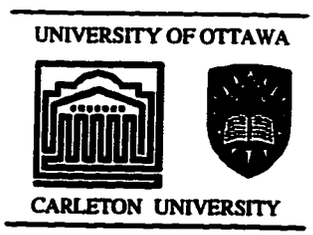
Université d'Ottawa · University of Ottawa

DRAWING ALGORITHMS FOR SURVIVABLE TELECOMMUNICATION NETWORKS

Ala Eddine Barouni

Thesis submitted to the School of Graduate Studies and Research
in partial fulfillment of the requirements for the degree of

Master of Computer Science



Ottawa-Carleton Institute of Computer Science
Department of Computer Science,
Faculty of Science
University of Ottawa,
Ottawa, Ontario, Canada

Abstract

Visualizing survivable telecommunication networks on the screen has proven to be useful and helpful for the network designers. In fact, they can easily identify rings, perceive the interaction between rings, and then rapidly spot possible problems.

Given a ring cover of survivable telecommunication networks, we provide three techniques for drawing a ring cover: the inside drawing, the outside drawing and the mixed drawing.

In chapter 1, we will introduce the problem and present some basic definitions related to our subject.

In chapter 2, we will present the inside drawing algorithm which consists of drawing each ring inside another one.

In chapter 3, we will deal with the outside drawing algorithm in which rings are drawn outside others.

In chapter 4, we will describe the mixed drawing algorithm which consists of drawing rings outside or inside each other.

Finally, a conclusion and open problems for future work will be presented in chapter 5.

We should mention that all these drawings should respect many criteria in order to preserve the readability of the drawing. These criteria are as follows:

Rings must be easily identifiable within the picture, no crossing is acceptable and the resolution rule should be respected. As in most of the graph drawing algorithms, the area used for the drawing is very important. Our proposed algorithms produce drawings that require $O(n^2)$ area, where n is the number of nodes in the ring cover.

Dedication

This work is lovingly dedicated to my mom, Fatma. For the priceless gift of her love, I thank her.

To my dad, Habib, I extend my deepest love and thanks for his constant moral support and encouragement.

Acknowledgments

I would like to thank all the people who assisted me in completing this work.

First of all, I would like to express my deepest appreciation to my supervisor, Dr. Nejib Zaguia, for his constant moral support, encouragement, and fruitful discussions throughout my graduate studies. The many discussions we have had since I started research work towards this thesis have been of great influence.

I wish to extend my thanks also to Dr. Ioannis G. Tollis for his help and for explaining to me his work.

I would like to thank Brahim Ghribi and Nabil Ben Saidane for they help in reviewing the drafts of the thesis and its presentation.

Finally, I must acknowledge with gratitude the financial support provided by the Tunisian Government and the Canadian International Development Agency (C.I.D.A).

Contents

Chapter 1: Introduction and Definitions

Chapter 2: Inside Drawing

Chapter 3: Outside Drawing

Chapter 4: Mixed Drawing

Chapter 5: Conclusion and Open Problems

Bibliography

Chapter 1

Introduction and Definitions

The problem of drawing a graph in the plane has received increasing attention recently due to the large number of applications stated in [BETI93]. For example automatic layout of pert diagrams [BPTI89], layout algorithm for data flow diagrams [BNT86] and layout algorithm of entity-relationships diagrams [BMT84].

The design and analysis of telecommunication network is a very important area (for more details see [SHAH92], [GVSM90] and [SNH90]). In this thesis, we study techniques for visualizing telecommunication networks. We are motivated to design a network which:

1. *Satisfies the traffic requirements.*
2. *Survives failures.*
3. *Has a minimum cost.*

A network is *survivable* if it can survive the failure of a link e, that is, the removal of the link e does not disconnect the network, and the traffic that was originally supported by the link e, can be accommodated by another path.

There are many architectures that can survive failures and satisfy the traffic requirements. The best architecture that meets the minimum cost requirement is called the *"Multi-Ring Architecture"* which is considered to be a cost-effective survivable network because of its simplicity and improved survivability (see [WULA92] and [WCA88]).

Consider a network N represented by a graph $G = (V, E)$, where V is the set of nodes representing the sites and E is a set of links representing the electrical links between nodes. A *ring* R in G is defined to be a cycle consisting of nodes $n_1, n_2, ..., n_p \in V$.

For some specific rings in a network, we may have a high traffic. The network designers may decide to add more equipment to the nodes (sites) of these rings in order to increase the performance of the network. Therefore, one of the most important properties of Multi-Ring Architecture, is that rings should be easily recognizable.

In this thesis, we focus on drawing networks on the computer screen so that properties of the Multi-Ring Architecture can easily be viewed.

Similar to graph drawing, many criteria should be considered in order to enhance the readability. Usually, a general optimization method are used, i.e., minimize the crossings, the area, the number of bends (in orthogonal drawings), and the number of slopes (in polyline drawings).

Readability is also our concern, because we have noticed, so far, that unnecessary crossings may create rings that do not exist in the original network.

In this thesis, we will address the issue of visualizing survivable telecommunication networks by presenting multi-ring architecture drawing techniques.

In section 1, we introduce the basic definitions. In section 2, we give more details about the main problem. Section 3 is devoted to the description of the related work. In

section 4, we explain what will be contributed to this work, and finally in section 5, we present the thesis outline.

1.1 Basic Definitions

The Ring Cover

A *Ring Cover C* is a set of rings that covers all links in a network [GHSITX94] (see figure 1.1 (a)).

The contact node

For a network N represented by a graph G, given a ring cover C, a *contact node* c is a node of G which belongs to the intersection of at least two rings of C (for instance, node c_2 in figure 1.1(a) is a contact node).

The ring-contact node graph

Let V' be a collection of contact nodes, R a ring, C a ring cover and c a contact node. In [TOXI94], Tollis and Xia define a graph $G' = (V' \cup C, E')$, such that the vertices of $V' \cup C$ are nodes corresponding to all rings $R \in C$ and contact nodes $c \in V'$, and each edge $e \in E'$ is the link (R, c), where the contact node c is in the ring R.

$E' = \{(R, c) : R \in C, c \in V' \text{ and } c \in R\}$.

The graph G' is called the *ring-contact node graph* which is considered to be the underlying structure of the ring cover, that contains all the relations that exist between rings within the ring cover. As an example, the ring-contact node graph of the ring cover shown in figure 1.1 (a) is depicted in figure 1.1 (b).

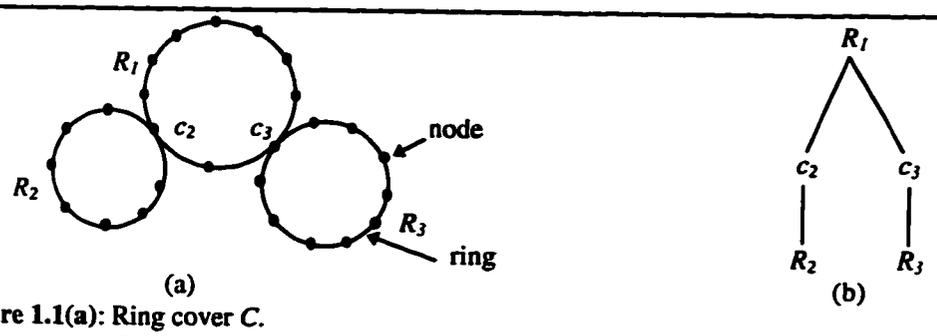


Figure 1.1(a): Ring cover C.
(b): Ring-contact node graph G' of C.

The resolution

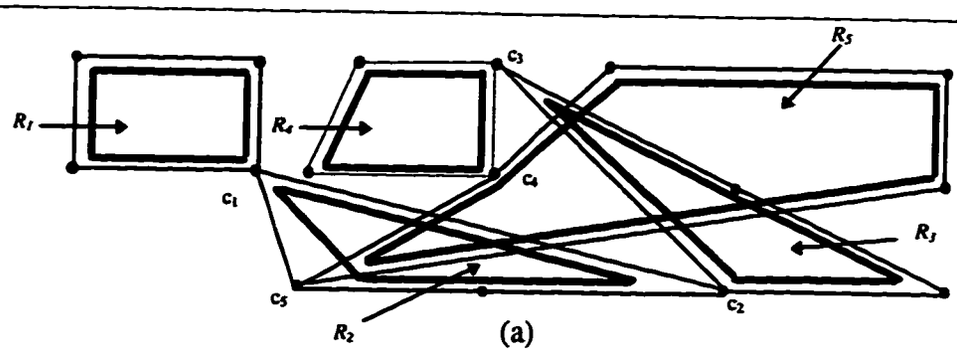
The *resolution* is a constant defined by the user which is considered to be the minimal distance between any two nodes in the drawing of the ring cover. The resolution rule is that the nodes must be kept far enough from each other, so that, the human eye can tell them apart. The resolution rule prevents the drawing algorithm from arbitrary scaling down the picture.

1.2 Problem definition

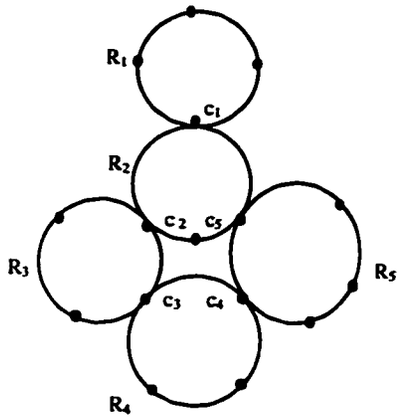
The problem is defined as follows: let N be a telecommunication network represented by a graph $G = (V, E)$, where V is the set of nodes representing the sites of switches and E is a set of links representing the electrical wires or optical fiber links between nodes. Corresponding to the graph G, a $|V| \times |V|$ matrix T is defined so that: the entries $T_{i,j}$ denote the amount of traffic between the node i and the node j, where $1 \leq i, j \leq |V|$.

In [GHSITX94], some techniques are provided to find a ring cover in survivable networks by using the matrix T as input. Thus, for a network N, we assume that the ring cover C is given and therefore our task is to deal with the **ring cover drawing**. We shall present three different techniques of ring cover drawing. They take into account many criteria in order to produce layouts that are clear and easy to understand.

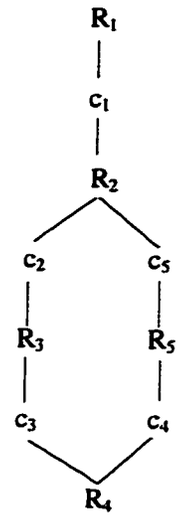
Since the nodes of the network correspond to sites, they have geographic coordinates. Hence, the network can be drawn naturally with little effort. However, for some complicated structure of networks, the important properties that designers are interested in, such as rings, are not displayed. For instance the ring cover drawn in figure 1.2(a), in which bold line polygons represent rings in the network, illustrates this matter.



(a)



(b)



(c)

: Bold line represents rings in the network.

Figure 1.2(a): A network N with the ring cover C.
(b): Example of a drawing ring cover of the network N.
(c): Ring-contact node graph G'.

1.3 Related Work

In [TOXI94] and [TOXI93], three techniques of ring cover drawing are provided:

1. Inside Drawing : Each ring is drawn inside other rings.
2. Outside Drawing : Each ring is drawn outside of other rings.
3. Mixed Drawing : Each ring can be drawn outside or inside other rings.

These three techniques are based on the following assumptions:

- Any two adjacent rings share only one contact node.
- The ring-contact node graph is a tree and therefore it contains no cycles.

They proposed two open problems:

1. Drawing a ring cover when any two adjacent rings share one contact node and the ring-contact node graph is not a tree.
2. Drawing a ring cover when rings share more than one contact node.

We remind our readers that these techniques will be described in detail in the following chapters.

1.4 Thesis Contributions

We focus on expanding the conditions of the ring cover drawing in order to deal with the above open problems.

Our aim is to present new techniques of ring covers drawing for inside drawing, outside drawing and mixed drawing.

Our research includes the case treated in [TOXI94] where any two adjacent rings share exactly one contact node, augmented by the following three major contributions:

1. First contribution : When any two adjacent rings share exactly one contact node and the ring contact node graph is not a tree.

2. Second contribution : When rings share more than one contact node.

3. Third contribution : A combination of both cases: the ring cover contains rings which share more than one contact node and the ring-contact node graph has cycles.

Other contributions:

The design and the implementation of a new system for drawing telecommunication networks based on the Multi-Ring Architecture. The system invites the user to enter the ring cover through its underlying structure (discussed in chapter 2) and then generates the picture of the network.

Finally we propose a new approach to find the optimum layout of the underlying structure which minimizes the area of the drawing.

1.5 Thesis Outline

In chapter 2, we will address the issue of the inside drawing algorithm. We present the related work which was limited only to the case where any two adjacent rings share exactly one contact node and the ring-contact node graph is a tree . We propose a new inside drawing algorithm in order to expand the drawing to other more general type of networks, such as rings sharing more than one contact node and networks where their ring-contact node graphs contain cycles. For each case of drawing, we propose a new specific underlying structure of the ring cover.

In chapter 3, we present another technique for drawing a ring cover called the outside drawing. We propose our optimal underlying structure which is the generalization of the specific underlying structure related to the cases studies in chapter 2, then we provide an improved version of the outside drawing algorithm.

Chapter 4, deals with the mixed drawing algorithm which uses both inside and outside drawings. We emphasize that our mixed drawing algorithm is totally different from the one presented in [TOXI94].

The conclusion of our dissertation and some open problems are provided in chapter 5.

Chapter 2

Inside Drawing

2.1 Introduction

In this chapter, we will describe the inside drawing technique which consists of drawing each ring inside the other one.

In the first part, we will present the previous work related to the inside drawing algorithm in [TOXI94].

In the second part, we will present our approach which consists of extending the drawing of the ring cover to other types of networks as follows:

First, we treat the case where any two adjacent rings in the ring cover share exactly one contact node, and we consider the possibility of having *ring-cycles* (discussed in section 2.3) in the ring cover. Then we define a new underlying structure to the ring cover which will be considered as the basis of the drawing. This underlying structure is a tree which we call *extended-ring-contact node graph.*

Second, we treat the case where any two adjacent rings can share more than one contact node, then we introduce another underlying structure of the ring cover which we call *ring-multi-contact node graph.* We assume that this underlying structure is a tree.

Finally, we consider the case where the ring cover has rings that share more than one contact node and it may contain *ring-cycles*. Then we introduce the general underlying structure of the ring cover called *generalized-ring contact node graph* which includes the previous two cases.

2.2 Case 1: Ring-contact node graph is a tree

In the inside drawing technique, the rings are presented by circles. The case where the ring cover has one ring, shows that the area required for drawing a ring cover has a lower bound $\Omega(n^2)$, where n is the number of the nodes in the network.

Drawing ring cover in general is rather difficult. Even if the rings are presented by circles or convex polygons, not all ring covers admit an appropriate drawing. The following lemma describes a necessary condition for a ring cover that admits a convex polygon representation, such that the face of the polygons presented in the plane are convex.

Pairwise compatible

Let a ring $R = \{n_1, n_2, ..., n_p\}$ be an ordered sequence of its nodes in clockwise order. Let $R_1 = \{n_1, n_2, ..., n_p\}$ and $R_2 = \{m_1, m_2, ..., m_q\}$ be two rings, and suppose that they have k contact nodes $\{c_1, c_2,.....,c_k\}$, where c_i appears before c_{i+1} in the clockwise order around R_1.

R_1 and R_2 are pairwise compatible, if and only if, there is a cyclic shift of the nodes of R_2, so that the contact nodes $\{c_1, c_2,.....,c_k\}$ appear in the order $1..k$ in R_2, or that the contact nodes $\{c_1, c_2,.....,c_k\}$ appear in the reverse order $k..1$ in R_2. Figure 2.1 illustrates such a case.

Lemma 1 [TOXI94]

Given a ring cover $C = \{R_1, R_2, ..., R_n\}$, if C has a convex polygon representation then all the rings in C are pairwise compatible.

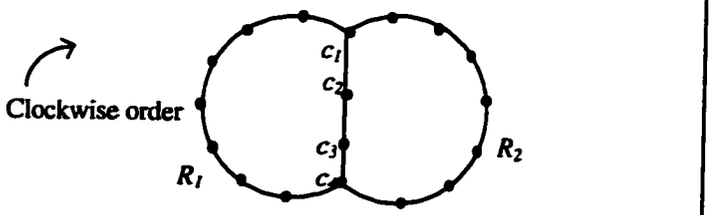


Figure 2.1: R_1 and R_2 are pairwise compatible.

Tollis and Xia provided an inside drawing of ring covers, only in the case where any two adjacent rings share exactly one contact node, and the ring-contact node graph G' is a tree. In the next paragraph, we are going to represent the inside drawing algorithm of the ring cover. Before doing so, we have to introduce some concepts:

Enclosing Cycle

For a ring node u and its subtree in the ring-contact node tree T, the enclosing cycle Γ_u of u must be large enough to place the drawing of u and its subtree inside Γ_u.

In the inside drawing, each ring is placed inside each other. This means that all the children rings are placed inside the parent ring. The enclosing cycle for a ring node u is actually the ring for u itself, since all of its children are placed inside u, see figure 2.2 (c).

Let u be a ring node in T and its children: u_1, u_2, ..., u_i. If the radius of all the circles Γ_j of u_j for $j = 1, 2, ..., i$, have been computed, then the radius of the circle Γ corresponding to the node u is computed using the following procedure:

Procedure *Compute_radius_ring*

First a circle Γ' is drawn, such that all the nodes of the ring u and the center of the circles Γ_j corresponding to u_j, are placed evenly on the perimeter of the circle Γ' (see figure 2.2(a)). The distance between the nodes and the circles is equal to the minimum distance required by the resolution rule. Obviously, the diameter of the circle Γ'

is larger than the maximum diameter of the circles Γ_j. Next, we draw a circle Γ which has the same center as the circle Γ' and a radius equal to the radius of the circle Γ' plus the maximum radius of the circles Γ_j.

Finally, each node and circle Γ_j is moved towards the perimeter of the circle Γ' following the vector (center(Γ'), center(Γ_j)), such that each circle Γ_j has one common contact point with the circle Γ (see figure 2.2(c)). Note that the resolution rule is preserved during this process.

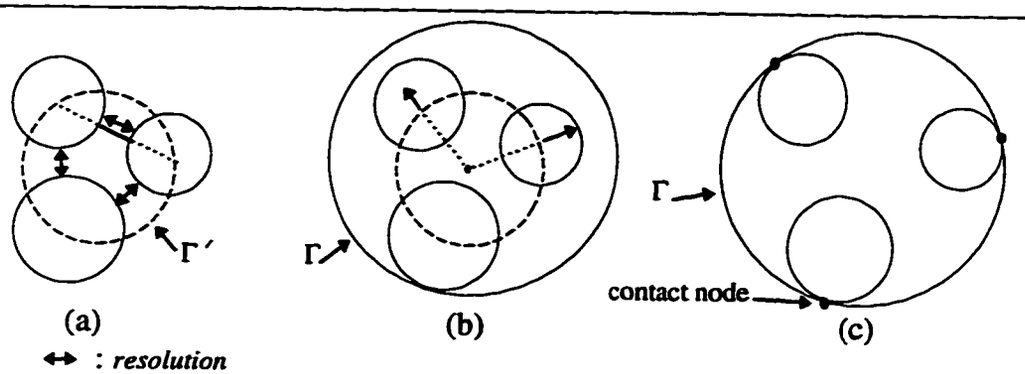


Figure 2.2 (a) : Placing the centers of the circles Γ_j on the perimeter of the circle Γ'.
(b) : Drawing of the circle Γ that encloses the circle Γ' and the circles Γ_j.
(c) : The final drawing of the circle Γ and all other circles Γ_j.

Using the above procedure, Tollis and Xia [TOXI94] proved the following lemma:

Lemma 2

For any ring node u and its subtree in T (ring-contact node tree), the radius of the corresponding circle Γ is $O(n)$, where n is the number of nodes in u and its subtree.

In the case where three rings share the same contact node, the rings are placed one inside the other, see figure 2.4(d). The transformation of the ring-contact node tree as shown in figure 2.3 and figure 2.4(c), consists of drawing a circle that corresponds to a subtree, inside the circle which corresponds to another subtree, so that they share the common contact point.

Let's take the example shown in figure 2.4(b), the contact node c_3 has two children R_4 and R_5. We move R_5 below R_4, so that the contact node c_3 will have only one child. The

circle corresponding to the ring R_5 will be drawn inside the circle corresponding to the ring R_4.

In figure 2.3, we present an overall diagram.

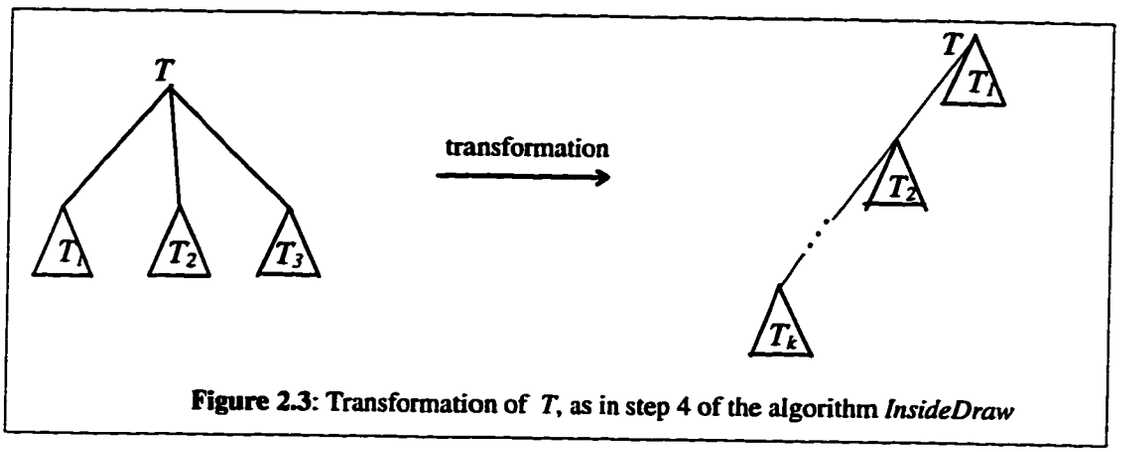


Figure 2.3: Transformation of T, as in step 4 of the algorithm *InsideDraw*

In figure 2.4, we present a detailed diagram.

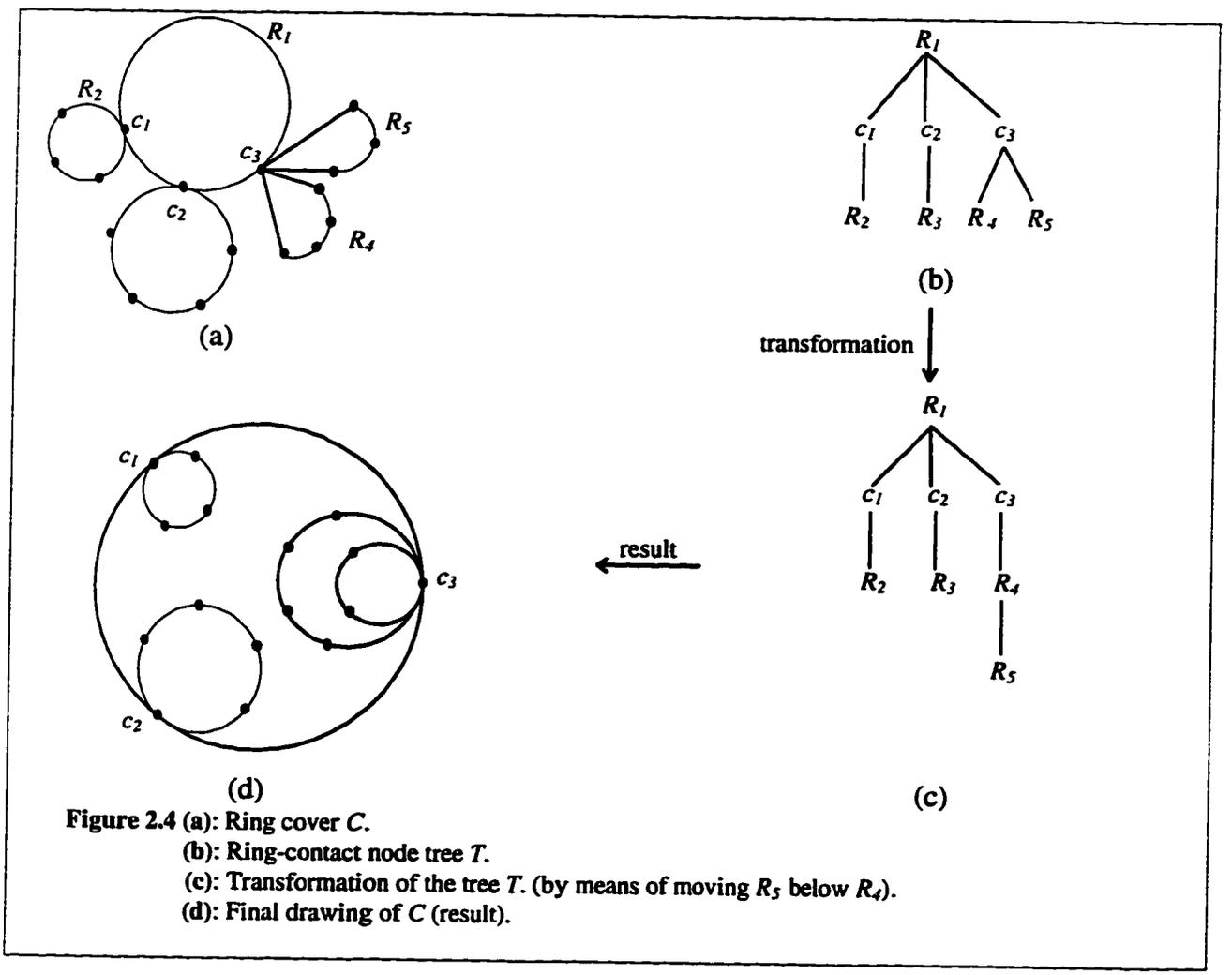


Figure 2.4 (a): Ring cover C.
(b): Ring-contact node tree T.
(c): Transformation of the tree T. (by means of moving R_5 below R_4).
(d): Final drawing of C (result).

2.2.1 Algorithm

Essentially, the algorithm traverses the ring-contact node tree T, and processes the nodes recursively in postorder fashion. After drawing all the children of a node, the algorithm draws their parent node (by computing the radius of the circle corresponding to the parent node). When a node u in the ring-contact node tree is a contact node, the algorithm ignores u and processes recursively its children.

Algorithm *InsideDraw(v)*;
Input : Ring cover C, and its ring-contact node tree T with root v.
Output : An inside Drawing of C.
Begin
1. **If** v is a leaf **then**
   - Compute the radius of the circle Γ corresponding to v, such that nodes are placed uniformly on the perimeter of the circle Γ, and are distant by *resolution*; **return;**

   EndIf;
2. **If** v is a node representing a ring **then**
   Begin
   - **For** all the children u_i of v **do** *InsideDraw(u_i)*;
   - Call *Compute_radius_ring(v)* (computation of the radius of the circle Γ corresponding to v as describe in section 2.2);

   EndIf;
3. **If** v is a node representing a contact node and v has only one child u' **then**
   - *InsideDraw(u')*;

   EndIf ;
4. **If** v is a node representing a contact node and v has more than one child **then**
   Begin
   - Perform a transformation as shown in figure 2.3.
   - *InsideDraw (T)*;

   EndIf;

End.

Using the above procedure, Tollis and Xia [TOXI94], proved the following theorem:

Theorem 1

Given a ring cover C and its corresponding ring-contact node tree T, the algorithm *InsideDraw*, produces a ring cover drawing such that:

1. Each ring is represented by a circle;
2. there are no crossings;

3. the area required by the drawing is $O(n^2)$, where n is the number of nodes in the ring cover C;
4. the time complexity of the algorithm is $O(n)$.

We have implemented the inside drawing algorithm described previously, our system called DSTN (Drawing Survivable Telecommunication Network) asks the user to enter as input: the ring cover through its underlying structure then the system generates a picture of the ring cover, such that the rings are presented by circles (see figure 2.5).

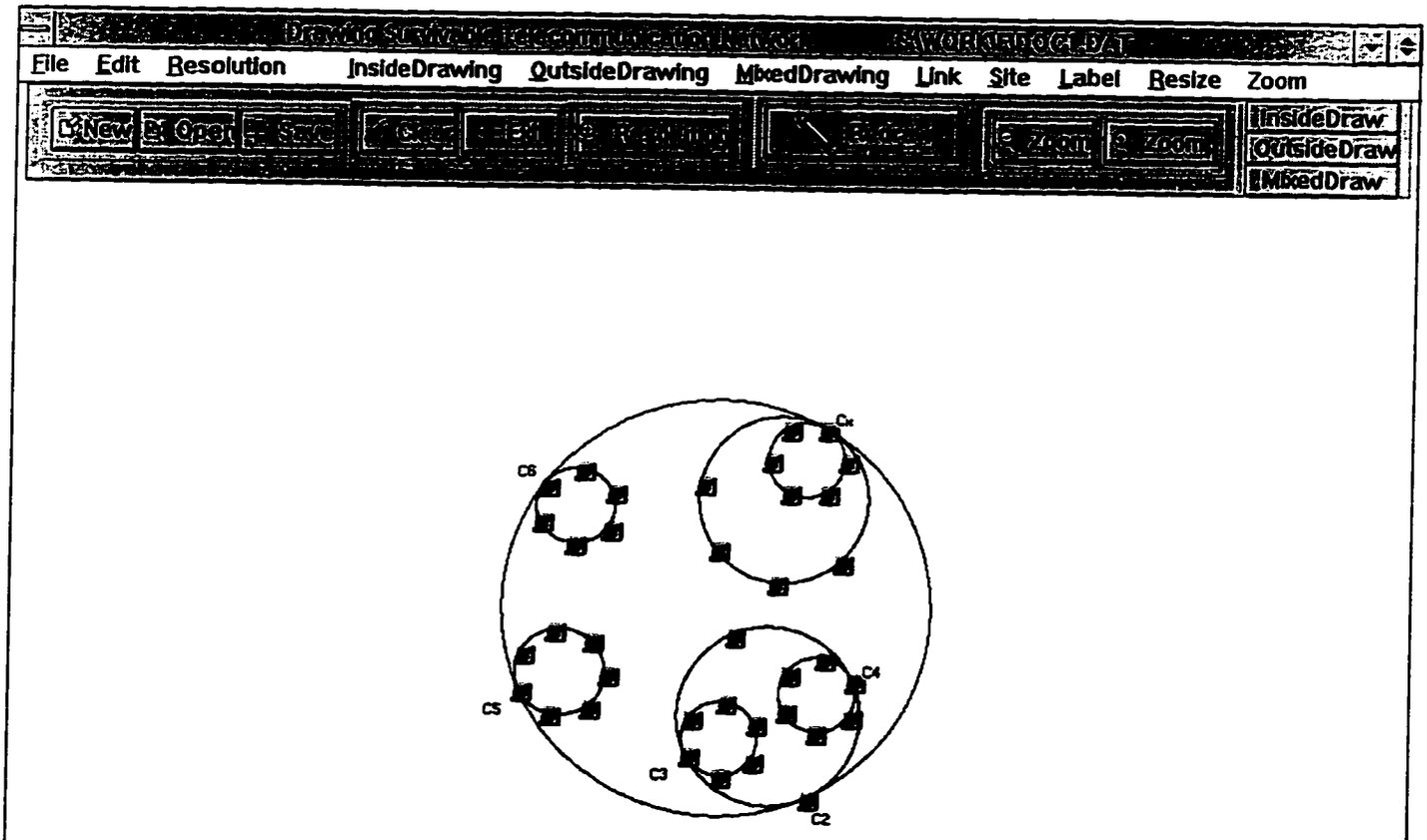


Figure 2.5: A picture representing the network layout produced by the system.

2.3 Case 2: Ring-contact node graph is not a tree

In this section, we present our approach to the inside drawing which consists of extending the drawing to other types of networks. We assume that any two adjacent rings in the ring cover share exactly one contact node, moreover it is possible to have *ring-cycles* in the ring cover.

The ring-cycle

A path P which connect two contact nodes c_i and c_j is called a *free* path if and only if c_i and c_j are the only contact nodes in this path.

A *ring-cycle* $RC = \{R_1, cn_1, R_2, cn_2, ..., R_i, cn_i, R_{i+1}, cn_{i+1}\}$, where $R_1 = R_{i+1}$ and $cn_1 = cn_{i+1}$, is a set of rings R's and special contact nodes cn's, so that rings form a cycle in the ring cover, that is:

- Any two adjacent rings R_j and R_{j+1} share exactly one contact node cn_j, $j = 1, ..., i$.
- And in each ring R_j, $j = 2, ..., i+1$, there exists at least one free path between two distinct special contact nodes cn_{j-1} and cn_j belonging to the ring R_j.

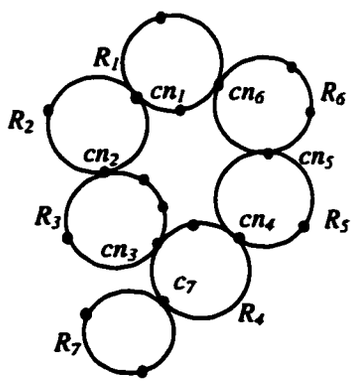


Figure 2.6: Ring Cover *C*.

For instance in figure 2.6, the ring cover C contains the ring cycle $RC = \{R_1, cn_1, R_2, cn_2, R_3, cn_3, R_4, cn_4, R_5, cn_5, R_6, cn_6, R_7, cn_7\}$, where $R_1 = R_7$ and $cn_1 = cn_7$. As we can see in the figure any two adjacent rings share exactly one contact node, and that each ring

R_i, $i = 2, ..., 7$ has two contact nodes cn_{i-1}, cn_i, and there exists a free path between those two contact nodes.

As another example one can notice that the ring cover shown in figure 2.7, does not contain a ring-cycle, because it exists at least the ring R_4 which has two special contact nodes cn_3, cn_4 that do not have a free path (because of c_7 and c_8).

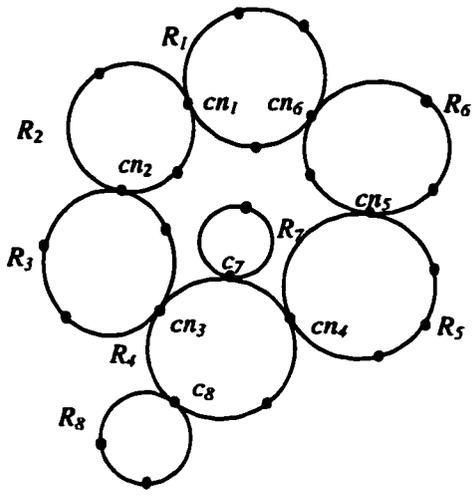


Figure 2.7: Ring Cover *C*.

In practice, it is possible to have a very complicated structure of ring covers. The only ring cover drawing treated, so far, deals with the case where the ring-contact node graph is a tree. In the case where the ring cover contains a ring-cycle, it is clear that the ring-contact node graph is not a tree, see figure 2.8(b).

Our approach is based on the idea used by Tollis and Xia which consists of specifying an hierarchical structure for the ring cover which will be considered later as the basis of our drawing. This new underlying structure called *the extended-ring-contact node graph* shows the relations between the rings and the relations between the rings and the ring-cycles.

2.3.1 The extended-ring-contact node graph

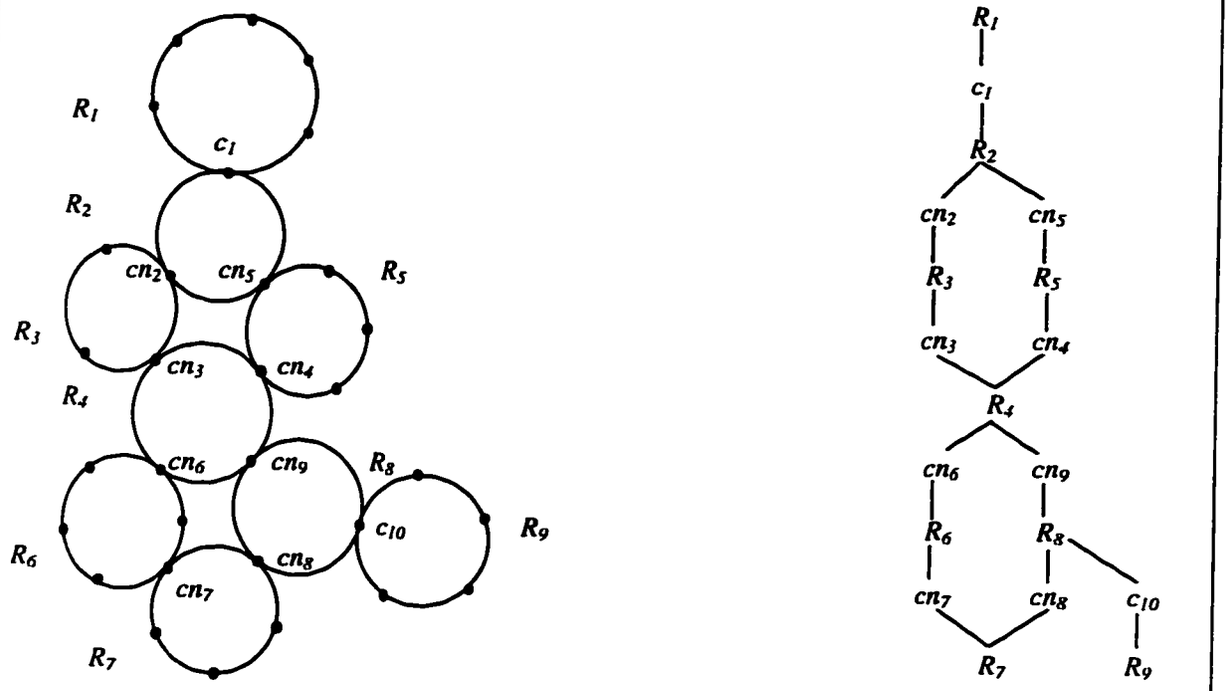


Figure 2.8(a): Ring cover *C*.
(b): Ring-contact node graph *G*.

Suppose that the ring cover contain at least one ring-cycle. We define an *extended-ring-contact node graph* as follows:

First of all, corresponding to each ring-cycle RC, we introduce a dummy vertex RC. This vertex is the root for a subtree, and the children of RC are the special contact nodes and the rings of the ring-cycle RC. Now, we construct the extended-ring-contact node graph G_I (V_I, E_I) such that:

A vertex in V_I could be one of the following:

1. Rings in the ring cover, denoted by R_i.
2. Special contact nodes that belong to the ring-cycle, denoted by cn_i.

3. Contact nodes that do not belong to any ring-cycle, denoted by c_i.
4. Ring-cycles in the ring cover, denoted by RC_i.

E_1: contains four types of edges :

1. $\{(R, c)$: $R \in C$, and $c \in R\}$.
2. $\{(RC, cn)$: $cn \in RC\}$.
3. $\{(cn, R)$: $R \in C$, $cn \in R$, $cn \in RC$ and $R \notin RC\}$.
4. $\{(RC, R)$: $R \in C$ and $R \in RC\}$.

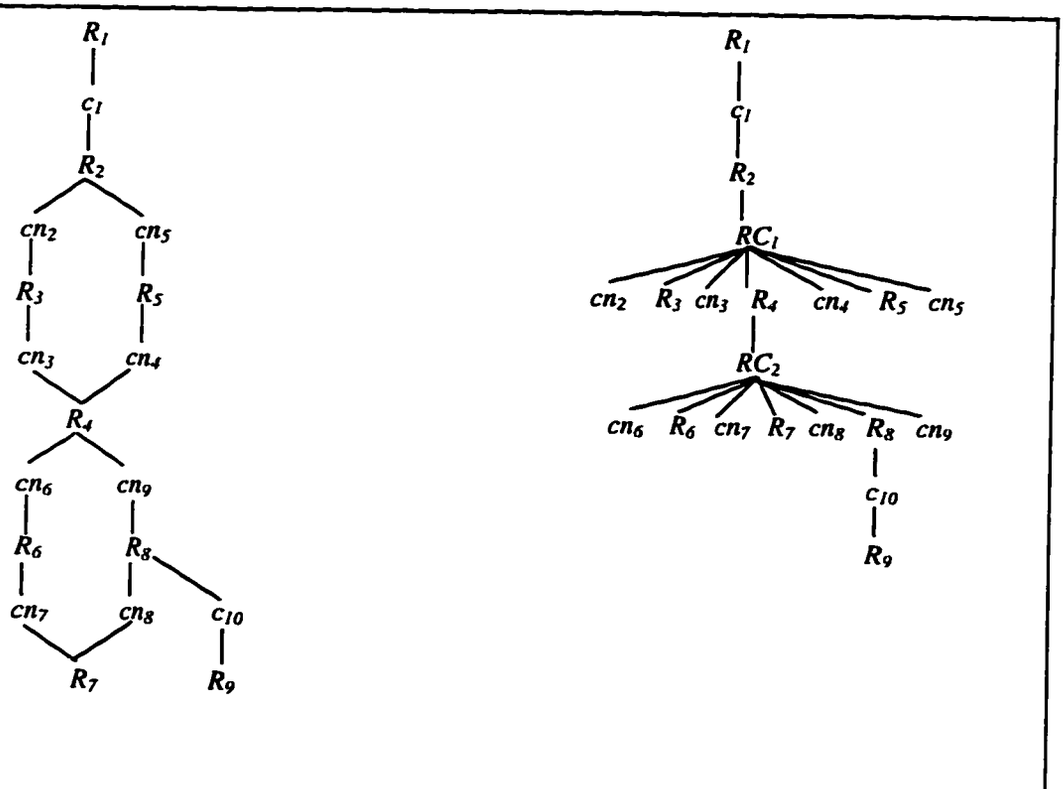


Figure 2.9(a): Ring-contact node graph.

(b): Extended-ring-contact node graph.

For instance in figure 2.8(a), The Ring cover C contains two ring-cycles RC_1 = $\{R_2, cn_2, R_3, cn_3, R_4, cn_4, R_5, cn_5\}$ and $RC_2 = \{R_4, cn_6, R_6, cn_7, R_7, cn_8, R_9, cn_9\}$. We note

that the ring-contact node graph corresponding to the ring cover C is not a tree, meanwhile the extended-ring-contact node graph is a tree (see figure 2.9(b)).

In our drawing, we focus only on the case where the intersection between any two ring-cycles in the ring cover is only one ring or empty. In this case, the extended-ring contact node graph is a tree, we call it T_I.

Let u be a node in T_I, and assume that u has i children, u_1, u_2, ..., u_i. Our algorithm traverses the extended-ring-contact node tree twice:

The first traversal of the tree T_I is in postorder fashion. It consists of computing the radius of the circles corresponding to the rings in the ring cover first. Second, computing the radius of the circles corresponding to the ring-cycles. When all the radius of the circles corresponding to all the children (rings or ring-cycles) of a node u in T_I are computed, the radius of the circle corresponding to the parent node u is computed recursively.

The second traversal of the tree T_I is in preorder fashion. It consists of placing the parent node and its children in the right position.

2.3.2 Radius Calculation

This phase uses standard trigonometric functions to calculate the radius of the circles corresponding to the rings and the ring-cycles.

2.3.2.1 Case 2.1: Father of the ring u is a contact node in T_I

We use the same inside drawing technique (mentioned in the section 2.2) to calculate the radius of the circle Γ corresponding to a node u in T_I. We only add a verification process to the calculation of the radius of the circle Γ' in order to avoid crossings and to preserve the resolution rule. The node u may or may not be a leaf in T_I.

First Case: Node u is a leaf in T_l.

The radius of the circle Γ corresponding to the ring u is computed, such that nodes are distributed uniformly on the perimeter of the circle Γ and are distant by *resolution* (see figure 2.10). The radius of the circle Γ of u is determined using the following procedure:

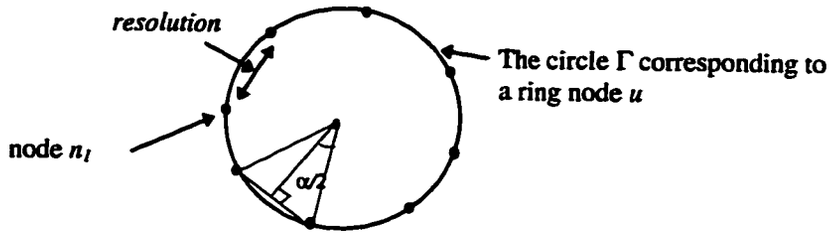


Figure 2.10: Placing nodes, distant by *resolution*, on the perimeter of the circle Γ.

Procedure *Compute_Radius_Ring1*

$Radius = (resolution / (2 \times sin(\pi / n)))$.

where n is the number of nodes in the ring u.

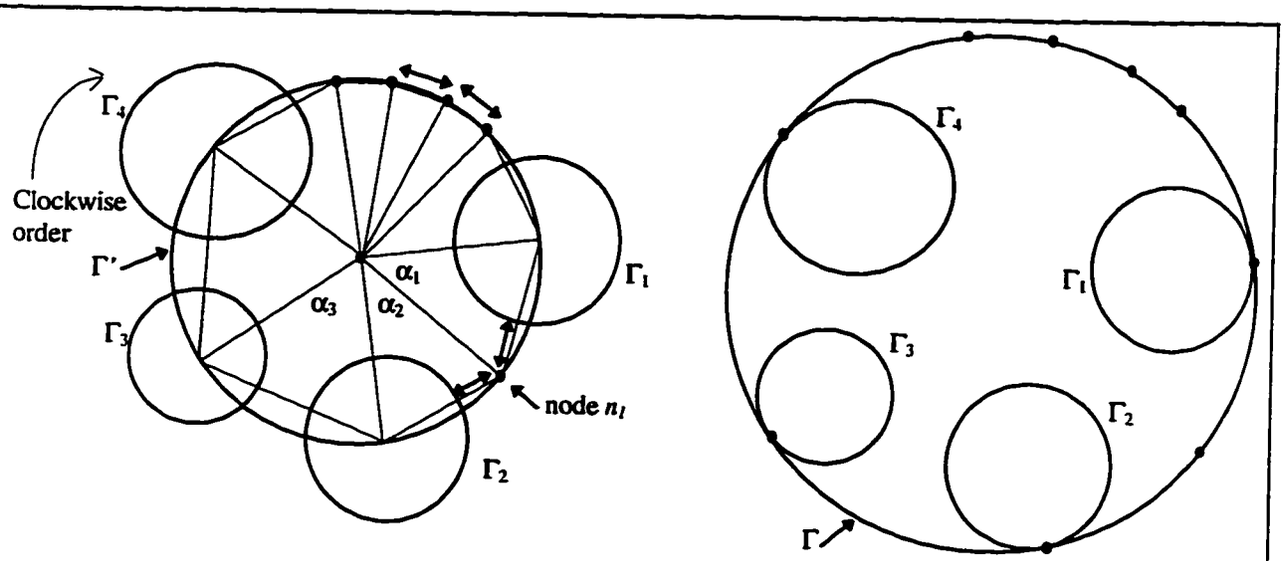
Second Case: the node u is not a leaf in T_l.

We assume that the ring node u has i children $u_1, u_2, \ldots, u_i$ in T_l. These children could be either rings or ring-cycles. After computing the radius of the circles Γ_j corresponding to the children u_j, for $j = 1, \ldots, i$, we compute the radius of the circle Γ corresponding to the parent ring node u as described in the following procedure:

Procedure *Compute_Radius_Ring2*

First, we compute the radius of a circle Γ', by placing each center of the circle Γ_j corresponding to u_j, and each node of u as follow:

We start by placing the center of the first circle or node on the perimeter of the circle Γ', then we place the second circle or node following the clockwise order, on the perimeter of the circle Γ', such that it is distant from the first circle or node by *resolution* (see figure 2.11(a)), then we place the following circles or nodes on the perimeter of the circle Γ', such that they are distant from the last node or circle by *resolution*.



↔ : *resolution*

Figure 2.11(a) : Placing the circles Γ_j on the perimeter of the circle Γ', such that all the circles and the nodes are distant by *resolution*.

(b) : Drawing of the circle Γ that encloses the circle Γ' and the circles Γ_j.

Let $R_{\Gamma'}$ be the radius of the circle Γ', in order to calculate the radius $R_{\Gamma'}$, we use the bisection method (for more details see [RIDO88]) to solve the following equation

$F(R_{\Gamma'}) = 0$, where $F(R_{\Gamma'})$ is equal to $2\pi - \sum_{i=1}^{n} \alpha_i$.

n is the number of nodes of u plus the number of the circles Γ_j.

α_i is the angle formed by a segment joining the center of the circle Γ' and the center of a circle or a node on the perimeter of the circle Γ' and another segment joining the center of the circle Γ' and the center of an adjacent node or circle (see figure 2.11(a)).

α_i is determined as follows:

$\alpha_i = 2\ arcsin\ (\ (\ (\ Radius_i + Radius_{i-1} + resolution)\ /2)\ /\ R_{\Gamma'})$.

Such that $Radius_i$ corresponds to the circle Γ_i and $Radius_{i-1}$ corresponds to the adjacent circle Γ_{i-1}. We remind that a node has a null radius.

In the first process of the calculation of the radius of the circle Γ', we can not ensure that there is no crossing between the circles placed on the perimeter of the circle Γ' (see figure 2.12(a)). The crossing may occurs when the circle Γ_1 and the circle Γ_2 have

both a maximal radius. In this case, we will not be able to pull further Γ_1 and Γ_2 (see section 2.2) since they already touch the perimeter of the circle Γ. That's why we are going to introduce a verification procedure in order to cope with this problem.

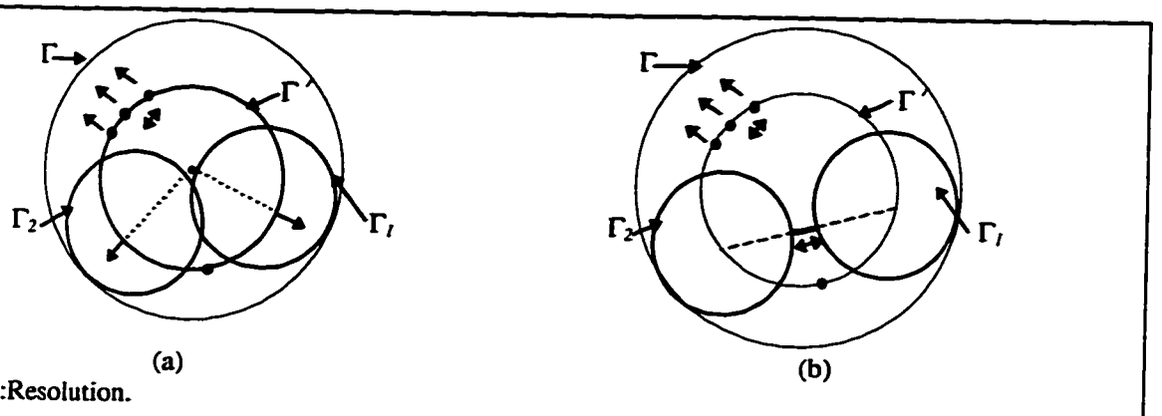


Figure 2.12 (a) : Radius of the circle Γ_1 and Γ_2 are maximal.
(b) : Computing the new radius of the circle Γ' such that the circle Γ_1 and the circle Γ_2 are distant by a resolution.

Procedure *Verification*

The idea is to apply the process of zoom out on all the circles already placed on the perimeter of the circle Γ'. Let S_1 be the set of all the circles placed on the perimeter of the circle Γ', and let S_2 be the set of values representing the radius of the circle Γ'.

For each circle Γ_j in S_1, we check if the circle Γ_j is not distant from all the other circles Γ_k ($k \neq j$) by at least *resolution*. If it is the case, we move Γ_j towards the perimeter of the circle Γ' following the vector (center(Γ'), center(Γ_j)) and we move Γ_k towards the perimeter of the circle Γ' following the vector (center(Γ'), center(Γ_k)), such that the circle Γ_j and the circle Γ_k are distant by a *resolution* see figure 2.12(b).

Next we add the new radius of the circle Γ' in S_2, at the same time we remove Γ_j from S_1, and we repeat this process for the remaining elements in S_1.

Finally, we pick up the maximum value among the elements in S_2 which will represent the final radius of the circle Γ'. Then, we pull each node and each circle Γ_j towards the perimeter of the circle Γ' following the vector (center(Γ'), center(Γ_j)). By

doing so we guarantee that there will be no crossings between all the circles placed on the perimeter of the circle Γ' and that they are distant by at least *resolution*.
Here is the layout of the verification algorithm:

Algorithm *Verification*
Begin
For all the circles Γ_j in S_1 **do**
Begin
For all the circles Γ_k in S_1 where $k \neq i$ **do**
Begin
If (the circle Γ_j is not distant from the circle Γ_k by at least a resolution) **then**
- Compute the new radius $R_{\Gamma'}$, such that the circle Γ_j and the circle Γ_k are distant by at least a resolution
- Add $R_{\Gamma'}$ to S_2;

EndIf
EndFor
- Delete Γ_j from S_1

{we delete Γ_j from S_1 because we have already checked with all the other circles}
EndFor
- $R_{\Gamma'} \longleftarrow$ Max(S_2)

{The radius of the circle Γ' is equal to the maximum radius in the set S_2}
End

The time complexity of the verification algorithm is $O(m^2)$, where m is the number of the circles placed on the perimeter of the circle Γ', because for each circle corresponding to a ring or ring-cycle placed on the perimeter of the circle Γ', we check if is far enough from all the other placed circles by *resolution*.

Next, we draw a circle Γ which has the same center as the circle Γ' and a radius equal to the radius of the circle Γ' plus the maximum radius of the circles Γ_j. Finally, each node and each circle Γ_j is moved out of the circle Γ' following the vector (center(Γ'), center(Γ_j)) towards the perimeter of the circle Γ (see figure 2.11(b)).

Let's consider the example in figure 2.11(a), we compute first β as follow:

$$\beta = \sum_{j=1}^{m} \varnothing(\Gamma_j) + n \times \textit{resolution}.$$

Where m is equal to the number of rings and ring-cycles corresponding to the children of the ring u and n is equal to m plus the number of nodes of the ring u. in figure 2.11(a) m is equal to 4 and n is equal to 9. We compute the default radius value of the circle Γ' by applying the bisection method using the two initial values a and b. Where a is equal to the: Max(Max(radius of the circles Γ_j) , (β/2π)) and b denotes a big number which is equal to β/2.

Second, we call the verification procedure to ensure that all the circles are far from each other by at least a *resolution*. Finally, we compute the radius of the circle Γ which enclose the circle Γ' and all the other circles Γ_j (see figure 2.11 (b)), such that the radius of the circle Γ is equal to the radius of the circle Γ' plus the maximum radius of the circles Γ_j.

2.3.2.2 Case 2.2: Father of the ring u is ring-cycle in T_l.

In this section, we will present the technique for computing the radius of the circle Γ corresponding to the ring node u which father is a ring-cycle in T_l (see figure 2.13(a)).

We assume that u has i children $u_1, u_2, \ldots, u_i$. If all the radius of the circles Γ_j of u_j for $j = 1,2, \ldots, i,$ have been determined, the radius of the circle Γ corresponding to the node u is computed using the following procedure:

Procedure *Compute_Radius_Ring3*

First, we compute the radius of a circle Γ', such that we place all the nodes of u and the centers of the circles Γ_j on the perimeter of the circle Γ'. The distance between the

nodes and the circles is equal to the minimum distance required by the resolution rule, see figure 2.13(b).

Let m be the number of nodes between the two special contact nodes cn_i and cn_{i+1} in the ring u. We compute the value d, such that it is equal to m minus one multiplied by the *resolution*. In figure 2.13(a), these nodes are $\{cn_1, n_1, n_2, cn_2\}$, therefore d is equal to ((4-1) × *resolution*). We place the set of nodes $\{cn_1, n_1, n_2, cn_2\}$ on the circle Γ' as shown in figure 2.13(b). We use the same process for computing the radius of the circle Γ' as described in the section 2.3.2.1 (second case).

Next, we draw a circle Γ which has the same center as the circle Γ' and a radius equal to the radius of the circle Γ' plus the maximum radius of the circles Γ_j.

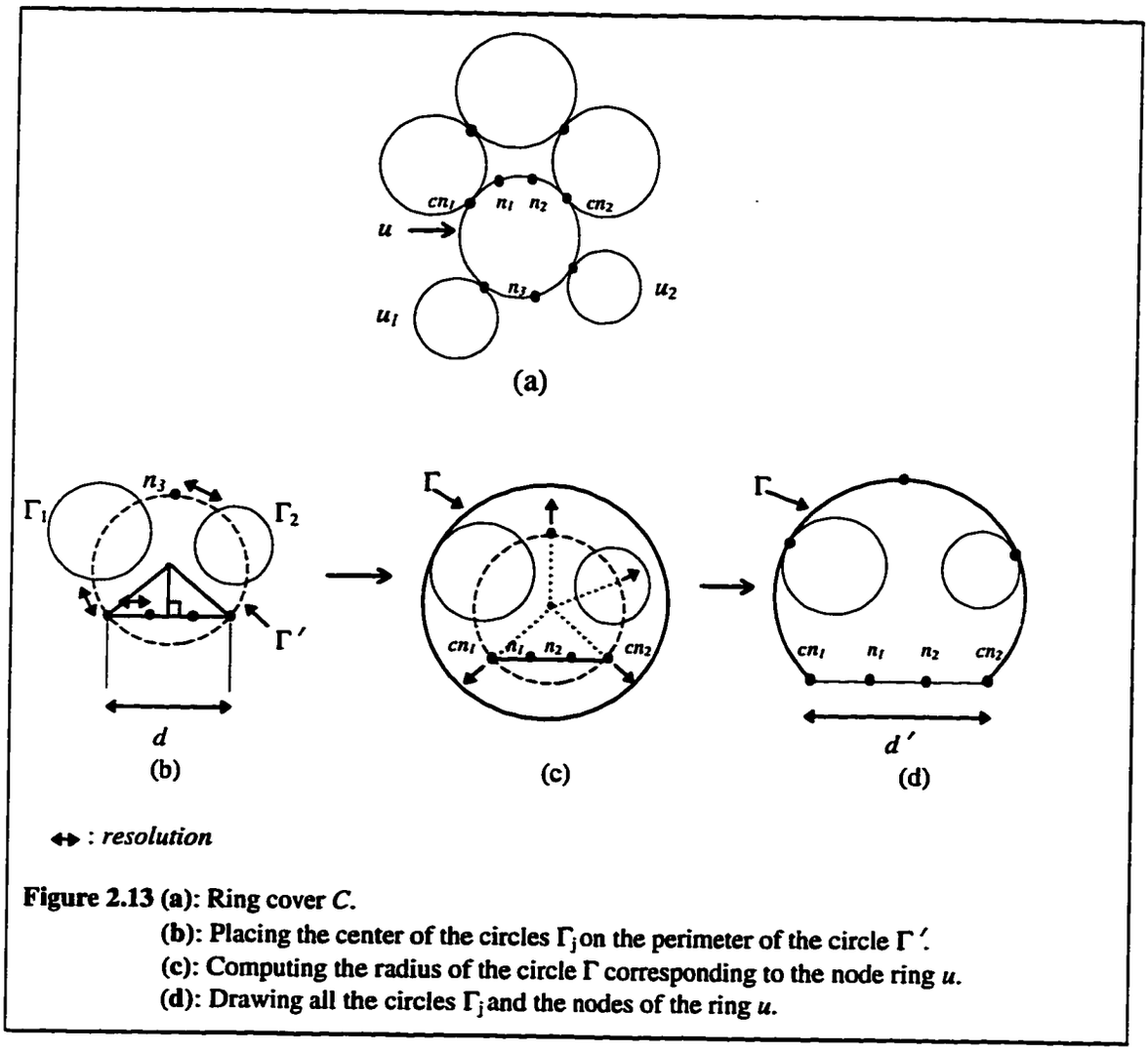


Figure 2.13 (a): Ring cover C.
(b): Placing the center of the circles Γ_j on the perimeter of the circle Γ'.
(c): Computing the radius of the circle Γ corresponding to the node ring u.
(d): Drawing all the circles Γ_j and the nodes of the ring u.

Finally, each node and circle Γ_j is moved out of the circle Γ' following the vector (center(Γ'), center(Γ_j)) towards the perimeter of the circle Γ. Also, we move the two special contact nodes cn_i and cn_{i+1} (in figure 2.13(a), these contact nodes are cn_1 and cn_2) until they touch the perimeter of the circle Γ. We calculate the new value d' as shown in figure 2.13(d). Note that the resolution rule is preserved all along this process. The radius of the circle Γ is equal to the radius of the circle Γ' plus the maximum radius of the circles Γ_j.

2.3.2.3 Case 2.3: Node u is ring-cycle in T_1

Due to the closure of the rings on themselves in the ring-cycle (see figure 2.14(a)), it is very difficult to draw these rings connected to each other, such that we respect the resolution rule, and we guarantee that there are no crossings in their drawing.

Our idea is to suppose that all these rings are detached, and that in the final drawing, for each two adjacent rings, we duplicate their contact node and we join them by a special arc which does not represent a real physical link. We modify the color and the width of these special arcs to emphasis that they are different from physical links in the network, see figure 2.14(d) and figure 2.17.

We compute the radius of a circle Γ corresponding to the ring-cycle u which will contains all the children of u in T_1, then we place the circle Γ inside its parent node in T_1 which is usually a ring. The radius of the circle Γ corresponding to the node u is computed as described in the following procedure:

Procedure *Compute_Radius_Ring-cycle*

First we compute the radius of a circle Γ', by placing all the centers of the circles Γ_j corresponding to the children of u, on the perimeter of the circle Γ'. The distance between the circles Γ_j is equal to the minimum distance required by the resolution rule, see figure 2.14(b).

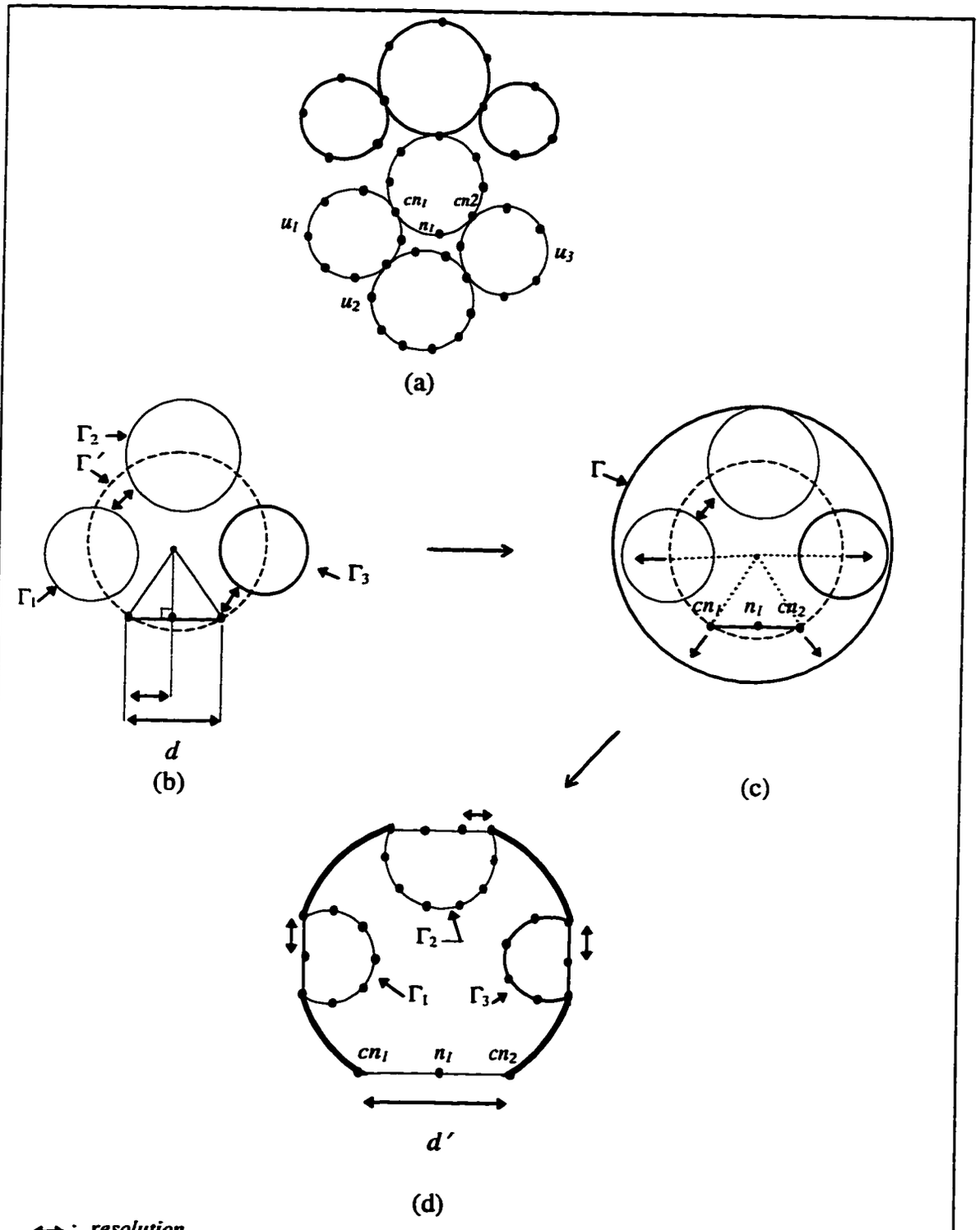


Figure 2.14(a): Ring cover C.
(b): Placing the center of the circles Γ_j on the perimeter of the circle Γ'.
(c): Computing the radius of the circle Γ corresponding to node u.
(d): Drawing all the circles Γ_j children of the ring-cycle u.(result).

Let m be the number of nodes between the two special contact nodes cn_i and cn_{i+1} that belong to the parent ring of u. We compute the value d, such that it is equal to m minus one multiplied by the *resolution*. In figure 2.14(a), the nodes that belong to the parent ring of u are $\{cn_1, n_1, cn_2\}$, therefore d is equal to ((3-1) × *resolution*). We place the set of nodes $\{cn_1, n_1, cn_2\}$ on the circle Γ' as shown in figure 2.14(b). We use the same process for computing the radius of the circle Γ' as described in the section 2.3.2.1 (second case).

Next, we draw a circle Γ which has the same center as the circle Γ' and a radius equal to the radius of the circle Γ' plus the maximum radius of the circles Γ_j.

Finally, we move each circle Γ_j out of the circle Γ' following the vector (center(Γ'), center(Γ_j)) to the perimeter of the circle Γ, such that the two extremity contact nodes of each circle Γ_j touch the perimeter of the circle Γ (see figure 2.14 (d)). Also, we move the two contact nodes cn_i and cn_{i+1} that belong to the parent ring of u (in figure 2.14(a), these contact nodes are cn_1 and cn_2) until they touch the perimeter of the circle Γ. Then we calculate the new value d' as shown in figure 2.14(d).

2.3.3 Algorithm

The following algorithm traverses the tree T_1 in postorder fashion, after computing the radius of all the circles corresponding to the children of a node in T_1 recursively, the algorithm computes the radius of the circle Γ corresponding to the parent node.

Algorithm *InsDraw1_Radius_Calculation (v)*;
Input : Ring cover C, and its extended-ring-contact node tree T_1 with root v.
Output : Radius of the circles corresponding to the rings and the ring-cycles in C.
Begin
1. If ((v is a node representing a ring) **and** (v is a leaf) **and** (father(v) is a contact node)) **then**
- Call *Compute_Radius_ring1(v)* (Computation of the radius of the circle Γ corresponding to v, such that nodes are placed uniformly on the perimeter of the circle Γ, and are distant by *resolution*, see section 2.3.2.1-first case), **return;**

EndIf;

2. **If** (((*v* is a node representing a ring) **and** (father(*v*) is a contact node)) **or** *v* is a root)
   Begin
   - **For** all the children u_i of *v* **do** *InsDraw1_RadiusCalculation* (u_i);
   - Call *Compute_Radius_ring2(v)* (Computation of the radius of the circle Γ corresponding to *v* as described in section 2.3.2.1- second case).

   EndIf;
3. **If** ((*v* is a node representing a ring) **and** (father(*v*) is a ring-cycle)) **then**
   Begin
   - **For** all the children u_i of *v* **do** *InsDraw1_RadiusCalculation* (u_i);
   - Call *Compute_Radius_ring3(v)* (Computation of the radius of the circle Γ corresponding to *v* as described in section 2.3.2.2.)

   EndIf;
4. **If** (*v* is a ring-cycle) **then**
   Begin
   - **For** all the children u_i of *v* **do** *InsDraw1_RadiusCalculation* (u_i);
   - Call *Compute_Radius_ring-cycle(v)* (Computation of the radius of the circle Γ corresponding to *v* as described in section 2.3.2.3.)

   EndIf;
5. **If** (*v* is a node representing a contact node and *v* has only one child *u* ') **then**
   - *InsDraw1_RadiusCalculation* (*u* ');

   EndIf;
6. **If** (*v* is a node representing a contact node and *v* has more than one child) **then**
   Begin
   - Perform a transformation to the tree T_1 as shown in figure 2.3.
   - *InsDraw1_RadiusCalculation* (T_1).

   EndIf;

End.

Analysis of the complexity

We use adjacency list to represent the ring cover (see figure 2.15(b)), as consequence, we give the following analysis of complexity.

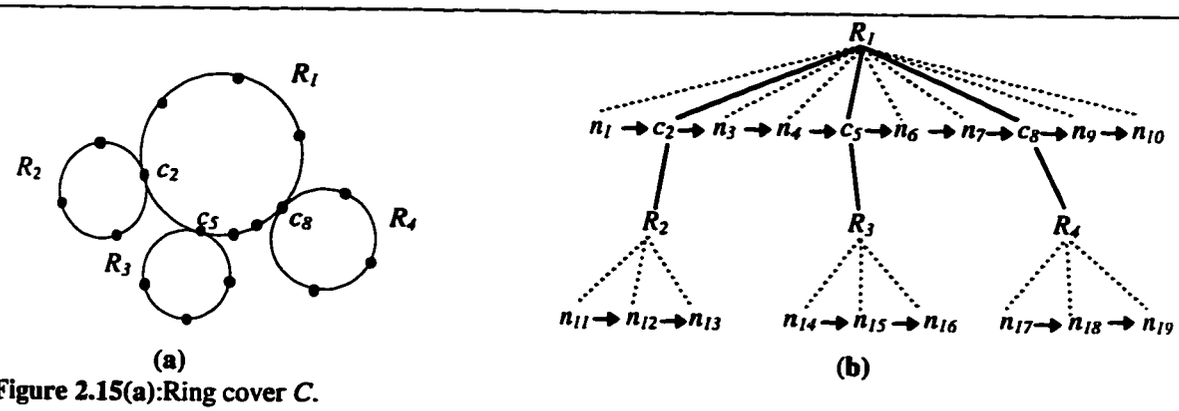


Figure 2.15(a):Ring cover *C*.
(b):Data structure of the ring cover, the continued line represent the underlying structure of *C*.

Let U be the set of nodes that represent the rings or the ring-cycles in the tree T_1. For each node u_i of U, our algorithm needs at most $n_i + m_i^2$ operations to compute the radius of the circle corresponding to u_i, n_i represents the number of nodes in u_i (if u_i is a ring), m_i is the number of the rings and the ring-cycles corresponding to the children of u_i. For each circle corresponding to a ring or a ring-cycle placed on the perimeter of the circle Γ' of u_i, we perform a verification process in order to check if it is far enough by *resolution* from all the other circles placed on the perimeter of the circle Γ' (see section 2.3.2.1 second case). So, we need m_i^2 verifications.

The *InsDraw1_RadiusCalculation* algorithm computes first, the children, of u_i then the parent node itself. We repeat the same process recursively with the parent of u_i, so that, at the end, the algorithm processes n times, where n is the number of nodes in the ring cover, and processes $m_1^2 + m_2^2 + \ldots + m_i^2$ times, where m_j (for $j = 1,2, \ldots, i$) is the number of the children of u_j, that correspond to the rings and the ring-cycles in T_1. It follows that, the time complexity is $O(n + m^2)$. where n is the number of nodes in the ring cover and m is the number of the rings in the ring cover.

2.3.4 Placement node and drawing phase

After computing the radius of each circle corresponding to a ring R or a ring-cycle RC of a node u in T_1. We traverse the tree T_1 in preorder fashion in order to place the nodes and the rings in their proper position.

We place the nodes and the rings based on the polar coordinate system, using the method described previously. This is the case for the other drawing techniques as well (i.e. in [KMG88]).

The coordinate of all the children u_i (nodes, rings, ring-cycles) with respect to their father v are computed as follow :

$u_x = v_x + Radius \times cos(\alpha)$.

$u_y = v_y + Radius \times sin(\alpha)$.

(v_x, v_y): is the coordinate of the center of the circle Γ corresponding to the parent node v which may be a ring or a ring-cycle in T_l.

(u_x, u_y): is the coordinate of the node u, children of v.

Radius: the radius of the circle Γ corresponding to the parent node v in T_l.

α : angle corresponding to the child u with respect to its father v in the circle Γ.

Algorithm *InsDraw1_PlacePhase* (v, v_x, v_y);

Input : Radius of the circles corresponding to the rings and the ring-cycles in C, computed by *InsDraw1_RadiusCalculation*, v represent the root of T_l.

Output : An inside drawing of the ring cover C.

Begin

1. **If** ((v is a node representing a ring) **and** (v is a leaf) **and** (father(v) is a contact node)) **then**
   - Draw the circle Γ of v at the coordinate (v_x, v_y). (The radius of the circle Γ was computed by procedure *compute_radius_ring1* see section 2.3.2.1-first case)

   EndIf;
2. **If** (((v is a node representing a ring) **and** (father(v) is a contact node)) **or** v is a root) **then**
   - Draw the circle Γ of v at the coordinate (v_x, v_y). (The radius of the circle Γ was computed by procedure *compute_radius_ring2* see section 2.3.2.1-second case).

   EndIf;
3. **If** (v is a ring-cycle) **then**
   - Draw the circle Γ of v at the coordinate (v_x, v_y). (The radius of the circle Γ was computed by procedure *compute_radius_ring-cycle* see section 2.3.2.3).

   EndIf;
4. **if** ((v is a node representing a ring) **and** (father(v) is a ring-cycle)) **then**
   - Draw the circle Γ of v at the coordinate (v_x, v_y).(The radius of the circle Γ was computed by procedure *compute_radius_ring3* see section 2.3.2.2).

   EndIf;
5. **If** (v is a node representing a contact node) **then**
   - *InsDraw1_PlacePhase* (u', v_x, v_y);

   EndIf;
6. **For** all the children u_i of v **do** *{u_i could be a node of v}*

   Begin
   - *Calculate_coordinate* (u_i, u_{ix}, u_{iy}); *{using polar technique}*
   - **If** u_i is a node **then**
     - Place the node at the coordinate (u_{ix}, u_{iy});

     Else *{contact node, ring or ring-cycle}*
     - *InsDraw1_PlacePhase* (u_i, u_{ix}, u_{iy});

     Endif;

   EndFor;

End.

Analysis of the complexity

The *InsDraw1_PlacePhase* algorithm runs in $O(n)$ time , where n is the number of nodes in the tree T_1.

In the first step (by means of *InsDraw1_RadiusCalculation*), we already computed the radius of all the circles corresponding to the rings and the ring-cycles. We have also computed the coordinate of all the children with respect to their father. In the second step we draw the parent node then recursively we draw the children.

Let us assume that u_i represents a ring node or a ring-cycle node in the tree T_2, for each node u_i, our algorithm needs $n_i + m_i$ operations to draw the children of u_i. Where n_i is the number of nodes in u_i and m_i is the number of the rings and the ring-cycles that correspond to the children of u_i.

At the end we need $n + m$ operations to draw the ring cover C, where n is the number of nodes in the ring cover and m is the number of the rings plus the number of the ring-cycles. Since $m < n$, the time complexity of *InsDraw1_PlacePhase* algorithm will be $O(n)$.

Using the above procedure, we conclude the following lemma:

Lemma 3

Given a ring cover C and its extended-ring-contact node tree T_1, the inside drawing algorithm (composed by *InsDraw1_RadiusCalculation* and *InsDraw1_PlacePhase*) produces a ring cover drawing such that:

1. There are no crossings;
2. the area required by the drawing is $O(n^2)$, where n is the number of nodes in the ring cover C;

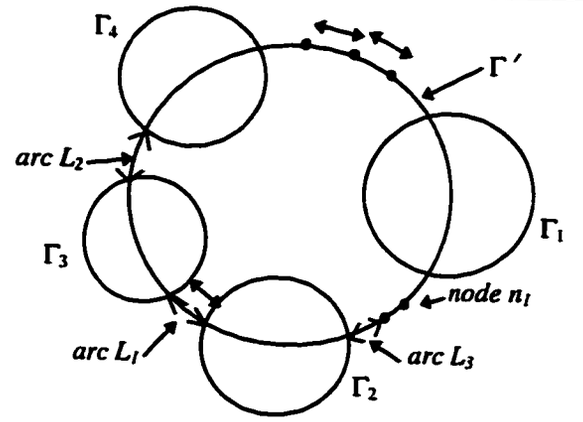


Figure 2.16: Placing the circle Γ_j and nodes on the perimeter of the circle Γ'.

Proof

1. Let u be a node in T_1, we assume that u has i children $u_1, \ldots, u_i$, which are leaves in T_1. We note that there are no crossings in the drawing between the circle Γ corresponding to the node u and the other circles Γ_j corresponding to nodes u_j, for $j = 1,2, \ldots, i$, as described in sections 2.3.2.1, 2.3.2.2, and in section 2.3.2.3. Since we use the same process recursively for the drawing of the parent of the node u, there will be no crossings.

2. Let u be a node in T_1, we assume that u has i children $u_1, \ldots, u_i$, and Γ_j represents a circle corresponding to each node u_j, for $j = 1,2, \ldots, i$. We consider first, that u_j are leaves in T_1, it follows that the order of the radius of these circles Γ_j is $O(n_j)$, where n_j is the number of nodes in each ring u_j.

The node u can be a ring or a ring-cycle, in both cases we use the basic drawing technique, that consists of placing each center of the circle Γ_j and nodes of u (if u is a ring) on the perimeter of the circle Γ' (see figure 2.16). We assume first that u is a ring . The perimeter of the circle Γ' respects the following inequality:

$$2\pi \times \text{radius}(\Gamma') \leq 2\pi \times \text{radius}(\Gamma_1) + \ldots + 2\pi \times \text{radius}(\Gamma_i) + (n + i) \times L.$$

where: i is the number of the circles Γ_j, for $j = 1,2, \ldots, i$.

n is the number of nodes in the ring u, and

L is the longest arc between two adjacent circles in Γ' (in figure 2.16, this arc could be L_1 or L_2), or between a circle or a node (arc L_3, in figure 2.16), or between two nodes.

$L = max(L_h)$, *where* $h = 1, \ldots, (n+i)$.

Thus

$$2\pi \times \text{radius}(\Gamma') \leq 2\pi \times \text{radius}(\Gamma_1) + \ldots + 2\pi \times \text{radius}(\Gamma_i) + (n + i) \times L.$$
$$\leq 2\pi (k_1 n_1 + \ldots + k_i n_i) + (n + i) \times L.$$

where $k_1, \ldots, k_i$ are constant, and $n_1, \ldots, n_i$ is the number of nodes in each ring u_j, for $j = 1, \ldots, i$,. The distance L is a constant that depends on the *resolution* (defined by the user), it follows that:

$$\text{radius}(\Gamma') \leq K (n_1 + \ldots + n_i + n + i).$$

where K is a constant which is bigger than k_j for $j = 1, \ldots, i$, and L.
We conclude that:

$$\text{radius}(\Gamma') \leq K (N + i).$$

where N is the number of nodes in the ring u plus the number of nodes in its children $u_1, \ldots, u_i$.

We know that i (number of the circles Γ_j) is less than N which is equal to the number of nodes in the subtree of u. We deduce that:

$$\text{radius}(\Gamma') \leq 2 K_2 N \leq K_3 N, \text{ where } K_2, K_3 \text{ are constant.}$$

The order of the radius of the circle Γ' is $O(N)$, where N is the number of nodes in the ring u and in its subtree. The radius of the circle Γ corresponding to the node u is

equal to the radius of the circle Γ' plus the maximum radius of the circles Γ_j, this means that the circle Γ has a radius $O(N)$.

If u is a ring-cycle, the same drawing technique is used for the ring-cycle. This technique consists of placing the center of the circles Γ_j corresponding to u_j on the perimeter of the circle Γ' (see figure 2.14(c)). If u is not a root of T_l, the circle Γ corresponding to u has an order $O(N)$, where N is the number of nodes in the subtree of u plus the number of nodes which exist between the two special contact nodes cn_i and cn_{i+1}. These contact nodes cn_i and cn_{i+1} belong to the father ring of u, see figure 2.14(a). If u is a root, N is equal to the number of nodes in the ring cover C.

We repeat the same process recursively for the drawing of the parent of node u. In our final drawing the radius of the circle Γ corresponding to the root v of T_l, has an order $O(N)$, where N is the number of nodes in the ring cover. It follows that, the drawing of the ring cover C has an area of order $O(N^2)$.

Figure 2.17, presents a ring cover drawn with the inside algorithm, we include the case where the ring cover contains ring-cycles.

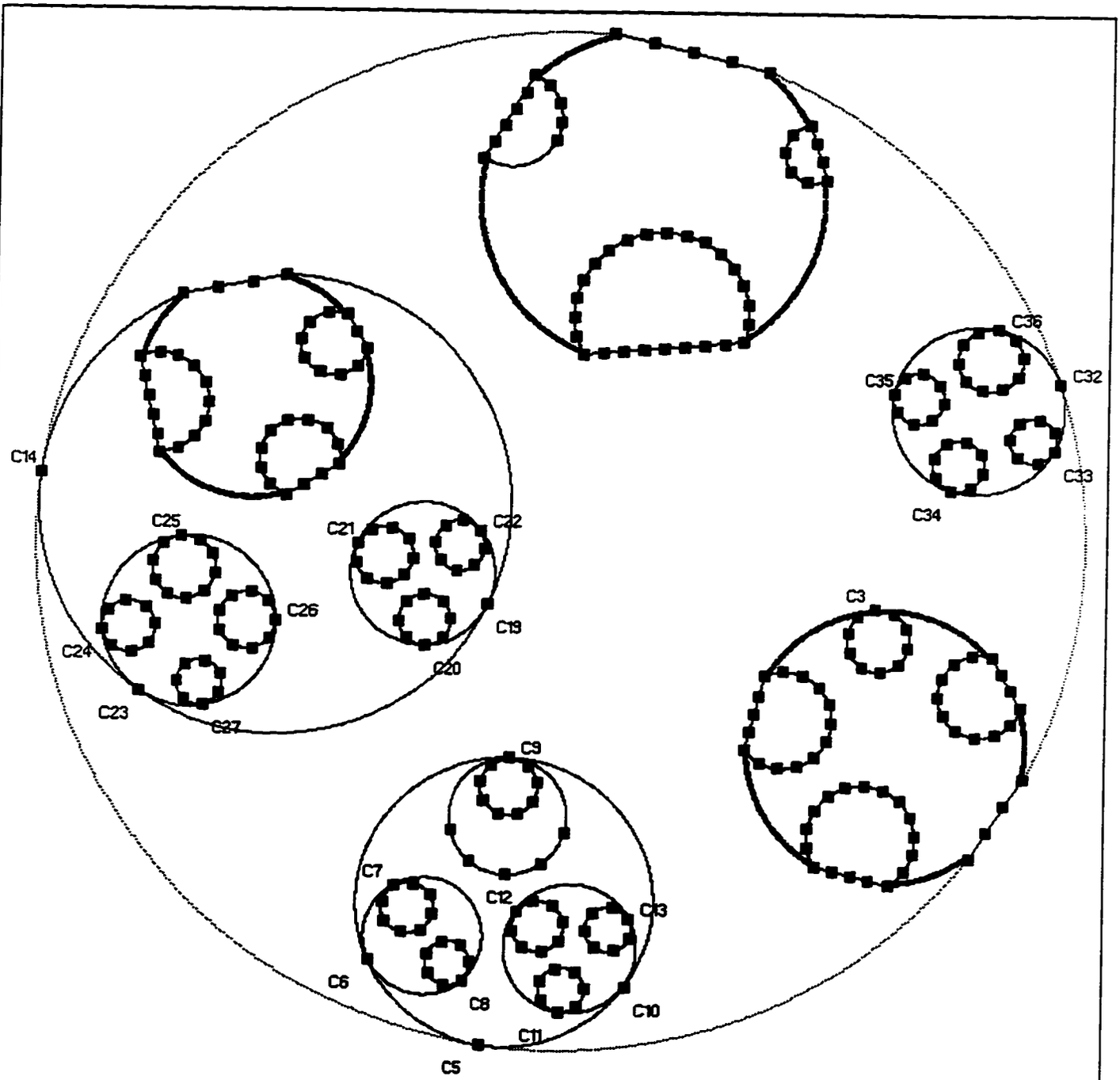


Figure 2.17: Snapshot of the system's screen. Arcs which are represented by a different color and width join the same contact node in the ring-cycle.

2.4 Case 3: Rings share more than one contact node

In this section, we are going to present a new approach for drawing other types of networks in which we consider the case where the rings could share more than one contact node. Before going into details, let us define some concepts:

2.4.1 Definitions

Let a ring $R = \{n_1, n_2, ..., n_p\}$ be an ordered sequence of its nodes, we call two nodes n_i and n_k $(i \neq k)$ adjacent in R, if there exists a direct link e between n_1 and n_2 within the ring R. In figure 2.18, c_1 and c_2 are adjacent nodes in R_1 and in R_2, but c_1 and c_3 are not adjacent nodes in R_1.

Let $R_X = \{n_1, n_2, ..., n_p\}$ and $R_Y = \{m_1, m_2, ..., m_q\}$ be two rings, such that R_X and R_Y are pairwise compatible (see section 2.2), and we suppose that they share k contact nodes $\{c_1, c_2, ..., c_k\}$. A multi-contact node mc is the set $\{c_1, c_2, ..., c_k\}$ of contact nodes, such that c_i, c_{i+1} are adjacent nodes in R_X and in R_Y, where $1 < i < k$-1.

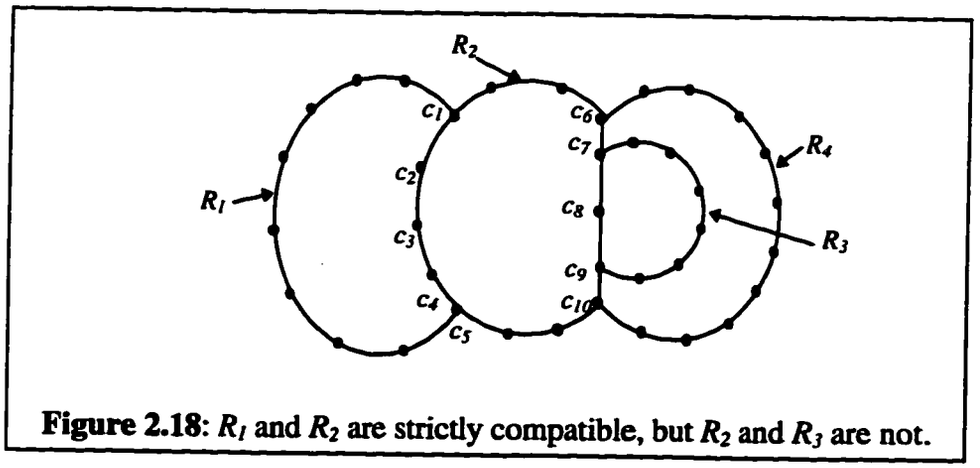


Figure 2.18: R_1 and R_2 are strictly compatible, but R_2 and R_3 are not.

Let $MC = \{mc_1, ..., mc_n\}$ be the set of all multi-contact nodes in the ring cover C and $mc_i \in MC$ be the multi-contact node between two rings R_X and R_Y. We say R_X and R_Y are *strictly compatible* if $mc_i \cap mc_j = \varnothing$ for $j = 1, 2, ..., n$ and $j \neq i$.

In figure 2.18, $MC = \{mc_1, mc_2, mc_3\}$, R_1 and R_2 share the multi-contact node $mc_1 = \{c_1, c_2, c_3, c_4, c_5\}$, R_2 and R_4 share the multi-contact node $mc_2 = \{c_6, c_7, c_8, c_9, c_{10}\}$,

R_2 and R_3 share the multi-contact node $mc_3 = \{c_7, c_8, c_9\}$. R_2 and R_3 are not strictly compatible, because there exists a multi-contact mc_2 where $mc_3 \cap mc_2 = \{c_7, c_8, c_9\}$. Meanwhile, R_1 and R_2 are strictly compatible, because all the contact node c_1, c_2, c_3, c_4, and c_5 belong only to one multi-contact node mc_1.

Our approach consists of specifying a new underlying structure of the ring cover which will be considered later on as the basis of our drawing. We assume first, that any two adjacent rings in the ring cover are strictly compatible, and we shall introduce the *ring-multi-contact node graph* considered to be the new underlying structure of the ring cover.

2.4.2 The ring-multi-contact node graph

We define the *ring-mutli-contact node graph* $G_2(V_2, E_2)$ as follows:

V_2 contains three types of vertices :

1. Rings in the ring cover, denoted by R_i.
2. Contact nodes, denoted by c_i.
3. Mutli-contact nodes, denoted by mc_i.

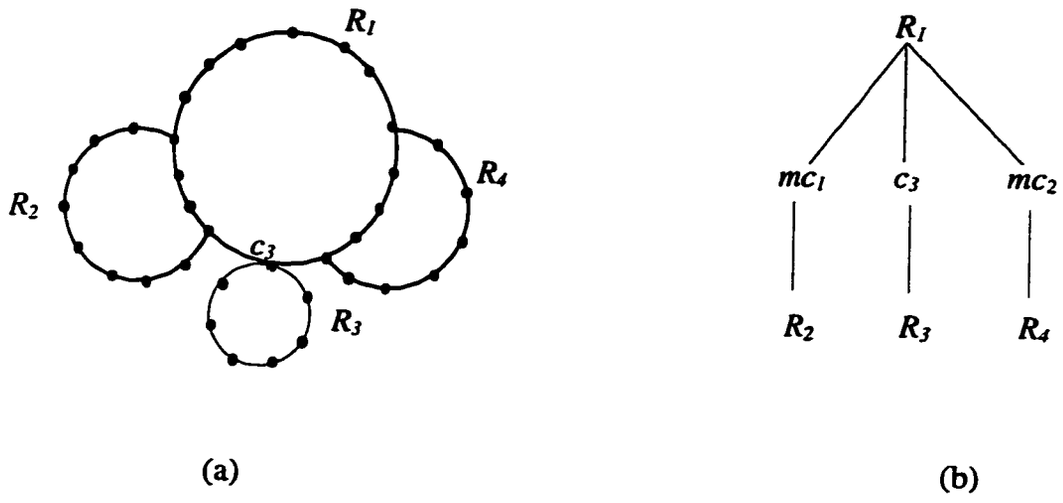
E_2 contains two types of edges :

1. $\{(R, c) : R \in C$, and $c \in R\}$.
2. $\{(R, mc) : R \in C$, and nodes of mc are in $R\}$.

We focus our attention only on the case where the ring-multi-contact node graph G_2 is a tree (see figure 2.19 (b)), we designate it by T_2. Our algorithm traverses the ring-multi-contact node tree two times :

The first traversal of the tree T_2 is in postorder fashion. It consists of computing the radius of the circles corresponding to the rings in the ring cover.

The second traversal of the tree T_2 is in preorder fashion. It consists of placing the parent node and its children in the right location.



(a) (b)

Figure 2.19 (a): Ring cover *C*.
(b): Ring-multi-contact node graph.

2.4.3 Radius Calculation

If the node u is a ring and the father of u is a contact node in T_2, we use the same technique to compute the radius of the circle corresponding to u as described in section 2.3.2.1. If the father of node u is a multi-contact node (see figure 2.20(a)), the drawing of the ring u will be described in the next section.

2.4.3.1 Case 3.1: Father of the ring u is a multi-contact node in T_2

We assume that u has i children $u_1, u_2, \ldots, u_i$. If all the radius of the circles Γ_j of u_j for $j = 1,2, \ldots, i$, have been determined, the radius of the circle Γ corresponding to the node u is computed using the following procedure:

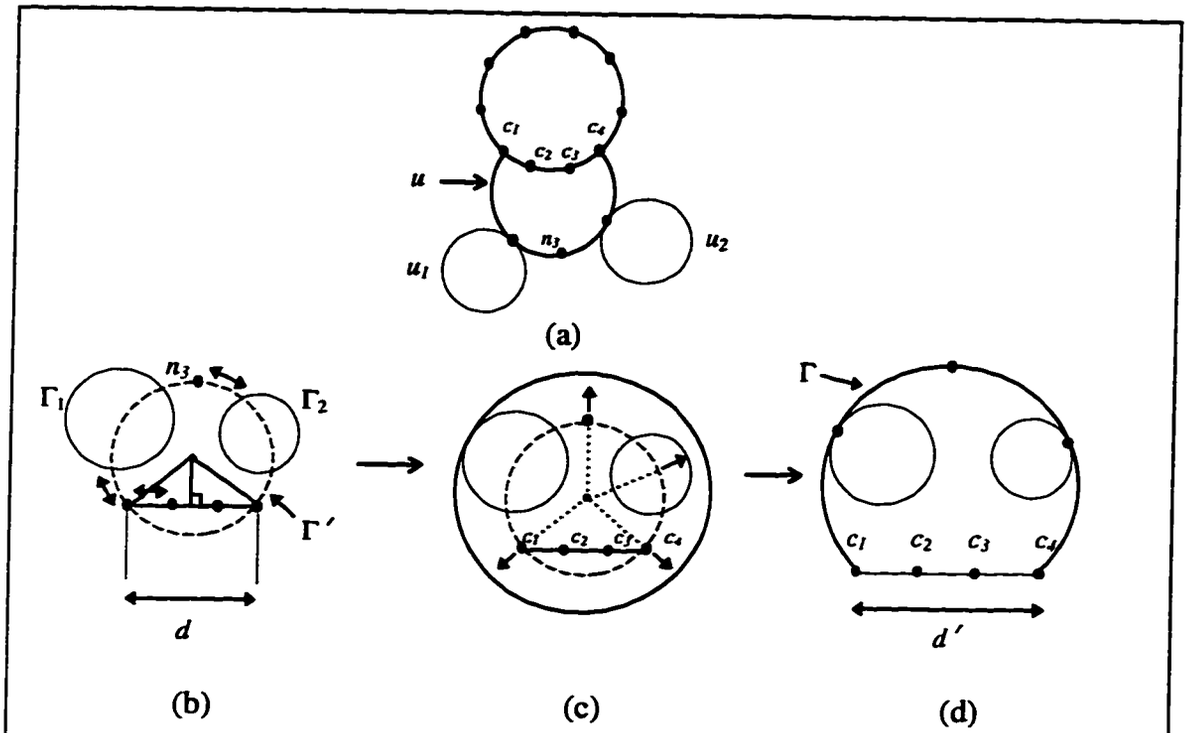


Figure 2.20 (a): Ring cover *C*.
(b): Placing the center of the circles Γ_j on the perimeter of the circle Γ'.
(c): Computing the radius of the circle Γ which corresponds to the node ring *u*.
(d): Drawing all the circles Γ_j and nodes of ring *u*.

Procedure *Compute_Radius_Ring4*

First, we compute the radius of a circle Γ', such that we place all the nodes of *u* and the center of the circles Γ_j on the perimeter of the circle Γ'. The distance between the nodes and the circles is equal to the minimum distance required by the resolution rule, see figure 2.20(b).

Let *m* be the number of contact nodes that belong to the multi-contact node *mc* which is the parent of *u* in T_3. We compute the value *d*, such that it is equal to *m* minus one multiplied by the *resolution*. In figure 2.20(a), the multi-contact node *mc* is $\{c_1, c_2, c_3, c_4\}$, therefore *d* is equal to ((4-1) × *resolution*). We place the set of nodes $\{c_1, c_2, c_3, c_4\}$ on the circle Γ' as shown in figure 2.20(b). We use the same process for computing the radius of the circle Γ' as described in the section 2.3.2.1 (second case).

Next, we draw a circle Γ which has the same center as the circle Γ' and a radius equal to the radius of the circle Γ' plus the maximum radius of the circles Γ_j.

Finally, each node and circle Γ_j is moved out of the circle Γ' following the vector (center(Γ'), center(Γ_j)) towards the perimeter of the circle Γ. Also, we move the two extremities contact nodes of *mc*, (in figure 2.20(a), these contact nodes are c_1 and c_4) until they touch the perimeter of the circle Γ. Then we calculate the new value d' as shown in figure 2.20(d). Note that the resolution rule is preserved all along this process.

In the case where three rings share the same multi-contact node *mc*, the rings are placed one inside the other. The transformation of the multi-contact node tree as shown in figure 2.21(c), consists of drawing a circle that corresponds to a subtree inside the circle which corresponds to another subtree, so that, they share the common multi-contact node *mc*.

Let's take the example shown in figure 2.21(a), the mutli-contact node mc_1 has two children R_4 and R_5. We move R_5 below R_4, so that, the mutli-contact node mc_1 will have only one child. The circle corresponding to the ring R_5 will be drawn inside the circle corresponding to R_4.

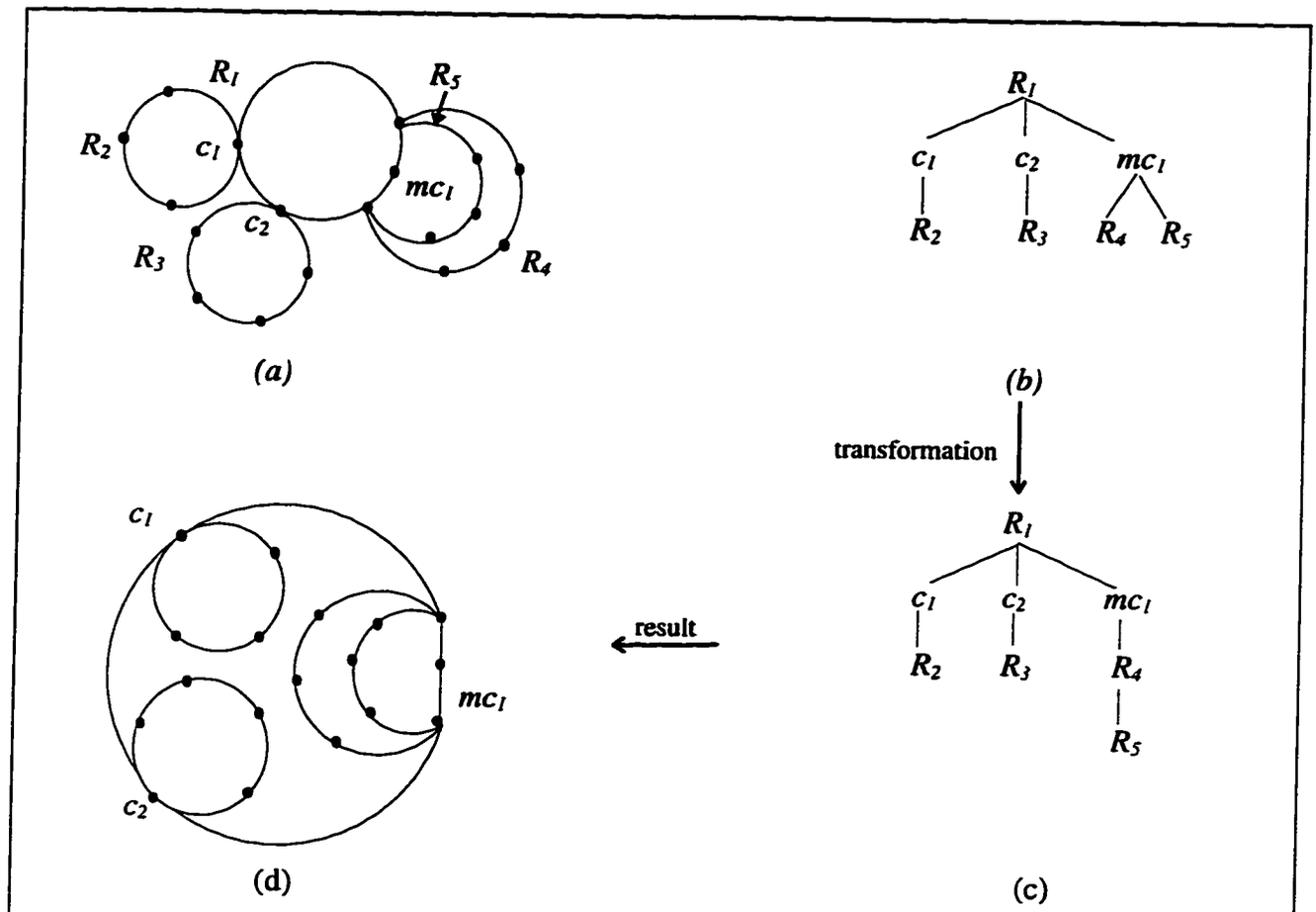


Figure 2.21 **(a)**: Ring cover C.
(b): Ring-multi-contact node tree T_2.
(c): Transformation of the tree T_2. (by means of moving R_5 below R_4).
(d): Final drawing of C, (result) .

2.4.4 Algorithm

Essentially, the algorithm traverses the tree T_2 in postorder fashion, after computing the radius of all the circles corresponding to the children of a node recursively, the algorithm computes the radius of the circle that corresponds to the parent node.

Algorithm *InsDraw2_RadiusCalculation(v)*;

Input : Ring cover C, and its ring-multi-contact node tree T_2 with root v.

Output : Radius of the circles corresponding to the rings in the ring cover C.

Begin

1. If ((v is a node representing a ring) **and** (v is a leaf) **and** (father(v) is a contact node)) **then**

- Call *Compute_Radius_ring1(v)* (Computation of the radius of the circle Γ corresponding to v, such that nodes are placed uniformly on the perimeter of the circle Γ, and are distant by *resolution*, see section 2.3.2.1-first case), **return;**

EndIf;

2. If (((v is a node representing a ring) **and** (father(v) is a contact node)) **or** v is a root) **then**

Begin

- **For** all the children u_i of v **do** *InsDraw2_RadiusCalculation* (u_i);
- Call *Compute_Radius_ring2(v)* (Computation of the radius of the circle Γ corresponding to v as described in section 2.3.2.1- second case).

EndIf;

3. If ((v is a node representing a ring) **and** (father(v) is a multi-contact node)) **then**

Begin

- **For** all the children u_i of v **do** *InsDraw2_RadiusCalculation* (u_i);
- Call *Compute_Radius_ring4(v)* (Computation of the radius of the circle Γ corresponding to v as described in section 2.4.3.1.)

EndIf;

4. If (v is a node representing a contact node **and** v has only one child u') **then**

- *InsDraw2_RadiusCalculation* (u');

EndIf;

5. If v is a node representing a multi-contact node and v has only one child u' **then**

- *InsDraw2_RadiusCalculation* (u');

EndIf;

6. If (v is a node representing a contact node **and** v has more than one child) **then**

Begin

- Perform a transformation to the tree T_2 as shown in figure 2.4(c).
- *InsDraw2_RadiusCalculation* (T_2).

EndIf;

7. If (v is a node representing a multi-contact node **and** v has more than one child) **then**

Begin

- Perform a transformation to the tree T_2 as shown in figure 2.21(c).
- *InsDraw2_RadiusCalculation* (T_2).

EndIf;

End.

The time complexity *InsDraw2_RadiusCalculation* is $O(n + m^2)$, where n is the number of nodes in the ring cover and m is the number of the rings in the ring cover. The same analysis of complexity is described for the *InsDraw1_RadiusCalculation* algorithm (see section 2.3.3).

2.4.5 Placement node and drawing phase

We will present the placement phase algorithm of the rings and the nodes of the ring cover, after computing the radius of the circles corresponding to the rings (see the previous algorithm), we traverse through the tree T_2 in preorder fashion in order to place the nodes and the circles that correspond to the rings in their proper position.

Algorithm *InsDraw2_PlacePhase* (v, v_x, v_y);
Input : Radius of the circles corresponding to the rings in C computed by the *InsDraw2_RadiusCalculation*, v represent the root of T_2.
Output: An inside drawing of the ring cover C.
Begin
1. **If** ((v is a node representing a ring) **and** (v is a leaf) **and** (father(v) is a contact node)) **then**
   - Draw the circle Γ of v at the coordinate (v_x, v_y). (The radius of the circle Γ was computed by procedure *compute_radius_ring1* see section 2.3.2.1-first case)

   EndIf;
2. **If** (((v is a node representing a ring) **and** (father(v) is a contact node)) **or** v is a root) **then**
   - Draw the circle Γ of v at the coordinate (v_x, v_y). (The radius of the circle Γ was computed by procedure *compute_radius_ring2* see section 2.3.2.1-second case).

   EndIf;
3. **If** ((v is a node representing a ring) **and** (father(v) is a multi-contact node)) **then**
   - Draw the circle Γ of v at the coordinate (v_x, v_y). (The radius of the circle Γ was computed by procedure *compute_radius_ring4* see section 2.4.3.1

   EndIf;
4. **If** (v is a node representing a contact node **and** u' is a child of v) **then**
   - *InsDraw2_PlacePhase* (u', v_x, v_y);

   EndIf;
5. **If** (v is a node representing a multi-contact node **and** u' is a child of v) **then**
   - *InsDraw2_PlacePhase* (u', v_x, v_y);

   EndIf;

6. For all the children u_i of v **do** *{u_i could be a node of v}*
Begin
- *Calculate_coordinate(u_i, u_{ix}, u_{iy});*
  {using polar technique described in section 2.3.4}
- **If** u_i is a node **then**
  - Place the node at the coordinate (u_{ix}, u_{iy});

  Else
  - *InsDraw2_PlacePhase (u_i, u_{ix}, u_{iy});*

  Endif;

EndFor;
End.

The time complexity of *InsDraw2_PlacePhase* algorithm is of order $O(n)$, where n is the number of nodes in the ring cover. First, we draw the parent node then recursively we draw the children, we process each node in T_2 only once (see section 2.3.4 for full detailed analysis).

Using the above procedure, we conclude the following lemma :

Lemma 4

Given a ring cover C and its ring-multi-contact node tree T_2. The inside drawing algorithm (composed by *InsDraw2_RadiusCalculation* and *InsDraw2_PlacePhase*) produces a ring cover drawing such that :

1. there are no crossings;
2. the area required by the drawing is $O(n^2)$, where n is the number of nodes in C.

Proof

1. Let u be a node in T_2, we assume that u has i children u_1, ..., u_i which are leaves in T_2, u can be a ring which shares one contact node with the other rings, or it can be a ring which shares more than one contact node with the other rings. In both cases, we use circles to represent the rings, such that the circle corresponding to u encloses u and its subtree. There are no crossings between u and its children (as described in section 2.3.2.1

and in section 2.4.3.1). We repeat the same drawing recursively with the parent of u, so that in the final drawing there will be no crossings.

2. Since we use circles to draw each ring either when the ring shares one contact node or shares more than one contact node with the other rings, and each circle that corresponds to a node u encloses the subtree of u has an area $O(n^2)$, where n is the number of nodes in u and in its subtree. This is due to the fact that we use the technique which consists of placing the center of the circles Γ_j corresponding to the children of u and nodes of u, on the perimeter of the circle Γ'(described in the proof of lemma 3). If u is the root of T_2, the area of the circle will be then $O(n^2)$, where n is the number of nodes in the ring cover.

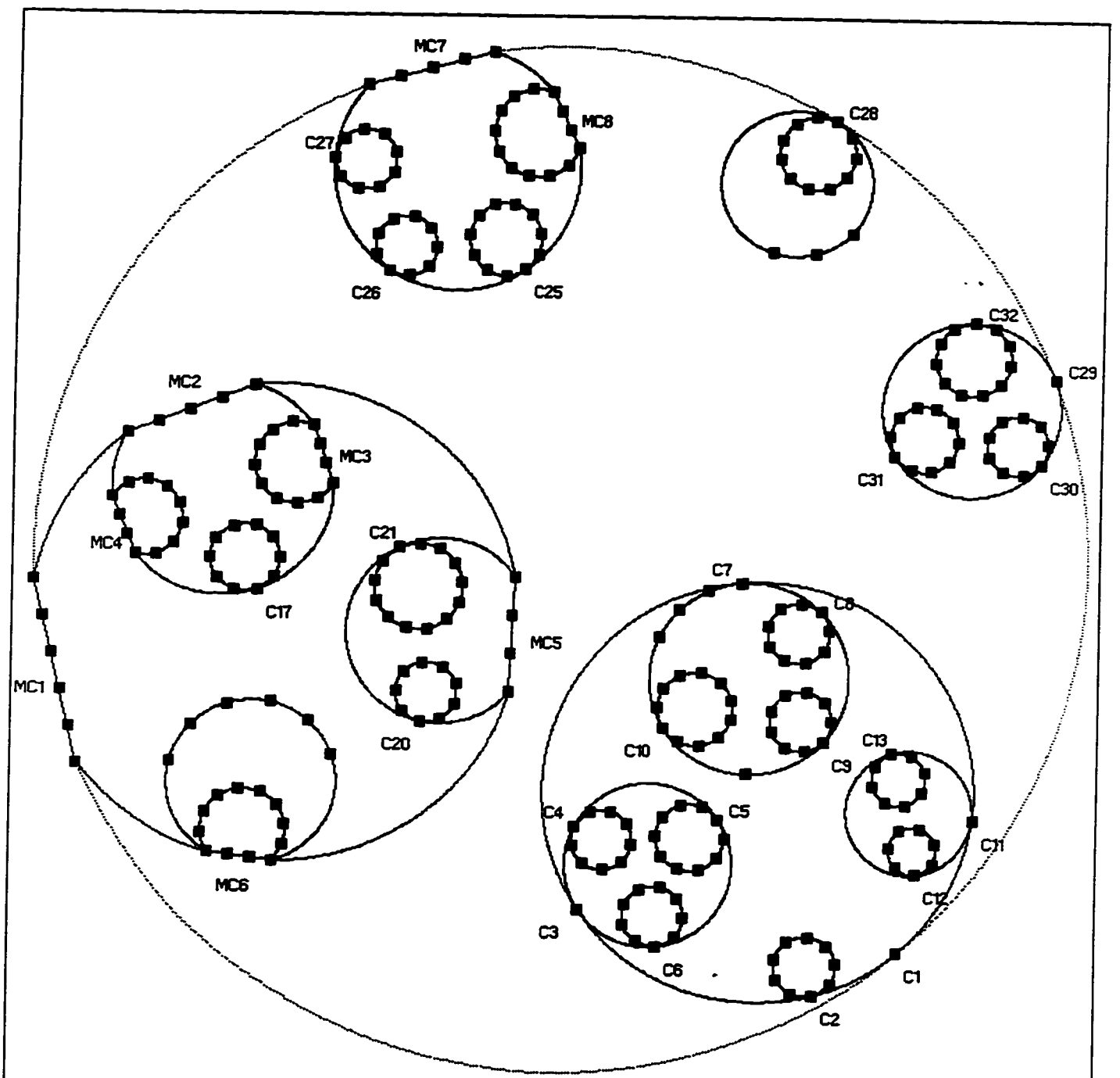


Figure 2.22: The system draws all arcs with the same color and width, these arcs represent physical links between sites. We note that the following ring cover contains rings which share more than one contact node with other rings.

2.5 Case 4 : Ring cover contains ring-cycles and rings share more than one contact node with other rings

In order to deal with a wider range of cases of drawing ring covers, we propose a drawing of other types of networks. We consider the case where the rings share more than one contact node and the ring cover contains ring-cycles. We assume first, that any two adjacent rings in the ring cover are strictly compatible (see section 2.4.1). Second, we assume that the new underlying structure (which will be presented later) of the ring cover is a tree.

We introduce the general underlying structure of the ring cover called *generalized-ring-contact node graph* which includes the previous two underlying structures. The *generalized-ring-contact node graph* will be considered later as the basis of our drawing which shows the relation between the rings themselves and between the rings and the ring-cycles.

2.5.1 The generalized-ring-contact node graph

We define the *generalized-ring-contact node graph* $G_3(V_3, E_3)$ as follows:

V_3 contains five types of vertices :

1. Rings in the ring cover, denoted by R_i.
2. Special contact nodes that belong to the ring-cycle, denoted by cn_i.
3. Contact nodes that do not belong to ring-cycle, denoted by c_i.
4. Ring-cycles in the ring cover, denoted by RC_i.
5. Mutli-contact nodes, denoted by mc_i.

E_3 contains five types of edges defined as follow:

1. $\{(R, c) : R \in C$, and $c \in R\}$.
2. $\{(RC, cn) : cn \in RC\}$.

3. $\{(cn, R) : R \in C, cn \in R, cn \in RC$ and $R \notin RC\}$.
4. $\{(RC, R) : R \in C$ and $R \in RC\}$.
5. $\{(R, mc) : R \in C$, and nodes of mc are in $R\}$.

In our drawing we focus only on the case where the generalized-ring-contact node graph G_3 is a tree (see figure 2.23 (b)), we call it T_3. Our algorithm traverses the tree T_3 two times :

The first traversal of the tree T_3 is in postorder fashion. It consists of computing the radius of the circles corresponding to the rings and the ring-cycles in the ring cover.

The second traversal of the tree T_3 is in preorder fashion. It consists of placing the parent node and its children in the right location.

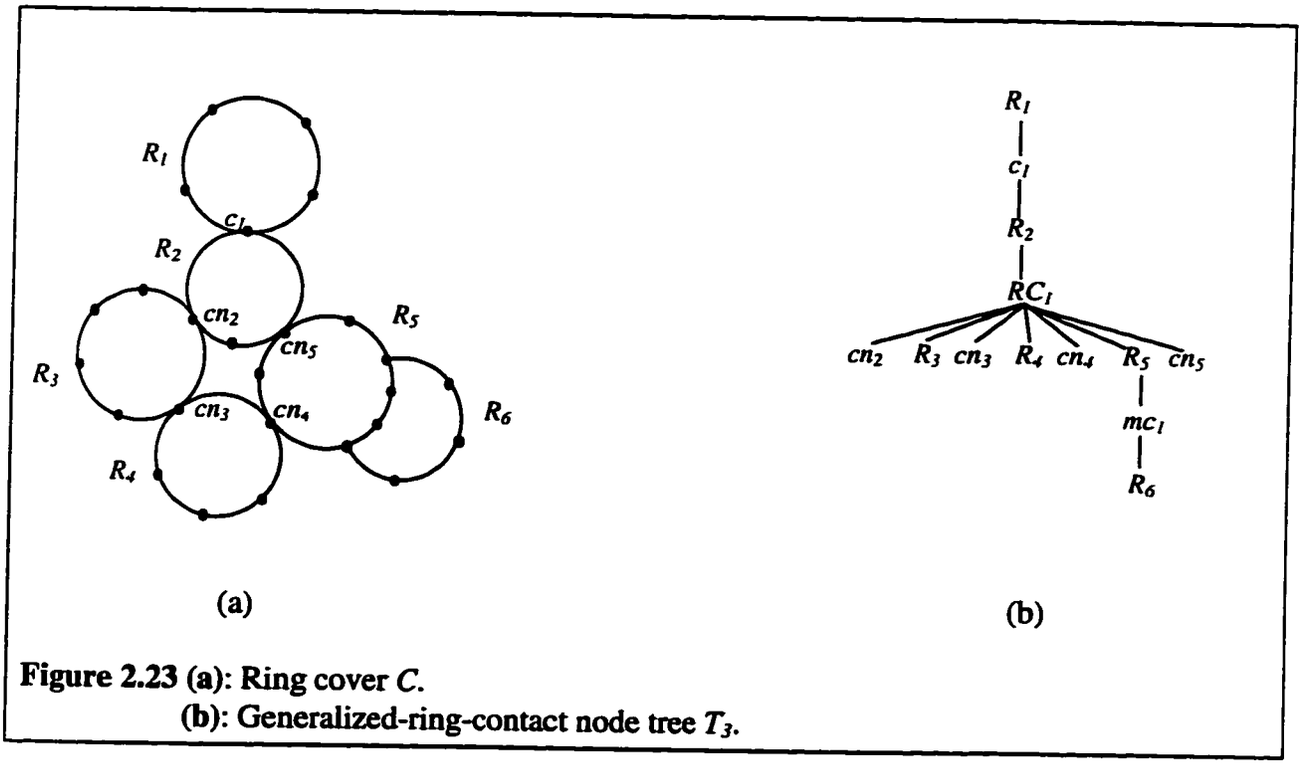


Figure 2.23 (a): Ring cover C.
(b): Generalized-ring-contact node tree T_3.

2.5.2 Algorithm

The following algorithm traverses the tree T_3 in postorder fashion. After computing the radius of all the circles corresponding to the children of u_i whether they are either rings sharing only one contact node with other rings, or rings sharing more than one

contact node, or ring-cycles. The algorithm computes the radius of the circle that corresponds to the parent node u.

Algorithm *InsDraw3_RadiusCalculation(v)*;
Input : Ring cover C, and its generalized-ring-contact node tree T_3 with root v.
Output : Radius of the circles corresponding to the rings and the ring-cycles in C.
Begin
1. **If** ((v is a node representing a ring) **and** (v is a leaf) **and** (father(v) is a contact node)) **then**
   - Call *Compute_Radius_ring1(v)* (Computation of the radius of the circle Γ corresponding to v, such that nodes are placed uniformly on the perimeter of the circle Γ, and are distant by *resolution*, see section 2.3.2.1-first case), **return;**

   EndIf;
2. **If** (((v is a node representing a ring) **and** (father(v) is a contact node)) **or** v is a root) **then**
   Begin
   - **For** all the children u_i of v **do** *InsDraw3_RadiusCalculation* (u_i);
   - Call *Compute_Radius_ring2(v)* (Computation of the radius of the circle Γ corresponding to v as described in section 2.3.2.1- second case).

   EndIf;
3. **If** ((v is a node representing a ring) **and** (father(v) is a ring-cycle)) **then**
   Begin
   - **For** all the children u_i of v **do** *InsDraw3_RadiusCalculation* (u_i);
   - Call *Compute_Radius_ring3(v)* (Computation of the radius of the circle Γ corresponding to v as described in section 2.3.2.2.)

   EndIf;
4. **If** (v is a ring-cycle) **then**
   Begin
   - **For** all the children u_i of v **do** *InsDraw3_RadiusCalculation* (u_i);
   - Call *Compute_Radius_ring-cycle(v)* (Computation of the radius of the circle Γ corresponding to v as described in section 2.3.2.3.)

   EndIf;
5. **If** ((v is a node representing a ring) **and** (father(v) is a multi-contact node)) **then**
   Begin
   - **For** all the children u_i of v **do** *InsDraw3_RadiusCalculation* (u_i);
   - Call *Compute_Radius_ring4(v)* (Computation of the radius of the circle Γ corresponding to v as described in section 2.4.3.1.)

   EndIf;
6. **If** (v is a node representing a contact node **and** v has only one child u') **then**
   - *InsDraw3_RadiusCalculation* (u');

   EndIf;

7. If (v is a node representing a multi-contact node **and** v has only one child u ') **then**
- *InsDraw3_RadiusCalculation* (u ');

EndIf;

8. If (v is a node representing a contact node **and** v has more than one child) **then**
Begin
- Perform a transformation to the tree T_3 as shown in figure 2.4(c);
- *InDraw3_RadiusCalculation* (T_3);

EndIf;

9. If (v is a node representing a multi-contact node **and** v has more than one child) **then**
Begin
- Perform a transformation to the tree T_3 as shown in figure 2.21(c);
- *InDraw3_RadiusCalculation* (T_3);

EndIf;

End.

The time complexity of the *InsDraw3_RadiusCalculation* is $O(n + m^2)$, where n is the number of nodes in the ring cover and m is the number of the rings in the ring cover. The same analysis of complexity is described for the *InsDraw1_RadiusCalculation* algorithm (see section 2.3.3).

2.5.3 Placement node and drawing phase

In the first step (by means of *InsDraw3_Radius Computation* algorithm), we computed the radius of all the circles corresponding to the rings and the ring-cycles. We traverse the tree T_3 in preorder fashion in order to place the nodes and the circles that correspond to the rings and the ring-cycles in their proper position. After drawing the parent node u, the algorithm *InsDraw3_PlacePhase* draws the children of u.

Algorithm *InsDraw3_PlacePhase* (v, v_x, v_y);
Input : Radius of the circles corresponding to the rings in C computed by the *InsDraw3_RadiusCalculation*, v represent the root of T_3.
Output: An inside drawing of the ring cover C.

Begin

1. If ((v is a node representing a ring) **and** (v is a leaf) **and** (father(v) is a contact node)) **then**
- Draw the circle Γ of v at the coordinate (v_x, v_y). (The radius of the circle Γ was computed by procedure *compute_radius_ring1* see section 2.3.2.1-first case)

EndIf;

2. **If** (((*v* is a node representing a ring) **and** (father(*v*) is a contact node)) **or** *v* is a root) **then**
   - Draw the circle Γ of *v* at the coordinate (v_x, v_y). (The radius of the circle Γ was computed by procedure *compute_radius_ring2* see section 2.3.2.1-second case).

   EndIf;
3. **If** (*v* is a ring-cycle) **then**
   - Draw the circle Γ of *v* at the coordinate (v_x, v_y). (The radius of the circle Γ was computed by procedure *compute_radius_ring-cycle* see section 2.3.2.3).

   EndIf;
4. **if** ((*v* is a node representing a ring) **and** (father(*v*) is a ring-cycle)) **then**
   - Draw the circle Γ of *v* at the coordinate (v_x, v_y).(The radius of the circle Γ was computed by procedure *compute_radius_ring3* see section 2.3.2.2).

   EndIf;
5. **If** (*v* is a node representing a ring **and** father(*v*) is a multi-contact node) **then**
   - Draw the circle Γ of *v* at the coordinate (v_x, v_y). (The radius of the circle Γ was computed by procedure *compute_radius_ring4* see section 2.4.3.1).

   EndIf;
6. **If** (*v* is a node representing a contact node **and** *u'* is a child of *v*) **then**
   - *InDraw3_PlacePhase (u', v_x, v_y)*;

   EndIf;
7. **If** (v is a node representing a multi-contact node **and** *u'* is a child of *v*) **then**
   - *InDraw3_PlacePhase (u', v_x, v_y)*;

   EndIf;
8. **For** all the children u_i of *v* **do** *{u_i could be a node of v}*

   Begin
   - *Calculate_coordinate (u_i, u_{ix}, u_{iy})*;
     {using polar technique described in section 2.3.4}
   - **If** u_i is a node **then**
     - Place the node at the coordinate (u_{ix}, u_{iy});

     Else
     - *InsDraw3_PlacePhase (u_i, u_{ix}, u_{iy})*;

     Endif;

   EndFor;

End.

The time complexity of *InsDraw3_PlacePhase* algorithm is of order *O(n)*, where *n* is the number of nodes in the ring cover. First, we draw the parent node then recursively we draw the children, each node in T_3 is processed only once. (see section 2.3.4 for full detailed analysis).

Using the above procedure, we conclude the following lemma :

Lemma 5

Given a ring cover C and its generalized-ring-contact node tree T_3, the inside drawing algorithm (composed by *InsDraw3_RadiusCalculation* and *InsDraw3_PlacePhase*) produces a ring cover drawing such that :

1. there are no crossings;
2. the area required by the drawing is $O(n^2)$, where n is the number of nodes in C;

Proof

1. Let u be a node in T_3, we assume that u has i children u_1, ..., u_i which are leaves in T_3. There are no crossings in the drawing of u and its children u_i (as described in sections 2.3.2.1, 2.3.2.2, 2.3.2.3 and in section 2.4.3.1). Our algorithm repeats recursively the same process of drawing with the parent node of u, so that, in the final drawing there will be no crossings between the rings.

2. Let u be a ring or ring-cycle node in T_3, we assume that u has i children u_1, ..., u_i. Since each node u in the tree T_3 is presented by a circle which encloses u and its subtree, the area of the circle corresponding to u is $O(n)$, where n is the number of nodes in u and in its subtree. This is due to the fact that we use the technique which consists of placing the center of the circles Γ_j corresponding to the children of u and nodes of u, on the perimeter of the circle Γ' (described in the proof of lemma 3). If u is the root of T_3, the area of the ring cover will be then $O(n^2)$, where n is the number of nodes in the ring cover.

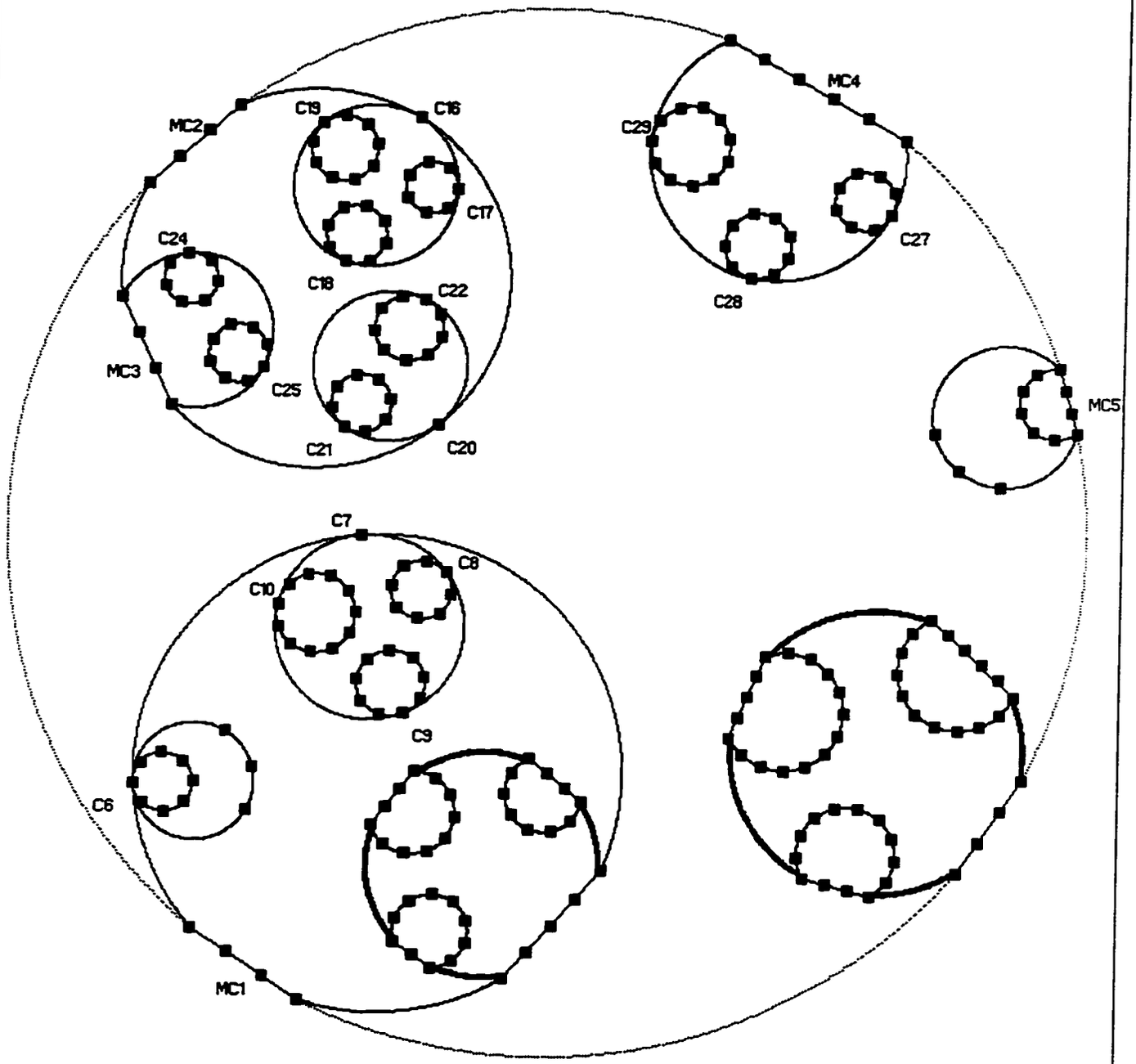


Figure 2.24: Snapshot of the system's screen. The ring cover contains ring-cycles and rings that share more than one contact node with other rings.

Chapter 3

Outside Drawing

3.1 Introduction

We present in this chapter the outside drawing technique which consists of drawing each ring outside the other rings.

In the first part of this chapter, we will present the previous work related to the outside drawing algorithm which was limited to the two following assumptions:

- Any two adjacent rings share only one contact node.
- The ring-contact node graph is a tree.

In the second part, we will present an improved version of the outside drawing algorithm, by taking into consideration other types of networks, for instance, rings which could share more than one contact node, and the ring cover may contains ring-cycles.

Given a ring cover C and its generalized-ring-contact node tree T_3 (presented in chapter 2, section 2.5.1), our algorithm draws rings outside each other, such that the resolution rule is respected, no crossings are generated, and a picture of the ring cover is produced in an optimal area.

3.2 Case 1: Ring-contact node graph is a tree

The structure of the ring-contact node tree T may be complex, and the number of nodes contained in different rings could be drastically diverse. It is important to position each circle corresponding to a ring in its proper position, otherwise, there will be some crossings.

For a ring node u and its subtree in the ring-contact node tree T, the enclosing cycle Γ_u of u is large enough to place the drawing of u and its subtree inside Γ_u as shown in figure 3.1(c). If u is a leaf, the radius of the circle Γ corresponding to u is computed, such that the nodes are placed uniformly on the perimeter of the circle Γ and are distant by *resolution*. The enclosing cycle Γ_u of u is actually the circle Γ itself.

Each node u and its subtree is drawn in a confined area, so that it does not intersect with rings of other subtrees of T. Each ring node u except for the root of T has a parent. Let P_u is the parent contact node of u. P_u is placed on the top half of the circle Γ corresponding to the ring u. All other nodes (including all other contact nodes) are placed on the bottom half perimeter of the circle Γ corresponding to u. By doing so, there are no intersections between the ancestors of u and its children, see figure 3.1 (b).

Let u be a ring node in T and its children: u_1, u_2, ..., u_i. If the radius of all the enclosing cycles Γ_j of u_j for $j = 1, 2, ..., i$, have been computed, then the radius of the circle Γ corresponding to the node u is computed using the following procedure:

Procedure *Compute_radius_ring*

First, a circle Γ' is drawn such that all the nodes of the ring u and the enclosing cycles Γ_j corresponding to u_j are placed evenly on the perimeter of the circle Γ', (see figure 3.1 (a)). The distance between the nodes and the enclosing cycles is equal to the minimum distance required by the resolution rule. Obviously, the diameter of the circle

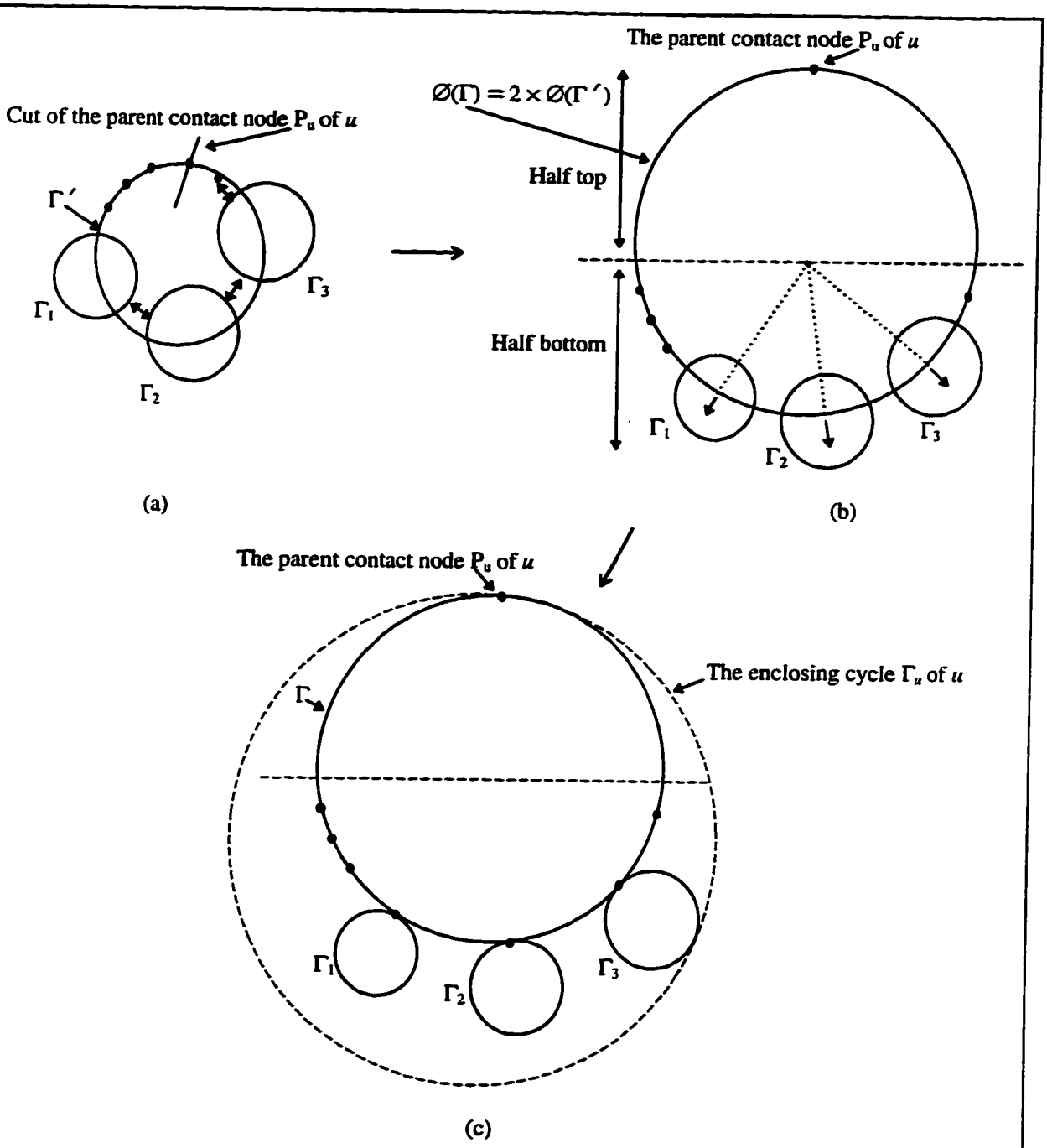


Figure 3.1 (a): Placing the center of the enclosing cycles Γ_j as well as the nodes of u on the perimeter of the circle Γ'.

(b): Drawing the circle Γ with a radius twice the radius of the circle Γ'.

(c): Each circle Γ_j is pulled downward until it touches the perimeter of the circle Γ.

Γ' is larger than the maximum diameter of the circles Γ_j. Second, the circle Γ corresponding to the node u is drawn with a radius twice the radius of the circle Γ'. The reason behind doing so, is to avoid crossings between the ancestors of u and its children. All the other enclosing cycles Γ_j, as well as all the nodes of u, except the parent contact node P_u of u, are placed on the bottom half perimeter of the circle Γ. This can be achieved easily by cutting the perimeter of the circle Γ' at the parent contact node P_u of u, and considering it as the bottom half perimeter of the circle Γ, see figure 3.1(a). The radius of the enclosing cycle Γ_u corresponding to u is equal to the radius of the circle Γ plus the maximum radius of the circles Γ_j.

Finally, each node and enclosing cycle Γ_j is moved towards the perimeter of the circle Γ following the vector (center(Γ), center(Γ_j)), such that each circle Γ_j has one common contact point with the circle Γ, (see figure 3.1(c)). Note that the resolution rule is preserved during this process.

Using the above procedure, Tollis and Xia [TOXI94] proved the following lemma:

Lemma 6

For any given ring node u and its subtree in T, the enclosing cycle Γ_u of u has radius $O(n)$, where n is the number of nodes in u and its subtree.

If three or more rings share a contact node, it is impossible to draw three disjoint circles, because in their drawing, we can have crossings which may visually create rings that do not exist in the original network. If such a case occurs, the outside drawing algorithm (will be presented later) assumes that these rings have distinguished contact nodes, and in their final drawing they will be presented by a geometric shape with a slight deviation from a circle, called *almost-cycle*.

An almost-cycle is a geometric shape that has all of its nodes placed on the perimeter of a circle, except a few of them, see figure 3.2.

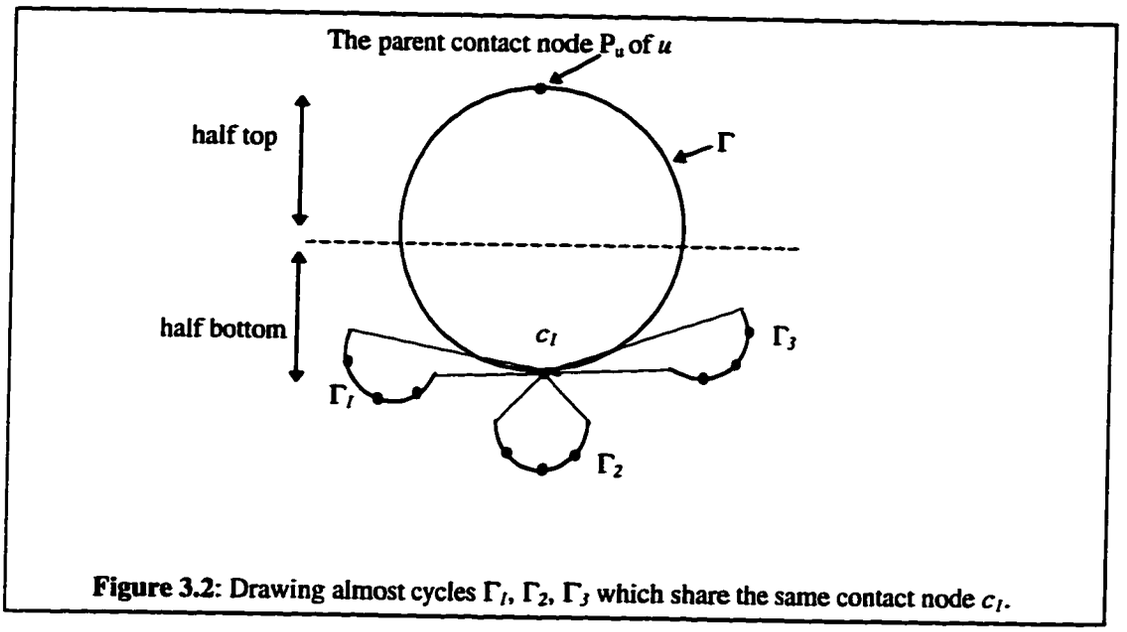


Figure 3.2: Drawing almost cycles Γ_1, Γ_2, Γ_3 which share the same contact node c_1.

If v is the root of T, it is not essential to have the radius of the circle Γ twice the radius of the circle Γ' in order to have a better utilization of the area. In this case the radius of the circle Γ is chosen to be equal to the radius of the circle Γ', and all the nodes of v and the center of the enclosing cycles Γ_j corresponding to the children of v are placed around the perimeter of the circle Γ corresponding to the root v, as shown in figure 3.3(a).

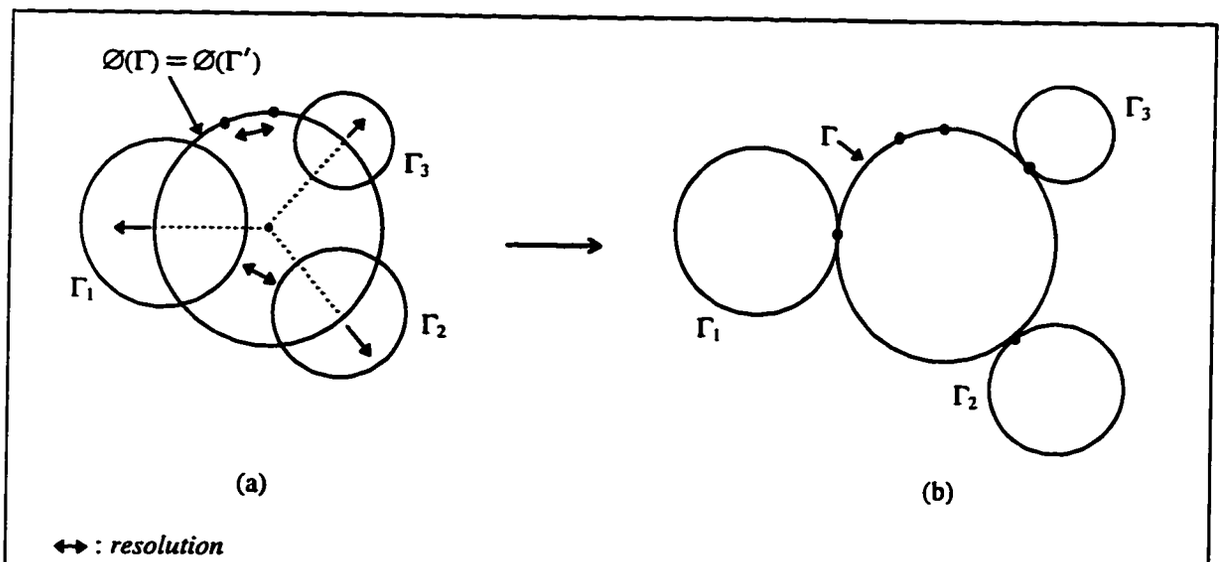


Figure 3.3 (a): Placing the nodes and the circles Γ_j around the perimeter of the circle Γ corresponding to the root.
(b): Each enclosing cycle Γ_j is pulled downward until it touches the perimeter of the circle Γ.

The outside drawing algorithm traverses the ring-contact node tree T in postorder fashion. After drawing all the children of a node recursively, the algorithm draws the parent node (by computing the radius of the circle corresponding to the parent node). If a node u in the ring-contact node tree T is a contact node, the algorithm ignores u and processes recursively its children.

Algorithm *OutsideDraw(v)*;
Input : Ring cover C, and its ring-contact node tree T with root v.
Output : An outside drawing of C.
Begin
1. **If** v is a leaf **then**
   - Compute the radius of a circle Γ corresponding to v, such that nodes are placed uniformly on the perimeter of the circle Γ, and are distant by *resolution*; **return;**

   EndIf;
2. **If** v is a node representing a ring **then**
   Begin
   - **For** all the children u_i of v **do** *OutsideDraw*(u_i);
   - Call *Compute_radius_ring* (computation of the radius of the circle Γ and the enclosing cycle Γ_v corresponding to v as describe in section 3.2);

   EndIf;
3. **If** v is a node representing a contact node **then**
   - **For** all the children u_i of v **do** *Outsidedraw*(u_j).

   EndIf ;

End.

Using the above procedure, Tollis and Xia [TOXI94], proved the following theorem:

Theorem 2

Given a ring cover C and its corresponding ring-contact node tree T, the algorithm *OutsideDraw* produces an outside drawing for the ring cover C, such that :

1. Each ring is presented by a circle or almost-cycle;
2. there are no crossings;
3. the area required by the drawing is $O(n^2)$, where n is the number of nodes in the ring cover C;
4. the time complexity of the algorithm is $O(n)$.

3.3 Case 2: Ring-contact node graph is not a tree.

As pointed out earlier, when rings share more than one contact node and when the ring cover contains ring-cycles, the ring-contact node graph is not appropriate enough to provide a tree structure to the ring cover. So, we propose the generalized-ring-contact node tree T_3 (which was presented in chapter 2, section 2.5.1), as the new underlying structure of the ring cover. The generalized-ring-contact node tree will be considered later as the basis of our outside drawing.

Given a ring cover C and its corresponding generalized-ring-contact node tree T_3, we will be dealing with the problem of drawing the ring cover on the screen, such that:

- the rings are drawn outside each other,
- there are no crossings,
- the resolution rule is respected,
- minimal area is used.

Our algorithm traverses the generalized-ring-contact node tree twice:

The first traversal of the tree T_3 is in postorder fashion. It consists of computing the radius of the circles corresponding to the rings in the ring cover first. Second, computing the radius of the circles corresponding to the ring-cycles. When all the radius of the circles corresponding to all the children (rings or ring-cycles) of a node u in T_3 are computed, the radius of the circle corresponding to the parent node u is computed recursively.

The second traversal of the tree T_3 is in preorder fashion. It consists of placing the parent node and its children in the right position.

3.3.1 Radius Calculation

We present in this phase different techniques related to different cases for computing the radius of the circles corresponding to the rings and the ring-cycles.

3.3.1.1 Case 2.1: Father of the ring u is a contact node in T_3

Let u be a ring node in T_3 and its children: u_1, u_2, ..., u_i. We propose a minor modification to the technique described in section 3.2 in order to provide a better optimization of the area used in our drawing.

The radius of the circle Γ corresponding to the node u is computed as described in the following procedure:

Procedure *Compute_Radius_Ring2*

First we compute the radius of a circle Γ', such that all the nodes of the ring u and the center of the enclosing cycles Γ_j corresponding to u_j, are placed evenly on the perimeter of the circle Γ'. The distance between the nodes and the enclosing cycles is equal to the minimum distance required by the resolution rule. We use the same process for computing the radius of the circle Γ' as described in the section 2.3.2.1 (second case).

In order to get the radius of the circle Γ that corresponds to the node u in T_3, it is enough to multiply the radius of the circle Γ' by two. This will allows us to place the parent contact node P_u of u on the top half of the circle Γ. This means that we do not need to include the parent contact node P_u in the calculation of the radius of the circle Γ', see figure 3.4 (b).

By doing so, the area of the drawing will be $O(n\text{-}m^2)$ (see proof of lemma 7); where n is the number of nodes in the ring cover and m is the number of the rings plus the number of the ring-cycles in the ring cover except for the root. In fact, for the root v of the tree T_3, the radius of the circle Γ is equal to the radius of the circle Γ'. This means that we include all the nodes of v in the computation of the radius of the circle Γ'.

When we implemented the algorithm of drawing almost-cycles, the results of the drawing were not aesthetic (see figure 3.2). It is due to the fact that the introduction of the new geometrical shape "almost-cycle" alters the uniformity of the drawing, since we use

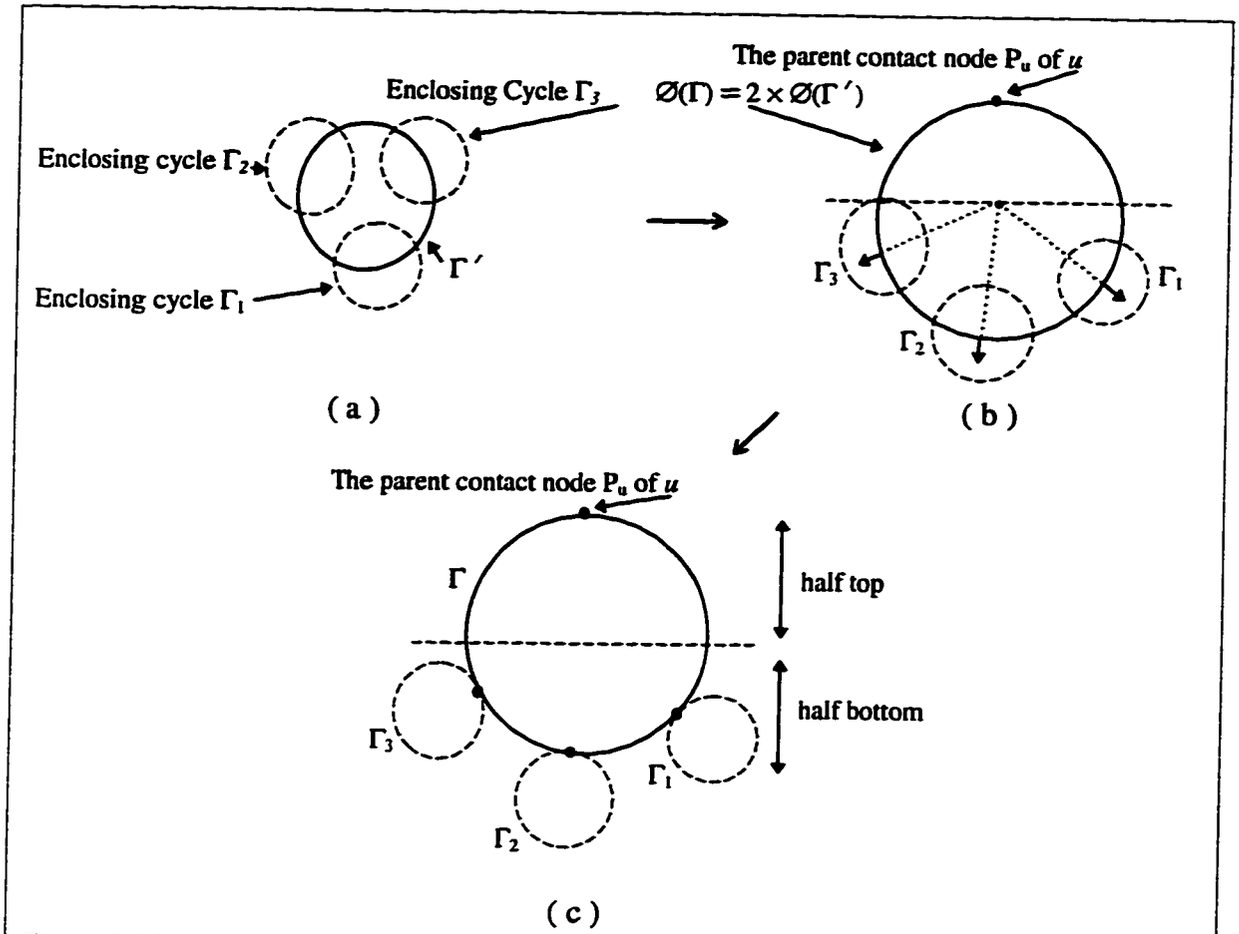


Figure 3.4 (a): Placing the enclosing cycles Γ_j of each subtree on the perimeter of the circle Γ', we notice that we did not include the parent contact node P_u of u in the calculation of the radius of the circle Γ'.

(b): The parent contact node P_u of u has enough space to be placed on the top half of the circle Γ.

(c): Each enclosing cycle Γ_j is pulled downward until it touches the perimeter of the circle Γ.

only circles to represent the rings in the ring cover. To cope with this problem, we introduce the following idea:

If more than two rings share the same contact node, we assume that these rings have distinguished contact nodes, and that in the final drawing, we join them by a special arc with different color and width, in order, for the user, to understand that this special arc connects two nodes which represent the same contact node, see figure 3.5. It amounts to duplicate some specific contact nodes.

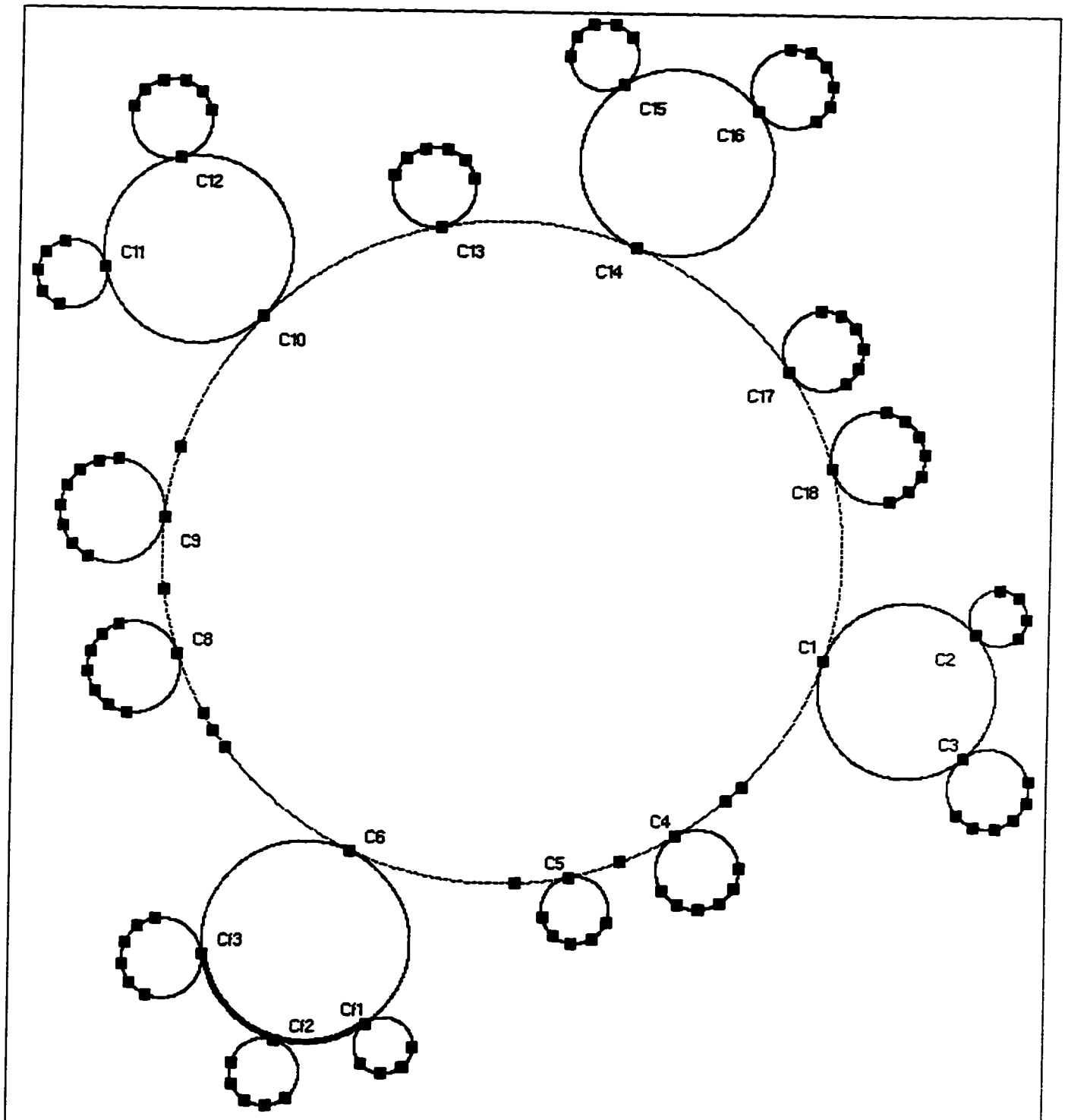


Figure 3.5: An outside drawing of the ring cover: note that the bold and colored arcs joining Cf_1, Cf_2 and Cf_2, Cf_3 means that Cf_1, Cf_2, Cf_3 represent one contact node.

3.3.1.2 Case 2.2: Father of the ring u is a multi-contact node in T_3.

In the next paragraph, we will present the technique for computing the radius of the circle Γ corresponding to the ring node u which father is a multi-contact node in T_3 (see figure 3.6 (a)).

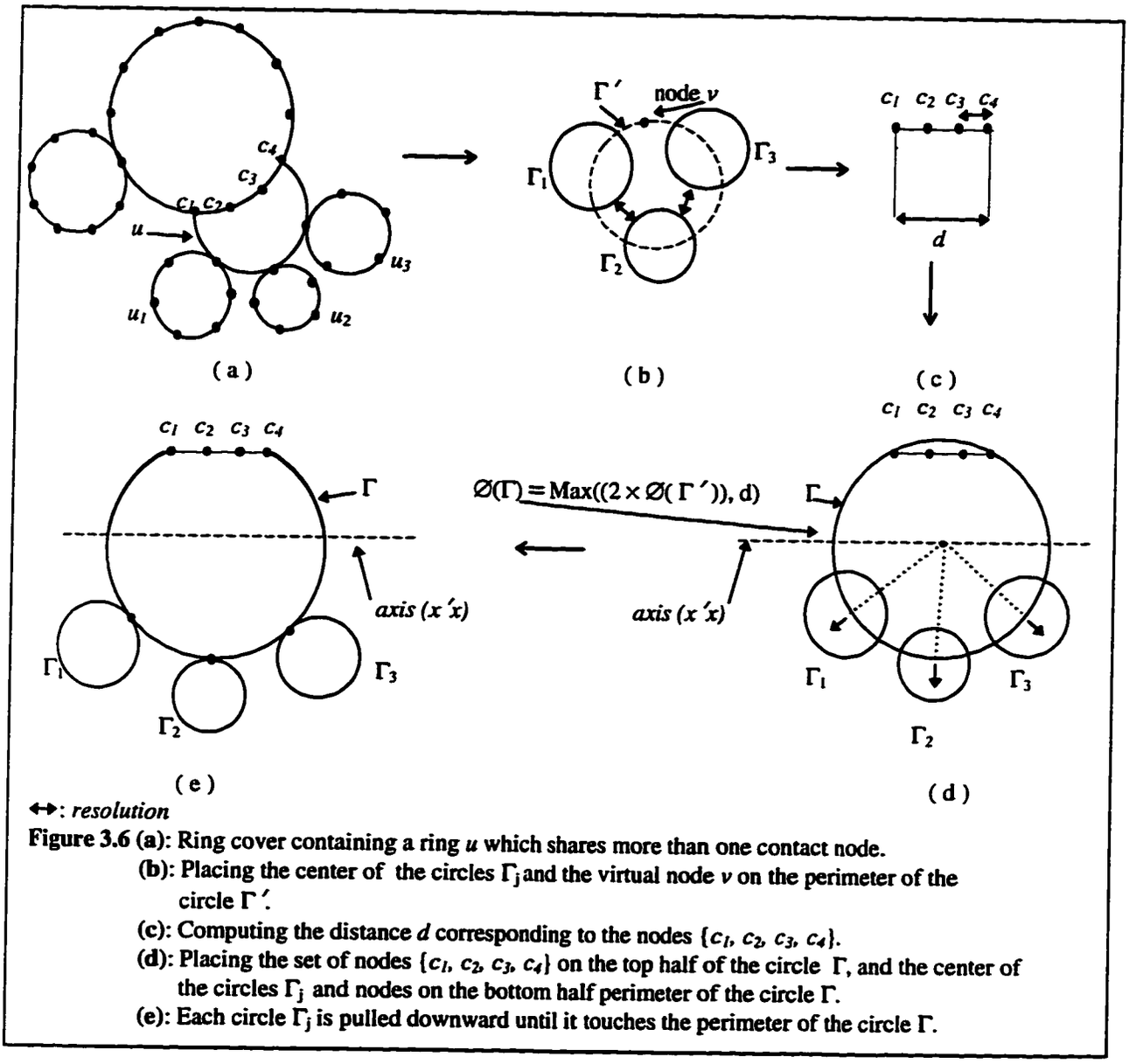


Figure 3.6 **(a)**: Ring cover containing a ring u which shares more than one contact node.
(b): Placing the center of the circles Γ_j and the virtual node v on the perimeter of the circle Γ'.
(c): Computing the distance d corresponding to the nodes $\{c_1, c_2, c_3, c_4\}$.
(d): Placing the set of nodes $\{c_1, c_2, c_3, c_4\}$ on the top half of the circle Γ, and the center of the circles Γ_j and nodes on the bottom half perimeter of the circle Γ.
(e): Each circle Γ_j is pulled downward until it touches the perimeter of the circle Γ.

We assume that u has i children $u_1, u_2, \ldots, u_i$. If all the radius of the circles Γ_j of u_j for $j = 1,2, \ldots, i$, have been determined, then the radius of the circle Γ corresponding to the node u is computed using the following procedure:

Procedure *Compute_Radius_Ring3*

First, in computing of the radius of the circle Γ', we replace the nodes of the multi-contact node *mc*, which is the parent of u in T_3, by one virtual node v (the purpose behind doing so, will be explained later, in the end of this section). Then we place on the perimeter of the circle Γ' the node v, as well as all the nodes of u and all the centers of the enclosing cycles Γ_j of u_j, for $j = 1,2, \ldots, i$. The distance between the nodes and the enclosing cycles Γ_j is the distance required by the *resolution*, see figure 3.6 (b). We use the same process for computing the radius of the circle Γ' as described in the section 2.3.2.1 (second case).

Let m be the number of contact nodes that belong to the multi-contact node *mc* which is the parent of u in T_3. We compute the value d, such that it is equal to m minus one multiplied by the *resolution*. In figure 3.6(a), the multi-contact node *mc* is $\{c_1, c_2, c_3, c_4\}$, therefore d is equal to $((4-1) \times resolution)$.

Second, we draw a circle Γ corresponding to u. The circle Γ has the same center as the circle Γ' and a diameter equal to the maximum of two times the diameter of the circle Γ' and the value d, that is :

$\varnothing(\Gamma) = \text{Max}((2 \times \varnothing(\Gamma')), d)$.

Finally, we place the set of the contact nodes of *mc* (father of u in T_3) on the top half of the circle Γ. The center of all the other enclosing cycles Γ_j and nodes of u, are placed on the bottom half perimeter of the circle Γ, see figure 3.6 (d). We pull each circle Γ_j downward until it touches the perimeter of the circle Γ. It is clear that each subtree corresponding to each node u_j has enough area to be drawn inside the enclosing cycle Γ_j, without intersecting the rings of the other subtrees of T_3.

By doing so, we guarantee that there are no crossings between the ancestors of u and the children of u, and no crossings between the rings of each subtree. We compute the diameter of the enclosing cycle Γ_u of u as follows:

$\varnothing(\Gamma_u) = \varnothing(\Gamma) + \text{Max}(\varnothing(\text{enclosing cycles } \Gamma_j))$.

The purpose of replacing the set of contact nodes of *mc* by one node is to have enough space in the perimeter of the circle Γ', so that if there exists a node placed on the bottom half of the circle Γ which is too close to the axis ($x'x$) dividing the circle Γ in two, see figure 3.6 (d), this node will be far enough by *resolution* from any node placed on the top half of the circle Γ.

In figure 3.6 (e), the diameter of the circle Γ corresponding to the node u, is equal to $(2 \times \varnothing(\Gamma'))$. All the other circles Γ_j, for $j = 1,2,3$ corresponding to the rings u_1, u_2, u_3 which are leaves in T_3, have an area $O((n_i-1)^2)$, where n_i is the number of nodes in each ring u_i. We decrease by one, because we do not need to include the parent contact node of each ring u_i.

The area of the circle Γ will be then $O((n-4)^2)$, where n is the number of nodes in u and in its subtree (see proof of lemma 3). We decrease by 4 since we have three children rings which are $\{u_1, u_2, u_3\}$ plus the ring u itself. In the calculation of the radius of the circle Γ', we do not include all the contact nodes $\{c_1, c_2, c_3, c_4\}$, we just replace these nodes by one virtual node v, see figure 3.6(b).

If the diameter of the circle Γ is equal to the value d, the area of the circle Γ could be better than $O((n-4)^2)$. Since the radius of the enclosing cycle Γ_u is equal to $\varnothing(\Gamma)$ plus Max($\varnothing$(enclosing cycles Γ_j)), the area of the circle Γ_u is also $O((n-4)^2)$.

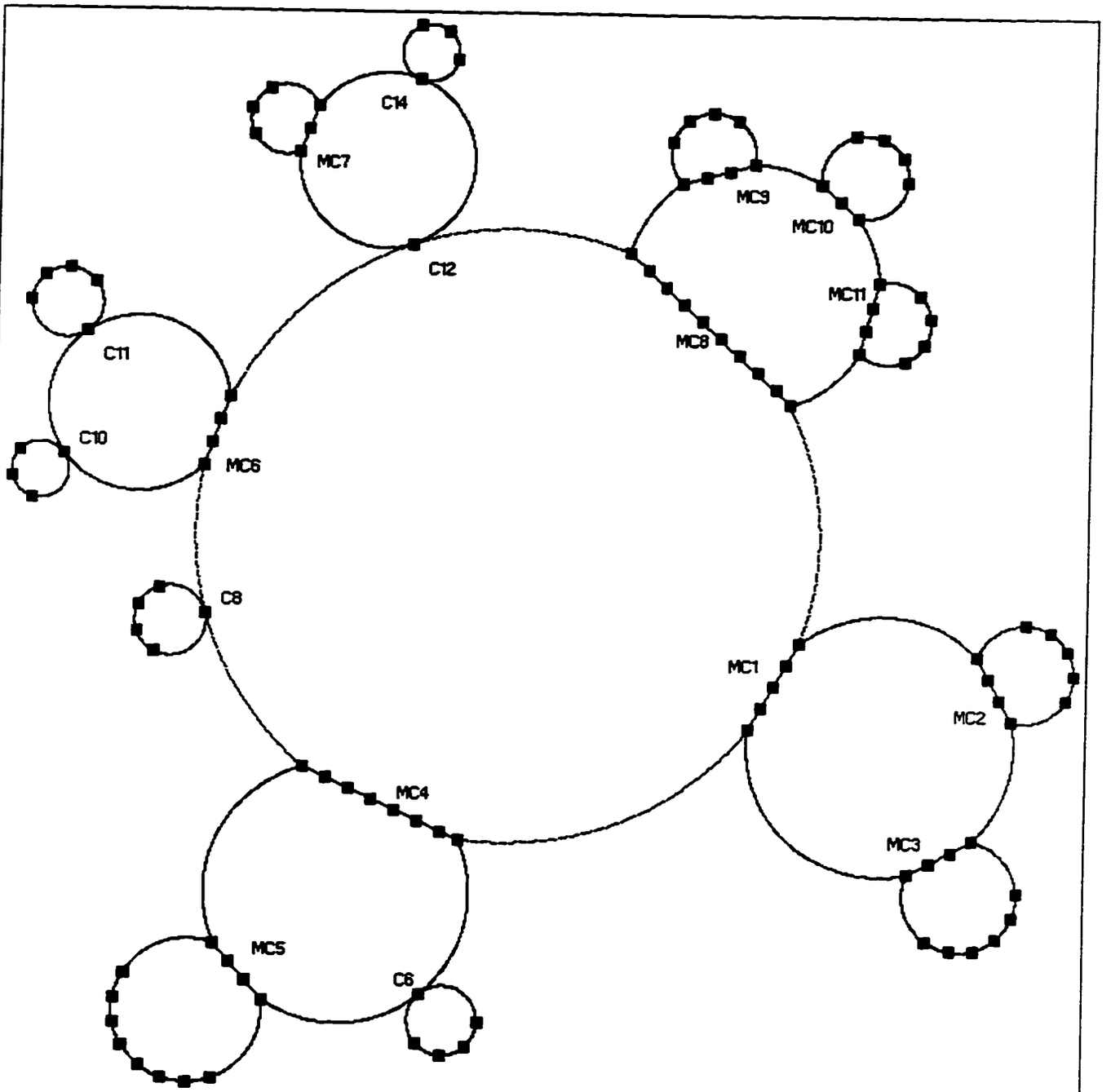


Figure 3.7: Snapshot of the system's screen. Ring cover contains rings that share more than one contact node.

3.3.1.3 Case 2.3: Father of the ring *u* is a ring-cycle in T_3

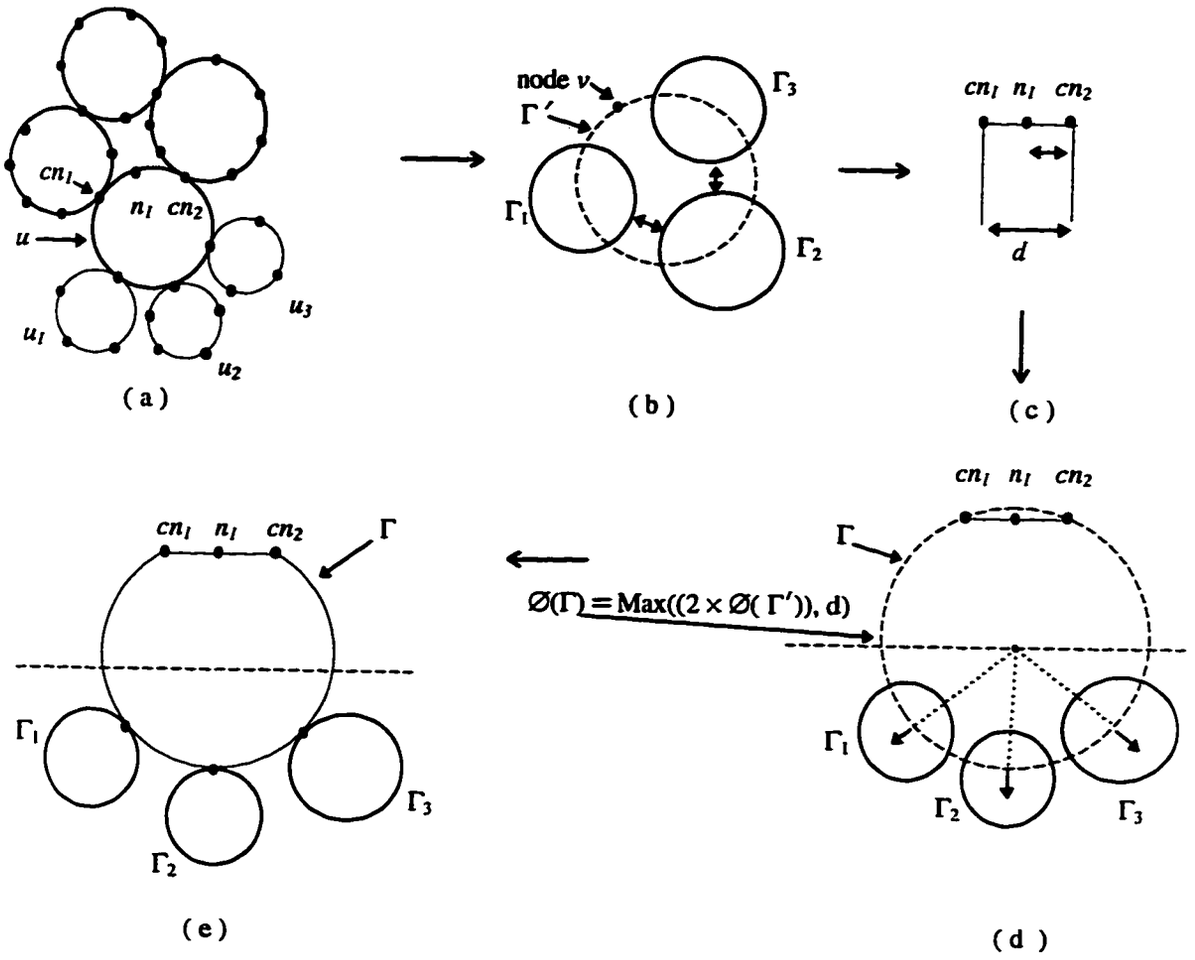


Figure 3.8 (a): Ring cover *C*.
(b): Placing the center of the circles Γ_j and the virtual node *v* on the perimeter of the circle Γ'.
(c): Computing the distance *d* which is equal to $2 \times$ *resolution*.
(d): Placing the set of nodes $\{cn_1, n_1, cn_2\}$ on the top half of the circle Γ, and the enclosing cycles Γ_j on the bottom half of the circle Γ.
(e): Each enclosing cycle Γ_j is pulled downward until it touches the perimeter of the circle Γ.

We use the same technique described in section 3.3.1.2, to compute the radius of the circles corresponding to the rings which have a ring-cycle as a father. The only difference is that we suppose that the set multi-contact node *mc* is the set of nodes between two special contact nodes cn_i and cn_{i+1} in the ring *u*, in figure 3.8 (a), these nodes are $\{cn_1, n_1, cn_2\}$.

The radius of the circle Γ corresponding to the node u is computed using the following procedure:

Procedure *Compute_Radius_Ring4*

First, in computing of the radius of the circle Γ', we replace the set of nodes between the two special contact nodes cn_i and cn_{i+1} in the ring u by one virtual node v. The purpose behind doing so, has been detailed at the end of section 3.3.1.2. Then we place on the perimeter of the circle Γ' the node v, as well as all the nodes of u and all the centers of the enclosing cycles Γ_j of u_j, for $j = 1,2, \ldots, i,$. The distance between the nodes and the enclosing cycles Γ_j is the distance required by the *resolution*, see figure 3.8 (b). We use the same process for computing the radius of the circle Γ' as described in the section 2.3.2.1 (second case).

Let m be the number of nodes between the two special contact nodes cn_i and cn_{i+1} in the ring u . We compute the value d, such that it is equal to m minus one multiplied by the *resolution*. In figure 3.8(a), the nodes between the two special contact nodes cn_i and cn_{i+1} in the ring u are $\{cn_1, n_1, cn_2\}$, therefore d is equal to ((3-1) $\times$ *resolution*).

Second, we draw a circle Γ corresponding to u. The circle Γ has the same center as the circle Γ' and a diameter equal to the maximum of two times the diameter of the circle Γ' and the value d, that is :

$\varnothing(\Gamma) = \text{Max}((2\times \varnothing(\Gamma')), d)$.

Finally, we place the set of nodes that exist between the two special contact nodes cn_i and cn_{i+1} in the ring u on the top half of the circle Γ. The center of all the other enclosing cycles Γ_j and nodes of u, are placed on the bottom half perimeter of the circle Γ (see figure 3.8 (d)). We pull each circle Γ_j downward until it touches the perimeter of the circle Γ.

The diameter of the enclosing cycle Γ_u of u is equal to the diameter of the circle Γ plus the maximum diameter of the enclosing cycles Γ_j. Thus:

$\varnothing(\Gamma_u) = \varnothing(\Gamma) + \text{Max}(\varnothing(\text{enclosing cycles } \Gamma_j))$.

3.3.1.4 Case 2.4: Node u is a ring-cycle in T_3

Since we use circles to represent the rings of the ring-cycles, it is too difficult to draw these circles connected to each other, such that we guarantee that there are no crossings in their drawing and that we respect the resolution rule. Our idea is to suppose that all these rings are detached, and in the final drawing, for each two adjacent rings that share the common contact node, we duplicate this contact node into two nodes, and we join them using an arc with a different color and a different width, in order to distinguish this arc from all the other physical links (see figure 3.9(e)).

The same process of drawing, described in section 3.3.1.3, is used to compute the radius of the circle corresponding to the ring-cycle. The only difference is that, in the final drawing we join the two nodes (that represent one contact node) between the two adjacent rings in the ring-cycle using a special arc, with a user defined color and a user defined width.

Let u has i children u_1, u_2, ..., u_i, such that u_i represents a ring that belongs to the ring-cycle u. If the radius of all the enclosing cycles Γ_j of u_j, for $j = 1, 2, ..., i$ have been computed, then the radius of the circle Γ corresponding to u is computed as described in the following procedure:

Procedure *Compute_Radius_Ring-cycle*

First, in computing of the radius of the circle Γ', we replace the set of nodes between the two special contact nodes cn_i and cn_{i+1} that belong to the parent ring of u (in figure 3.9 (a) these nodes are $\{cn_1, n_1, cn_2\}$), by one virtual node v. The purpose behind doing so, has been detailed at the end of section 3.3.1.2. Then we place on the perimeter of the circle Γ' the node v, as well as all the enclosing cycles Γ_j of u_j, for $j = 1,2, ..., i$. The distance between the enclosing cycles Γ_j is the distance required by the *resolution*, see figure 3.9 (b). We use the same process for computing the radius of the circle Γ' as described in the section 2.3.2.1 (second case).

Let m be the number of nodes between the two special contact nodes cn_i and cn_{i+1} that belong to the parent ring of u. We compute the value d, such that it is equal to m minus one multiplied by the *resolution*. In figure 3.9(a), the nodes that belong to the parent ring of u are $\{cn_1, n_1, cn_2\}$, therefore d is equal to $((3\text{-}1) \times resolution)$.

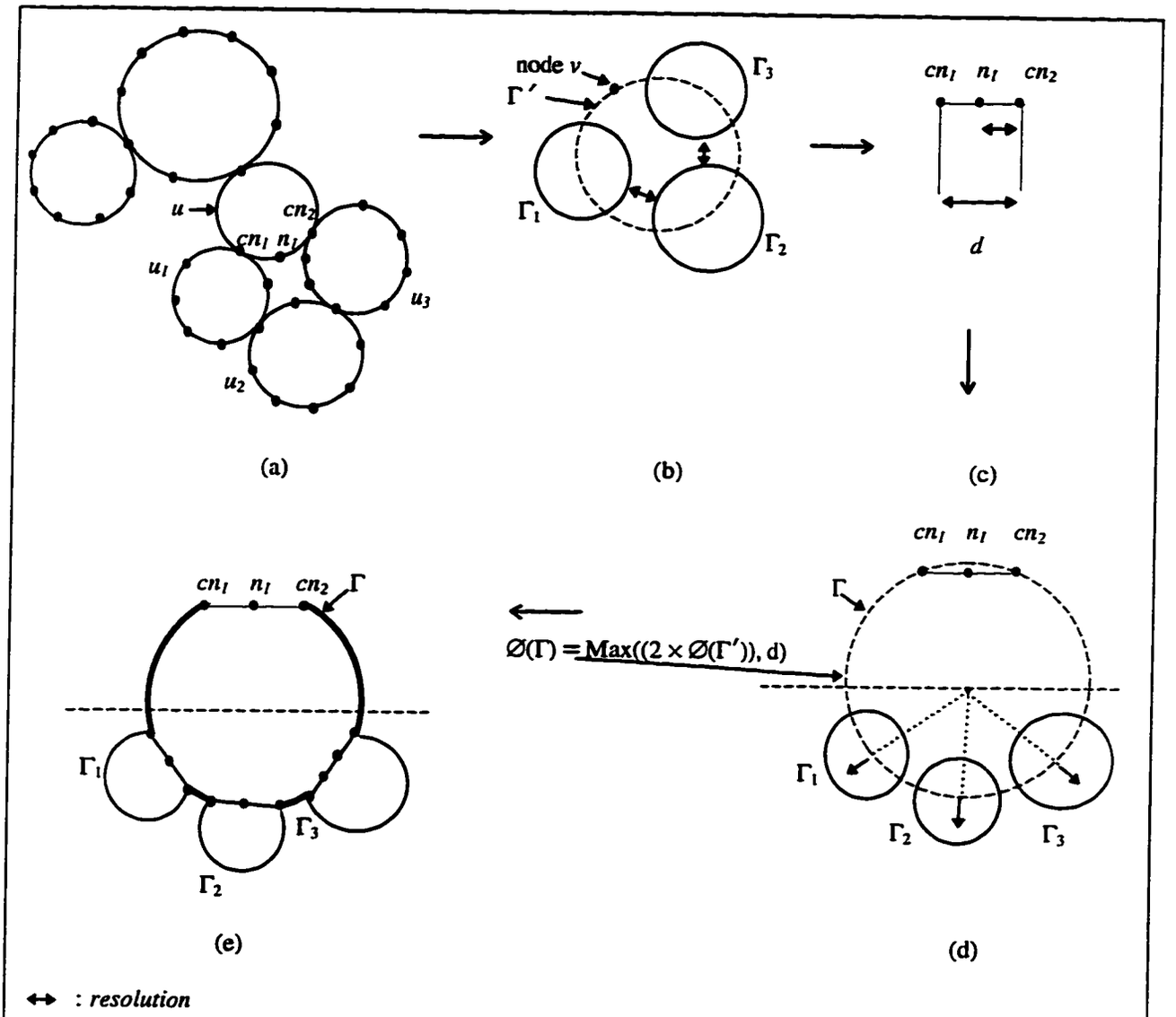


Figure 3.9 **(a)**: Ring cover containing a ring-cycle u.
(b): Placing the centers of the enclosing cycles Γ_j and the virtual node v on the perimeter of the circle Γ'.
(c): Computing the distance d which is equal to $2 \times resolution$.
(d): Placing the set of nodes $\{cn_1, n_1, cn_2\}$ on the top half of the circle Γ, and the centers of the enclosing cycles Γ_j on the bottom half of the circle Γ.
(e): Each enclosing cycle Γ_j is pulled downward until it touches the perimeter of the circle Γ by the two extremity contact nodes.

Second, we draw a circle Γ corresponding to u. The circle Γ has the same center as the circle Γ' and a diameter equal to the maximum of two times the diameter of the circle Γ' and the value d, that is :

$\varnothing(\Gamma) = \text{Max}((2\times \varnothing(\Gamma')), d)$.

By doing so, we guarantee that there are no crossings between the ancestors of u and the children of u, and no crossings between the rings of each subtree.

We compute the radius of the enclosing cycle Γ_u of u as follows:

$\varnothing(\Gamma_u) = \varnothing(\Gamma) + \text{Max}(\varnothing(\text{enclosing cycles } \Gamma_j))$.

Finally, we place the set of nodes that belong to the parent ring of u (in figure 3.9(a), these nodes are $\{cn_1, n_1, cn_2\}$) on the top half of the circle Γ, and all the other enclosing cycles Γ_j and the nodes of u on the bottom half perimeter of the circle Γ (see figure 3.9 (d)). We pull each circle Γ_j downward until it touches the perimeter of the circle Γ.

3.3.2 Algorithm

The algorithm traverses the tree T_3 in postorder fashion. After computing the radius of the circles and the enclosing cycles corresponding to the rings and the ring-cycles recursively, the algorithm computes the radius of the circle and the radius of the enclosing cycle corresponding to the parent node.

Algorithm *OutsDraw_Radius_Computation (v)*;
Input : Ring cover C, and its generalized-ring-contact node tree T_3 with root v.
Output : Radius of the circles corresponding to the rings and the ring-cycles in C.
Begin
1. **If** ((v is a node representing a ring) **and** (v is a leaf) **and** (father(v) is a contact node)) **then**
   - Call *Compute_Radius_ring1(v)* (Computation of the radius of the circle Γ corresponding to v, such that nodes are placed uniformly on the perimeter of the circle Γ, and are distant by *resolution*, see section 2.3.2.1-first case), **return;**

EndIf;

2. **If** (((*v* is a node representing a ring) **and** (father(*v*) is a contact node)) **or** *v* is a root) **then**
   Begin
   - **For** all the children u_i of *v* **do** *OutsDraw_RadiusCalculation* (u_i);
   - Call *Compute_Radius_ring2(v)* (Computation of the radius of the circle Γ and the enclosing cycle Γ_v corresponding to *v* as described in section 3.3.1.1).

   EndIf;
3. **If** ((*v* is a node representing a ring) **and** (father(*v*) is a multi-contact node)) **then**
   Begin
   - **For** all the children u_i of *v* **do** *OutsDraw_RadiusCalculation* (u_i);
   - Call *Compute_Radius_ring3(v)* (Computation of the radius of the circle Γ and the enclosing cycle Γ_v corresponding to *v* as described in section 3.3.1.2.)

   EndIf;
4. **If** ((*v* is a node representing a ring) **and** (father(*v*) is a ring-cycle)) **then**
   Begin
   - **For** all the children u_i of *v* **do** *OutsDraw_RadiusCalculation* (u_i);
   - Call *Compute_Radius_ring4(v)* (Computation of the radius of the circle Γ and the enclosing cycle Γ_v corresponding to *v* as described in section 3.3.1.3.)

   EndIf;
5. **If** (*v* is a ring-cycle) **then**
   Begin
   - **For** all the children u_i of *v* **do** *OutsDraw_RadiusCalculation* (u_i);
   - Call *Compute_Radius_ring-cycle(v)* (Computation of the radius of the circle Γ and the enclosing cycle Γ_v corresponding to *v* as described in section 3.3.1.4)

   EndIf;
6. **If** (*v* is a node representing a contact node) **then**
   - **For** all the children u_i of *v* **do** *OutsDraw_RadiusCalculation* (u_i);

   EndIf;
7. **If** (*v* is a node representing a multi-contact node (*v* will have only one child *u* ′)) **then**
   - *OutsDraw_RadiusCalculation* (*u* ′);

   EndIf;

End.

The time complexity of *OutsDraw_RadiusCalculation* is $O(n + m^2)$, where *n* is the number of nodes in the ring cover and *m* is the number of the rings in the ring cover. We use the same analysis of complexity described for the *InDraw1_RadiusCalculation* algorithm (see chapter 2 section 2.3.3).

3.3.3 Placement node and drawing phase

In the first step (by means of the *OutsDraw_Radius Computation* algorithm), we computed the radius of all the circles corresponding to the rings and the ring-cycles. We have also computed the radius of their enclosing cycles. We traverse the tree T_3 in preorder fashion in order to place the nodes, the rings and the ring-cycles, in their proper position.

Algorithm *OutsDraw_PlacePhase* (v, v_x, v_y);
Input : Radius of circles corresponding to the rings in C computed by the *OutsDraw_RadiusCalculation*, v represent the root of T_3.
Output: An Outside drawing of the ring cover C.
Begin
1. **If** ((v is a node representing a ring) **and** (v is a leaf) **and** (father(v) is a contact node)) **then**
   - Draw the circle Γ of v at the coordinate (v_x, v_y). (The radius of the circle Γ was computed by procedure *compute_radius_ring1* see section 2.3.2.1-first case)

   EndIf;
2. **If** (((v is a node representing a ring) **and** (father(v) is a contact node)) **or** v is a root) **then**
   - Draw the circle Γ of v at the coordinate (v_x, v_y). (The radius of the circle Γ was computed by procedure *compute_radius_ring2* see section 3.3.1.1)

   EndIf;
3. **If** ((v is a node representing a ring) **and** (father(v) is a multi-contact node)) **then**
   - Draw the circle Γ of v at the coordinate (v_x, v_y).(The radius of the circle Γ was computed by procedure *compute_radius_ring3* see section 3.3.1.2).

   EndIf;
4. **If** (v is a ring-cycle) **then**
   - Draw the circle Γ of v at the coordinate (v_x, v_y).(The radius of the circle Γ was computed by procedure *compute_radius_ring-cycle* see section 3.3.1.4).

   EndIf;
5. **if** ((v is a node representing a ring) **and** (father(v) is a ring-cycle)) **then**
   - Draw the circle Γ of v at the coordinate (v_x, v_y).(The radius of the circle Γ was computed by procedure *compute_radius_ring4* see section 3.3.1.3).

   EndIf;
6. **If** (v is a node representing a contact node) **then**
   - **For** all the children u_i of v **do** *OutsDraw_PlacePhase* (u_i, v_x, v_y); **Return**;

   EndIf;
7. **If** (v is a node representing a multi-contact node and u' is a child of v) **then**
   - *OutsDraw_PlacePhase* (u', v_x, v_y); **Return**;

   EndIf;

8. For all the children u_i of v **do** *{u_i could be a node of v}*
Begin
- *Calculate_coordinate* (u_i, u_{ix}, u_{iy});
  {using polar technique described in section 2.3.4}
- **If** u_i is a node **then**
  - Place the node at the coordinate (u_{ix}, u_{iy});

  Else
  - *OutsDraw_PlacePhase* (u_i, u_{ix}, u_{iy});

  Endif;

EndFor;
End.

The time complexity of the *OutsDraw_PlacePhase* algorithm is of order $O(n)$, where n is the number of nodes in the ring cover. First, we draw the parent node then recursively we draw the children, we process each node in T_3 only once, (see chapter 2 section 2.3.4 for full detailed analysis).

Using the above procedure, we conclude the following lemma :

Lemma 7

Given a ring cover C and its generalized-ring-contact node tree T_3, the outside drawing algorithm (composed by *OutsDraw_Radius_Computation* and *OutsDraw_PlacePhase*) produces a ring cover drawing such that:

1. There are no crossings;
2. the area required by the drawing is $O((n-m)^2)$, where n is the number of nodes in C, and m is the number of the rings plus the number of the ring-cycles minus one (the root);

Proof

1. Let u be a ring or a ring-cycle node in T_3, we assume that u has i children u_1, ..., u_i which are leaves in T_3. We place all the children of u on the bottom half of the circle Γ corresponding to u, and the parent of u is placed on the top half of the circle Γ. By doing so we avoid intersections between the children of u and the ancestors of u. Also, we place the node u with its subtree, in an enclosing cycle, such that it does not intersect the rings

of the other subtrees in T_3. We repeat recursively the same process of drawing with the parent node of u, so that, in the final drawing there will be no crossings.

2. Let u be a node in T_3, and let u has i children u_1, u_2, ..., u_i. We assume first, that the children u_i are leaves in T_3. The circles corresponding to u_i can represent either rings that share one contact node or rings that share more than one contact node.

If the child node u_i of u represents a ring that shares exactly one node with other rings (as described in section 3.3.1.1), we do not include the parent contact node in the calculation of the radius of the circle Γ_i' of u_i. The reason is that the parent contact node P_u of u has enough area to be placed in the top half of the circle Γ_i corresponding to u_i. It follows that the circle Γ_i has a radius $O(n_i - 1)$, where n_i is the number of node in the ring u_i.

If the child node u_i of u represents a ring that shares more than one contact node with other rings, the radius of the circle Γ_i corresponding to the node u_i has still a radius $O(n_i - 1)$, where n_i is the number of nodes in u_i. The reason, is that the diameter of the circle Γ_i is equal to the Max(d, $2 \times \varnothing(\Gamma_i')$), (as described in section 3.3.1.2). This means that if the diameter of the circle Γ_i is equal to twice the diameter of the circle Γ_i', we do not include all the nodes of the ring u_i for the calculation of the radius of the circle Γ'. Also if the diameter of the circle Γ_i is equal to the distance d, we do not include all the nodes of the ring u_i for the calculation of distance d. In general the area of the circle Γ_i corresponding to the ring u_i can be better than $O((n_i - 1)^2)$.

From the previous analysis, we conclude that all leaves in the tree T_3 have an area of $O((n_i - 1)^2)$, where n_i is the number of nodes in each ring u_i. We build our drawing from the bottom-up in T_3, and we repeat the same drawing recursively for the parent node u which could be either a ring or a ring-cycle, we assume first, that u is a ring and u is not the root of T_3, the radius of the circle Γ corresponding to u, is $O((n-m)^2)$, where n is the number of nodes in u and its subtree, m is the number of the rings in the subtree of u including u itself. This is due to the fact that we use the technique which consists of

placing the center of the circles Γ_i corresponding to the children of u and nodes of u, on the perimeter of the circle Γ' (described in the proof of lemma 3).

If u is a ring-cycle and u is not the root of T_3, the radius of the circle Γ corresponding to u is $O((n-m)^2)$, where n is the number of nodes in the subtree of u plus the number of nodes which exist between the two special contact nodes cn_i and cn_{i+1}. These contact nodes cn_i and cn_{i+1} belong to the father ring of u, see figure 3.9(a). m is the number of the rings in the subtree of u including u itself.

If u is a root in T_3, the enclosing cycle of u will have an area of $O((n-m)^2)$, where n is the number of nodes in the ring cover and m is the number of the rings and the ring-cycles in the ring cover minus one, so that, m = |rings| + |ring-cycles| - 1, the reason we decrease by one, is that in the calculation of the radius of the circle Γ corresponding to the root u of T_3, where the radius of the circle Γ is equal to the radius of the circle Γ', we include all the nodes of u.

For example in figure 3.11, m is equal to 21 because we have 19 rings (except for the root) and 2 ring-cycles.

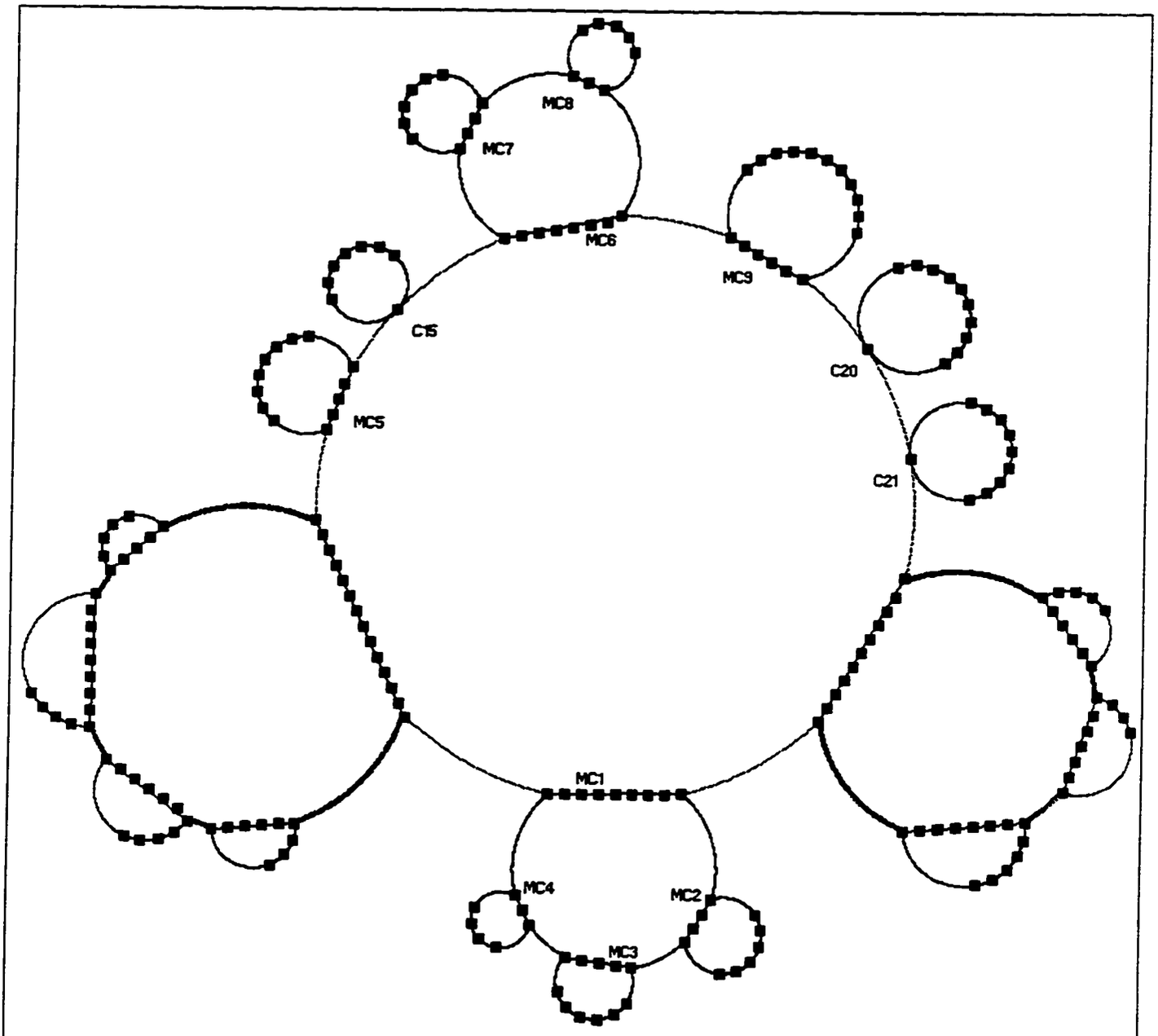


Figure 3.10: Ring cover contains rings which share more than one contact node and contains two ring-cycles. The bold and colored arcs combine the same contact node.

Chapter 4

Mixed Drawing

4.1 Introduction

In the first part of this chapter, we will present the mixed drawing technique which uses both inside and outside drawing algorithm for the ring cover drawing. We emphasize the fact that our mixed drawing algorithm is totally different from the one presented in [TOXI94].

We introduce the mixed drawing for the following reason. First, because it is impossible to draw rings outside each other, when more than two rings share the same multi-contact node (see figure 2.21 (a), in chapter 2 section 2.4.3.1). The second reason, is that, the mixed drawing algorithm uses less area than the outside drawing algorithm.

In the second part, we will present a new method to find the optimum layout of the generalized-ring-contact node tree T_3 in order to minimize the area of our drawing.

4.2 Basic idea

In the inside drawing, we are able to draw rings when more than two rings share a multi-contact node (see figure 2.21(a)). Our idea consists of drawing some subtrees of T_3

by the inside algorithm, and other subtrees by the outside algorithm. First, we traverse the tree T_3 in preorder fashion, if we encounter a ring node u, such that it contains a child which is a multi-contact node or a ring-cycle, then the ring corresponding to u and its subtree will be drawn by the inside drawing algorithm, else, if u does not have any child which is either a multi-contact node or a ring-cycle, then u and its children will be drawn one outside the other.

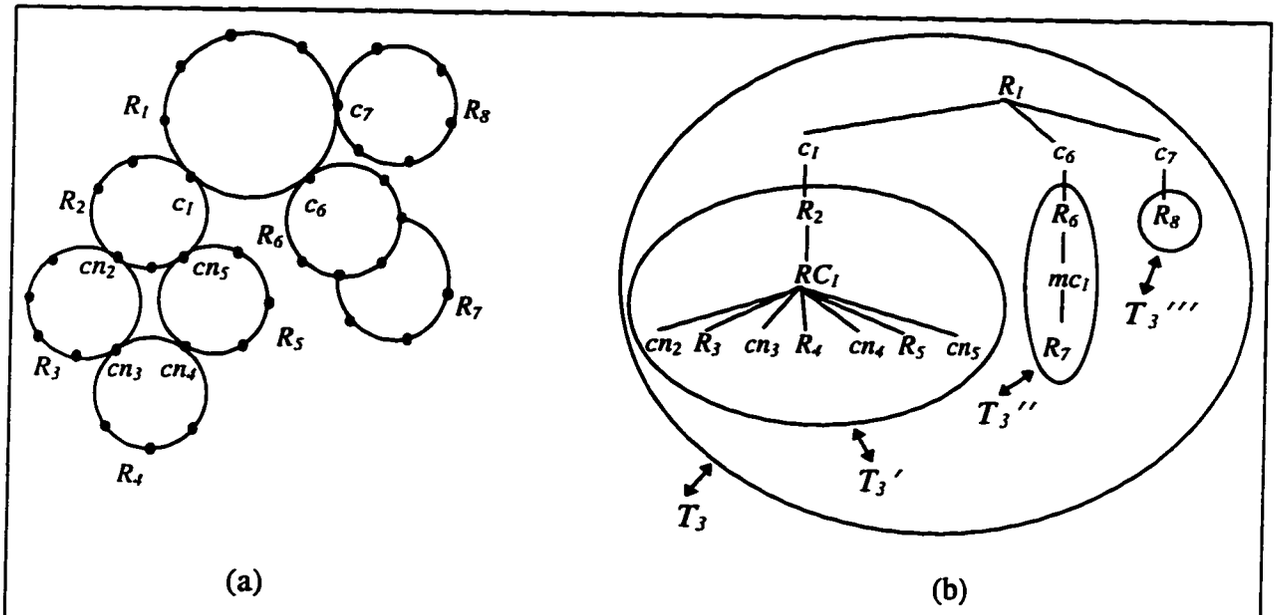


Figure 4.1 (a): Ring cover C.

(b): The generalized-ring-contact node tree T_3.

Let's consider the example in figure 4.1 (b), the ring node R_2 and its subtree T_3' will be drawn by the inside drawing algorithm, because R_2 has a child which is a ring-cycle RC_1. The node R_6 and its subtree T_3'' will be drawn by the inside drawing algorithm, because R_6 has a child which is a multi-contact node mc_1. Since all the children of R_1 which are $\{c_1, c_6, c_7\}$ are different from a ring-cycle and a multi-contact node, R_1, R_2, R_6 and R_8 will be drawn each one outside the others.

Let *INSIDE* be a Boolean label associated with each node ring or ring-cycle in T_3, we traverse the tree T_3 three times. The first traversal of T_3 is in preorder fashion, it is

done in order to assign a value {true or false} to the label *INSIDE* corresponding to each ring node and ring-cycle node in T_3. The reason is to decide for each node (ring or ring-cycle) which drawing technique inside or outside will be considered.

The second traversal of T_3 is in postorder fashion which aims to compute the radius of the circles and the radius of the enclosing cycles corresponding to the rings and the ring-cycles.

Finally, the last traversal of T_3 is in preorder fashion, it is done in order to place the rings and the nodes in their proper position, such that there are no crossings.

4.3 Algorithm

The algorithm *MixDraw_LabelNode* will traverse the tree T_3 in preorder fashion. If we encounter a node u which has at least one child which is either a multi-contact node or a ring-cycle, we assign a value true to *INSIDE* associated to the node u and all the nodes in the subtree of u. This means that, all the rings and the ring-cycles of the subtree of u will be drawn inside each other. If all the children of u are different than a ring-cycle and a multi-contact node, we assign the value false to *INSIDE* associated to the node u.

Algorithm *MixDrawLabelNode(v)*;
Input: Ring cover C, and its generalized-ring-contact node tree T_3 with the root v.
Output: An update of the label *INSIDE* associated to the rings and the ring-cycles.
Begin
1. If ((v is a ring that contains at least one child which is a multi-contact node **or** ring-cycle) **or** v is a ring-cycle))
then
- Assign the value true to *INSIDE* associated to the node v and to all the nodes in the subtree of v; **return;**

EndIf;
2. If (v is a ring) **and** (all the children of v are contact nodes) **then**
Begin
- Assign the value false to *INSIDE* associated to the node v;
- **For** all the children u_i of v **do** *MixDraw_LabelNode* (u_i);

EndIf;
3. If (v is a contact node) **then**
- **For** all the children u_i of v **do** *MixDraw_LabelNode* (u_i);

EndIf;
End.

The time complexity of *MixDraw_LabelNode* algorithm is of order $O(n)$, where n is the number of nodes in the ring cover. We process (by means of updating the *INSIDE* label) each node in T_3 only one time.

In the second traversal of T_3, the algorithm starts with the root v of T_3. If v is a ring or a ring-cycle which *INSIDE* value is true, then we call the *InsDraw3_RadiusCalculation* algorithm to compute the radius of the circles corresponding to the rings and the ring-cycles in T_3. If the value of *INSIDE* is equal to false, then the node v and all its children will be drawn each one outside the other. By doing so we mix both drawings: the inside drawing and the outside drawing to obtain a new drawing called the mixed drawing.

Algorithm *MixDraw_RadiusCalculation(v)*;
Input: Ring cover C, and its generalized-ring-contact node tree T_3 with the root v.
Output : Radius of the circles corresponding to the rings and the ring-cycles in C.
Begin
1. If ((v is a ring **or** v is a ring-cycle) **and** (*INSIDE(v)* is true))
then
- *InsDraw3_RadiusCalculation(v)*; **Return**; *{see chapter 2, section 2.5.2}*

EndIf;
2. If ((v is a ring) **and** (*INSIDE(v)* is false)) **then**
Begin
- **For** all the children u_i of v **do** *MixDraw_RadiusCalculation* (u_i);
- Call *Compute_Radius_ring2(v)* (Computation of the radius of the circle Γ and the enclosing cycle Γ_v corresponding to v as described in section 3.3.1.1).
{ we place v and its children one outside the other }

EndIf;
3. If (v is a node representing a contact node) **then**
- **For** all the children u_i of v **do** *MixDraw_RadiusCalculation* (u_i);

EndIf;
End.

The time complexity of *MixDraw_RadiusCalculation* is of order $O(n + m^2)$, where n is the number of nodes in the ring cover and m is the number of the rings in the ring cover. We use the same analysis of complexity described for the *InDraw1_Radius-Calculation* algorithm (see chapter 2 section 2.3.3).

We have already the radius of all the circles corresponding to the rings and the ring-cycles by the *MixDraw_RadiusCalculation* algorithm. We traverse the tree T_3 in preorder fashion, such that if the label *INSIDE* of a node has the value true, we call *InsDraw3_PlacePhase* to draw v and its subtree, else we draw v and all its children one outside the other.

Algorithm *MixDraw_PlacePhase* (v, v_x, v_y);
Input : Radius of the circles corresponding to the rings in C computed by the *MixDraw_RadiusCalculation*, v represent the root of T_3.
Output: An Mixed drawing of the ring cover C.
Begin
1. If (((*v* is a ring) **and** (*INSIDE* = true)) **or** (*v* is a ring-cycle))
then
- *InsDraw3_PlacePhase (v)*; **Return**; *{see chapter 2, section 2.5.3}*

EndIf;
2. If ((*v* is a ring) **and** (*INSIDE* = false)) **then**
Begin
- Draw the circle Γ of v at the coordinate (v_x, v_y). (The radius of circle Γ was computed by procedure *compute_radius_ring2* see section 3.3.1.1)
- **For** all the children u_i of v **do** *{u_i could be a node of v}*
Begin
  - *Calculate_coordinate* (u_i, u_{ix}, u_{iy}); *{using polar technique}*
  - **If** u_i is a node **then**
    - Place the node at the coordinate (u_{ix}, u_{iy});

  Else
    - *MixDraw_PlacePhase* (u_i, u_{ix}, u_{iy});

  Endif;

EndFor;
EndIf;
3. If v is a node representing a contact node
then
- **For** all the children u_i of v **do** *MixDraw_PlacePhase* (u_i, v_x, v_y);

EndIf;
End.

Since we traverse the tree T_3 in preorder, and we process each node in T_3 only once, such that, first, we draw the parent node then recursively we draw the children. The time complexity of *MixDraw_PlacePhase* algorithm will be $O(n)$, where n is the number of nodes in the ring cover (see chapter 2 section 2.3.4 for full detailed analysis).

Using the above procedure, we conclude the following lemma :

Lemma 8

Given a ring cover C and its generalized-ring-contact node tree T_3, the algorithm mixed draw produces a ring cover drawing such that:

1. There are no crossings;
2. the area required by the drawing is $O(n^2)$, where n is the number of nodes in C.

Proof

1. There are no crossings, since we use only the inside and the outside drawings techniques. We start our drawing from the root v of T_3, if the *INSIDE* value of v is true then the ring cover will be drawn by the inside algorithm, in this case there are no crossings as proved in lemma 5. If the *INSIDE* value of v is false, then v and all its children will be drawn one outside the other, in this case there are also no crossings, since we use the outside drawing algorithm as proved in lemma 7. The same technique of drawing will be applied to the children of the node v, so that in the final stage, there will be no crossings.

2. Let u be a node in T_3, we assume first that u is a ring in T_3. Since we use either the inside drawing algorithm or the outside drawing algorithm, the area of the enclosing cycle corresponding to u is $O(n^2)$, where n is the number of nodes in u plus the number of nodes in its subtree (as described in proof of lemma 5 and lemma 7). If u is a ring-cycle, such that u is not a root in T_3, the area of the enclosing cycle corresponding to u is $O(n^2)$, where n is the number of the nodes in the subtree of u plus the number of the nodes existing between the two special contact nodes cn_i and cn_{i+1} which belong to the father ring of u (as described in proof of lemma 5 and lemma 7). It follows, that if u is a root of T_3, the area of the enclosing cycle corresponding to u is $O(n^2)$, where n is the number of nodes in the ring cover.

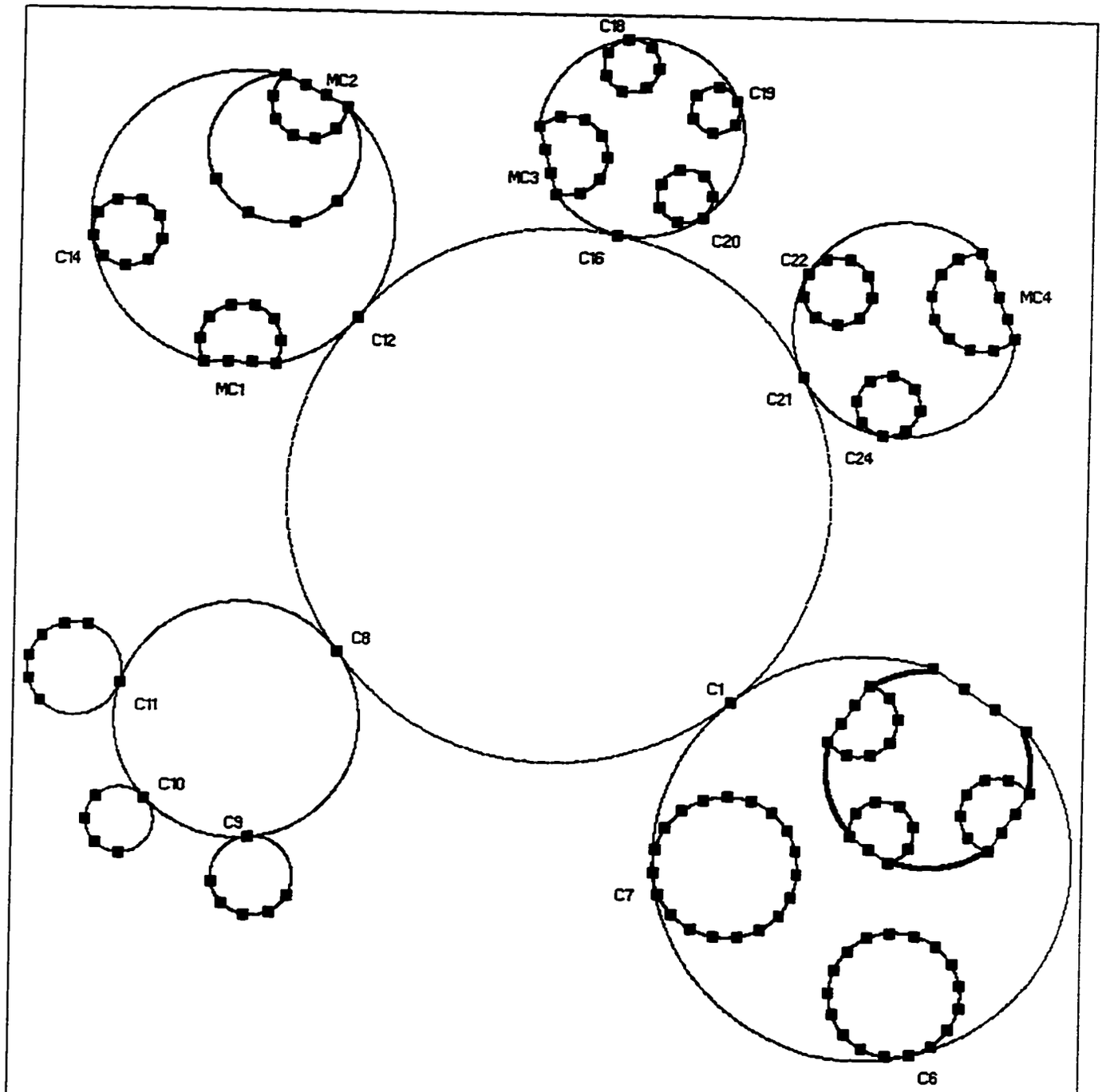


Figure 4.2: A picture of a ring cover generated by the system. We notice that if a ring u has at least one child which shares more than one contact node with other rings or a ring-cycle, then u and all the nodes of its subtree will be drawn one inside the other.

4.4 Optimum layout of the generalized-ring-contact node tree

It is interesting to observe that the order of the area occupied by the drawing does not depend on the starting point. In other words, the order of the area remains unchanged, no matter which node is considered as the root of the tree T_3, but the exact value of the area of the drawing depends on the starting point in the tree T_3. In order to minimize the area of our drawing, we propose the *Optimum_Layout_Tree* algorithm which determines the node that we have to start with as the root of T_3.

Given a ring cover and its generalized-ring-contact node tree T_3, our algorithm provides the optimum layout of the generalized-ring-contact node tree T_3 which occupies the minimum area.

First, we traverse the tree T_3 in preorder fashion, such that in our traversal, we save the location of each node of type ring or ring-cycle that exists in the tree T_3 on a set S. The length of the set S is the number of the rings plus the number of the ring-cycles in the ring cover. This first traversal of the tree T_3 is of order $O(n)$, where n is the number of nodes in the ring cover, because each node in the tree T_3 is visited only once.

Second, for each node of type ring or ring-cycle of the set S, we make a transformation to the tree T_3, such that the actual node visited in the set S becomes the root of the tree T_3.

Let's consider the example in figure 4.3, let's take R_2 the candidate node visited in the set S. We make a transformation to T_3, such that R_2 becomes the new root of T_3, we compute then the area used for this structure and we repeat the same process to all the other nodes in the set S. At the end we keep apart the optimum layout of the tree T_3 which uses the minimal area in our drawing.

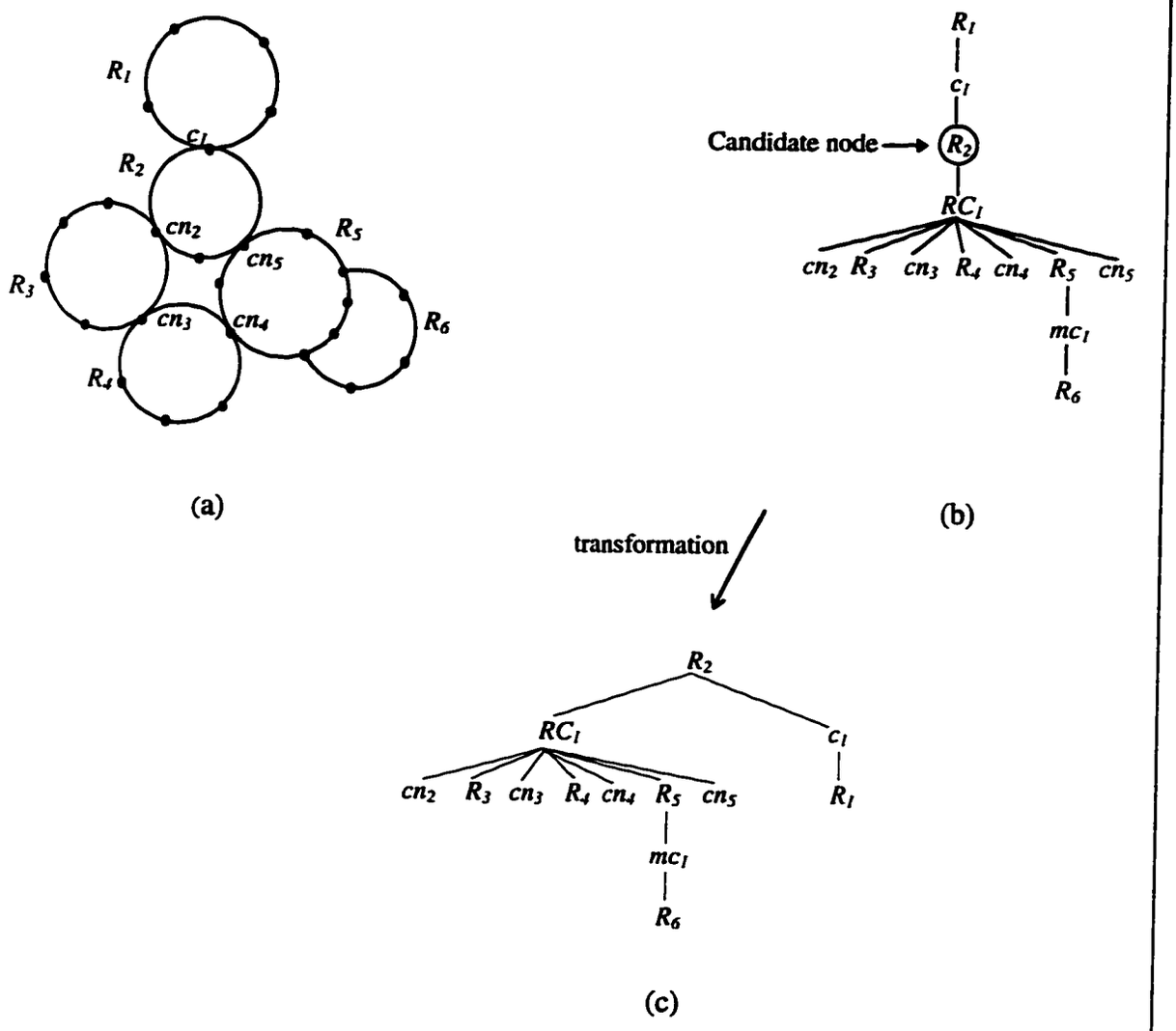


Figure 4.3 **(a)**: Ring cover C.
(b): Generalized-ring-contact node tree T_3.
(c): Transformation of the tree T_3, such that R_2 becomes the new root of the tree T_3.

Before presenting the *Optimum_Layout_Tree* algorithm, we need to define the following variable used in our algorithm.

CN: represents the candidate node chosen in the set S,

Area: represents the area required for each new layout of the tree T_3,

TN: represents the root of the optimum layout of the tree T_3,

Minimum_area: represents the area required for the optimum layout of the tree T_3.

Algorithm *Optimum_Layout_Tree*;
Input : The Generalized-ring-contact node tree T_3 and the set S.
Output : The optimal layout tree T_3'.
Begin
1. **CN** ⟵ root of T_3;
2. **TN** ⟵ **CN**;
3. **Minimum_area** ⟵ area required for the tree T_3;
   {using one of the previous algorithm (Inside , outside , Mixed)}
4. **For** all the nodes u_i of type ring or ring-cycle that exist in the set S **do**
   Begin
   - **CN** ⟵ u_i;
   - Copy the tree T_3 to the tree T_3';
   - Transform T_3' such that **CN** becomes the root of T_3' as shown in figure 4.3(c);
   - **AREA** ⟵ area required for the tree T_3';
     { using one of the previous algorithm (Inside , outside , Mixed)}
   - **if AREA < Minimum_Area then**
     Begin
     - **Minimum_Area** ⟵ **AREA**;
     - **TN** ⟵ **CN**;

     EndIf;

   EndFor;
5. Copy the tree T_3 to the tree T_3';
6. Transform T_3' such that **TN** becomes the root of T_3' as shown in figure 4.3(c);

End.

The Optimum_Layout_Tree algorithm has a time complexity $O(m \times n+m^3)$, where n is the number of nodes in the ring cover and m is the number of the rings in the ring cover.

For each node of type ring or ring-cycle of the set S, we use the inside, outside or mixed drawing algorithm that has a complexity of order $O(n + m^2)$ in order to compute the area needed for the drawing. Let m' be the size of the set S. Thus m' is equal to the number of the rings plus the number of the ring-cycles. We know that m' is of order $O(m)$. The time complexity of Optimum_Layout_Tree algorithm will be then $O(m \times n+m^3)$.

Chapter 5

Conclusion and Open Problems

In our dissertation, we presented three techniques for drawing a ring cover of survivable telecommunication networks: the inside drawing, the outside drawing and the mixed drawing. Our contribution to this subject was to expand the drawing of the ring cover to more complicated structures of survivable telecommunication networks.

In fact, we presented an improved version for the outside and the inside drawing in order to deal with the case where the rings share more than one contact node. We proposed the ring-multi-contact node graph as a new underlying structure for the ring cover. Then we have considered the case where the ring cover contains ring-cycles in which we have presented the extended-ring-contact node graph as the new underlying structure for the ring cover. Finally we considered the case where the ring cover contains ring-cycles and rings that share more than one contact node. We also introduced the generalized-ring-contact node graph as the final underlying structure for the ring cover.

While dealing with the outside drawing technique, we encountered the problem that occurs when at least three rings share the same multi-contact node, that is why we proposed a new approach for the mixed drawing algorithm different from the one presented in [TOXI94].

In order to guarantee the readability of the drawing, we have to respect the following criteria:

1. No crossing is allowed, because if it is the case, we may visualize rings that do not exist in the original network.
2. Rings should be easily recognizable in the picture.
3. The resolution rule should be respected (for visibility purposes).
4. Use of minimal area.

Our proposed drawing algorithms produce drawings that require $O(n^2)$ area, where n is the number of nodes in the ring cover.

The design of these three algorithms (inside drawing, outside drawing and mixed drawing), allowed us to introduce a new method which produces the optimum layout of the underlying structure corresponding to the ring cover, in order to minimize the area of the drawing.

Besides our theoretical research, we have done an experimental work which consist of implementing the previously designed algorithm. For efficiency purposes, we used the object-oriented language called "Borland Delphi".

There are still some issues which have not been addressed yet and that we consider them as being open problems, two of them are described here:

1. Ring cover drawing when the intersection of two ring-cycles consists of more than one ring.
2. Ring cover drawing when the generalized-ring-contact node graph is not a tree.

Bibliography

[BPTI89] G. Di Battista, E. Pietrosanti, R. Tamassia, and I.G. Tollis, "Automatic Layout of PERT Diagrams with XPERT," Proc. IEEE Workshop on Visual Languages (VL'89).

[BETI93] G. Di Battista, P. Eades, R. Tamassia and I.G. Tollis. "Algorithms for Automatic Graph Drawing: An Annotated Bibliography," Department of computer Science, Brown University, Technical Report, 1993.

[BNT86] C. Batini, E. Nardelli, and R. Tamassia, "A Layout Algorithm for Data Flow Diagrams". IEEE Trans Software Eng., vol. 12 1986.

[BMT84] C. Batini, M. Talamo, and R. Tamassia, "Computer Aided Layout of Entity-Relationships Diagrams" The Journal of Systems and Software, vol. 4, 1984.

[GHSITX94] L.M. Gardner, M. Heydari, J. Shah, I.H. Sudborough, I.G. Tollis, and C. Xia, "Techniques for Finding Ring Covers in Survivable Networks" Proceedings of IEEE GLOBECOM 1994.

[GVSM90] W.D.Grover, B.D.Venables, J.H. Sandham, and A.F. Milne, "Performance Studies of a Self-Healing Network Protocol in Telecom Canada Long Haul Networks " Proceedings of IEEE GLOBECOM 1990.

[KMG88] G. Kar, B. Madden and R.S. Gilbert, "Heuristic layout Algorithms for Network Management Presentation Services" IEEE Network, November, 1988.

Bibliography

[TOXI94] Ioannis G. Tollis and Chunliang Xia. "Drawing Telecommunication Networks". DIMACS International Workshop, Graph drawing'94. Princeton, New Jersey, USA, October 1994. Proceedings.

[TOXI93] Ioannis G. Tollis and Chunliang Xia. "Graph drawing algorithms for the design and analysis of telecommunication networks". Graph Drawing'93, Proceedings of the ALCOM International workshop on graph drawing Sèvres France.

[SHAH92] J.C.Shah , "Restoration Network Planning Tool", Proc. 8th Annual Fiber Optic Engineers Conf. April 21, 1992.

[SNH90] H.Sakauchi, Y.Nishimura, and S.Hasegawa, " A Self-Healing Network with an economical Spare-Channel Assignment " IEEE GLOBECOM 1990.

[RIDO88] Richard L. Burden and j.Douglas Faires " Numerical analysis " fourth edition, 1988.

[WULA92] Tsong-Ho Wu and R. C. Lau , "A Class of Self-Healing Ring Architectures for SONET Network Application" IEEE Trans. on Communication, Vol. 40, No. 11 Nov. 1992.

[WCA88] T.H. Wu, D.J. Collar, and R.H. Cardwell, "Survivable Network Architectures for Broad-band Fiber Optic Networks: Model and Performance Comparison" IEEE journal of Lightwave Technology, Vol. 6, No. 11, November 1988.