Modeling the Geometry of the Endoplasmic Reticulum Network

Laurent Lemarchand[1], Reinhardt Euler[1], Congping Lin[2], and Imogen Sparkes[3]

[1] Lab-STICC UMR 6285, UBO-Université Européenne de Bretagne, Brest, France,
{Laurent.Lemarchand,Reinhardt.Euler}@univ-brest.fr
[2] Mathematics, University of Exeter, UK,
c.lin@exeter.ac.uk
[3] Biosciences, University of Exeter, UK,
I.Sparkes@exeter.ac.uk

Abstract. We have studied the network geometry of the endoplasmic reticulum by means of graph theoretical and integer programming models. The purpose is to represent this structure as close as possible by a class of finite, undirected and connected graphs the nodes of which have to be either of degree three or at most of degree three. We determine plane graphs of minimal total edge length satisfying degree and angle constraints, and we show that the optimal graphs are close to the ER network geometry. Basically, two procedures are formulated to solve the optimization problem: a binary linear program, that iteratively constructs an optimal solution, and a linear program, that iteratively exploits additional cutting planes from different families to accelerate the solution process. All formulations have been implemented and tested on a series of real-life and randomly generated cases. The cutting plane approach turns out to be particularly efficient for the real-life testcases, since it outperforms the pure integer programming approach by a factor of at least 10.

Keywords: endoplasmic reticulum, plane graph, 0-1 programming, separation procedure

1 Problem

The endoplasmic reticulum (ER) is a membrane-bound organelle that forms a highly complicated interconnected network of tubules and flattened sacs (known as cisternae) [10, 6]. As the cortical ER in plant cells occupies a very thin, almost two-dimensional, layer of cytoplasm beneath the plasma membrane, our study of the ER network will be based on 2D approximations of the ER network in Tobacco leaf epidermal cells. Figure 1 (a) shows an instance of live cell images of an ER network [13]. Transition between tubules and cisternae can be highly dynamic and tubules can also dynamically change their polygonal network [10]. The dynamic ER shape is suggested to be adaptable to the cells requirements for ER function; for example, ER cisternae may be the preferred site of protein translocation while tubules might be the preferred site for ER vesicle budding. As an expanding number of proteins have been identified that mediate the generation and shape of the ER network, the stage is now set for investigations into the mechanisms regulating ER morphology within the cell. To be able to carry out such investigations, better tools are required to quantify the morphology and dynamic rearrangements that the ER undergoes. It is important to consider that whilst the proteins and stresses on the system driving formation and changes in the remodelling may differ between eukaryotic systems, the network geometry of the ER appears to be fairly well conserved in terms of it consisting of a polygonal network of interconnected tubules and cisternae. Therefore, the problem considered here has universal appeal towards understanding the constraints placed on ER network formation in all systems.

A quantitative analysis [11] of the ER network in tobacco leaf epidermal cells suggests that it is a perturbed Euclidean Steiner network between terminals, where terminals are persistent nodes (static elements of tubules) and degree-1 nodes. A Euclidean Steiner network is a locally minimal network, i.e., a network in which any local perturbation of non-terminal nodes would increase the total length of the network [11]. This is analogous to Steiner trees in Euclidean space [8] in which the non-terminals (called Steiner points) have degree 3. Note that local optima such as Euclidean Steiner networks might not give a unique network topology, which is essential for modelling its dynamics. In this paper, we reanalyze live cell confocal microscopy data of native ER networks from [13] in an attempt to understand whether there is an optimization principle governing the network shape beyond local minimization. For quantitative analysis, we abstract ER networks into geometric graphs using the image processing method from [11]; examples of abstracted graphs are shown in Figure 1 (b,c).

The ER membrane surface, serving as a transport network, is intuitively suspected to be a minimal film [12]. Due to the existence of cycles commonly observed in networks, minimal spanning trees do

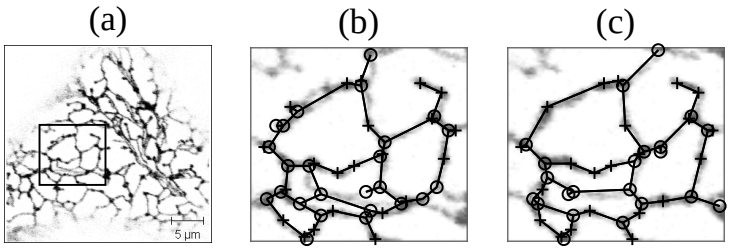

Fig. 1. Illustration of the ER network and abstracted geometric graphs. (a) shows a static ER network, where rectangle regions highlight a region with no ER cisternae. (b)-(c) show zoomed local networks in the rectangle region and corresponding abstracted geometric graphs from two images. The abstracted graphs are obtained using the image processing method introduced in [11], where markers '+' and 'o' represent persistent and non-persistent nodes, respectively, and lines represent edges (only the largest connected components in the chosen rectangle regions are considered, and tubules extending outside the chosen region are not included in the abstracted graphs). The experimental ER images are taken from [13] (www.plantcell.org, Copyright American Society of Plant Biologists)

not sufficiently explain the shape of the ER network. Also, ER tubules generate 3-way junctions when branching and angles at these degree-3 nodes follow a normal distribution with mean around 120^o [11]. Here, we include a degree constraint and an angle constraint for the nodes while minimizing total edge-length, and we test whether the optimal graph under these constraints could mimic the ER network geometry. To test the optimal solutions from our model, we quantitatively compare them to the abstracted ER networks from a tobacco leaf epidermal cell.

2 Model

We construct two models: one basic model with only degree constraints and one full model with both degree and angle constraints.

Basic Modelization Given a set of nodes V (corresponding to all the nodes in an abstracted ER network), a subset V_b of V (corresponding to the degree-3 branching nodes in the network), the problem is to find an undirected connected plane graph, whose nodes in V_b have degree 3, those outside V_b have degree at most 3, and which minimizes the sum of the Euclidean distances associated with the connecting edges.

Full Modelization A full model is the basic model with an additional angle constraint. More precisely, for a branching node $u \in V_b$, and its neighbours v_1, v_2, v_3, we add the constraint that any two angles at the node u formed from edges $uv_i (i = 1, 2, 3)$ have a sum no less than θ where θ is a given angle around 180^o. The idea comes from the concern of force balance acting on the branching node. For each degree-3 branching node (tubule junction), as modeled in [11], each of the three ER filaments is assumed to apply a membrane tension force on this tubule junction. Thus, it is unlikely that the tension forces are in the same direction of a half plane, i.e., the sum of two angles of the branching node is less than 180^o. This angle constraint means that none of two angles of a branching node form a sum less than $\theta \approx 180^o$ in the optimal solution.

3 Problem Formulation and Resolution

Given the complete edge-weighted graph $G = (V, E, w)$ and a set $V_b \subseteq V$, the basic problem $\mathcal{BP}$ can be formulated as follows:

$$\text{minimize} \sum_{x_{uv} \in E} w_{uv} x_{uv} \tag{1}$$

$$\text{subject to} \quad 1 \leq \sum_{v \neq u} x_{uv} \leq 3 \qquad \forall u \in V \setminus V_b, \tag{2}$$

$$\sum_{v \neq u} x_{uv} = 3 \qquad \forall u \in V_b, \tag{3}$$

$$\delta(W) \geq 1 \qquad \forall W \subset V, |W| \geq 2 \tag{4}$$

$$x_{uv} + x_{wz} \leq 1 \qquad \forall u, v, w, z \in V, \text{edges } uv \text{ and } wz \text{ cross} \tag{5}$$

$$x_{uv} + x_{uw} + x_{uz} \leq 2 \qquad \forall u \in V_b, v, w, z \in V, \text{angle}(vuw) + \text{angle}(wuz) < \phi \tag{6}$$

$$x_{uv} \in \{0, 1\}, \qquad \forall uv \in E. \tag{7}$$

We are looking for a minimum-weight, connected and plane, spanning subgraph of G, where:

- (1) the objective function represents the total Euclidean distance of the connecting subgraph;
- (2) and (3) represent the constraints on nodes including degree-3 nodes;
- (4) ensures the connectivity of the resulting subgraph, where $\delta(W) = \sum_{i \in W, j \notin W} x_{ij}$;
- Angle equations (6) are set with a degree of $\phi = 180^o$. u is a branching node, and v, w, z are its neighbours in a solution;
- x_{uv} and x_{vu} represent the same variable, and $x_{uv} = 1$ iff edge uv is selected in the solution.

We remark that the complexity status of our problem still seems to be open. The particular case of finding a minimum-weight 3-regular connected spanning subgraph of G is known to be NP-hard [3]. Moreover and throughout our calculations, we have observed only few solutions containing a pair of crossing edges. Therefore, instead of adding constraints (5) at start, we check after each iteration whether two edges cross, in case of which the corresponding constraint (5) is added on the fly.

Similar incompatibility constraints are added for the angle restrictions of the full model.

3.1 Binary Linear Programming Resolution

Problem $\mathcal{BP}$ is solved using the CPLEX MIPS solver as follows. The initial $0-1$ programming problem is solved without connectivity constraints (4). Cuts are added iteratively in order to discard connected components which cover only a subset of the nodes. More precisely, if $W \subset V$, with $1 < |W| < |V|$, is the node set of a connected component obtained as a result, we add the following constraint to $\mathcal{BP}$:

$$\delta(W) \geq 1. \tag{8}$$

This process is repeated until the resulting graph is connected. As described above, a plane embedding is checked for at each iteration.

3.2 Linear Programming Formulation

Let $\mathcal{RP}$ denote the linear relaxation of $\mathcal{BP}$, i.e., the linear program obtained by replacing the constraints (7) by $0 \leq x_{uv} \leq 1 \; \forall uv \in E$. Relaxing $\mathcal{BP}$ to $\mathcal{RP}$ allows to replace the IP-solver by an LP-one, but we are now faced with the possibility of fractional optimal solutions.

Three types of search algorithms have been used to "cut off" such fractional solutions, leading to 4 different *separation* procedures.

The first one (*2-cut* procedure) is based on a min-cut algorithm. If an $s-t$-cut leads to a cut value < 1, the connectivity constraint is violated. This situation is detected using the Stoer-Wagner algorithm [14]. If V_1, V_2 is the corresponding partition of V, we add the constraint

$$\delta(V_1) \geq 1 \tag{9}$$

The second one relies on a p-partition of the node set V, $P = \{V_1, V_2, ..., V_p\}$. In this case we add the constraint

$$\frac{1}{2} \sum \delta(V_i) \geq (p-1) \tag{10}$$

The problem is now to find a partition of V, whose inequality is violated by the current optimal (and fractional) solution.

We have implemented 2 separation procedures for this type. The *r-cut* procedure is based on a recursive splitting of partitions using a min-cut algorithm, whereas the *k-cut* procedure is a 1-edge-like contraction procedure.

- The *r-cut* procedure for multi-cuts is inspired by [1]. Given a p-partition, find a minimum cut in each part of it. For the cut with minimum value among all these min-cuts, break the associated component to obtain a $(p+1)$-partition.
- The *k-cut* procedure for multi-cuts goes as follows. The partition is induced by the components of the graph $G' = (V, E' = \{e \in E | x_e \geq \alpha\})$, with $\alpha \in]0..1]$ as given by the current optimal solution. Case $\alpha = 1$ corresponds to the component search of $\mathcal{BP}$. This separation procedure is applied for $\alpha = 0.8, 0.6, 0.4$.

The third one is a *b-cut* procedure based on *blossom* inequalities arising from Edmonds' description of matching-polytopes [5]. For this we just recall that the incidence vectors of *u-capacitated b-matchings* are the solutions of the following constraints:

$$\sum_{e \in \delta(i)} x_e \leq b_i \qquad \forall i \in V, \tag{11}$$

$$0 \leq x_e \leq u_e \qquad \forall e \in E, \tag{12}$$

$$x_e \in \mathbb{Z} \qquad \forall e \in E, \tag{13}$$

and if we let $b_i = 3\ \forall i \in V$ and $u_e = 1\ \forall e \in E$, the following *blossom* inequalities

$$\sum_{e \in E(W)} x_e + \sum_{f \in F} x_f \leq \left\lfloor \frac{3|W| + |F|}{2} \right\rfloor, \forall W \subset V, F \subset \delta(W) \text{ with } 3|W| + |F| \text{ odd} \tag{14}$$

are valid for any solution of our basic problem.

The separation procedure presented in [9] is based on cut-trees. In our implementation we use Gusfield's algorithm [7] for cut-tree computations, and we also make use of the igraph C library [4].

The initial binary algorithm is thus split into two phases :

- *Linear phase* : Solve $\mathcal{RP}$ with an LP-solver, and add constraints for all disconnected components. When the obtained solution is connected but fractional, find iteratively minimal cuts (2-cuts of value less than one, k-cuts or b-cuts, whose corresponding inequality is currently violated) and add the corresponding constraints.
- *Binary phase* : When no more efficient cuts are found, solve the resulting 0-1 programming problem with an MIPS solver.

According to the kind of linear constraints we add in the linear phase, this leads to 5 versions of the mixed linear/binary algorithms:

- *BP*: the binary formulation,
- *LP*: the linear formulation,
- *LPr*: the linear formulation with recursive cuts,
- *LPrk*: the linear formulation with both recursive and parametric cuts.
- *LPrkb*: the linear formulation with recursive, parametric and blossom cuts.

4 Tests

All formulations have been implemented using the CPLEX API in C. A set of 50 real-life testcases has been provided. Randomly generated testcases, with distance values in the range [1..100] are also used for the tests, especially with a large number of nodes. We look at the comparative results in terms of runtimes for both the binary and the linear formulation as well as the different cutting techniques for the basic problem solution. Random sets are used to evaluate the behaviour of the different algorithms with respect to problem characteristics such as nodes and percentage of branching nodes.

Finally, real-life testcases are used to evaluate the behaviour of the algorithms in view of the different pieces of the model.

4.1 Runtimes for the Solution of the Basic Problem

We examine the runtimes of 5 different algorithms for solving the basic model: (1) the binary formulation *BP*, (2) the linear formulation *LP*, (3) the linear formulation with recursive cuts *LPr*, (4) the linear

Algorithm 1: General Algorithm. BP ($solveBinary = True$), LP ($solveBinary = False$), LPr ($solveBinary = False$, $checkRcuts = True$), $LPrk$ ($solveBinary = False$, $checkRcuts = True$, $checkKcuts = True$) and $LPrkb$ ($solveBinary = False$, $checkRcuts = True$, $checkKcuts = True$, $checkBlossom = True$) are derivated from this general algorithm

Data: a set of points V with a subset of branching nodes V_b, a set I of incompatible edge pairs $\{e, e'\}$, a set of boolean variables $\{solveBinary, checkPlan, check2cuts, checkRcuts, checkKcuts, checkBlossom\}$

Result: a connected plane subgraph

```
1  begin
2      graph G := (V, E), with E := V^2, w_uv the Euclidean distance between u and v
3      generate problem P with constraints (2) and (3)
4      add constraints (5) for all edge pairs in I
5      if solveBinary = True then
6          add constraints (7) to P
7      end
8      f := solve (P)
9      if G_f = (V, f) is connected and binary then
10         if checkPlan = True then
11             if checkPlanarity(f) = OK then goto 31
12             else add corresponding constraints (5)
13         end
14     end
15     if G_f = (V, f) is not connected then
16         compute component set W = {W_1, ..., W_p}
17         add corresponding constraints (4)
18     end
19     if solveBinary = False then
20         if check2cuts = True then check and add corresponding constraints (9)
21         if checkRcuts = True then check and add corresponding constraints (10)
22         if checkKcuts = True then check and add corresponding constraints (10)
23         if checkBlossom = True then check and add corresponding constraints (14)
24         if no new constraint added then
25             add constraint (7) to P
26             solveBinary := True
27         end
28     end
29
30     goto 8
31 end
```

formulation with both recursive and parametric cuts *LPrk* with $\alpha = \{0.4, 0.6, 0.8\}$, and finally (5) *LPrkb*, that is *LPrk* plus the blossom-cut separation procedure.

We have generated a set of problems of increasing size (in number of nodes) with distance values in the range [1..100] and with 30 % of branching nodes. Problem sizes of Figure 2(a) are comparable to real-life testcases : the average percentage of branching nodes is 27.54 %, and the number of nodes is in the range [12..76] for the testcase set of Figure 3. We measure the average runtimes of 10 trials for the different algorithms, according to problem size. Corresponding results are presented in Figure 2 (a). Figure 2 (b) shows results for much larger problems obtained with *BP* and, the most efficient procedure, *LPrk*.

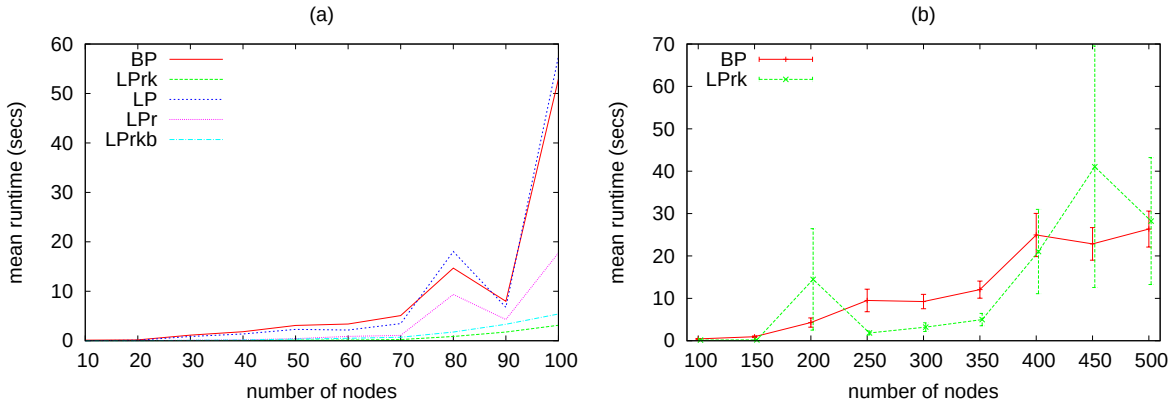

Fig. 2. (a) *BP* and *LPx* formulation runtimes for series of 10 random testcases with distances in [10..100] and 30 % branching nodes. (b) *BP* and *LPrk* runtimes (and standard error) on large testcase series with 45% of branching nodes

For random testcases of Figure 2 (a), *LPrk* outperforms slightly the *BP* approach. Adding blossom inequalities in *LPrkb* is time-consuming but does not improve the average results of *LPrk*. *BP* runtimes dramatically increase with problem size.

For large instances (Figure 2 (b)), the efficiency of *LPrk* is clearly visible for problems with up to 400 nodes, number which is much greater than that of the real testcases. For more than 400 nodes this advantage disappears, because the solver swaps into Binary programming mode rather early in relation to the runtime of the whole optimization process. Time won in early steps does not lead to significant improvements in cost for later steps. For those large cases, *LPrk* is often faster, but suffers from a lack of stability, with some instances degrading heavily the average performance, as shown by the large standard error of the runtimes for this procedure.

Figure 3 shows the runtimes for the different real-life testcases (*frames*) and the different procedures. The average runtimes are, respectively, 102.29 s, 98.96 s, 6.67 s, 0.33 s and 0.63 s for *BP*, *LP*, *LPr*, *LPrk* and *LPrkb*.

Figure 3 clearly shows that *LPrk* is very efficient for real-life testcases when compared to other approaches for solving the basic problem. Its maximum runtime over the 50 cases is 5.58 s, compared to 9.68 s for *LPrkb*, 78 s for *LPr* and two non-terminated cases for *BP* and *LP*.

Runtimes according to the number of branching nodes We have generated a set of problems of increasing size (in number of nodes), and we have measured the total runtimes of *LPrk* for different percentages of branching nodes within each instance.

As shown in Figure 4, the runtimes of *LPrk* depend heavily on the number of branching nodes within the problem, that we recall to have a fixed degree of 3. For large size random problems, the algorithm behaves correctly provided the percentage of branching nodes is above 40.

Conclusion The *LPrk* approach is very efficient for solving real-life problems with the basic model, in comparison to other methods, especially *BP*. *LPrk* is also the most efficient for a large part of the random problems, i.e., those with up to 400 nodes. However, it gives bad runtimes for a few instances.

The percentage of branching nodes has a great influence on the runtimes whatever the solution algorithm is. The higher this percentage, the better the runtimes.

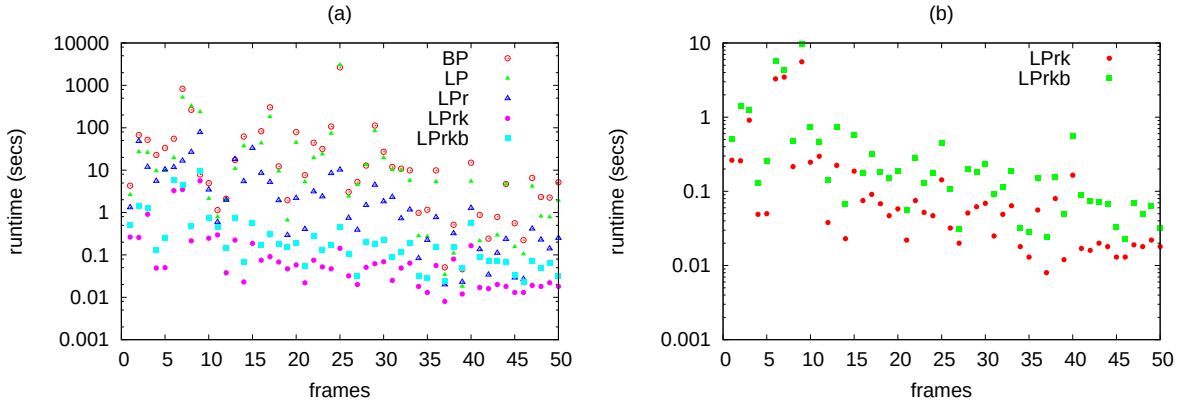

Fig. 3. (a) *BP* and all *LP* formulation runtimes for real-life testcases. (b) *LPrk* and *LPrkb* only

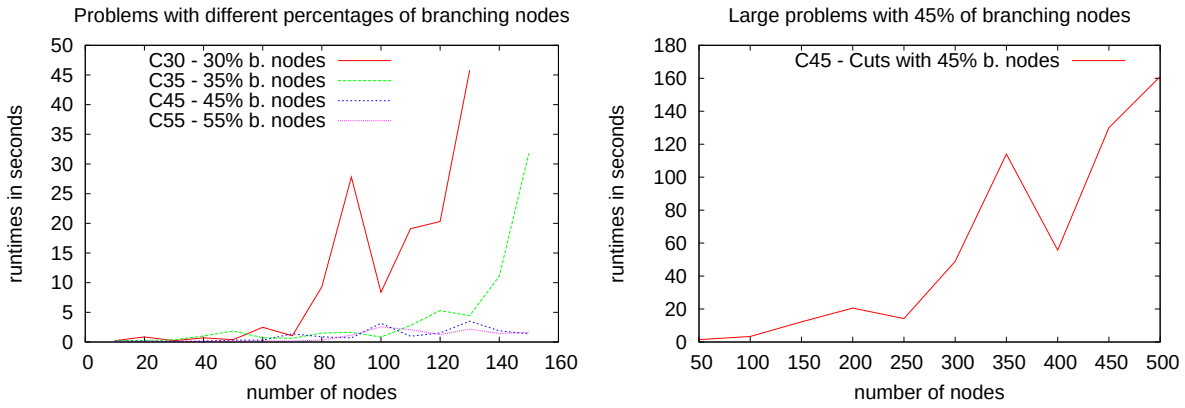

Fig. 4. Runtimes of *LPrk* according to the size of the problem and the number of branching nodes. Distances are in the range [1..100]

4.2 Runtimes of Real-life Testcases with the Full Model

Figure 5 shows the runtimes for the 50 testcases and the full model (see section 2) solved with both the *BP* and *LPrk* algorithms. As opposed to the basic model, constraints on angles are added. Planarity problems arise only in two cases. Thus, it is not necessary to check for a plane embedding at each iteration. Instead, it suffices to check it at the end, and the solution process is restarted after the addition of appropriate constraints whenever needed.

With the *LPrk* solver, the full model is solved in 46.52 s on average compared to 0.33 s on average for the basic model. One case is very costly and represents half of the total runtime. If this case is removed, the average runtime is 21.75 s, which is 2 orders of magnitude more than the basic model runtime. With *BP*, mean runtime was 96.92 s for the full model. In this series, one case (different from the one already mentioned) was very time consuming again. If removed, the mean runtime is 64.24 s, 3 times the average runtime for *LPrk*.

These last results show that the complexity as induced by the model's advanced constraints leads to a runtime explosion for the *LPrk* algorithm, although the problem is still solvable with *LPrk* in reasonable time. *BP* runtimes remain stable when taking into account the full model. But those runtimes are still more than two times higher than those of the *LPrk* approach on average for the 50 frames testbench. Finally, notice that sometimes one of the algorithms behaves well while the other spends a lot of time for solving a particular case.

In the next section, we show, however, that those additional constraints are mandatory in order to obtain results that are close to real-life networks. That means that the runtime aspect would be crucial if larger networks are to be computed. Instead of relying on the MILP solver branch-and-bound algorithm when cutting planes are not found anymore, we could exploit these cuts further by embedding them into a branch-and-cut procedure. However, this implies the development of a specific branching strategy. This could help improving the runtimes, especially for those instances where the MILP solver is called early.

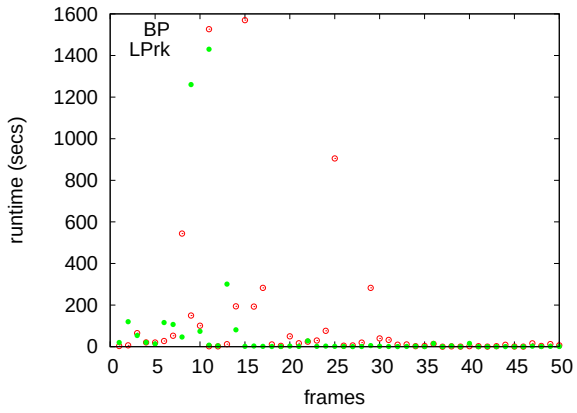

Fig. 5. For the full model, *BP* and *LPrk* runtimes for real-life testcases

4.3 Real-life Testcase Results and a Comparison with Actual Topologies

We have implemented the proposed technique to find optimal graphs for $N = 50$ problem instances both for the basic model (with degree-3 constraints on nodes $V_b \subset V$) and the full model (with angle constraints as illustrated in Section 2 in addition to the degree-3 constraints). The nodes are abstracted from native ER networks in chosen regions with no cisternae using the image processing method given in [11]. Figure 6 shows different optimal graphs for given sets of nodes and degree-3 nodes together with the abstracted ER network. To quantify the difference between these optimal graphs and the abstracted ER network we may use the concepts of *effective similarity*, *error correcting matching* and *angle distribution*.

We define an effective similarity $s(G_1, G_2)$ between two graphs G_1, G_2 as the average of two percentages: percentage of edges in G_1 that do appear in G_2 and percentage of edges in G_2 that do appear in G_1. This measurement can be calculated via the adjacency matrices of the two graphs. The measurement $s(G_1, G_2)$ ranges from 0 to 1; it equals 0 if none of the edges coincides in the two graphs and it equals 1 if the connecting structures between the two graphs are the same. Note that the adjacency matrix Adj_G is symmetric whenever G is.

As a complement, and in analogy to the notion of error correcting graph matching [2], we define a normalized error correcting matching $m(G_1, G_2)$ between two graphs G_1, G_2 over the same set of nodes as the ratio of the minimum number of edit operations (edge addition and edge deletion) necessary to transform one graph into the other, to the number of edges of the complete graph with the same set of nodes. This measurement can be calculated via the adjacency matrix as well. It also ranges from 0 (no correction needed) to 1.

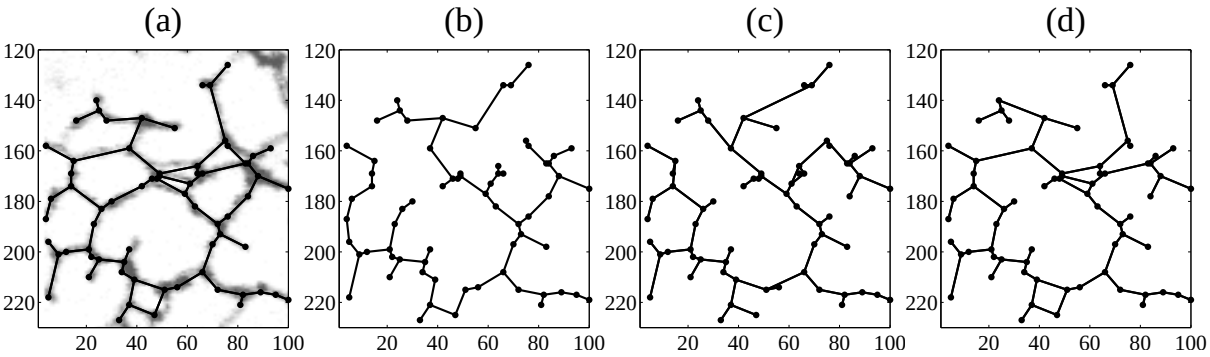

Fig. 6. A comparison between: an abstracted ER network (a), a minimal spanning tree (b), an optimal graph from the basic model (c), an optimal graph from the full model with $\theta = 180^o$ (d). Their total lengths are $627.8, 508.3, 558.0$ and 602.3, respectively, in units of pixel. The effective similarities of (b-d) with respect to the abstracted network in (a) are $0.726, 0.812$ and 0.938, respectively, while the normalized error correcting matchings with respect to the abstracted network in (a) are $0.120, 0.012$ and 0.004, respectively. The underlying ER image in (a) is from the imaging data in [13]

Table 1. A comparison of optimal solutions G_1 from different models with the abstracted ER network G in terms of similarity $s(G, G_1)$ and matching $m(G, G_1)$. Data indicate the mean±error of the mean (N=50). T tests show significant differences on $s(G, G_1), m(G, G_1)$ between the case where G_1 is a minimal spanning tree and the case where G_1 is an optimal graph from the basic model ($p < 0.0001$ for both measurements), and slight differences between the basic and the full model with $\theta = 160$ ($p = 0.0058, p = 0.0136$ for similarity and matching, respectively). One-way ANOVE tests show no significant differences between graphs from the full model ($p = 0.1083, p = 0.0869$ for similarity and matching, respectively)

N=50	MSP model	basic model	full model ($\theta = 160^o$)	full model ($\theta = 170^o$)	full model ($\theta = 180^o$)
$s(G, G_1)$	0.7376 ± 0.0015	0.9065 ± 0.0103	0.9185 ± 0.0092	0.9270 ± 0.0081	0.9271 ± 0.0084
$m(G, G_1)$	0.2930 ± 0.0195	0.0106 ± 0.0011	0.0096 ± 0.0011	0.0087 ± 0.0010	0.0087 ± 0.0011

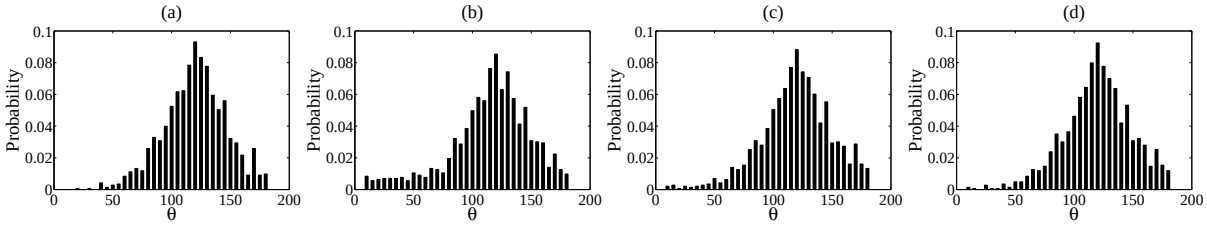

Fig. 7. A comparison of angle distributions from an abstracted ER network (a), an optimal graph from the basic model (b) and an optimal graph from the full model with $\theta = 160^o$ (c) and with $\theta = 180^o$ (d). Observe, that the distributions in (c) and (d) are much closer (with p-values of 0.863 and 0.975, respectively) to that in (b) (with a p-value of 0.111) when compared to the distribution in (a). The p-values are the asymptotic p-values for the null hypothesis that distribution in (b) ((c) and (d)) and that in (a) are from the same distribution. Angles are taken from all 3 angles of degree-3 nodes in all 50 problem instances. The sample size is $N = 1428$ in each distribution

Figure 6 illustrates an example of an ER network in comparison to different optimal graphs (including minimal spanning tree, optimal graph from the basic model and optimal graph from the full model). The optimal graphs from the full model show a higher similarity and a lower error correcting matching. Moreover, we show in Table 1 that overall the ER network is closer to an optimal graph from the full model than that with/without degree constraints, and the θ in the angle constraint would not lead to a significant difference in terms of similarity and error correcting matching when compared to the ER network. In addition, Figure 7 shows that the distribution of angles of degree-3 nodes in the model with both degree and angle constraints is much closer to that of the abstracted ER networks than that in the minimal spanning trees and in the optimal graphs with only degree constraints. This suggests that beyond degree constraints, angle constraints are necessary for understanding the principles governing the ER network geometry.

In addition, considering that the nodes abstracted from the ER network using the image processing method in [11] may have errors in their position, we have analyzed the sensitivity of optimal solutions with respect to perturbations of the node positions by randomly modifying one pixel among the 8 connected neighbours. Note, that in the presence of angle constraints, optimal solutions should change when a pair of nodes forming a sum of angles close to the critical θ-value are perturbed. Figure 8 shows examples of optimal graphs with perturbed nodes together with those without such a perturbation. However, statistical analysis on similarity and error correcting matching with the ER networks for 10% perturbed nodes gives $m(G, G_1) = 0.9200 \pm 0.0029 (N = 500), s(G, G_1) = 0.0096 \pm 000039 (N = 500)$. This indicates that there is no significant difference in these measurements between optimal graphs with/without node perturbation.

5 Discussion

In this article, a quantitative comparison between optimal graphs and the ER network geometry suggests that the ER tubule network in the native state minimizes the total tubule length between branching nodes and non-branching nodes. The difference between an optimal graph and the corresponding ER network might be that there are other principles behind the ER network geometry. For instance, Figure 9 shows that the ER network contains a cycle while the optimal graph for this instance does not have a cycle. We suspect this ER network to be more robust in structure. Indeed, we have compared the number of

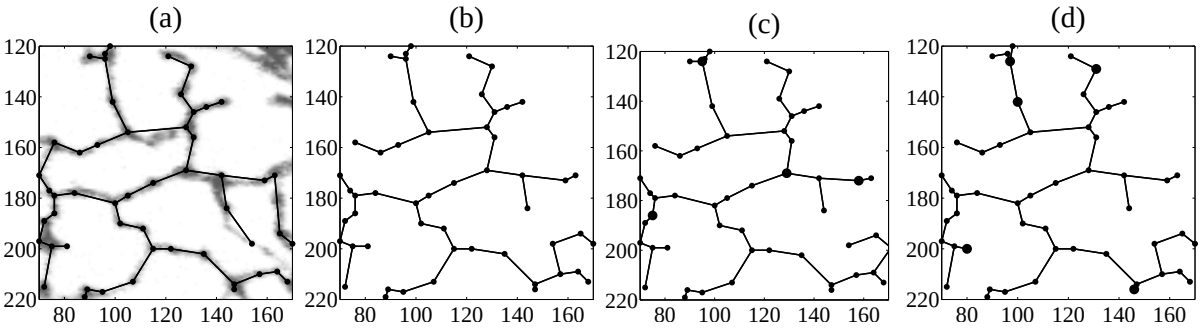

Fig. 8. A comparison of graphs with perturbed node positions. (a) shows an abstracted ER network. (b) shows the optimal graph from the full model with $\theta = 180^o$. (c) and (d) show optimal graphs with perturbed node positions (shown as larger bold dots) from the full model with $\theta = 180^o$. The underlying ER image in (a) is from the imaging data in [13]

cycles and the natural connectivity for all 50 times points in the data set, and we show in Figure 10 that both the number of cycles and the natural connectivity in the abstracted ER networks are overall larger than in corresponding optimal graphs. The natural connectivity $\lambda(G)$ [15] is a measurement of structural robustness of a graph G in terms of a weighted sum of numbers of closed walks. More precisely, $\lambda(G) = \log(\sum_{k=0}^{\infty} \frac{n_k}{k!}/|V|)$, where n_k is the number of closed walks of length k. Another reason for cycles disappearing in an optimal solution might be due to the choice of the region, where branching junctions in the global ER network might only have one or two edges (branching junctions connecting with tubules outside the chosen regions are not included in the abstracted graph) and thus are not in the set of degree-3 nodes. To avoid this, one may wish to choose a non-rectangle region where branching junctions are all of degree 3 in the abstracted graph. However, these branching junctions might be connected with ER cisternae and thus techniques need to be developed to distinguish ER cisternae and ER tubules.

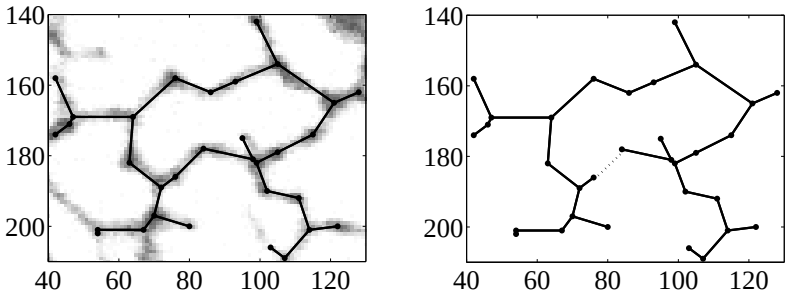

Fig. 9. Left: an abstracted ER network. Right: the optimal graph from the full model ($\theta = 180^o$). They only differ in one edge shown as a dotted line in the right panel. The underlying ER image in the left panel is from the imaging data in [13]

The study of the optimal principles behind ER network geometry could, in the future, allow predictions of network dynamics as a consequence of node movement. This would require image processing methods to track the node movements and one challenge for this is the dramatic dynamics of ER networks themselves where nodes may appear and disappear. We leave these aspects for future study.

As to algorithmic efficiency, there should be place for improvement. Setting $V = V_b$ and omitting crossing and angle constraints leads to the minimum-weight d-regular connected spanning subgraph problem with $d = 3$. For $d = 2$, we encounter the traveling salesman problem, well-studied both for its polyhedral structure and algorithmic aspects. A suitable transfer of this knowledge to our case, combined with the modifications indicated at the end of section 4.2, could considerably improve the efficiency of our method.

ACKNOWLEDGMENTS

CL thanks grant BB/J009903/1 for support of the time.

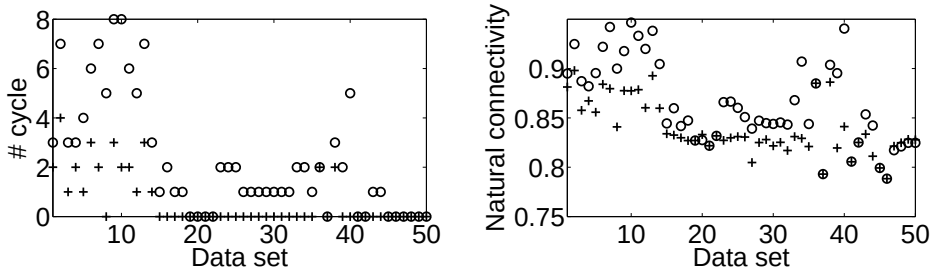

Fig. 10. A comparison of the number of cycles (left) and natural connectivity (right) between abstracted ER networks (circles) and optimal graphs (crossings) from the full model with $\theta = 180^o$ for each data set

References

1. Barahona, F.: On the k-cut problem. Oper. Res. Lett. 26, 99–105 (2000)
2. Bunke, H.: Error correcting graph matching: on the influence of the underlying cost function. IEEE Trans. Pattern Anal. Mach. Intell. 21, 917–922 (1999)
3. Cheah, F., Corneil, D.: The complexity of regular subgraph recognition. Discrete Appl. Math. 27, 59–68 (1990)
4. Csardi, G., Nepusz, T.: The igraph software package for complex network research. InterJournal, Complex Systems 1695 (2006), http://igraph.sf.net
5. Edmonds, J.: Maximum matching and a polyhedron with 0-1 vertices. J. Res. Nat. Bur. Standards 69B, 125–130 (1965)
6. Goyal, U., Blackstone, C.: Untangling the web: Mechanisms underlying er network formation. Biochim. Biophys. Acta 1833, 2492–2498 (2013)
7. Gusfield, D.: Very simple methods for all pairs network flow analysis. SIAM J. Comput. 19(1), 143–155 (1990)
8. Hwang, F.K., Richards, D.S., Winter, P.: The steiner tree problem. Ann. Discrete Math. 53 (1992)
9. Letchford, A., Reinelt, G., Theis, D.: Odd minimum cut sets and b-matchings revisited. SIAM J. Discrete Math. 22(4), 14801487 (2008)
10. Levine, T., Rabouille, C.: Endoplasmic reticulum: one continuous network compartmentalized by extrinsic cues. Curr. Opin. Cell. Biol. 17, 362–368 (2005)
11. Lin, C., Ashwin, P., Sparkes, I.A., Zhang, Y.: Structure and dynamics of er networks and perturbed euclidean steiner networks. *(in preparation)* (2014)
12. Sparkes, I.A., Hawes, C., Frigerio, L.: FrontiERs: movers and shapers of the higher plant cortical endoplasmic reticulum. Curr Opin Plant Biol. 14(6), 658–65 (2011)
13. Sparkes, I.A., Runions, J., Hawes, C., Griffing, L.: Movement and remodeling of the endoplasmic reticulum in nondividing cells of tobacco leaves. Plant Cell 21, 3937–3949 (2009)
14. Stoer, M., Wagner, F.: A simple min-cut algorithm. J. ACM 44, 585–591 (1997)
15. Wu, J., Barahona, M., Tan, Y.J., Deng, H.Z.: Spectral measure of structural robustness in complex networks. IEEE Trans. Syst., Man, Cybern.A, Syst., Humans 41, 1244–1252 (2011)