triangular solver can be improved by expressing the data dependencies of the triangular solver for the lower triangular matrix $L$ as a direct acyclic graph.

A dependence tree that represents the elements relationship is defined as follows: there is an edge from vertex $i$ to vertex $j$ is the result of $x[i]$ influences the value of $x[j]$. A simple way to check this condition is to look at row $j$ in matrix $L$. For each column $i$ with a non zero in row $j$, an edge from vertex $i$ to vertex $j$ is created [22].

Due to the parallel operation of the GPU, it is inevitable that there will be some contention when writing data. For example, if two elements share a node, the threads performing computation for each of these elements will both need to update the same location. In order to prevent data races, techniques that can be used include the use of atomic operations, or coloring schemes.

Coloring schemes may be used to remove the need for atomic operations, by ensuring that entities whose updates may be in conflict have different colors. The execution proceeds in parallel over each color, but each color is processed sequentially. This prevents any contention.

A graph is colored such that no two adjacent nodes share the same color. Such an algorithm guarantee that adjacent nodes in the mesh will have distinct colors. By reordering the elements, vertices with the same color are grouped and sequentially ordered, such that unknown values associated to these vertices can be calculated in parallel in the triangular system solver [23].

Consider a simple unidimensional FEM model with 5 nodes shown in Fig. 6. Its nodes are numbered sequentially, the original stiffness matrix represented by a sparse matrix is shown in the left, and the corresponding lower triangular portion is shown in the right. The dependence tree shows that no parallelization is allowed at all. Fig. 7 shows the same unidimensional FEM with renumbered nodes, the stiffness matrix is in the left, and the corresponding lower triangular portion is shown in the right. The dependence tree, in the renumbered FEM case, shows that the processing of the triangular solver needs just two passes with 3 threads.

It was implemented a recursive coloring algorithm for planar graphs with a maximum of five colors similar to the one proposed by [24]. The nodes associated with one specific color can be processed in parallel. The limitation is the available number of threads. The nodes associated to different colors are sequentially processed. The triangular linear system solver is implemented with code shown in Alg. 5. Three index vectors are used: two input index vectors ($ii$ and $di$) and one output index vector ($li$).
**Algorithm 5** Triangular solver implemented with CUDA.

```c
int tid = blockDim.x * blockIdx.x + threadIdx.x;
if tid < cont[3] then
  if ii[tid] >= 0 then
    x[ii[tid]] = x[ii[tid]] - v[di[tid]] * x[i[tid]];  
  end if
end if
__syncthreads();
```

6. Results

For the evaluation of the proposed EIT reconstruction method, a simple model was built, composed of a cylindrical container made of acrylic measuring 300 mm in diameter, with 32 electrodes equally spaced in the outer wall. The electrodes are prismatic, that is, their cross-sections are invariant. The container was filled with saline water up to a height of 25 mm. The current was applied to the model in “jump-three” patterns, that is, the current was applied on electrodes that were separated by three electrodes. Those patterns were applied to each of the 32 electrodes. The applied current was of about 10 mA (actual values varied at each application) AC at 125 Hz. In order to produce observable phantoms, three cucumber slices (see [25] for a study of cucumber as a material for EIT evaluation), slightly thicker than the solution height (but not enough to produce significant 3D effects) were inserted in a triangular pattern, as seen in Fig. 8.(a). Data was collected for both the empty saline solution and for the solution with cucumber slices.

For the simulated model, a FEM model was created with 750 triangular elements and 450 nodes (see Fig. 8.(b)). In a first step, the reconstruction process was executed with the
empty medium data, in order to obtain the impedance for each electrode. It was presumed uniform, yet unknown, conductivity inside the simulated model and independent conductivity impedance values for each electrode, yielding an optimization problem of $32 + 1$ unknowns. To reconstruct the cucumber phantoms, the conductivity distribution in the simulated model was parameterized using the same 1st-order interpolating functions used for the potential in the FEM (the parameters being the conductance value at each node). The conductance of the electrodes was fixed at the values obtained in the first step, and so was the conductance for the “outer ring” elements.

The FEM problems were solved using CG method with Incomplete Choleski decomposition preconditioning [26]. The neighborhood heuristic used by the SA was taken from [18], changing only a single conductivity parameter at each iteration and reducing the modifications on parameters that lead to rejected solutions. The divergence probability $P_{err}$ was arbitrarily defined as $1/100$. The number of layers was set to 5.

The reconstructed phantom can be seen in Fig. 8 (c). Despite being heavily constrained by the mesh coarse discretisation, the reconstructed conductance distribution is a reasonable image of the physical phantom. The impact of the outside-in heuristic, and also the impact of using partial evaluation objective function, in the process performance can be seen in Fig. 9(a),
showing the average number of CG iterations used by the process at each temperature, with and without the heuristic. Even without the outside-in heuristic, and considering that the system has over 450 nodes, it is interesting to note that the system is able to achieve those results while using in average less than 35 iterations of CG. The adoption of the proposed heuristic reduces the number of iterations even more, for most of the processing.

Finally, as it is shown in Fig. 9(b), there is also an increase in the acceptance rate of solution candidates. Considering that the number of accepted candidates defines the length of the internal loop, the increase in the acceptance rate leads to a decrease in computing time. Actually, on an Intel i7 2.8 GHz CPU, and using an initial temperature of 1.0 and final temperature of 0.0001, the image in Fig. 8 (c) was reconstructed in about 205 minutes when using the outside-heuristic. Without it, the reconstruction processing needed about 220 minutes, an increase of 7.3%.

This preliminary study shows that the outside-in heuristic can improve the speed of the SA with objective function incomplete evaluation. However, further research is necessary to verify if other implementations of the outside-in heuristic can have better performance.

The parallel sparse matrix-vector multiplication was tested using CUDA 4.1 on a NVIDIA GeForce GTX 480, containing 480 cores. The presented algorithm is compared with libraries available: CUSPARSE \(^2\) [27] and Eigen \(^3\). The CUSPARSE library represents sparse matrices in the CSR format, and it has parallelized versions for the matrix vector product and triangular solver. The Eigen library has serial implementation for matrix vector product and triangular solver.

### 6.1. Matrix Vector Multiplication

Table 1 shows the result for the matrix vector multiplication. It is possible to observe that the proposed representation and method for the sparse matrix \(K(\sigma)\) showed to be much faster mainly when the number of iterations increases.

### 6.2. Triangular Solver

The proposed triangular solver was tested with the original and reordered matrices. The test was also executed with the CUSPARSE triangular solver as it is also implemented with two phases: dependence tree determination and triangular solver.

Figure 10(a) shows the original matrix and Fig. 10(b) shows the reordered matrix. The results are summarized in Table 2. Using the original matrix, 135 iterations were necessary to solve the triangular linear system. After reordering, with 455 nodes, the parallelization level increased and only 10 iterations were necessary to solve the same triangular system.

In the proposed method, it is possible to separate the two phases, and the first phase is executed just once. This fact reduces the impact of its computational cost. The performance

---

2 http://developer.download.nvidia.com/compute/DevZone/docs/html/CUDALibraries/doc/CUSPARSE_Library.pdf
3 http://eigen.tuxfamily.org/
Figure 10: (a) Original matrix. (b) Reordered matrix.

Table 2: Execution time for the Triangular Solver with CUSPARSE and the proposed method with original and reordered matrices. The pre-analysis creates the dependence tree explained in section 5.2

| Method   | Pre-Analysis (ms) | Solver (ms) |
|----------|-------------------|-------------|
| CUSPARSE | 3.442             | 0.027       |
| Original | 3.727             | 0.638       |
| Reordered| 8.174             | 0.032       |

Table 3: Execution time (ms) and error for Conjugate Gradient Algorithm with and without the incomplete Cholesky preconditioner

| Method   | Iterations | Time | Error ($10^{-11}$) |
|----------|------------|------|--------------------|
| Eigen    | with       | 348  | 471                | 4.7                 |
|          | without    | 106  | 167                | 2.0                 |
| CUSPARSE | with       | 348  | 110                | 3.4                 |
|          | without    | 106  | 50                 | 2.4                 |
| New      | with       | 348  | 35                 | 3.3                 |
|          | without    | 106  | 23                 | 1.3                 |

reached by the proposed method is similar to the CUSPARSE library. However, the CUSPARSE library requires that the pre-analysis to be executed every time with the triangular solver. This fact is a large drawback for the CUSPARSE library.

A conjugate gradient algorithm was implemented using the proposed parallel algorithms with and without the incomplete Cholesky preconditioner. A similar test was executed with the Eigen and CUSPARSE libraries. The results are in Table 3, and one can observe that the proposed method showed compatible convergence rates with smaller computational costs. The use of preconditioners in the conjugate gradient method improves the convergence rate by reducing the number of iterations to reach convergence. The execution time does not reduce in the same proportion because the preconditioner has a high computational cost.
7. Conclusions
The presented outside-in heuristic improved the computation time of the SA with objective function incomplete evaluation. The new heuristic selectively picks a layer with higher probability, such that the solution is found from external to internal layers, improving the efficiency of the SA algorithm without compromising its convergence, reducing the computing time by increasing the acceptance rate of candidates.

Also, it is shown how the use of GPU parallelization of matrix operations can effectively reduce the computational time. The use of the general purpose libraries might not be always the best approach. The proposed parallelization methods showed compatible convergence rates with smaller computational costs when compared to available libraries as CUSPARSE and Eigen.

Acknowledgements
This work was supported by FAPESP (Grant 2010/19380–0). R.S. Tavares was supported by FAPESP (Grant 2010/18658–4). M.S.G. Tsuzuki was partially supported by CNPq (Grant 309.570/2010–7).

References
[1] F. C. Trigo, R. G. Lima, M. B. P. Amato, Electrical impedance tomography using the extended Kalman filter, IEEE Transactions on Biomedical Engineering 51 (2004) 72–81.
[2] D. C. Barber, B. H. Brown, Applied potential tomography, Journal of Physics E: Scientific Instruments 17 (1984) 723–733.
[3] M. T. Clay, T. C. Frey, Weighted regularization in electrical impedance tomography with applications to acute cerebral stroke, IEEE Transactions on Medical Imaging 21 (2002) 629–637.
[4] T. J. Kao, D. Isaacson, J. C. Newell, G. J. Saulnier, A 3D reconstruction algorithm for EIT using a handheld probe for breast cancer detection, Physiological Measurements 27 (2006) S1–S11.
[5] B. M. Eyüboglu, B. H. Brown, D. C. Barber, In vivo imaging of cardiac related impedance changes, IEEE Engineering in Medicine and Biology Magazine 8 (1989) 39–45.
[6] P. Hua, E. J. Woo, J. G. Webster, W. J. Tompkins, Finite element modeling of electrode-skirt contact impedance in electrical impedance tomography, IEEE Transactions on Biomedical Engineering 40 (1993) 335–343.
[7] M. B. P. Amato, C. S. V. Barbosa, D. M. Medeiros, R. B. Magalí, G. P. Schettino, G. Lorenzi-Filho, R. A. Kairalla, D. Deheinzelin, C. Munoz, R. Oliveira, T. Y. Takagaki, C. R. R. Carvalho, Effects of a protective ventilation strategy on mortality in the acute respiratory distress syndrome, The New England Journal of Medicine 338 (1998) 347–354.
[8] T. C. Martins, E. D. L. B. Camargo, R. G. Lima, M. B. P. Amato, M. S. G. Tsuzuki, Image reconstruction using interval simulated annealing in electrical impedance tomography, IEEE Transactions on Biomedical Engineering 59 (2012) 1861–1870.
[9] B. H. Brown, A. Seagar, The Sheffield data collection system, Clinical Physics and Physiological Measurements 8 (1987) A91–A97.
[10] P. Hua, J. G. Webster, W. J. Tompkins, A regularised electrical impedance tomography reconstruction algorithm, Clinical Physics and Physiological Measurements 9 (1988) 137–141.
[11] F. S. Moura, K. C. C. Aya, A. T. Fleury, M. B. P. Amato, R. G. Lima, Dynamic imaging in electrical impedance tomography of the human chest with online transition matrix identification, IEEE Transactions on Biomedical Engineering 57 (2010) 422–431.
[12] G. H. Golub, C. F. van Loan, Matrix Computations, 3rd Edition, John Hopkins, 1996.
[13] L. M. Mello, C. R. Lima, M. B. P. Amato, R. G. Lima, E. C. N. Silva, Three-dimensional electrical impedance tomography: a topology optimization approach, IEEE Transactions on Biomedical Engineering 55 (2008) 531–540.
[14] C. N. L. Herrera, M. F. M. Vallejo, F. S. Moura, J. C. C. Aya, R. G. Lima, Electrical impedance tomography algorithm using simulated annealing search method, in: Proceedings of International Congress of Mechanical Engineering, ABCM, Brasilia, 2007.
[15] H. D. S. Lakshmanan, Simultaneous parameter estimation and segmentation of Gibbs random fields using simulated annealing, IEEE Transaction on Pattern Analysis and Machine Intelligence 11 (1989) 799–813.
[16] T. C. Martins, M. S. G. Tsuzuki, Simulated annealing with partial evaluation of objective function applied to electrical impedance tomography, in: Proceedings of the 18th IFAC World Congress, Milano, Italy, 2011, pp. 4989–4994.
[17] E. Somersalo, M. Cheney, D. Isaacson, E. Isaacson, Layer stripping: a direct numerical method for impedance imaging, Inverse Problems 7 (1991) 899–926.
[18] T. C. Martins, M. S. G. Tsuzuki, Placement over containers with fixed dimensions solved with adaptive neighborhood simulated annealing, Bulletin of the Polish Academy of Sciences Technical Sciences 57 (2009) 273–280.
[19] R. S. Tavares, T. C. Martins, M. S. G. Tsuzuki, Simulated annealing with adaptive neighborhood: a case study in off-line robot path planning, Expert Systems with Applications 38 (2011) 2951–2965.
[20] R. Nath, S. Tomov, T. T. Dong, J. Dongarra, Optimizing symmetric dense matrix-vector multiplication on GPUs, in: Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis, SC ’11, ACM, New York, NY, USA, 2011, pp. 6:1–6:10.
[21] C.-Y. Lin, Y.-C. Chung, J.-S. Liu, Efficient data compression methods for multidimensional sparse array operations based on the EKMR scheme, IEEE Transactions on Computers 52 (12) (2003) 1640 – 1646.
[22] P. Kumar, M. Kumar, A. Basu, Parallel algorithms for sparse triangular system solution, Parallel Computing 19 (2) (1993) 187 – 196.
[23] P. Berger, P. Bronaye, J. C. Syre, A mesh coloring method for efficient mimd processing in finite element problems, in: Proc. of the International Conference on Parallel Processing, IEEE Computer Society, 1982, pp. 41–46.
[24] G. N. Frederickson, On linear-time algorithms for five-coloring planar graphs, Information Processing Letters (1984) 219–224.
[25] A. R. D. S. Holder, Y. Hanquan, Some practical biological phantoms for calibrating multifrequency electrical impedance tomography, Physiological Measurement 17 (4A) (1996) A167–77.
[26] S. Kershaw. The incomplete Cholesky-conjugate solution of systems, Journal of Computational Physics 65 (1978) 43–65.
[27] NVIDIA, CUDA CUSPARSE library, Tech. rep., NVIDIA (Aug. 2010).