2
Algoritmo para Segmentação e Contagem de Clusters de Máximos Locais

Este capítulo apresenta o método proposto para segmentação e contagem de *clusters* de máximos locais (CML) em uma imagem digital. Este método pode ser dividido em duas grandes etapas: a etapa de segmentação e a etapa de contagem.

Na etapa de segmentação, através da informação de luminância uma representação da imagem baseada em grafo é construída. O CML é definido como um subgrafo e um algoritmo de mineração é utilizado para encontrar os *clusters* no grafo que representa a imagem. Em situações onde o objeto de interesse apresenta vários pontos brilhantes que possuam alturas muito diferentes, um algoritmo de clusterização de grafos pode ser incorporado ao processo para agrupar estes pontos, melhorando o resultado final. A etapa de contagem dos objetos é um resultado direto do algoritmo de mineração e de clusterização, quando este último é aplicado.

O método como um todo é composto de seis passos: cinco passos obrigatórios (aplicação do filtro Gaussiano na etapa de pré-processamento, partição do histograma, detecção de componentes conectadas, construção do grafo e mineração do grafo) e um passo opcional (clusterização do grafo) para lidar com o caso de vários pontos brilhantes. A Figura 2.1 fornece uma visão geral do método proposto.

(a)

Figura 2.1: Visão geral do método proposto: cada passo é representado por uma caixa. Os ítens em cinza (clusterização do grafo e segmentação do fundo) representam os passos opcionais.

No primeiro passo (pré-processamento) é aplicado um filtro Gaussiano para diminuir o ruído e se necessário o plano principal da imagem (objetos) é sepa-

rado do fundo. No segundo passo, o eixo horizontal do histograma da imagem é particionado em intervalos de tamanho fixo. Em seguida, no terceiro passo, as componentes conectadas são detectadas e no passo seguinte um grafo de adjacência, o qual representa a imagem, é construído. Finalmente, no quinto passo, um processo de mineração de grafos é aplicado para encontrar os CML. Se necessário, um sexto passo (clusterização do grafo) pode ser adicionado ao processo.

Para facilitar o entendimento do método proposto são necessárias algumas definições, as quais são apresentadas a seguir.

Seja $I : \Omega \in \mathbb{Z}^2 \longrightarrow [0, 255]$ uma função que representa uma imagem em tons de cinza. Esta imagem pode ser visualizada geometricamente como o gráfico $G(I)$ da função I, $G(I) = \{(x, y, z); (x, y) \in \Omega \text{ e } z = I(x, y)\}$, considerando os valores de intensidade como a altura $z = I(x, y)$ em cada ponto (x, y) do domínio [9]. Este fato é ilustrado na Figura 1.2. Esta interpretação geométrica permite uma visão mais intuitiva de certos aspectos da imagem.

Neste trabalho, uma faixa de nível de I em $z = c$ é definida como o conjunto $L_c = \{(x, y) \in \Omega; (c - 1)\varepsilon \leq I(x, y) < c\varepsilon\}$ para um número inteiro positivo ε fixo e $c = 1, 2, 3, \ldots, 256/\varepsilon$. Grandes valores para ε correspondem a poucas curvas de nível e vice-versa. As Figuras 2.2(a) e 2.2(c) apresentam exemplos de curvas de nível para valores de $\varepsilon = 64$ e $\varepsilon = 32$, respectivamente. O gráfico 3D mostrado nesta figura corresponde à imagem exibida na Figura 1.2(a). Os planos coloridos representam as curvas de nível e os números a esquerda do eixo z correspondem ao valor de c. Como descrito anteriormente, o topo de um CML pode conter vários pontos de máximos locais. Assim, ao traçar estas curvas de nível, os pontos do domínio que possuem uma diferença de altura inferior a ε são agrupados em *clusters*.

Um modo equivalente de se obter os conjuntos L_c acima é dividir o eixo horizontal do histograma da imagem em intervalos de tamanho ε e agrupar os pixels de acordo com sua luminância, conforme ilustrado nas Figuras 2.2(b) e 2.2(d). Estas figuras mostram o histograma da imagem apresentada na Figura 1.2(a) e os respectivos intervalos para valores de $\varepsilon = 64$ e $\varepsilon = 32$. Cada intervalo do histograma corresponde a uma classe de luminância e os números de cada intervalo equivalem ao nível da curva que ele representa.

As próximas seções deste capítulo apresentam cada passo do método proposto em detalhes.

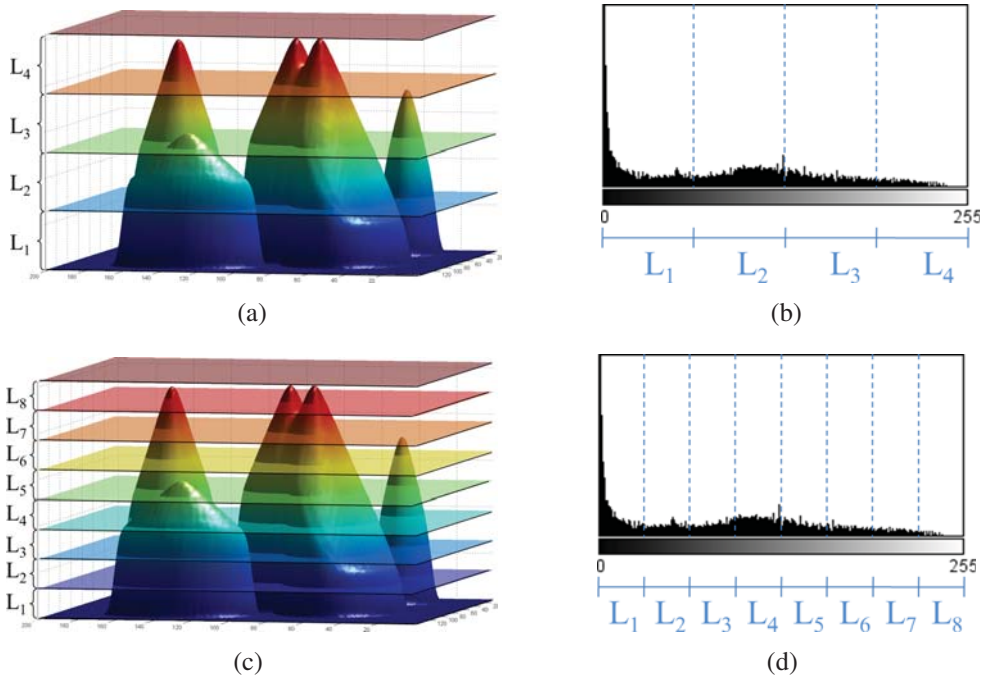


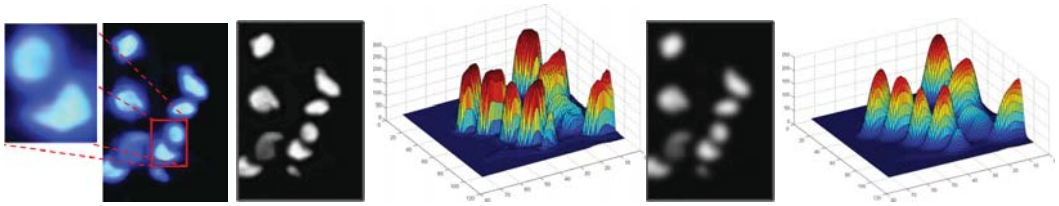
Figura 2.2: Curvas de nível e partição do histograma para a imagem apresentada na Figura 1.2(a) para dois valores de ε: (a) e (b) apresentam os resultados para $\varepsilon = 64$ e; (c) e (d) para $\varepsilon = 32$, respectivamente.

2.1 Pré-processamento

O passo de pré-processamento é composto por duas etapas: aplicação de um filtro Gaussiano para reduzir o ruído e, se necessário, a segmentação do fundo onde o plano principal de imagem (objetos) é separado do fundo através de um *threshold* global simples.

O objetivo da aplicação do filtro Gaussiano é reduzir o número de falsos picos sem borrar ou fundir objetos (CML) diferentes em um. Quando a imagem apresenta objetos com mais de um ponto brilhante, esta etapa raramente os mescla, conforme ilustrado na Figura 2.3(e). Estes picos extras serão tratados pela última etapa do algoritmo. A Figura 2.3(a) apresenta a imagem de entrada e a imagem no canto superior esquerdo destaca o fundo não uniforme. A Figura 2.3(b) mostra a mesma imagem em tons de cinza (informação de luminância) e a Figura 2.3(c) apresenta o gráfico 3D correspondente. As Figuras 2.3(d) e 2.3(e) mostram o resultado obtido após a aplicação do filtro Gaussiano na imagem e na superfície, respectivamente.

A segmentação do fundo é opcional. Esta etapa é, geralmente, aplicada quando a imagem possui fundo não uniforme, como as imagens de microscopia fluorescente (Figura 2.4(b)), por exemplo. De modo geral, as imagens são capturadas em um ambiente controlado e, portanto, o plano principal da imagem (objetos) pode ser separado do fundo da imagem através de um *threshold* global simples $t = \mu + x\delta$,

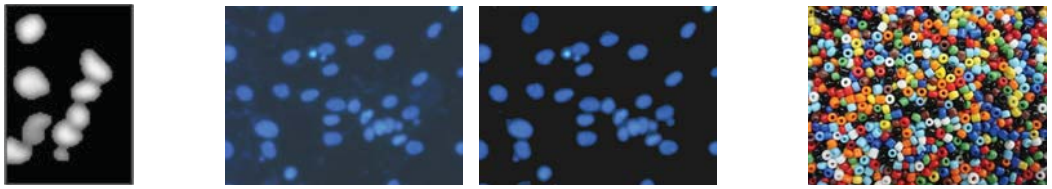


(a) (b) (c) (d) (e)

Figura 2.3: Resultado do filtro Gaussiano: (a) imagem de entrada; (b) e (c) informação de luminância e superfície topológica correspondente; (d) e (e) resultado do filtro Gaussiano sobre a imagem e sobre a superfície, respectivamente.

onde μ é o valor médio da imagem, δ é o desvio-padrão da imagem, e x é uma constante definida experimentalmente. A parte final desta etapa consiste em zerar os pixels que possuem valor de luminância menor que t.

A Figura 2.4(a) mostra o resultado da etapa de segmentação de fundo para a imagem da Figura 2.3(a). As Figuras 2.4(b) e 2.4(d) exemplificam quando a etapa de segmentação do fundo deve, ou não, ser aplicada. A imagem apresentada na Figura 2.4(b) possui um fundo não uniforme. Nesta situação, o plano principal da imagem deve ser separado do fundo da imagem. O resultado obtido é apresentado na Figura 2.4(c). Por outro lado, a imagem da Figura 2.4(d) não possui fundo. Logo, para este tipo de imagem a etapa de segmentação não deve ser aplicada.



(a) (b) (c) (d)

Figura 2.4: Resultado da etapa de segmentação do fundo e exemplos de quando ela deve ou não ser aplicada: (a) resultado desta etapa para imagem apresentada na Figura 2.3(a); (b) e (c) imagem de microscopia fluorescente e respectivo resultado da segmentação do fundo; e (d) imagem de miçangas, a qual não possui fundo.

2.2 Partição do Histograma

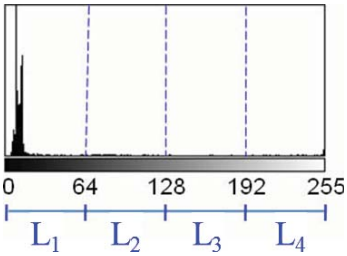
O segundo passo do método proposto consiste basicamente em particionar o eixo horizontal do histograma da imagem em intervalos de tamanho ε fixo e agrupar os pixels de acordo com o seu valor de luminância. Cada intervalo da partição define uma classe de luminância. Como conseqüência desta partição, os CML são decompostos em regiões, as quais pertencem a classes diferentes de acordo com sua intensidade.

A variável ε recebe valores inteiros positivos divisores de 256 e valores típicos para ε são 8, 16 e 32. Valores muito pequenos para ε, tais como 1, 2, ou 4, geralmente produzem super-segmentação como resultado. Como estes valores são muito pequenos, os pontos de máximo não são agrupados nas classes de luminância e praticamente todos os pontos de máximo local são segmentados. Como conseqüência, o mesmo objeto é segmentado em vários pedaços e contado mais de uma vez. Além disso, o método pode se tornar extremamente lento, pois o grafo que representa a imagem se torna muito grande e denso. Por outro lado, valores muito grandes para ε como, por exemplo, 64 ou 128, de modo geral impedem que objetos sobrepostos ou fortemente agrupados sejam segmentados individualmente uma vez que pontos com alturas bem diferentes são agrupados no mesmo intervalo.

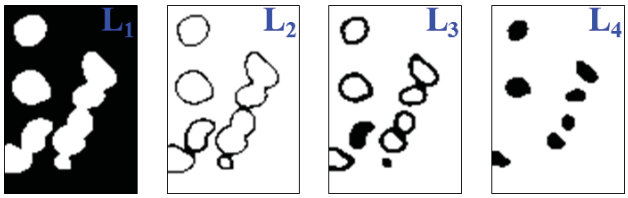
Assim, ajustando o tamanho do intervalo de modo adequado, é possível detectar objetos que tenham mais de um ponto brilhante, ou seja, mais de um ponto de máximo local e também aqueles que estejam muito próximos ou até mesmo sobrepostos.

Cada intervalo $[a, b)$ da partição é representado por uma imagem binária onde os *pixels* pretos correspondem àqueles cujo valor de luminância pertencente à $[a, b)$. Como conseqüência desta representação, os CML são decompostos em componentes conectadas, as quais pertencem à imagens binárias distintas de acordo com sua intensidade. A Figura 2.5 apresenta a partição do histograma e as respectivas imagens binárias para a imagem exibida na Figura 2.3(a). As Figuras 2.6(a) e 2.6(b) mostram as imagens binárias correspondentes à partição do histograma apresentada nas Figuras 2.2(b) e 2.2(d), respectivamente. Os L_c ($c = 1, 2, 3, 4$) no canto superior direito de cada imagem binária correspondem ao número do intervalo da partição que esta imagem representa. Nestas figuras, o histograma da imagem foi particionado em intervalos de tamanho $\varepsilon = 64$ e $\varepsilon = 32$. Estes valores de ε são apenas para propósitos de ilustração. Os reais valores para o tamanho do intervalo da partição é menor como mostrado no decorrer deste trabalho.

Na Figura 2.6(a) pode-se observar que os CML que estão muito próximos foram, incorretamente, detectados como um só. Isto acontece quando valores grandes para ε são utilizados em imagens onde os objetos estão sobrepostos ou fortemente agrupados. No entanto, ao se diminuir o tamanho do intervalo para $\varepsilon = 32$ (Figura 2.6(b)) estes CML são detectados corretamente. Logo, quanto mais sobrepostos ou fortemente agrupados estiverem os objetos, menor deve ser o tamanho (ε) do intervalo. Ou seja, o valor de ε varia de acordo com as características da imagem.

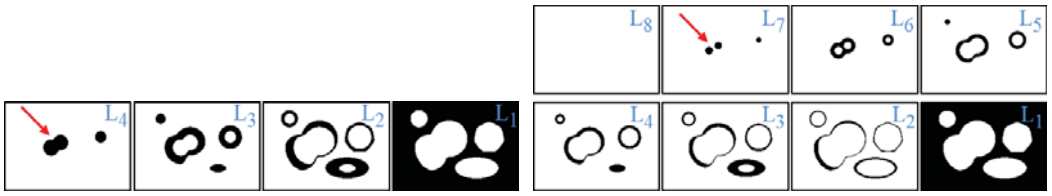


(a)Histograma da imagem particionado em intervalos de tamanho $\varepsilon = 64$



(b)Imagens binárias representado cada um dos intervalos da partição

Figura 2.5: Partição do histograma e imagens binárias correspondentes para a imagem apresentada na Figura 2.3(a).



(a) (b)

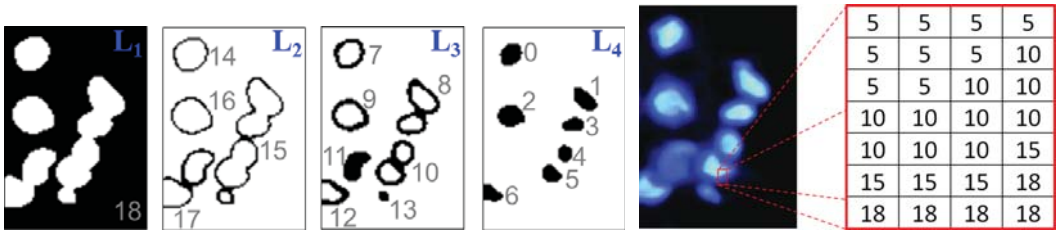
Figura 2.6: Imagens binárias representando cada intervalo da partição do histograma apresentado na Figuras 2.2(b) e 2.2(d), respectivamente. As setas na cor vermelha indicam as mudanças decorrentes de valores diferentes para o tamanho do intervalo

2.3 Detecção das Componentes Conectadas

Após a partição do histograma, o próximo passo é dividir o plano principal da imagem em regiões 8-conectadas que pertençam à mesma classe de luminância, ou seja, à mesma imagem binária. Uma vizinhança 4-conectada também poderia ter sido utilizada, no entanto, aumentaria o tamanho do grafo de modo considerável.

Para detectar as componentes, as imagens binárias são percorridas em ordem decrescente de luminância, isto é, a partir da imagem que contém os pixels mais claros, a qual representa o intervalo $[256 - \varepsilon, 256)$, até aquela que contém os pixels mais escuros representando o intervalo $[0, \varepsilon)$. Em cada uma dessas imagens é aplicado o algoritmo apresentado em [3], projetado para encontrar componentes conectadas em imagens binárias. A medida que as componentes são detectas, elas passam a ser identificadas com números naturais consecutivos, tais que regiões com maior luminância recebem números menores. Esta ordem numérica é importante na etapa de mineração do grafo. Ao adicionar esta condição à definição clássica de caminho simples [26] um CML fica caracterizado por um subgrafo conforme será mostrado na Seção 2.5. A Figura 2.7(a) apresenta as componentes detectadas em cada imagem binária e seus respectivos identificadores, apresentados na cor cinza, para a imagem da Figura 2.3(a).

O resultado final da etapa de detecção das componentes conectadas é uma matriz M com as mesmas dimensões da imagem de entrada, onde cada entrada $M(i,j)$ da matriz contém o identificador da componente que o pixel $p(i,j)$ pertence. Ou seja, $M(i,j) = k \Leftrightarrow p(i,j) \in C_k$, $1 \leq k \leq R$, onde $p(i,j) \in \Omega$ é um pixel da imagem e R é o número total de componentes detectadas. A Figura 2.7(b) mostra uma pequena parte da matriz M, a qual corresponde a parte da imagem destacada em vermelho. Uma vez criada a matriz M, o próximo passo é construir a representação da imagem baseada em grafo conforme descrito a seguir.



(a)Componentes detectadas para cada imagem binária e seus respectivos identificadores (números na cor cinza).

(b)Parte da matriz M, a qual corresponde a parte da imagem destacada em vermelho.

Figura 2.7: Componentes conectadas com seus respectivos identificadores e parte da matriz M para a imagem apresentada na Figura 1.2(a).

2.4 Construção do Grafo

Um grafo G é um par de conjuntos $G = (V, E)$, onde V representa o conjunto de vértices (ou nós) e E contém as arestas (ou linhas) do grafo [78]. Um grafo de adjacência (*region adjacency graph* – RAG) é uma estrutura de dados que fornece uma visão espacial da imagem. Este tipo de grafo consiste em associar um vértice a cada região da imagem e uma aresta a cada par de regiões adjacentes [62].

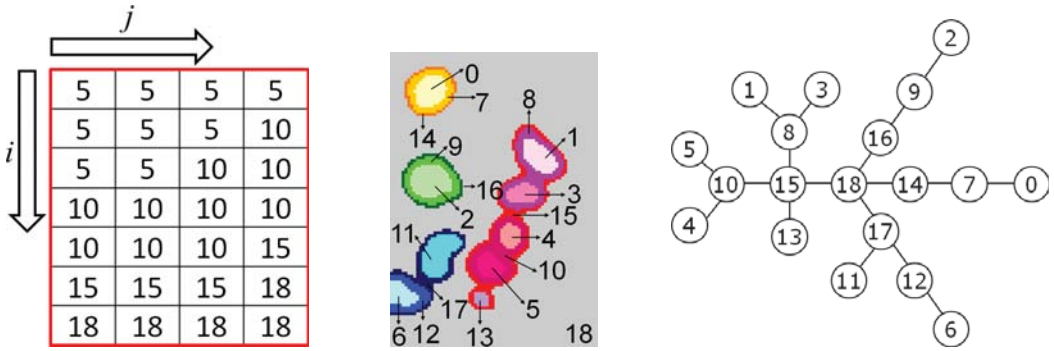
O objetivo final deste passo é construir um RAG baseado na matriz M. Cada vértice $v_i \in V$ representa uma região (componente conectada) da imagem e seu índice i corresponde diretamente ao identificador da região que ele representa. Por exemplo, o vértice v_3 no grafo representa a componente conectada que possui o identificador de número 3. As arestas em E conectam pares de regiões adjacentes. Com o intuito de reduzir o número de arestas no grafo, uma vizinhança 4×4 foi utilizada, como descrito na definição a seguir:

Definição 2.1 (Componentes Adjacentes) *Duas componentes C_i e C_j são adjacentes se existir pelo menos um par de pixels $p_i \in C_i$ e $p_j \in C_j$, tais que p_i e p_j são vizinhos 4×4.*

O grafo de adjacência G é construído varrendo-se a matriz M de cima para baixo e da esquerda para a direita. Para cada elemento $m(i,j) \in M$ são

verificados o vizinho superior $m(i+1,j)$ e o da esquerda $m(i,j-1)$. Se pelo menos o valor em um deles for diferente do valor em $m(i,j)$ então estes *pixels* pertencem à componentes diferentes. Logo, de acordo com a Definição 2.1 acima, estas componentes são adjacentes e portanto deve existir uma aresta em G ligando os vértices correspondentes.

A Figura 2.8 mostra uma pequena parte da matriz M (Figura 2.8(a)), as componentes conectadas com seus respectivos identificadores (Figura 2.8(b)), os quais correspondem diretamente ao índice do vértice que a representa, e o RAG que representa esta imagem (Figura 2.8(c)). Nesta figura pode-se observar que as componentes 5, 10, 15 e 18 são adjacentes e como conseqüência os vértices v_5, v_{10}, v_{15} e v_{18} aparecem conectados no grafo.

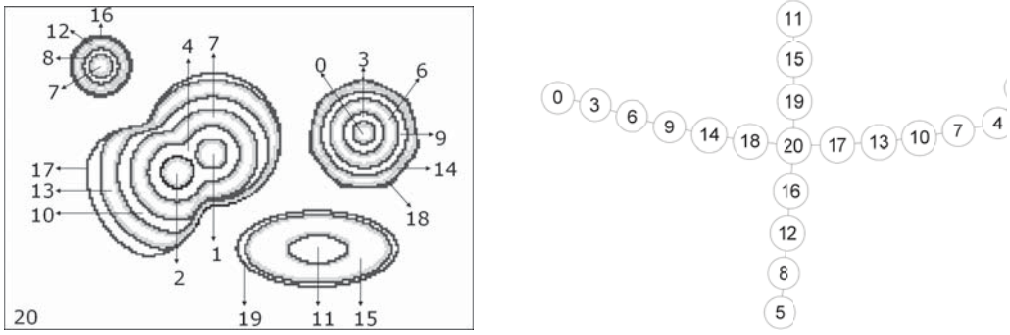


(a) (b) (c)

Figura 2.8: Matriz M e representação da imagem baseada em grafo para a imagem da Figura 1.2(a): (a) pequena parte da matriz M; (b) componentes detectadas com respectivos identificadores; e (c) RAG correspondente.

A Figura 2.9 apresenta as componentes detectadas e o respectivo grafo de adjacência para a imagem da Figura 2.3(a). Exemplos de representação da imagem baseada em grafo apresentada neste trabalho para imagens naturais são apresentados na Figuras 2.10 e 2.12.

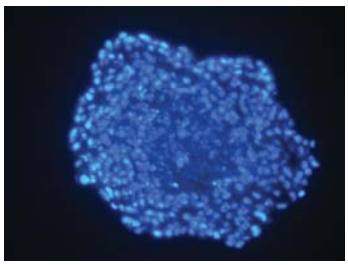
Nos grafos que representam a imagem como, por exemplo, aqueles mostrados nas Figuras 2.9(b), 2.10 e 2.12 têm-se que os vértices com índices menores, isto é, aqueles que correspondem as regiões mais brilhantes, estão localizados na extremidade do grafo. Tomando como centro do grafo o vértice de valor máximo, pode-se observar também que estes grafos se tornam mais densos a medida que se caminha em direção ao seu centro. Esta característica deve-se ao fato dos vértices com índices maiores, de modo geral, corresponderem a regiões vizinhas a várias outras regiões. Além disso G é um grafo não direcionado, simples e que não contém vértices isolados uma vez que todos os vértices estão conectados, pelo menos, ao vértice que representa o fundo da imagem e/ou aquele que representa a maior componente conectada.



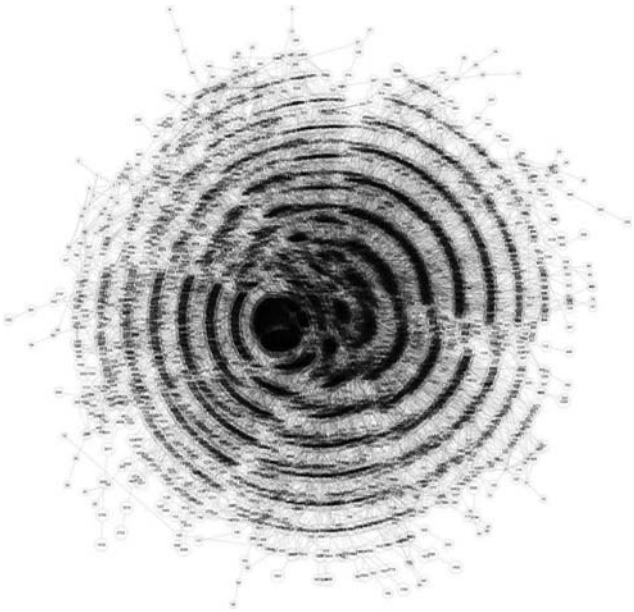
(a)Componentes detectadas e respectivos identificadores

(b)Grafo de adjacência correspondente

Figura 2.9: Componentes conectadas e representação da imagem baseada em grafo para a imagem da Figura 2.3(a).



(a)Imagem de microscopia fluorescente.



(b)Grafo de adjacência correspondente para $\varepsilon = 8$.

Figura 2.10: Exemplo de representação da imagem baseada em grafo para uma imagem natural.

2.5 Processo de Mineração do Grafo

A mineração de grafos tem como objetivo encontrar padrões (subgrafos) de interesse que estejam imersos em um único grafo ou em uma coleção de grafos. Subgrafos de interesse incluem subgrafos freqüentes, de um tipo especial como completos ou bi-partidos, e ainda qualquer subgrafo que represente um padrão relevante para aplicação [22, 11].

O método proposto nesta tese tem como objetivo segmentar e contar CML em uma imagem digital. Desta forma, para esta aplicação em particular, o subgrafo de interesse é aquele capaz de representar um CML em G. Como as imagens utilizadas

neste trabalho geralmente contêm vários objetos que são representados por CML, este subgrafo além de representar um padrão relevante é também um subgrafo freqüente, uma vez que aparece várias vezes no grafo que representa a imagem.

A tarefa de mineração de grafos pode ser classificada como *transaction graph-mining* ou *single graph-mining*. Em se tratando de *transaction graph-mining*, a base de dados a ser minerada compreende uma coleção de grafos pequenos (*transactions*) e o objetivo é descobrir subgrafos recorrentes dentro da base de dados. Já em *single graph-mining*, a entrada para a tarefa de mineração é um único grafo, geralmente grande, e o objetivo é descobrir um tipo de subgrafo freqüente que ocorre dentro do grafo. Mais detalhes sobre mineração de grafo podem ser encontrados em [43].

O processo de mineração de grafos descrito neste trabalho adota um algoritmo do tipo *single graph-mining*, uma vez que o objetivo é encontrar um subgrafo específico - aquele que caracteriza um CML - dentro de um único grafo, o qual representa a imagem de entrada.

Na etapa de partição do histograma um CML é decomposto em componentes adjacentes, as quais correspondem à uma seqüencia de vértices conectados. Além disso, uma componente que pertença a um CML pode ser adjacente, no máximo, à outras duas componentes. Desta forma, esta seqüência contém somente vértices com grau um ou dois. Na Seção 2.3 ficou estabelecido que quanto maior a luminância de uma região menor é o número do seu identificador, o qual corresponde diretamente ao índice do vértice que a representa. Assim, esta seqüencia de vértices deve estar em ordem crescente. Logo, um CML em G fica caracterizado como um caminho simples, isto é uma seqüencia de vértices tal que de cada um dos vértices existe uma aresta para o próximo vértice da seqüencia [78], cujos vértices estão em ordem crescente.

De maneira formal, um subgrafo S de G que representa um CML é definido como:

Definição 2.2 (CML como um subgrafo) *Um CML em um grafo G é representado por um subgrafo (caminho simples) S que satisfaz:*

1. *os vértices de S são todos distintos;*
2. *se S contém mais de um vértice, então de cada um deles existe uma aresta para o próximo vértice da seqüencia;*
3. *cada vértice de S tem, no máximo, grau dois;*
4. *os índices dos vértices devem estar em ordem crescente.*

Assim, um CML em G é identificado por uma seqüencia de vértices $S = v_0, v_1, \ldots, v_{k-1}, v_k$, onde as arestas não estão explicitamente representadas dado que todos os vértices são distintos e portanto todas as arestas são também distintas.

Para encontrar os CML presentes em G, foi desenvolvido o processo de mineração de grafo apresentado no Algoritmo 1. Dado que em um grafo típico, como aquele mostrado nas Figuras 2.12 e 2.10, geralmente os CML (caminhos simples) aparecem na extremidade do grafo, o processo de mineração desenvolvido utiliza uma busca em ordem. A entrada é um grafo finito $G = (V, E)$ e a saída é um conjunto $L = \{S_1, \ldots, S_k\}$ onde cada $S \in L$ representa um caminho simples. Todos os vértices de G são inicializados como pertencente a nenhum caminho e como não visitado. Partindo do vértice v_0, o conjunto V é percorrido iterativamente e somente os vértices com grau 1 ou 2 são avaliados, pois de acordo com a Definição 2.2 um subgrafo que representa um CML possui vértices com grau máximo 2. Estes vértices são classificados como um novo caminho simples ou como parte de algum caminho detectado anteriormente. Os vértices com grau um e dois são avaliados por procedimentos distintos como descrito a seguir.

Algorithm 1

```
1:  function GRAPHMINING(G = (V, E))
2:      S = {}
3:      Set all vertices in G as not visited
4:      for i ← 0, |V| do
5:          Let v_i ∈ V be the node associated with the i^th
6:          connected component detected.
7:          if v_i has degree 1 then
8:              EVALUATENODE1(S, v_i)
9:          else if v_i has degree 2 then
10:             EVALUATENODE2(S, v_i)
11:         end if
12:         Set v_i as visited
13:     end for
14:     return S
15: end function
```

O Algoritmo 2 é responsável por avaliar e classificar os vértices que têm um único vizinho. Sendo G um grafo conexo, estes vértices estão localizados na extremidade do grafo. Assim, um vértice de grau 1 é classificado como um novo caminho simples se: (a) o seu vizinho ainda não tiver sido visitado (avaliado) ou (b) se o seu vizinho não pertencer à caminho simples algum. Caso contrário, o vértice corrente é adicionado ao mesmo caminho simples que seu vizinho pertence. Ao final da avaliação, os vértices são definidos como visitados.

Os vértices com grau dois são avaliados pelo Algoritmo 3. Para estes vértices tem-se três possibilidades: (1) ambos os vizinhos estão definidos como visitados; (2) somente um vizinho está definido como visitado e; (3) ambos os vizinhos estão definidos como não visitados. A seguir, o procedimento para cada uma dessas possibilidades é apresentado. Ao final da avaliação, os vértices são definidos como visitados.

Algorithm 2

```
1: procedure EVALUATENODE1(S, v)
2:     Let v' be the neighbor of v
3:     if v' is set as visited and ∃ s ∈ S | v' ∈ s then
4:         s = s ∪ {v}
5:     else
6:         s = {v}
7:         S = S ∪ s
8:     end if
9: end procedure
```

Ambos os vizinhos estão definidos como visitados Se pelo menos um dos vizinhos pertencer a algum caminho simples, então o vértice corrente corresponde a uma região que pertence ao mesmo CML que este caminho representa. Logo, o vértice corrente é adicionado à este caminho simples (linhas 4-6 do Algoritmo 3). Por outro lado, se os vértices vizinhos pertencem a caminhos diferentes, então uma região da imagem que é adjacente a dois CML distintos foi encontrada. Logo, o vértice corrente não pode ser adicionado a caminho algum.

Apenas um vizinho está definido como visitado Sendo V avaliado iterativamente e em ordem crescente, se o vértice vizinho pertencer a algum caminho simples, então o vértice corrente corresponde a uma região que pertence ao mesmo CML que este caminho representa. Logo, o vértice corrente é adicionado a este caminho simples (linhas 8-10 do Algoritmo 3).

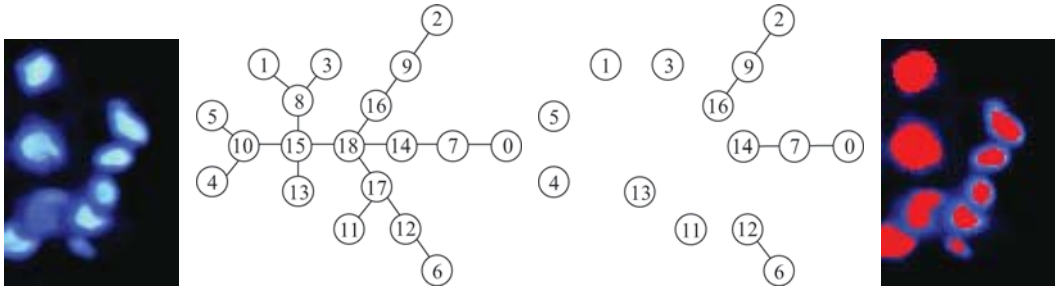
Ambos os vizinhos estão definidos como não visitados Quando ambos os vizinhos estão definidos como não visitados, significa que um novo CML foi encontrado. Logo, o vértice corrente torna-se um novo caminho simples (linhas 12-14 do Algoritmo 3). De modo geral, a iluminação da imagem não é uniforme e, como conseqüência, objetos embora idênticos podem aparecer com intensidades diferentes. Como resultado, os CML que representam estes objetos podem não conter pontos tão altos quantos os dos demais CML e assim, aparecem distante da extremidade do grafo.

As Figuras 2.11, 2.12 e 2.13(a) apresentam os resultados obtidos com algoritmo de mineração. Na Figura 2.11, as regiões em vermelho corresponde aos CML encontrados. A tabela apresentada junto com a Figura 2.12(d) mostra os caminhos simples encontrados no grafo exibido na Figura 2.12(e), o qual representa a imagem apresentada na Figura 2.12(a). Para a construção deste grafo foi utilizado intervalos de tamanho $\varepsilon = 32$ e os vértices em cor cinza representam os caminhos simples encontrados. A Figura 2.12(b) mostra as componentes conectadas representando cada

Algorithm 3

procedure EVALUATENODE2($\mathcal{S}$, v)
  Let v' and v'' be the neighbors of v
  if v' **and** v'' are set as visited **then**
    if $\exists\ s \in \mathcal{S} \mid v' \in s$ **and** $\forall\ s \in \mathcal{S}, v'' \notin s$ **or**
    $\exists\ s \in \mathcal{S} \mid v'' \in s$ **and** $\forall\ s \in \mathcal{S}, v' \notin s$ **or**
    $\exists\ s \in \mathcal{S} \mid v', v'' \in s$ **then**
      $s = s \cup \{v\}$
    end if
  else if v' is set as visited **and** $\exists\ s \in \mathcal{S} \mid v' \in s$ **or**
  v'' is set as visited **and** $\exists\ s \in \mathcal{S} \mid v'' \in s$ **then**
    $s = s \cup \{v\}$
  else if v' and v'' ware not visited **then**
    $s = \{v\}$
    $\mathcal{S} = \mathcal{S} \cup s$
  end if
end procedure

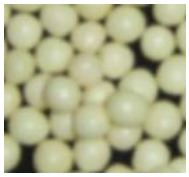
objeto detectado. Nesta figura, os números em cor azul correspondem ao menor índice dentre os vértices que compõe o caminho (subgrafo) que forma esta região. Por último, a Figura 2.12(c) apresenta os objetos detectados, os quais estão indicados por pequenos pontos azuis.



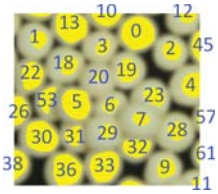
(a) (b) (c) (d)

Figura 2.11: Resultado obtido com o algoritmo de mineração para a imagem da Figura 2.3(a): (a) imagem de entrada; (b) representação da imagem baseada em grafo; (c) caminhos simples encontrados; e (d) resultado do algoritmo sobre a imagem.

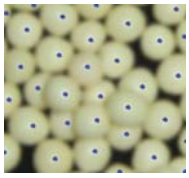
Para as imagens das Figuras 2.11 e 2.12 pode-se notar que o algoritmo apresentado provê resultados satisfatórios. De fato, este método funciona bem na maioria dos casos. No entanto, ele é propenso a erros quando muitos objetos contêm mais de um ponto brilhante. A Figura 2.13(a) ilustra esta situação e a Figura 2.13(b) apresenta o RAG correspondente com os caminhos simples encontrados identificados por linhas tracejadas. As setas em cor rosa indicam os pontos brilhantes e as regiões amarelas representam os caminhos simples, com os maiores índices dos vértices que o compõe em vermelho. Os pontos vermelhos indicam os objetos encontra-



(a)



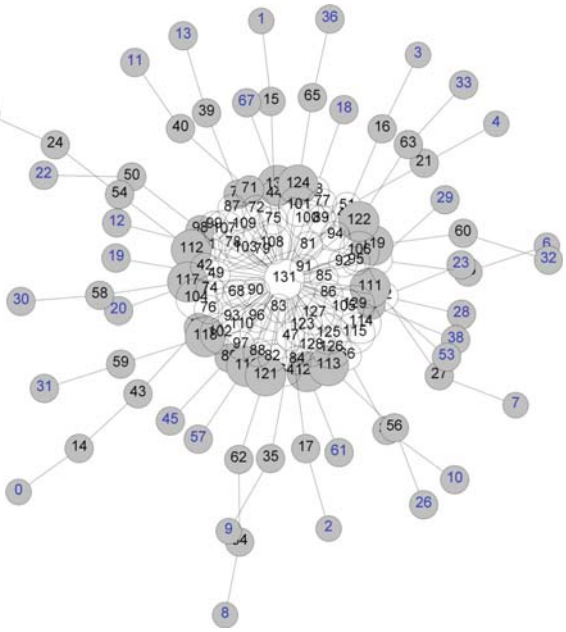
(b)



(c)

SIMPLE PATHS DETECTED	
0, 14, 43, 73	20
1, 15	22, 50, 98
2, 17	23
3, 16	26, 56, 113
4, 21	28
5, 24, 54, 112	29
6, 25	30, 58, 117
7, 27	31, 59, 118
8, 34, 62, 121	32, 60, 119
9, 35	33, 63, 122
10, 37	36, 65, 124
11, 40, 71	38, 69
12	45, 80
13, 39, 70	53, 111
18	57, 116
19	61, 120
	67, 130

(d)



(e)

Figura 2.12: Resultado do algoritmo de mineração de grafo: (a) imagem de entrada; (b) regiões conectadas; (c) objetos identificados; (d) caminhos simples encontrados durante a etapa de mineração do grafo; e (e) representação da imagem de entrada baseada em grafo.

dos. Nestas figuras, pode-se observar que embora os caminhos simples tenham sido segmentados corretamente, foram encontrados 14 objetos (células) ao invés de 5.

Os passos 1–5 do método proposto assumem que os pontos brilhantes adicionais presentes no objeto de interesse foram agrupados na etapa de partição do histograma. Logo, é assumido que os objetos possuem apenas um ponto brilhante e que sua luminância decai monotonicamente deste ponto em direção à fronteira do objeto. Entretanto, devido à presença de ruído, iluminação inadequada e até mesmo parâmetros intrínsecos dos objetos, nem sempre é possível agrupar estes pontos corretamente e o passo 6 se faz necessário quando a imagem apresenta esta situação.

2.6 Algoritmo para Clusterização do Grafo

Para reduzir o número de falsos positivos, ou seja, elementos que são incorretamente classificados como objetos, o passo 6 do método proposto executa uma clusterização hierárquica do grafo [65]. Nesta etapa, os caminhos simples que representam o mesmo objeto (CML) são agrupados em super conjuntos. Estes caminhos são identificados através da distância Euclidiana e topologia do grafo. Assume-se que se um conjunto de caminhos simples representa um único objeto, então tais caminhos devem ter um vizinho (vértice) em comum. A Figura 2.13(b) ilustra esta

situação. De acordo com a Figura 2.13(a) pode-se notar que os caminhos S_5 S_6 e S_7 representam o mesmo objeto e têm o vértice S_{15} como vizinho comum.

Para facilitar a descrição do algoritmo de clusterização as seguintes definições devem ser consideradas:

- a **distância** entre dois caminhos simples S_i e S_j é dada por $d_E(c_i, c_j)$ onde c_i e c_j são os centros das *bounding boxes* das regiões que são representadas pelos vértices com maiores índices, em S_i e S_j respectivamente.
- dois caminho simples S_i e S_j são ditos **vizinhos** se existir pelo menos uma aresta ligando um vértice de S_i a um vértice de S_j.
- a **vizinhança de um caminho simples** S_i é definida pelos vértices vizinhos aos vértices de S_i que não estejam em S_i.
- um **super conjunto** $\mathbb{S}$ é um conjunto de vértices que contém caminhos simples e vértices adicionais.
- de maneira análoga, **a vizinhança de um super conjunto** $\mathbb{S}$ é dada pelos vértices vizinhos aos vértices de $\mathbb{S}$ que não estão em $\mathbb{S}$ e a **distância** entre dois super conjuntos é dada pela distância euclidiana entre os centros das *bounding boxes* das regiões que são representadas pelos vértices com maiores índices.

Este algoritmo baseia-se no seguinte fato: dois caminhos simples S_i e S_j representam o mesmo objeto se e somente se $d_E(c_i, c_j) < \lambda$, onde λ é a metade do diâmetro médio dos objetos, medido a partir da imagem de entrada.

Dado um grafo G e uma lista $L = \{S_1, \ldots, S_k\}$ de caminhos simples encontrados durante o processo de mineração do grafo, para por em prática o passo 6 do método proposto e assim agrupar os caminhos simples que representam um mesmo objeto em super conjuntos, os cinco itens descritos a seguir devem ser executados.

1. Agrupar os caminhos simples de L, que possuam um vizinho em comum, em conjuntos de vértices.

2. Para cada conjunto, combinar os caminhos que representem o mesmo objeto ($d_E(c_i, c_j) < \lambda$) em super conjuntos. Se todos os caminhos simples de um dado conjunto pertencem ao mesmo super conjunto e o vértice comum não representar o fundo da imagem, então este vértice deve ser adicionado a este super conjunto. Os caminhos simples que, nesta etapa do processo, não foram adicionados a super conjunto algum se tornam um super conjunto.

3. Avaliar a vizinhança de cada super conjunto e adicionar os vértices que: (a) não pertençam a nenhum outro super conjunto; (b) não sejam vizinhos de outros super conjuntos e; (c) não represente o fundo da imagem.

4. Unir os super conjuntos que representam o mesmo objeto, isto é, super conjuntos vizinhos ou com um vizinho em comum, e que possuam $d_E(c_i, c_j) < \lambda$).

5. Repetir as etapas 3 e 4 até que nenhum super conjunto possa ser unido com outro super conjunto e nenhum vértice possa ser adicionado a algum super conjunto.

O resultado do passo de clusterização é uma coleção de super conjuntos no grafo G, cada um representando um objeto (CML) da imagem. Os resultados para cada etapa do algoritmo de clusterização são exemplicados como transições nas Figuras 2.13(c) – 2.14(e).

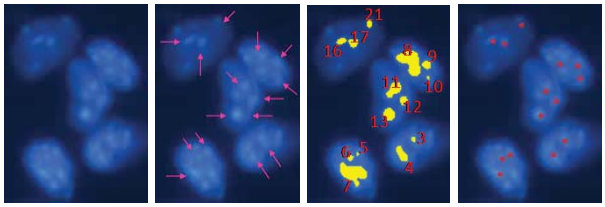
A Figura 2.13(b) ilustra a entrada do algoritmo de clusterização, ou seja, o grafo G e a lista L. Nesta figura, os caminhos simples (S_i) encontrados durante o processo de mineração estão identificados por linhas tracejadas.

O resultado da primeira etapa do algoritmo de clusterização é mostrado na Figura 2.13(c). Como resultado desta etapa foram obtidos cinco conjuntos de vértices (A, B, C, D e E), os quais estão representados por linhas tracejadas.

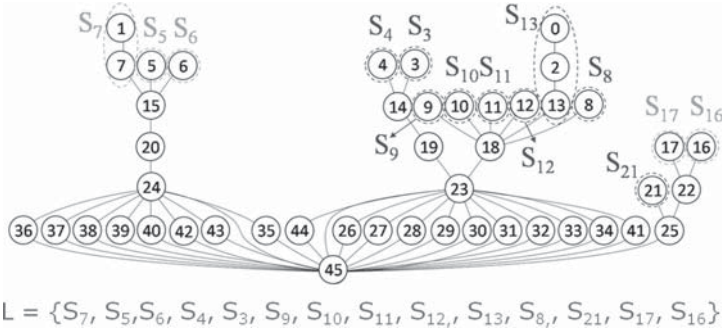
A Figura 2.13(d) apresenta o resultado parcial da segunda etapa. Nesta figura pode-se observar que o conjunto C foi transformado em dois super conjuntos, os quais são apresentados na cor vermelha e verde, respectivamente. Como a segunda etapa ainda não foi finalizada, os vizinhos em comum (indicados por setas coloridas) ainda não foram avaliados. O resultado final para esta etapa é apresentado na Figura 2.13(e). Note que o vértice 18 não foi adicionado. Nesta figura, os super conjuntos são representados por cores diferentes e a vizinhança de cada super conjunto é indicada pelas setas de mesma cor.

A Figura 2.14(a) mostra o resultado obtido após a etapa 3. Como os vértices 18 e 25 são vizinhos de super conjuntos distintos, eles não foram adicionados. Após o quarto passo, pode-se observar que os super conjuntos em rosa e azul foram unidos (Figura 2.14(b)), pois estão a uma distância um do outro inferior a λ.

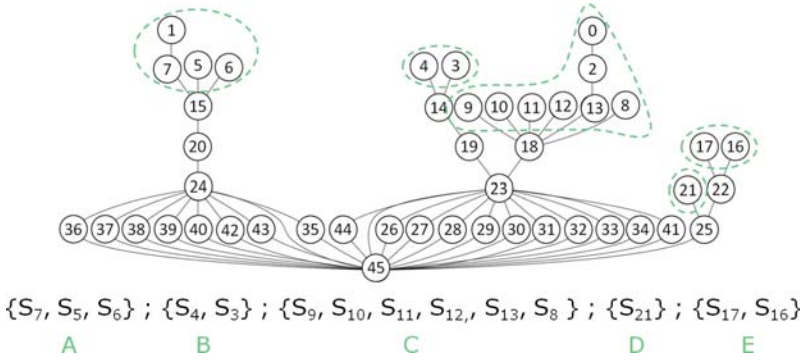
A Figura 2.14(c) mostra o resultado parcial da etapa 5. Como resultado de uma nova iteração da etapa 3, os vértices 23, 24 e 25 foram adicionados aos super conjuntos representados pelas cores laranja, azul e rosa, respectivamente. O resultado final da etapa 5 é apresentado na Figura 2.14(d), a qual exibe uma coleção de super conjuntos, cada um representado um objeto (neste caso uma célula) da imagem.



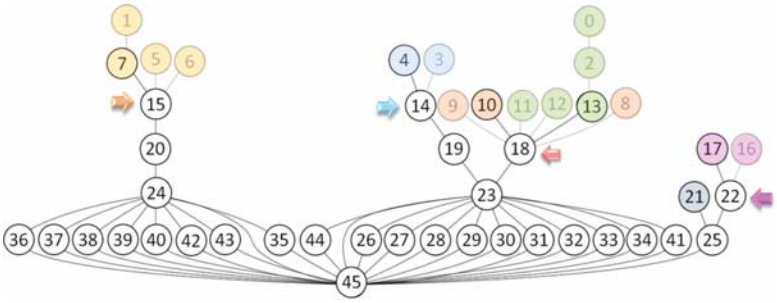
(a)Resultado do processo de mineração



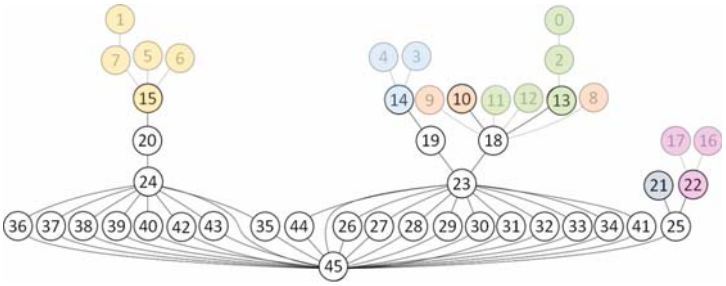
(b)Entrada do algoritmo de clusterização



(c)Resultado da etapa 1

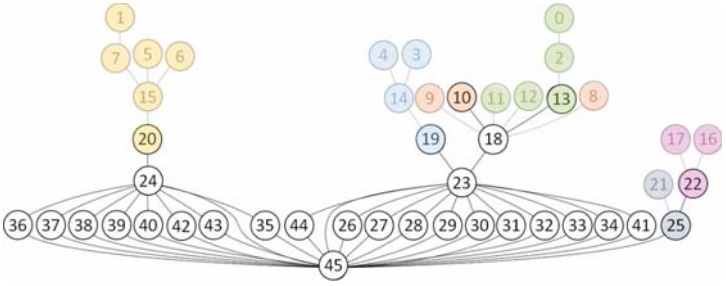


(d)Resultado parcial da etapa 2

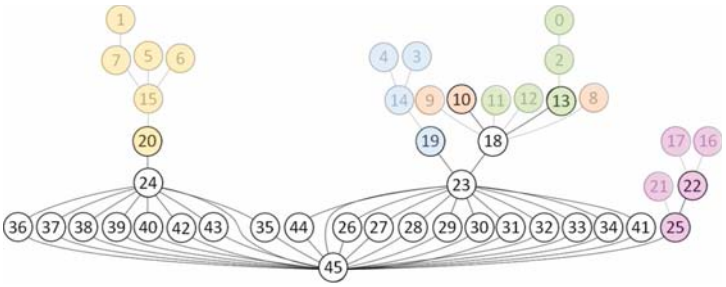


(e)Resultado final da etapa 2

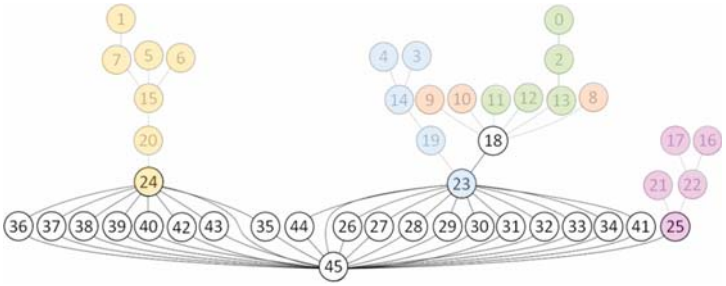
Figura 2.13: Resultado obtido com o algoritmo de mineração e resultados das etapas 1 e 2 do algoritmo de clusterização.



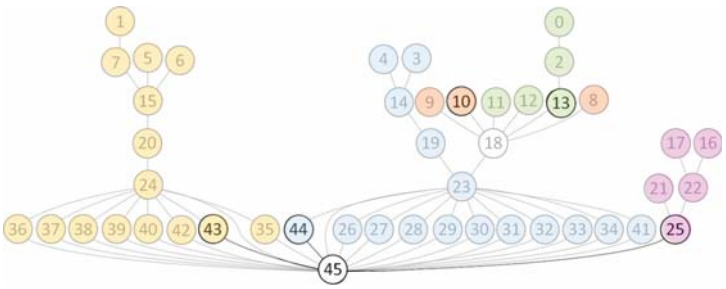
(a)Resultado da etapa 3



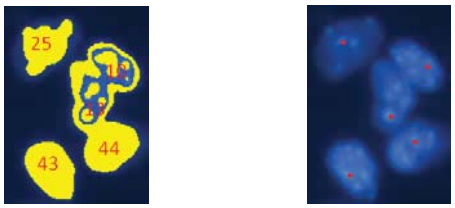
(b)Resultado da etapa 4



(c)Resultado parcial da etapa 5



(d)Resultado final da etapa 5



(e)Resultado do algoritmo de clusterização sobre a imagem de entrada

Figura 2.14: Resultados das etapas 3, 4 e 5 do algoritmo de clusterização.