FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO


# Geração Assistida de Diagramas Esquemáticos de Rede Eléctrica

**Ricardo José Fonseca de Oliveira Paulo**

Mestrado Integrado em Engenharia Informática e Computação

Orientador: Professor José Rui da Rocha Pinto Ferreira

Julho de 2010

# Resumo

É propósito deste estudo apresentar um conjunto de funcionalidades e respectiva especificação, que sirvam de suporte à construção de diagramas esquemáticos de rede eléctrica, tendo como base um modelo de conectividade da rede. O modelo de conectividade refere-se a uma base de dados que contém a especificação dos equipamentos da rede, indicando quais os equipamentos que estão conectados entre si e através de que terminais.

Nos diagramas esquemáticos não é – contrariamente a outros diagramas de rede eléctrica – importante a localização geográfica dos equipamentos, daí a sua denominação. De facto, os diagramas tratados são desenvolvidos de forma a complementar diagramas geográficos da rede, apresentando maior visibilidade em relação a estes.

A abordagem proposta passa por especificar um conjunto de ferramentas a introduzir num editor de diagramas esquemáticos de rede eléctrica que permita eliminar os processos repetitivos ao utilizador, reduzindo-lhe o tempo necessário para desenhar um diagrama e diminuindo os erros introduzidos durante a edição manual.

As ferramentas propostas para o editor passam sobretudo, por funcionalidades que permitam gerar automaticamente partes de diagramas esquemáticos. No entanto, são também sugeridas ferramentas de interacção simples, que facilitem a manutenção dos diagramas e consigam determinar incoerências destes relativamente ao modelo de conectividade. A possibilidade de gerar automaticamente partes de diagramas, dispondo os seus componentes no espaço, é contudo o foco principal do documento.

Pelo facto do tema se centrar essencialmente em geração automática de componentes de um diagrama, a aproximação feita passou sobretudo pela pesquisa na área da teoria dos grafos, principalmente desenho de grafos.

Após apresentar a revisão bibliográfica, o documento começa por apresentar um conjunto de ferramentas que se crê, seriam úteis para auxiliar um utilizador humano a desenvolver um diagrama esquemático e posteriormente apresenta uma possível abordagem para permitir gerar automaticamente partes de esquemáticos.

De entre os resultados obtidos destaca-se a criação de uma *framework* de testes, que prova a possibilidade de gerar automaticamente ramificações de diagramas esquemáticos segundo o princípio indicado previamente para o efeito.

# Abstract

The scope of this study is present a group of features and their specification to provide support to the construction of schematic diagrams of electrical networks based on a connectivity model. The connectivity model refers to a database that contains the specification of the network equipments, information about which equipments are interconnected and the terminal through each connection occurs.

In schematic diagrams it is not important - unlike other electrical network diagrams - the geographic location of the components, which explains its denomination. Indeed, the schematics are developed as a complement to geographical network diagrams offering better visibility compared with them.

The followed approach has provided the specification of a group of features to add to a schematic diagrams editor that should eliminate most of the repetitive processes to the user, reducing the current time required to produce a diagram.

Those features are mostly tools to allow the automatic generation of parts of diagrams. Despite, the proposed tools also include simple interaction features to make easier the maintenance of schematic diagrams finding inconsistencies between the diagrams and the connectivity model database. However, the possibility of automatically generate parts of diagrams, arranging the components in space is the main focus of the document.

The study of literature emphasis in graph theory (mostly in the area of graph drawing) due to the fact that the main target of this research is the automatic generation of parts of a diagram.

After presenting a literature review, the document begins with the presentation of some tools that could be useful to assist a human user to develop a schematic diagram. Then it introduces a possible method to allow the automatic generation of ramifications in schematic diagrams.

Among the final results, highlight to the creation of a framework to test the possibility of automatic generate ramifications of schematic diagrams using the followed principles.

# Agradecimentos

Dedico este espaço a todos que de alguma forma deram a sua contribuição para que esta dissertação fosse realizada, incluindo aqueles que, por equívoco meu, me olvide referir.

Começo por agradecer ao professor José Rui Ferreira e ao engenheiro Pedro Manuel Silva, respectivamente orientador da dissertação e responsável da instituição proponente do tema, a forma como orientaram o meu trabalho. A cordialidade e disponibilidade apresentada sempre que solicitados foram características dominantes de ambos. Expresso-lhes a minha gratidão pela liberdade de acção que me permitiram, demonstrando sempre confiança no trabalho desenvolvido, atitude que foi fundamental para manter os meus níveis de motivação elevados.

Agradeço também aos colaboradores da EFACEC Engenharia, S.A. que me acompanharam de perto ao longo dos meses em que decorreu o estudo e que embora não tenham contribuído directamente para o trabalho realizado, tiveram papel preponderante na integração ao ambiente de trabalho e à instituição, ajudando-me a manter níveis de satisfação elevados.

Gostaria ainda de agradecer aos meus colegas de curso e companheiros de Erasmus, particularmente aqueles pertencentes a um grupo cuja denominação não deve ser citada neste documento, mas o qual os seus destinatários facilmente identificarão. Não tendo sido a sua presença regular durante o decorrer da dissertação, foi uma constante durante o curso que culminou com esta, tendo em muito contribuído para o meu desenvolvimento não apenas pessoal, mas também académico. Destes, perdoem-me os restantes, gostaria de distinguir os meus amigos Francisco Richardson, Luís Carneiro e Rui Costa por terem marcado fases diferentes do meu percurso e pela confiança que neles deposito.

Por último mas não menos importante, quero agradecer aos meus familiares que me apoiaram ao longo dos últimos anos. De facto, a atenção e tempo que lhes dispensei durante estes anos não foi, por diversos motivos, a devida, embora tema que tal possa vir a agravar-se. Particularmente ao meu irmão e ao meu pai, por terem sido os últimos leitores não oficiais deste documento, o meu muito obrigado.

Ricardo Paulo

# Conteúdo

# Lista de Figuras

# Lista de Tabelas

# Lista de Expressões

# Abreviaturas e Símbolos

| | |
|---|---|
| **SCADA** | Supervision Control And Data Acquisition Station<br>(Sistemas de Supervisão e Aquisição de Dados) |
| **RTU** | Remote Terminal Unit<br>(Unidade Terminal Remota) |
| **PLC** | Programmable Logic Controller<br>(Controlador Lógico Programável) |
| **MTU** | Master Terminal Unit<br>(Unidade Terminal Principal) |
| **DMS** | Distribution Management System<br>(Sistemas de Gestão à Distribuição) |
| **GIF** | Graphics Interchange Format<br>(Formato para Intercâmbio de Gráficos) |

# 1.Introdução

Tendo sido realizada em ambiente empresarial, esta dissertação tem interesse prático para a instituição proponente, apresentando por isso um enquadramento específico no seio da organização. Por essa razão, as motivações e objectivos exigem soluções de aplicação prática quase imediata, não sendo este estudo de interesse meramente académico. O trabalho envolve, contudo, investigação profunda apoiada na literatura existente e centra-se essencialmente em apresentar possíveis soluções para os objectivos propostos e não resultados finais.

Neste capítulo apresenta-se primeiramente o contexto e enquadramento desta dissertação, onde é apresentado o interesse do tema para a instituição que o propôs. De seguida, expõem-se as motivações e objectivos que levaram à sua realização. No final do capítulo é ainda definida a estrutura seguida pelo documento.

## 1.1 Contexto/Enquadramento

A EFACEC Engenharia, S.A. é uma empresa Portuguesa que actua na área da engenharia, particularmente nas áreas de energia eléctrica e transportes. [1]

Um dos produtos desenvolvidos pela empresa para o sector energético, o projecto *Scatex*, inclui um conjunto de aplicações de monitorização e aquisição de dados de redes de distribuição eléctrica.

De entre os módulos desenvolvidos para o *Scatex*, incluem-se um visualizador de diagramas de rede eléctrica e um editor de diagramas esquemáticos. O visualizador e o editor de diagramas são módulos diferentes do mesmo produto.

Seguidamente, apresentam-se duas figuras de partes de diagramas retiradas do visualizador. A primeira figura apresenta um diagrama geográfico de uma rede eléctrica, a segunda, um esquemático.

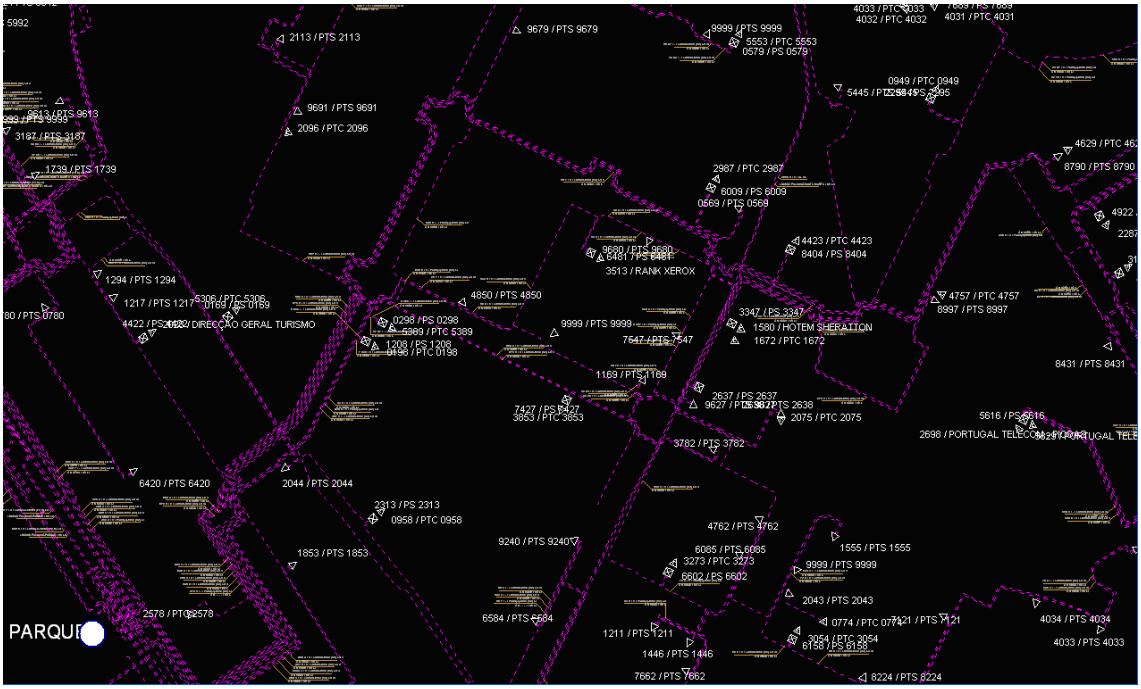


**Figura 1 - Diagrama geográfico de rede eléctrica**

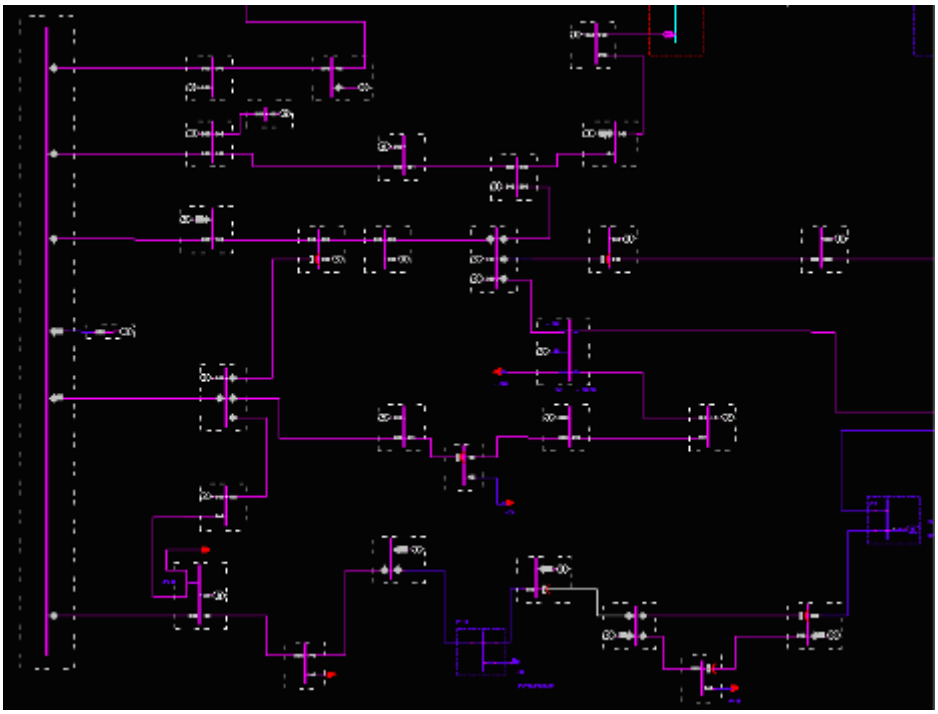

**Figura 2 - Diagrama esquemático de rede eléctrica**

A principal diferença entre os dois tipos de diagrama, é que enquanto que os geográficos têm os equipamentos dispostos consoante a sua localização real, os diagramas esquemáticos privilegiam a visualização dos componentes em detrimento da sua localização.

Crendo na importância dos diagramas esquemáticos para a monitorização dos componentes de uma rede, a empresa desenvolveu um editor para permitir a criação deste tipo de diagramas. No editor, o desenho dos diagramas é feito pelo utilizador, introduzindo símbolos e associando-os manualmente a equipamentos reais especificados numa base de dados. Essa base de dados baseia-

se num modelo de conectividade, guardando não apenas a especificação de cada componente da rede, mas também as ligações entre componentes e sobre que terminais ocorrem.

O tema da dissertação passa por especificar ferramentas que permitam reduzir as tarefas repetitivas ao utilizador no processo de criação de diagramas esquemáticos no editor, tornando a sua tarefa mais simples, rápida, iterativa e fiável.

## 1.2 Motivação e Objectivos

Cientes de que o editor de diagramas esquemáticos desenvolvido carecia de ferramentas de suporte ao utilizador no desenho de diagramas esquemáticos e que parte do processo deveria ser automatizado, os responsáveis da instituição proprietária da aplicação, colocaram como objectivos da dissertação definir e especificar um conjunto de ferramentas de suporte ao utilizador na construção e manutenção de diagramas esquemáticos.

Foi definido que as ferramentas a desenvolver deveriam permitir essencialmente o desenho automático de partes de um diagrama esquemático através de uma leitura à base de dados. Não obstante, foi também colocado como objectivo, a sugestão de algumas ferramentas que permitissem evitar erros na construção dos diagramas, bem como detectar informação incoerente nestes, relativamente à informação contida na base de dados dos equipamentos.

Apesar de ter sido definido como objectivo principal a geração automática de partes do diagrama, nunca foi dada como realista a possibilidade de gerar automaticamente a totalidade de um diagrama esquemático sem intervenção do utilizador. De facto, dado as grandes dimensões dos diagramas, na ordem das dezenas de milhar de componentes, uma geração automática integral seria extremamente difícil de obter, especialmente por haverem vários critérios visuais na sua construção muito difíceis de sistematizar. O objectivo colocado foi permitir gerar apenas partes do diagrama da melhor forma possível, dando seguidamente a possibilidade ao utilizador de fazer ajustes às partes geradas. Sempre se acreditou que as decisões mais difíceis deveriam continuar, por isso, a ser tomadas por um utilizador humano.

A motivação máxima da dissertação é tornar o produto *Scatex*, particularmente o módulo de edição de diagramas esquemáticos, mais competitivo e enquadrado com a tecnologia e exigências do mercado actual. De facto, era de crer pelos responsáveis pelo desenvolvimento do editor que o processo de desenho de diagramas deveria ser tornado mais iterativo, poupando trabalho ao utilizador, aumentando-lhe a produtividade e ao mesmo tempo tornando os diagramas mais fiáveis, reduzindo a probabilidade de erro humano.

## 1.3 Estrutura da Dissertação

De forma a cumprir os objectivos propostos, a dissertação está dividida em cinco capítulos para além da introdução.

Assim, primeiramente será apresentada a revisão bibliográfica realizada, onde será descrito o estado da arte do tema em estudo e estará centrada sobretudo na área de desenho de grafos. Seguidamente, será apresentado um capítulo que incorpora um conjunto de ferramentas e respectivas interfaces que se julga que seriam uma mais valia no editor de diagramas esquemáticos. Terminada a exposição das ferramentas propostas, o estudo passa a centrar-se em facultar um possível método que permita gerar automaticamente partes de diagramas esquemáticos. Finalmente, serão apresentados os resultados de *layout* obtidos para os métodos de geração automática propostos e as conclusões finais onde é avaliado o nível de satisfação com os resultados obtidos e indicadas possíveis melhorias futuras.

# 2.Revisão Bibliográfica

Este capítulo destina-se a descrever a revisão do estado da arte na área em estudo, sendo apresentada a literatura existente que poderá ser relevante para o problema proposto, na medida em que evitará trabalho sobre matérias já desenvolvidas e oferecerá uma maior credibilidade ao estudo.

O capítulo apresenta também uma revisão tecnológica às principais ferramentas que utilizam os princípios que se pretendem aplicar, o que poderá trazer uma ideia melhor da sustentabilidade dos estudos teóricos.

## 2.1 Introdução

Uma vez que o tema e objectivos da dissertação têm uma amplitude vasta, do estado da arte constam vários subcapítulos centrados em áreas de conhecimento diferentes.

O sistema *Scatex*, produto do qual o editor de diagramas esquemáticos de rede eléctrica é um módulo, trata-se de um Sistema de Supervisão e Aquisição de Dados (SCADA). Por essa razão, é importante conhecer melhor do que se trata um sistema deste tipo, assim como identificar o estado actual de desenvolvimento destes sistemas. Atendendo a esse facto, a parte inicial deste capítulo irá centrar-se num pequeno estudo sobre este tipo de sistema.

A segunda parte da revisão bibliográfica estará centrada em estudos sobre teoria de grafos, particularmente desenho de grafos. Esta será eventualmente a parte mais importante deste estado da arte, já que o principal objectivo do trabalho é o desenho automático de partes de um diagrama, área abrangida pela teoria dos grafos. Começará por se apresentar os princípios e terminologia usados na área da teoria dos grafos antes de se partir para o tema mais específico de desenho de grafos.

Finalmente, será estudada a aplicabilidade da literatura sobre desenho de grafos em trabalhos práticos, preferencialmente relacionados com o desenho de componentes eléctricos, ainda que dado a sua especificidade, não tenha sido possível analisar a sua aplicação directa no desenho de diagramas esquemáticos de rede eléctrica.

## 2.2 Sistemas SCADA

Os Sistemas de Supervisão e Aquisição de Dados, comummente designados SCADA são sistemas que utilizam software para a monitorização e supervisão de equipamentos em sistemas industriais. [2]

Nos seus primórdios, os SCADA permitiam informar periodicamente o estado de um determinado processo industrial, monitorizando sinais representativos de medidas e estados de dispositivos, utilizando apenas um painel de lâmpadas e indicadores, sem qualquer interface aplicacional com o operador. [3]

Actualmente, os SCADA utilizam tecnologias de computação e comunicação para automatizar a monitorização e controlo de processos industriais, efectuando recolha de dados em ambientes complexos, eventualmente dispersos geograficamente, e a respectiva apresentação ao utilizador com recurso a interfaces gráficos amigáveis. [3]

Os SCADA melhoram a eficiência do processo de monitorização e controlo, disponibilizando em tempo útil o estado actual do sistema, através de um conjunto de previsões, gráficos e relatórios, de modo a permitir a tomada de decisões operacionais apropriadas, quer automaticamente, quer por iniciativa do operador. [3]

Estes sistemas são constituídos por três elementos chave. A unidade terminal remota (RTU) ou, por vezes, controladores lógicos programáveis (PLCs), o centro de controlo (MTU) e o sistema de comunicação. A RTU, estando ligada ao equipamento físico, monitoriza-o e recolhe métricas como a pressão, o fluxo, a tensão ou a corrente. O MTU por seu lado reúne, armazena e apresenta informação ao utilizador, processando dados da RTU. Finalmente, o sistema de comunicação reúne uma panóplia de equipamento de transporte de dados (cabos de cobre, fibra ou mesmo *wireless*) bem como equipamentos responsáveis pelo controlo do envio/recepção desses dados, tais como modems, transmissores, receptores ou rádios. [2]

Os SCADA actuais incluem ainda um conjunto de aplicações de software personalizáveis, que permitem não apenas a apresentação de dados ao utilizador mas também um conjunto de outras funcionalidades. Estas incluem monitorização e processamento de eventos, funções de controlo, obtenção e análise de dados, assim como relatórios e cálculos complementares. [4]

A aplicação dos SCADA é múltipla, pelo que estes sistemas revelam-se de crucial importância na estrutura de gestão das empresas que os detêm, tendo deixado de ser vistos como meras ferramentas operacionais ou de engenharia, para passarem a ser considerados como uma importante fonte de informação. Estão inseridos num ambiente industrial cada vez mais complexo e competitivo, onde os factores relacionados com a disponibilidade e segurança da informação assumem elevada relevância, tornando-se necessário garantir que a informação está disponível e segura, quando necessária, independentemente da localização geográfica. Actualmente, estes sistemas estão já presentes em áreas como a indústria de celulose, a indústria petrolífera, a indústria têxtil, a indústria metalúrgica, a indústria automóvel, a indústria electrónica, entre outras. [3]

Até tomarem a forma actual, estes sistemas percorreram um longo caminho. No entanto, apesar do nível de evolução actual, os SCADA continuam a apresentar uma boa margem de desenvolvimento, surgindo como uma boa oportunidade para futuros investimentos. [2]

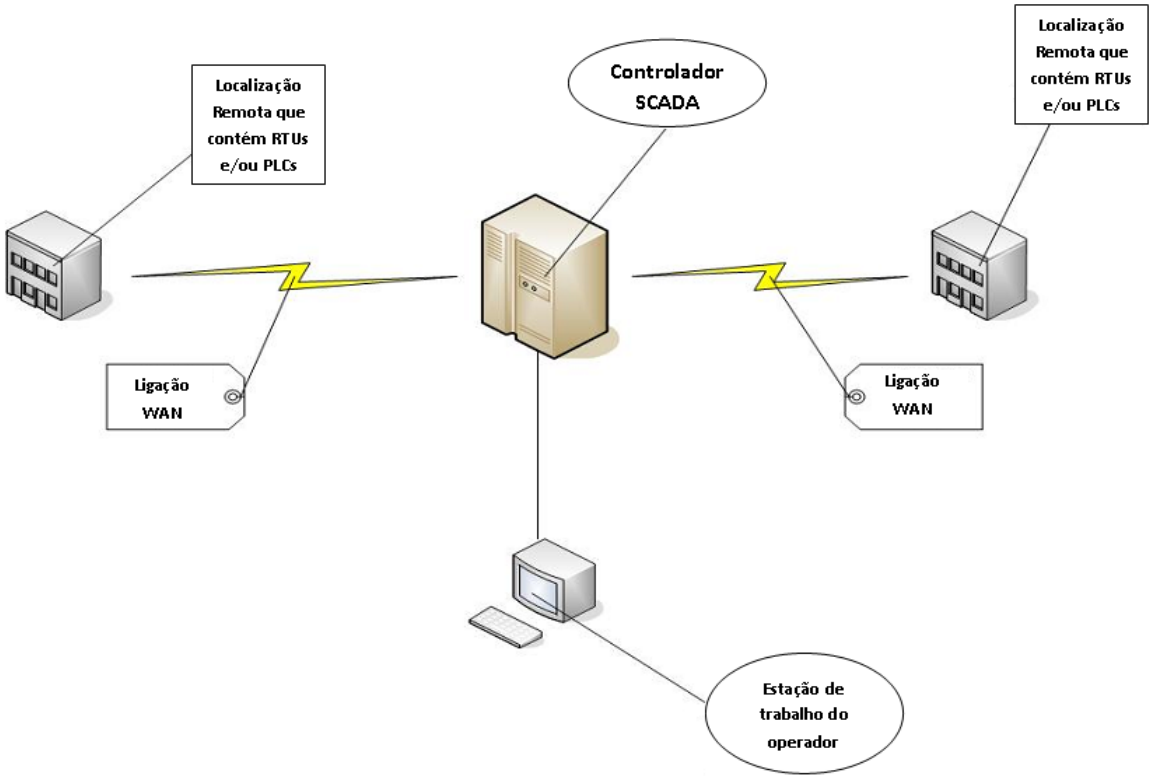


**Figura 3 - Funcionamento de sistema SCADA típico [5]**

### 2.2.1 SCADA em Redes de Distribuição Eléctrica

Uma das múltiplas aplicações dos SCADA é a monitorização de equipamentos de redes de distribuição de energia eléctrica. [3]

Os sistemas de gestão de redes de distribuição (DMS), são um conjunto de aplicações de software que suportam operações em redes de distribuição de energia eléctrica, nomeadamente supervisão e aquisição de dados dos seus equipamentos constituintes.

Os SCADA monitorizam equipamentos da rede, possibilitando também o seu controlo. Não têm, no entanto, noção da interligação existente entre equipamentos. Os SCADA/DMS são vocacionados para gestão de redes de distribuição eléctricas, suportando todas as funcionalidades dos SCADA e dispondo, para além disso, de informação sobre o modelo de conectividade da rede. Essa informação permite disponibilizar uma série de ferramentas que ajudam na supervisão de uma rede de distribuição eléctrica. [25]

As aplicações incluídas nos SCADA/DMS fornecem recursos avançados de apoio a decisões, que conduzem a uma operação segura e fiável de uma rede de distribuição de energia eléctrica. Estas oferecem um ambiente amigável e funcionalidades necessárias para melhorar a gestão de redes de distribuição, o que permite melhorar o planeamento e gestão dos recursos da rede.

O projecto *Scatex*, onde se inclui o editor de diagramas esquemáticos, é um SCADA/DMS, apresentando as características dos sistemas mais actuais. Nesse contexto, inserem-se os diagramas representativos da rede, como forma de facilitar a monitorização e supervisão dos componentes de uma rede de despacho eléctrica.

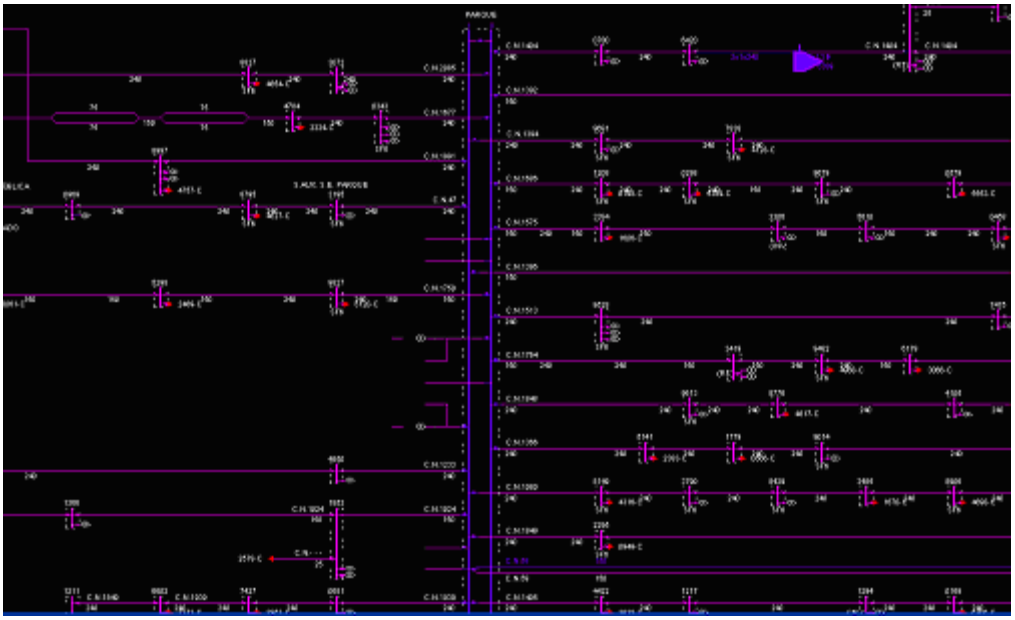

**Figura 4 - Diagrama esquemático de rede eléctrica que facilita a monitorização e supervisão dos componentes da rede**

## 2.3 Teoria dos Grafos

Em 1736, o matemático suíço **Leonhard Euler** apresentou uma teoria geral, que incluiu uma resposta para uma questão conhecida como o problema da **ponte de Königsberg**. Tal problema acendeu um estudo aprofundado até hoje, conhecido como **teoria dos grafos**. A figura 5 apresenta o problema que Leonhard Euler se propôs a resolver. [6]

O problema colocado consistia, em partindo de qualquer ponto do mapa, atingir todos os locais assinalados (A, B, C e D), passando em cada ponte assinalada exactamente uma vez. Para simplificar o problema, este foi modelado num **grafo** de acordo com o apresentado na figura 6. Euler acabou por provar que não era possível traçar um caminho seguindo tais restrições.

Desde então, investigadores de todo o mundo têm efectuado várias pesquisas ao tema, enquanto engenheiros as têm vindo a aplicar nas mais diversas áreas, tendo-se revelado de grande utilidade no campo da engenharia eléctrica e electrónica. [6]

Estes estudos sobre teoria dos grafos podem ser aplicados na solução de vários problemas, tais como, determinar se um dado circuito pode ser implementado numa placa de circuito integrado sem o uso de pontes ou túneis (que surgem na impossibilidade de se desenhar um circuito sem sobreposição de ligações). [6]


**Figura 5 - Problema da ponte de Königsberg retirado do Google Earth a 8 de Março de 2007**

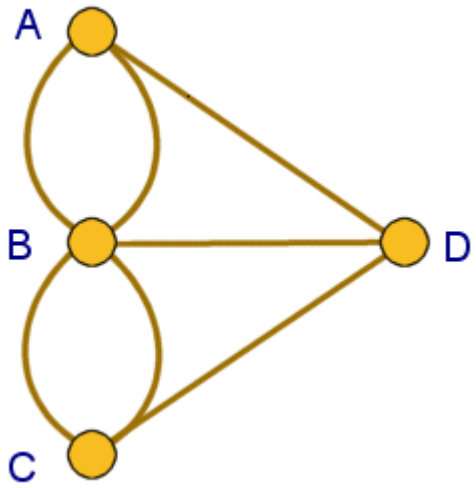


**Figura 6 - Esquema do problema da ponte de Königsberg [6]**

### 2.3.1 Terminologia usada na teoria dos grafos

Em termos básicos, um **grafo** é a representação de um sistema através de um esquema que utiliza dois elementos básicos: **nós** (por vezes também designados **vértices**) e **arestas**.

Esquematicamente, um **nó** é geralmente representado por um círculo embora possa tomar outras formas. [6]

Uma **aresta** é uma linha que **liga dois nós**, podendo ainda conter um texto correspondente, designado *label* (etiqueta) tal como se apresenta na figura 10. [6]

Um grafo diz-se **dirigido** quando as suas arestas têm um sentido, ou **não dirigido** caso contrário. Na figura 7 apresenta-se um grafo dirigido, analogamente nas figuras 6, 8, 9 e 10 encontram-se exemplos de grafos não dirigidos. [6]

Um grafo diz-se **simples** se for **não dirigido** e entre cada par de vértices distintos existir no máximo uma aresta e não contiver **laços.** Um **laço** é uma aresta que liga um vértice a si mesmo, como demonstrado na figura 8. Todo o grafo que não seja simples, diz-se **multigrafo**. [7]

Dois vértices dizem-se **adjacentes** se tiverem uma aresta que os ligue e uma aresta diz-se **incidente** a um **vértice** se este fizer parte do par que a define. [6]

Num grafo, duas ou mais arestas são **paralelas** se estão associadas ao mesmo par de vértices. Por exemplo, as duas arestas que ligam os vértices *A* e *B* na figura 6 são paralelas. Pelo que já foi dito, pode por isso dizer-se que **um grafo simples não contém arestas paralelas**. [6] O grafo citado apresenta arestas paralelas logo trata-se de um **multigrafo**. [7]

Um vértice que não é **incidente** a nenhuma aresta, ou seja, não faz parte de nenhum dos vértices que compõem os pares das arestas de um grafo, diz-se **isolado**. O grafo da figura 8 apresenta um vértice isolado. [6]

Um grafo diz-se **completo** se é simples e existe uma aresta entre cada par de vértices distintos, tal como o grafo da figura 9. [6]

Atribui-se a denominação de **grafo planar** a qualquer grafo que possa ser desenhado no plano sem que as suas arestas se cruzem, como é exemplo o representado na figura 10. [6]

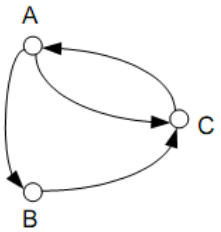


**Figura 7 - Exemplo de grafo dirigido [6]**

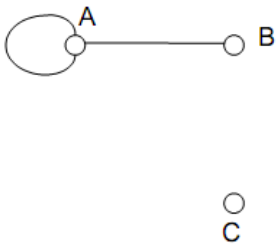


**Figura 8 - Grafo não dirigido com um vértice isolado e um laço [6]**

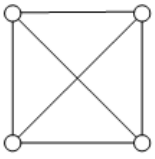

**Figura 9 - Grafo completo [6]**

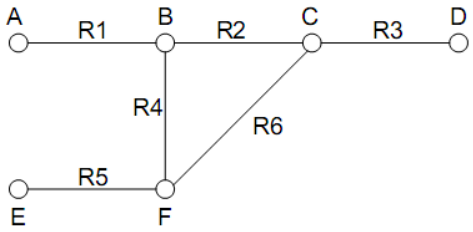

**Figura 10 - Grafo com *labels* (etiquetas) nas arestas [6]**

### 2.3.2 Representação de grafos

Alternativamente à versão esquemática (gráfica) já apresentada, os grafos podem também ser representados **algebricamente**, através de **expressões matemáticas** ou **matrizes**.

Matematicamente, um **grafo sem arestas paralelas** pode ser representado por um par de conjuntos (V, A), em que ***V*** é o conjunto de vértices do grafo e ***A*** o conjunto das suas arestas. Uma aresta ***A*** é definida pelo par dos dois vértices que conecta, por exemplo a = (v1, v2.) [6]

No caso de o grafo ser dirigido, então, os dois vértices do conjunto devem estar ordenados de tal forma, que a aresta definida tem o sentido do primeiro vértice do conjunto para o segundo. Se o grafo não for dirigido, torna-se indiferente a ordem dos vértices. [6]

Tomando como exemplo o grafo dirigido *G* apresentado na figura 7, a sua definição matemática através dos seus conjuntos de vértices e arestas seria:

$$V(G) = \{A, B, C\}$$
$$A(G) = \{(A, B), (A, C), (B, C), (C, A)\}$$

**Expressão 1 - Definição de um grafo através dos seus conjuntos de vértices e arestas**

- Em que V(G) é o conjunto de nós do grafo e A(G) o conjunto das suas arestas.

Um **grafo completo** pode ser definido pela expressão K*n*, em que *n* determina o número de vértices do grafo. O grafo completo da figura 9 pode ser assim definido pela expressão *k4*. [6]

Um grafo pode também ser representado através do sistema matricial. A **matriz de adjacência** de um grafo *G* é a matriz A = [*αij*], de ordem n × n, em que n é o número de arestas que ligam o vértice *vi* ao vértice *vj*. [7]

Assim, a **matriz de adjacência** *A* do grafo dirigido apresentado na figura 7 seria:

$$\begin{matrix} \mathbf{0} & \mathbf{1} & \mathbf{1} \\ \mathbf{0} & \mathbf{0} & \mathbf{1} \\ \mathbf{1} & \mathbf{0} & \mathbf{0} \end{matrix}$$

Ambas as representações apresentadas poderão vir a ser utilizadas no decorrer deste documento.

### 2.3.3 Desenho de grafos

O desenho de grafos, vulgarmente designado ***graph layout***, é um ramo da teoria dos grafos que aplica conceitos de **topologia** e **geometria** para produzir representações gráficas bidimensionais de grafos. [8]

Já foram apresentados vários exemplos de representação gráfica de grafos, através de diagramas nas figuras 6 a 10.

Existem várias formas de representar graficamente o mesmo grafo sendo que umas podem tornar-se mais funcionais e/ou mais legíveis que outras, dependendo do fim a que se destinam. Essas formas variam principalmente na disposição (*layout*) dos seus vários nós e arestas. É por isso importante estudar as várias formas de dispor os componentes de um grafo, de forma a potenciar a sua utilização.

O estudo sobre o desenho de grafos aplica-se a várias áreas do conhecimento. No caso da engenharia electrónica pode, por exemplo, permitir descobrir qual a melhor forma de desenhar um circuito integrado numa placa de forma a evitar ao máximo o uso de túneis ou pontes.

Estudos realizados na área têm-se revelado igualmente importantes no desenho de diagramas relacionais em engenharia de *software* (permitindo representações que facilitam a visualização ao utilizador), desenho de tabelas de bases de dados, gestão de projectos, gestão de informação, telecomunicações entre outras. [8] Neste caso concreto, pretende-se potenciar os estudos sobre desenho de grafos para melhorar o nível de visualização de diagramas esquemáticos de redes de distribuição eléctricas.

Nos próximos capítulos, serão apresentadas algumas das principais estratégias de disposição utilizadas para desenhar grafos. Existindo várias formas de *layout* possíveis, serão exploradas algumas das mais genéricas. Antes disso, no entanto, é importante conhecer-se as convenções geométricas utilizadas na representação dos vértices e arestas, que serão faladas primeiramente.

#### 2.3.3.1 Convenções geométricas utilizadas na representação geométrica de grafos

Na representação geométrica de grafos, existem alguns tipos de convenção utilizados, particularmente no que diz respeito aos formatos tomados pelas arestas nas ligações aos vértices e pela disposição destes. [8]

Neste capítulo, serão estudadas quatro das principais formas que uma representação geométrica de grafos pode tomar. Estas serão utilizadas nas estratégias de *layout* apresentadas nos capítulos seguintes.

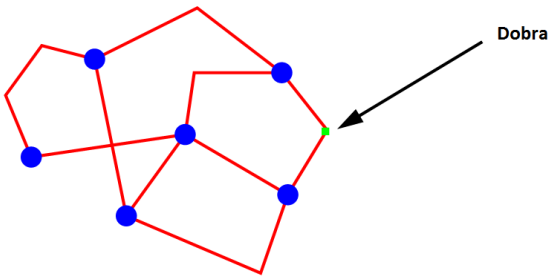


**Figura 11 - Representação poligonal de um grafo [8]**

Uma das hipóteses é o desenho poligonal tal como na representação da figura 11. A característica principal deste tipo de desenho é a utilização de arestas com "dobras" para ligar os vértices. Estas, não são mais que ângulos entre os vértices que permitem mudar o seu sentido.

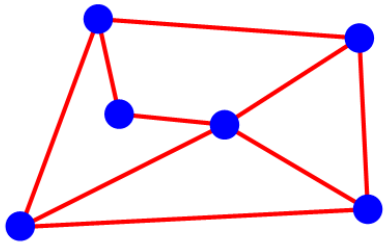

**Figura 12 - Desenho de grafo usando apenas arestas rectas [8]**

Outra das estratégias que se pode considerar, é o desenho utilizando apenas linhas rectas para representar as arestas tal como apresentado na figura 12. Neste caso, contrariamente ao que acontecia no desenho poligonal, as arestas são sempre linhas rectas, não tendo ângulos que mudem a sua direcção.

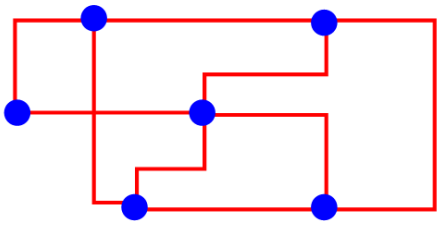

**Figura 13 - Desenho ortogonal de grafo [8]**

Também pode ser utilizado desenho ortogonal para representar os grafos, como é mostrado na figura 13. Neste caso, tal como no desenho poligonal, existem dobras nas arestas, no entanto estas apresentam sempre **ângulos de 90º**. [8]

Geralmente, neste tipo de desenho deve tentar-se colocar o máximo número de vértices sobre as mesmas linhas verticais ou horizontais, de forma a minimizar o número de cruzamentos e dobras entre arestas. A colocação dos vértices em grelha (*grid drawing*) é o procedimento mais comum como será mostrado posteriormente. [10]

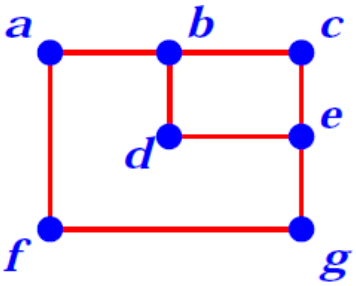


**Figura 14 - Desenho ortogonal de grafo utilizando apenas linhas rectas [8]**

O desenho ortogonal de um grafo utilizando apenas linhas rectas, tal como se vê na figura 14, é um tipo de desenho ortogonal em que são utilizadas apenas linhas rectas para representar as arestas do grafo, devendo os seus vértices ser colocados de forma a possibilitar o desenho sem que haja necessidade de utilizar arestas com dobras.

A utilização de cada um dos tipos de desenho dependerá do fim que se pretende e do tipo de *layout* a considerar. Nos capítulos seguintes, onde se apresentam algumas possibilidades para dispor automaticamente os componentes de um grafo, serão utilizados os princípios de alguns dos tipos de desenho apresentados.

### 2.3.3.2 Algoritmos de desenho de grafos

Seguindo as convenções de desenho já apresentadas, a literatura sobre desenho de grafos apresenta um conjunto de metodologias que permitem o desenho de grafos mediante determinados critérios estéticos.

Os princípios a aplicar para desenhar um grafo dependem do tipo de visualização que se pretende, bem como das características do grafo. Por essa razão, os vários algoritmos de *layout* de grafos apresentam princípios de desenho diferentes, levando a representações alternativas de um mesmo grafo.

Geralmente, o princípio fundamental que deve levar à escolha de um determinado algoritmo de disposição, é garantir que este oferece a melhor visualização possível do problema que irá representar.

Alguns dos critérios mais seguidos pelos algoritmos de *layout*, sendo que cada algoritmo privilegia uns em detrimento de outros, são os seguintes [8]:

- Apresentação de **simetria**, se existente
- **Planaridade** (desenho de grafo sem cruzamento de arestas)
- **Linearidade** (evitar arestas poligonais ou mesmo ortogonais)
- **Uniformidade** entre nós (procurar manter distâncias uniformes entre vértices)
- **Uniformidade de arestas** (manter arestas com comprimentos semelhantes)
- **Manter sentido** (para grafos dirigidos é conveniente que as arestas apontem todas na mesma direcção)

Geralmente, os algoritmos de desenho de grafos tentam conjugar estes critérios, embora dificilmente seja possível mantê-los todos, principalmente para casos de grafos maiores e mais complexos.

Na escolha de um algoritmo de desenho para representar um problema específico, deve ter-se em conta, quais dos critérios apresentados esse algoritmo se propõe tratar e quais os que realmente são importantes para a visualização do grafo em causa. Mediante isso, deve escolher-se o algoritmo mais apropriado.

Outro factor que deve ser tido em conta, é o tempo de processamento necessário para aplicar alguns algoritmos, principalmente se o grafo que se pretende desenhar for de grandes dimensões, já que alguns algoritmos têm tempos de processamento elevados.

De seguida, apresentam-se três algoritmos de *layout* que se propõem garantir alguns dos critérios expostos acima.

### Layout através de algoritmos *force based*

O *layout* através de algoritmos baseados em forças é uma das estratégias possíveis para desenhar um grafo no plano. [9]


**Figura 15 - Comportamento repulsivo de duas partículas de carga eléctrica ligadas através de uma mola [9]**

Uma das variantes deste tipo de algoritmo baseia-se no comportamento das partículas de carga eléctrica quando ligadas por meio de uma mola, como mostrado na figura 15. Supondo que as duas partículas da imagem são de sinal idêntico, a força efectuada pela mola no sentido de

juntar as partículas é contrabalançada pelas suas forças repulsivas. De facto, se não existisse mola, as partículas tenderiam a distanciar-se. Partindo deste princípio, a ideia é simular o comportamento dos nós de um grafo no das partículas, actuando as arestas do grafo como "molas". [9]

Desta forma, nós adjacentes tendem, por acção das "molas", a permanecerem mais próximos. O objectivo é fazer evoluir o sistema para um estado em que as forças se anulam, ou seja, os nós ficarem dispostos de tal forma que se actuassem como um sistema de partículas eléctricas estas permaneceriam imóveis, em equilíbrio. [9]

A força aplicada em cada vértice seria dada através da seguinte expressão [9]:

$$F(v) = \sum_{(u,v)\in E} \left[ \vec{F}_{atractiva} + \vec{F}_{repulsiva} \right]$$

**Expressão 2 - Força aplicada a cada vértice num algoritmo de disposição *force based* [9]**

Um exemplo da aplicação deste algoritmo encontra-se na figura 16, onde é possível ver a mudança causada após a aplicação do algoritmo num grafo com os vértices dispostos aleatoriamente.

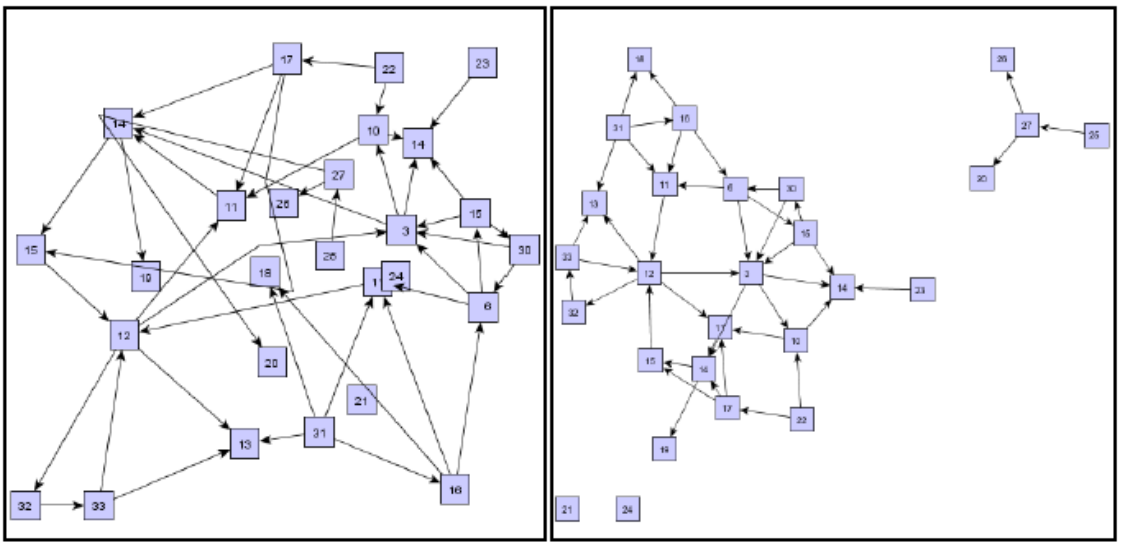

**Figura 16 - Aplicação de um algoritmo *force based* a um grafo, antes e após [10]**

Os resultados da aplicação deste tipo de algoritmo costumam ser razoáveis para várias situações. As distâncias entre vértices tendem a ser uniformes, sendo que nós sem ligações entre si são colocados mais distantes, como se vê no exemplo dado. Outra vantagem é o facto de geralmente as arestas terem tamanho similar [9]. A linearidade de arestas é mantida, embora tal leve a que haja cruzamentos entre arestas e mesmo arestas a atravessar nós, não sendo a planaridade um dos critérios seguidos pelo algoritmo. Não existe qualquer preocupação em manter simetrias ou com o sentido das arestas.

O tempo de execução do algoritmo é elevado, na ordem de $O(V^3)$. [9]

O seu uso dependerá muito do tipo de informação que se pretende tratar no grafo, sendo que a principal vantagem deste algoritmo é que mantém a uniformidade entre nós.

Este algoritmo é ideal para grafos com zonas dorsais altamente povoadas, com a presença de anéis periféricos ou estruturas em estrela. Estas regiões, estruturalmente diferentes de uma rede, podem ser facilmente identificadas ao olhar para um desenho produzido por esta via. [10]

Algumas aplicações destes algoritmos foram usadas na bioinformática, representação de conhecimento ou gestão de sistemas. [10]

### *Layout* Hierárquico

O estilo de **layout hierárquico** visa **destacar a direcção do fluxo principal de um grafo dirigido**. No resultado da sua aplicação, os nós de um grafo são colocados em **camadas hierarquicamente organizadas** de tal forma, que a maioria das arestas do grafo apresenta a mesma orientação. Por exemplo, maioritariamente orientadas **verticalmente** de cima para baixo ou **horizontalmente** da esquerda para a direita. Além disso, a ordenação dos nós dentro de cada camada é escolhida de maneira a que o número de **sobreposições de arestas é minimizado** [10].

A estratégia é particularmente **eficiente em grafos acíclicos** já que nesse caso, é sempre possível que todas as arestas tenham a mesma orientação. Ainda assim, num grafo que contenha dependências cíclicas entre nós, estas deverão ser automaticamente **detectadas e resolvidas**, embora nem todas as arestas possam seguir a orientação do fluxo principal nesse caso. [10]

Um exemplo de um grafo acíclico desenhado segundo esta perspectiva ilustra-se na figura seguinte.

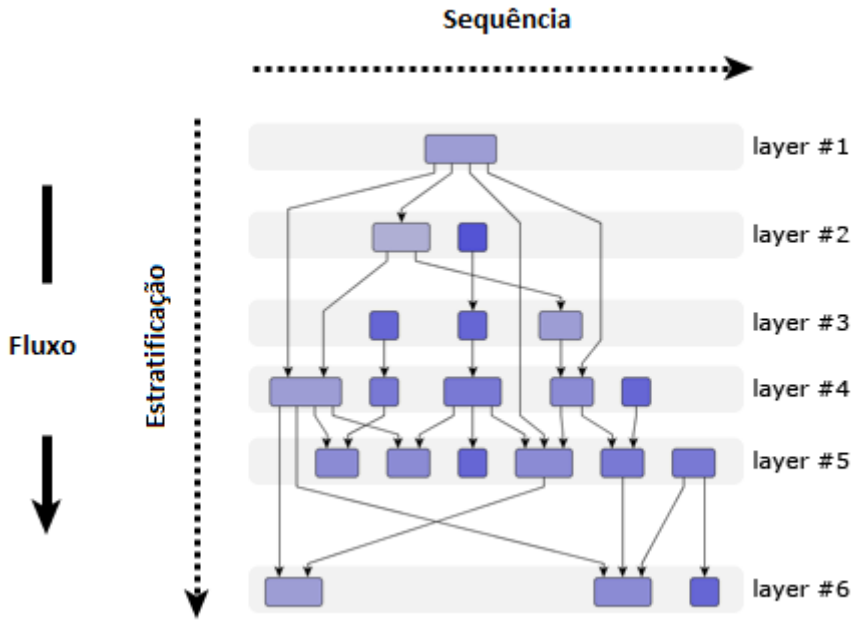


**Figura 17 - Grafo desenhado através de um algoritmo hierárquico, com linhas poligonais [10]**

A ordem dos nós numa determinada camada (*layer*) é chamada **sequência**. O processo de ordenar as camadas de um grafo é chamado **estratificação**. [10]

Como se pode ver na figura 17, a **orientação** de todas as arestas do grafo **é a mesma que a do fluxo**. No entanto, por vezes tal pode não se verificar devido à presença de ciclos que impossibilitem uma orientação igual em todos os casos, como ocorre na aplicação do algoritmo apresentado na figura 18. [10]

O processo para dispor os elementos do grafo segundo este ponto de vista decorre em três passos [10]:

- Determinar a estratificação para todos os vértices do grafo;
- Encontrar uma sequência para os nós de cada camada;
- Atribuir coordenadas a cada um dos vértices e mapear as arestas. Esta fase é normalmente denominada fase de desenho.

Se o grafo não for acíclico, é ainda necessário um primeiro passo adicional que remova esses ciclos, fazendo com que seja possível que as arestas apontem todas no mesmo sentido. Nesse caso, no final da aplicação do algoritmo, o sentido original das arestas deve ser reposto.

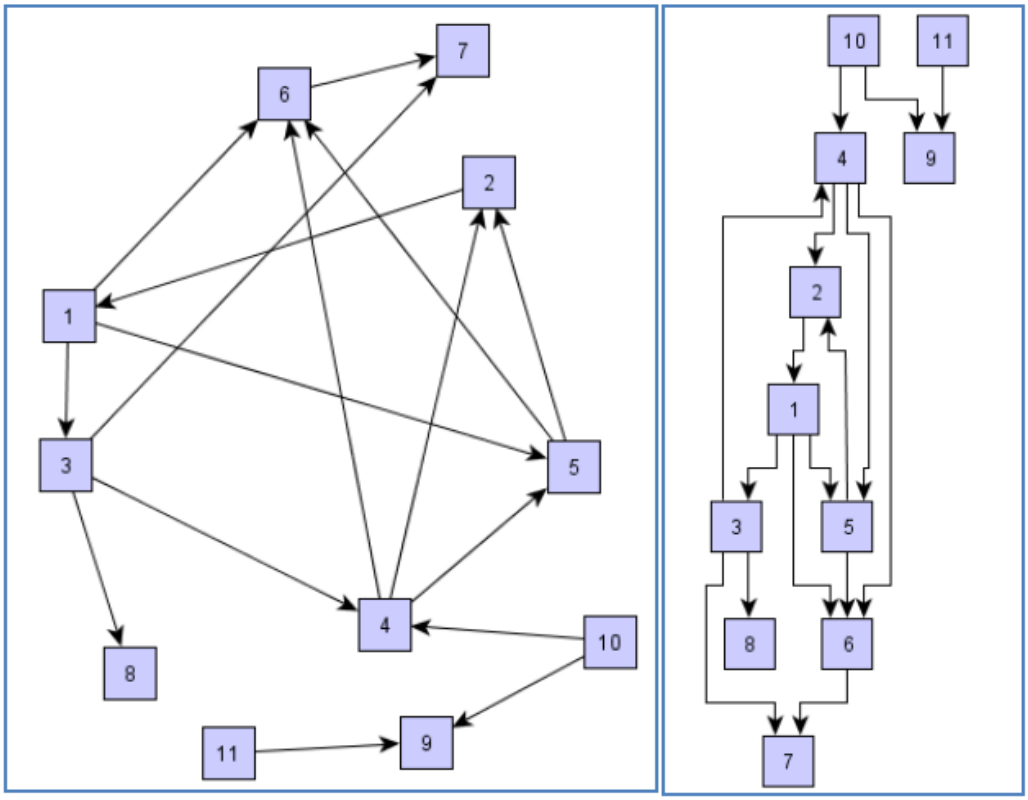


**Figura 18 - Aplicação de um algoritmo de *layout* hierárquico utilizando arestas ortogonais, antes e depois [10]**

Como se vê na figura 18, onde se demonstra a aplicação do algoritmo a um grafo previamente criado, são eliminadas as sobreposições de arestas e a maioria destas encontram-se alinhadas na direcção do fluxo. Devido a alguns ciclos, não é no entanto possível evitar que algumas arestas apresentem **sentido oposto ao fluxo**, caso das arestas **A(3, 4)** e **A(5, 2)**.

Atente-se ainda noutro detalhe: enquanto no grafo representado na figura 17 as arestas estão desenhadas num **formato poligonal**, na aplicação do algoritmo apresentado na figura 18 estas estão **desenhadas ortogonalmente** (apenas linhas horizontais ou verticais). De facto, embora seja mais comum a utilização de arestas poligonais na disposição hierárquica, tal não invalida que se aplique um passo que permita ortogonalizar as arestas.

O *layout* hierárquico é ideal para casos em que seja fundamental que as relações de **dependência entre entidades estejam claramente visíveis**, particularmente se essas relações formarem uma cadeia. Nestes casos, esta perspectiva oferece grande legibilidade. [10]

Os critérios de desenho que procura manter são **planaridade** (evitar cruzamentos entre arestas), **uniformidade** nas distâncias entre nós (embora dependente das ligações que possuem entre si) e tamanhos semelhantes entre arestas. Procura também **manter o sentido principal do fluxo**, ou seja, todas as arestas a apontar na mesma direcção.

Este algoritmo já foi aplicado a grafos representantes de fluxo de trabalho, diagramas de actividade em engenharia de *software*, modelagem de processos e de bases de dados, redes de computadores e diagramas de decisão. [10]

### *Layout* Ortogonal em grelha utilizando algoritmo *topology-shape-metrics*

*O Layout* ortogonal em grelha é uma estratégia que permite desenhar representações de grafos com o intuito de produzir uma visualização clara de redes complexas. [10]

A característica fundamental do desenho ortogonal, é que as arestas podem tomar apenas direcção vertical ou horizontal sendo que as mudanças de direcção são feitas através de **dobras** [8] cujo número deve ser limitado ao mínimo [11].

Neste caso será aplicada a estratégia de *layout* ortogonal em grelha, que é um tipo de representação ortogonal em que os vértices e as dobras das arestas têm coordenadas inteiras (daí o termo grelha) [11].

A título de exemplo, atente-se no diagrama desenhado ortogonalmente, segundo este princípio, da figura seguinte.

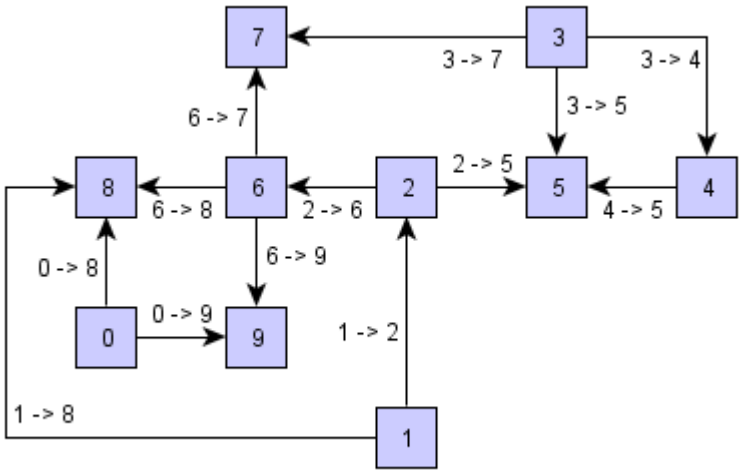


**Figura 19 - Grafo desenhado ortogonalmente [10]**

Uma aproximação que pode ser utilizada para efectuar uma disposição ortogonal em grelha é o algoritmo ***topology-shape-metrics*** (topologia-forma-métricas), que compreende um conjunto de três passos [12]:

- **Planarização** (determina a topologia), neste passo deve tentar-se produzir uma representação do grafo a tratar onde seja reduzida ao mínimo a ocorrência de cruzamentos entre arestas. Caso no final continue a haver sobreposição de arestas, em cada cruzamento deve ser colocado um vértice fictício de forma que fique representado um grafo planar.

- **Ortogonalização** (determina a forma), consiste em produzir uma representação ortogonal do resultado dado na planarização. Para tal, as arestas passarão a conter **dobras**, sempre com ângulos de 90º para evitar sobreposição.

- **Compactação** (determina as métricas), tenta minimizar a área de desenho, diminuindo espaço desnecessário, determinando a coordenada final de cada vértice e a das dobras nas arestas. Neste momento os vértices fictícios colocados para simular as dobras devem ser removidos.

A figura 20 apresenta a aplicação da aproximação indicada passo por passo, o que ajudará a compreender o procedimento efectuado em cada uma das fases indicadas e o seu resultado final.

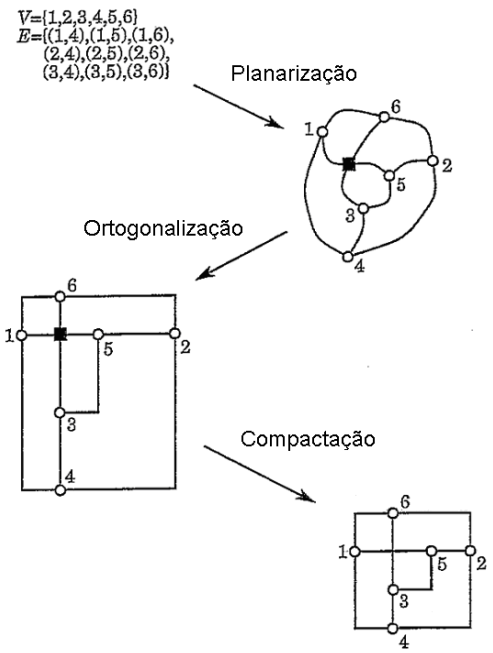


**Figura 20 - Layout ortogonal através da aproximação topology-shape-metrics [12]**

O resultado final é uma representação **sem sobreposição de nós**, **poucos cruzamentos entre arestas** e **número de dobras** nas arestas **reduzido**. Não há grande uniformidade na distância entre nós e arestas. Deve ser, por isso, usado quando for necessária uma representação compacta e com poucos cruzamentos entre arestas.

Já foi testada a estratégia em áreas como a engenharia de *software*, esquemas de bases de dados, gestão de sistemas, representação de conhecimento e na engenharia electrotécnica. [10]

## Combinação de algoritmos de *layout* de grafos

Existem vários tipos de algoritmos de *layout* além dos especificados. Alguns são baseados em princípios diferentes aos apresentados, outros são o resultado da combinação destes.

A combinação de algoritmos de *layout*, poderá oferecer resultados interessantes na representação de grafos mais complexos, que possam ser decompostos em várias partes.

Veja-se a figura seguinte que decompõe o problema do desenho de um grafo em vários grupos, onde a ordenação é primeiramente aplicada sobre os grupos e posteriormente sobre a totalidade do grafo. [10]

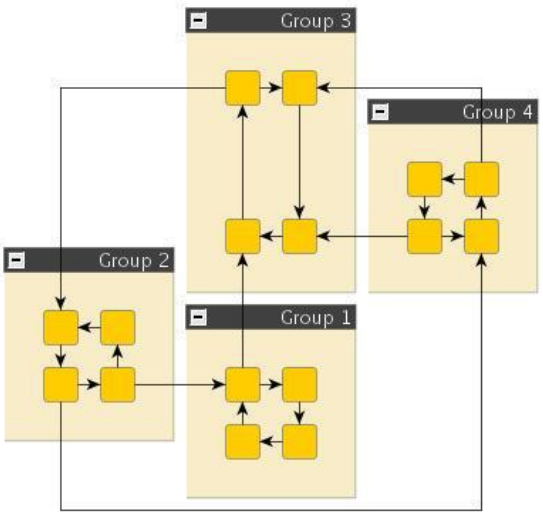


**Figura 21 - Desenho de grafo dividindo o problema em grupos [10]**

A combinação entre algoritmos para produzir resultados que ofereçam melhor visualização de um grafo, poderá ser uma forma de resolver problemas de disposição em que os elementos de um diagrama detêm diferentes características, podendo ser agrupados e dispostos de forma independente, antes de ser feito o desenho da totalidade do grafo.

## 2.4 Ferramentas de desenho de diagramas com funcionalidades de *layout* automático

Actualmente, já estão disponíveis diversas ferramentas de suporte à construção de diagramas, pelo que, o levantamento das suas funcionalidades poderá ser importante para retirar alguns conceitos que podem ser úteis na aplicação que se pretende aprimorar.

No entanto, dado o nicho de mercado de uma aplicação SCADA/DMS (como o ***Scatex)*** ser muito reduzido, o que leva a que as aplicações de *software* desenvolvidas na área sejam destinadas a clientes específicos, não foi possível a análise de sistemas de suporte à criação de diagramas de distribuição de redes de energia eléctrica.

Ainda assim, dado existirem algumas semelhanças entre estes diagramas e os criados para o desenvolvimento de circuitos impressos, estando disponíveis para este último caso várias aplicações que permitem a disposição automática dos componentes, foi feita uma análise a um sistema deste tipo. Foi escolhida a ferramenta **orcad** pelo facto de ser uma das que permite disposição automática dos componentes.

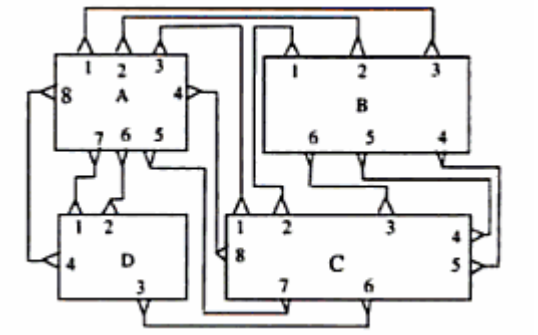


**Figura 22 - Desenho ortogonal de circuito integrado [13]**

Uma análise sobre as funcionalidades de disposição automática apresentadas pela ferramenta de desenho de circuitos integrados (ver figura 22) **Orcad** será feita de seguida.

### 2.4.1 Orcad

O **orcad** é uma ferramenta utilizada para o desenho de circuitos impressos. Além das funcionalidades associadas à inserção e remoção de componentes num diagrama, possui duas funcionalidades a salientar, relacionadas com a colocação automática de componentes num diagrama e com o encaminhamento automático de ligações a componentes. [14]

A ferramenta de ***autoplacement*** permite dispor automaticamente componentes (sem ligações) sobre uma área de desenho delimitada previamente pelo utilizador. Este pode também optar por inicialmente fixar a posição de determinado componente, atribuindo-lhe uma localização pré-definida, utilizando um comando denominado *lock* (fixar). Desta forma, todos os componentes que estiverem marcados como *lock* mantêm a sua localização de origem. Veja-se a figura 23, onde é aplicada a disposição sobre componentes pré-colocados pelo utilizador. [14]

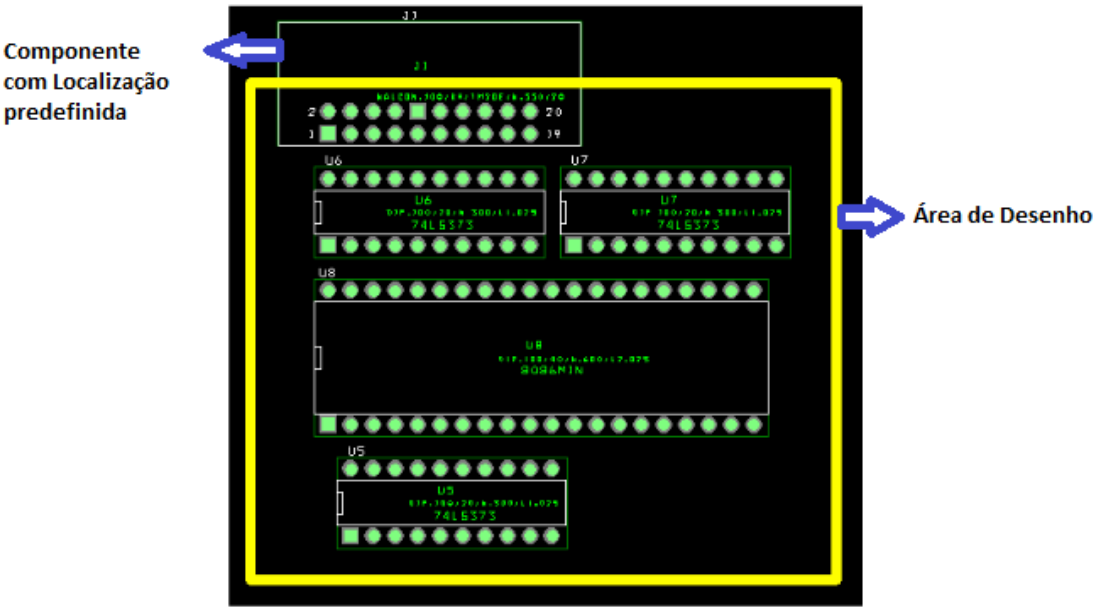


**Figura 23 - Funcionalidade de *Autoplacement* do Orcad [14]**

O programa incorpora ainda uma funcionalidade denominada ***autorouting*** que faz automaticamente a ligação dos componentes do circuito, previamente indicadas pelo utilizador, evitando cruzamentos entre estas. O utilizador começa por desenhar as ligações, sem se preocupar com a sobreposição destas e, após aplicar o utilitário de ***autorounting***, todas as ligações dentro de uma área delimitada são dispostas de forma a terem o mínimo de cruzamentos possíveis. Veja-se a figura 24 que mostra a aplicação da funcionalidade.

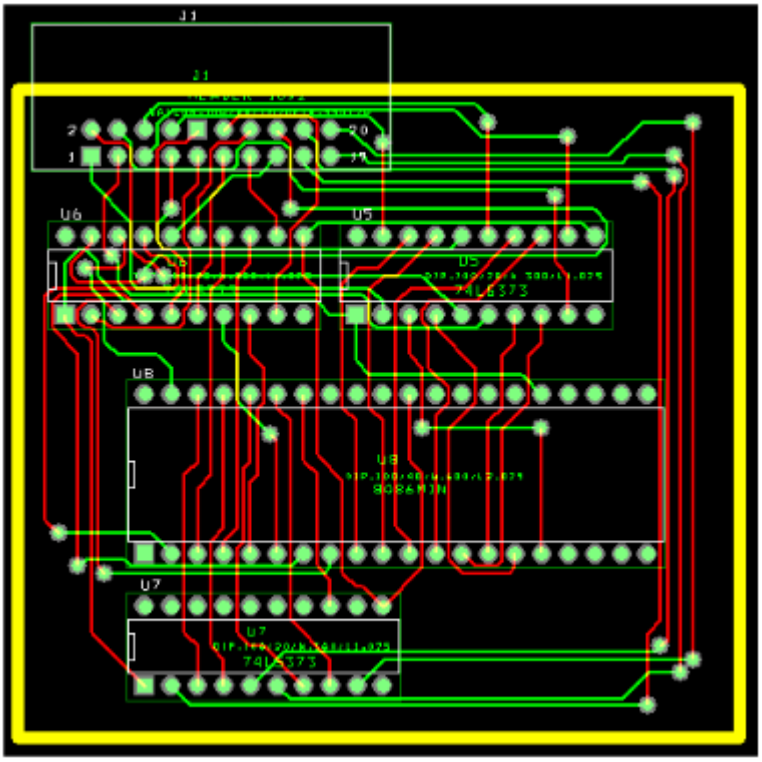


**Figura 24 - Funcionalidade de *Autorouting* do Orcad [14]**

Repare-se que o tipo de desenho utilizado no encaminhamento das arestas é poligonal, utilizando dobras (quebras de linha) de forma a mudar o sentido das ligações.

Os componentes, por seu lado, são colocados em grelha, de forma a ocuparem o mínimo de espaço possível.

Tendo em conta que a funcionalidade de ***autoplacement*** e de ***autorouting*** parecem actuar separadamente, é de crer que a colocação dos componentes na área de desenho não leva em conta as ligações destes a outros componentes, o que poderia ajudar a evitar cruzamentos entre arestas. Esta preocupação parece estar associada unicamente à funcionalidade de ***autorouting***, o que leva a que nem sempre se encontre uma disposição tão boa como aconteceria se os componentes tivessem sido colocados considerando as ligações.

# 3. Ferramentas propostas para o editor de Diagramas Esquemáticos

## 3.1 Introdução

Os objectivos propostos para a dissertação passam pela sugestão de ferramentas que facilitem a tarefa de desenhar e manter (actualizar) diagramas esquemáticos de rede eléctrica. Foi colocado como objectivo não apenas identificar possíveis métodos de geração automática de partes de um diagrama (através da leitura da base de dados), mas também, possíveis formas de interacção destes com o utilizador, bem como de outras ferramentas que lhe possam ser igualmente úteis.

Dado que as ferramentas concretas a integrar no editor de esquemáticos não foram definidas à partida, faz parte do estudo da dissertação defini-las, assim como propor métodos de interacção do utilizador com estas.

Este capítulo destina-se a identificar ferramentas de suporte ao utilizador que se crê que deveriam fazer parte do editor. Será também descrito através de protótipos a *interface* que se julga que deveriam apresentar para comunicar com o utilizador.

Não é objectivo deste capítulo indicar detalhes técnicos de como tais ferramentas poderiam operar, apenas métodos de interacção e resultados que se pretenderia obter.

As ferramentas a propor foram divididas em dois capítulos. O primeiro apresenta ferramentas a implementar que incorporem funcionalidades de desenho automático de partes do diagrama. O segundo sugere ferramentas que ajudem o utilizador na manutenção de diagramas esquemáticos, indicando erros, ou informação incoerente nestes.

## 3.2 Ferramentas de Geração Automática

A geração automática de partes de um diagrama esquemático evitaria procedimentos morosos e repetitivos ao utilizador, gerando ou modificando de forma automática partes de um diagrama, incluindo componentes e ligações. Permitiria ainda evitar erros relacionados com a introdução de dados errados pelo utilizador.

Da geração automática deveriam constar ferramentas que permitissem a inserção e disposição automática de objectos e ligações de um diagrama. As ferramentas visariam assistir (auxiliar) o utilizador a construir os seus diagramas evitando-lhe o processo repetitivo de colocar símbolo por símbolo os equipamentos no diagrama.

Neste capítulo, será apresentada a *interface* de uma hipotética ferramenta a introduzir no editor de diagramas, que permitiria a geração automática de partes destes, bem como o resultado esperado da sua aplicação.

O diagrama seguinte apresenta os casos de uso que se pretende para uma ferramenta de geração automática.

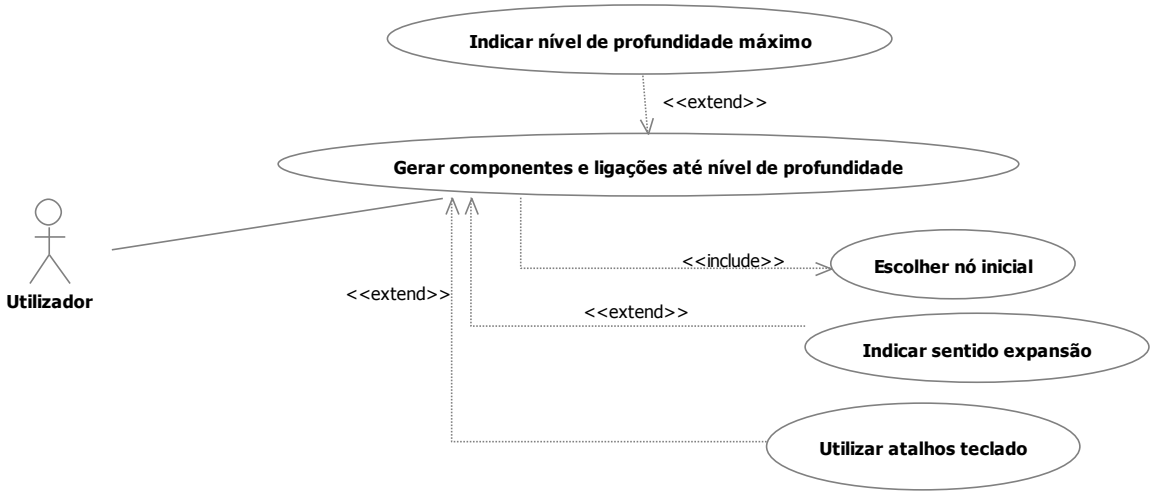


**Figura 25 - Casos de uso respeitantes a geração automática de partes do diagrama**

A ferramenta tratar-se-ia de uma funcionalidade que gerasse todos os componentes e respectivas ligações de um determinado componente até dado nível de profundidade, indicado pelo utilizador. Até esse nível, seriam gerados todos os componentes e respectivas ligações intermédias.

Dado que os diagramas podem ser muito extensos (na ordem dos milhares de equipamentos), se não fosse indicada uma profundidade máxima de expansão, surgiria o problema do diagrama ser todo gerado, possivelmente com um aspecto pouco satisfatório (apresentando má visibilidade dos componentes). Assim, é necessário aplicar critérios de paragem numa expansão automática. O critério de paragem mais lógico é utilizar as subestações apresentadas nos diagramas, por representarem estruturas principais deste. Na figura seguinte apresenta-se um extracto de um diagrama que demonstra a representatividade que uma subestação tem num diagrama.

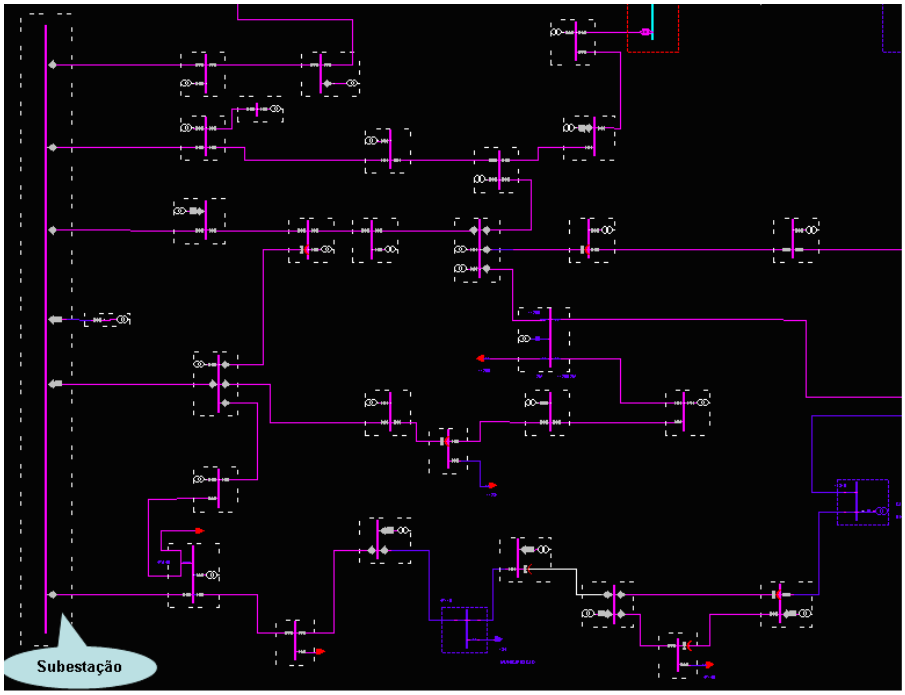


**Figura 26- Subestação de diagrama esquemático**

Mediante os factos descritos, seria desejável a geração ocorrer até que a profundidade máxima indicada pelo utilizador fosse atingida ou até ao surgimento de uma subestação. No último caso, devem ser indicados ao utilizador os locais onde foi efectuada uma paragem na expansão automática devido ao critério de corte.

Há ainda um outro critério de paragem natural, no caso de, no decurso de uma expansão, serem encontrados equipamentos já existentes no diagrama. Nesse caso, esses equipamentos deveriam também servir como critério de paragem na expansão, visto já se encontrarem no diagrama. Tais paragens deveriam igualmente ser indicadas ao utilizador.

A figura seguinte apresenta uma possível *interface* a implementar no editor da ferramenta sugerida.

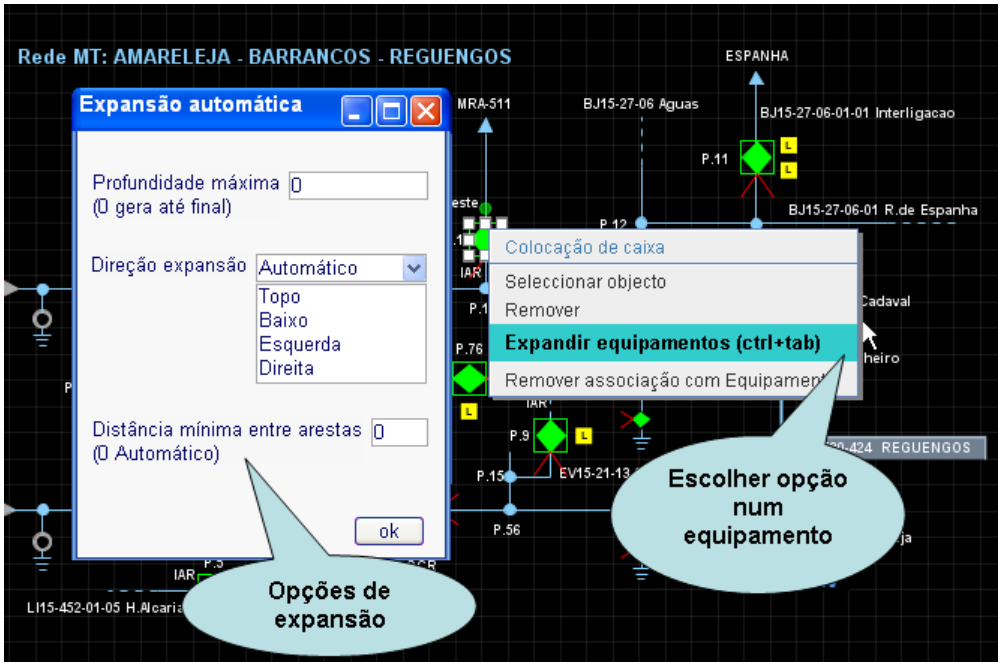


**Figura 27 - Possível *interface* de ferramenta de geração automática no editor de diagramas**

**Indicar sentido da expansão**

Como caso de uso adicional, poderia ser dada a possibilidade ao utilizador de indicar o sentido para o qual se deveria expandir a "nuvem" de equipamentos e ligações a partir do componente escolhido, tal como ilustrado na *interface* proposta.

**Utilizar atalhos do teclado**

Uma funcionalidade simples que seria interessante dispor ao utilizador, era dar-lhe a possibilidade de gerar rapidamente apenas um nível de profundidade num determinado equipamento, ou seja, gerar os componentes que lhe estão directamente ligados.

Uma possibilidade é facultar um **atalho no teclado** que, estando um componente seleccionado, adicionaria todos os componentes que lhe estão ligados. Poderia ser usado para o efeito a combinação de teclas **ctrl+tab**. Pressionando a combinação diversas vezes, seriam dadas várias opções de disposição para esses equipamentos.

Como se depreende da informação apresentada, a ferramenta proposta serviria essencialmente para gerar automaticamente a maioria das **ramificações** de um diagrama esquemático. Caberia depois ao utilizador pegar nos resultados devolvidos e efectuar-lhes correcções. Ao mesmo tempo iria controlando o processo de geração automática uma vez que, embora grandes partes do diagrama pudessem ser geradas de uma vez, estariam todas ligadas a uma mesma estrutura principal (subestação).

## 3.3 Ferramentas de detecção de Erros ou Incoerências

Este capítulo serve para introduzir um conjunto de funcionalidades que se crê, seriam úteis no editor de diagramas, visando evitar a criação de diagramas com erros e por outro lado, facilitar a manutenção de diagramas esquemáticos, já que podem ocorrer modificações na base de dados real dos equipamentos levando a que surjam incoerências na informação entre esta e os diagramas.

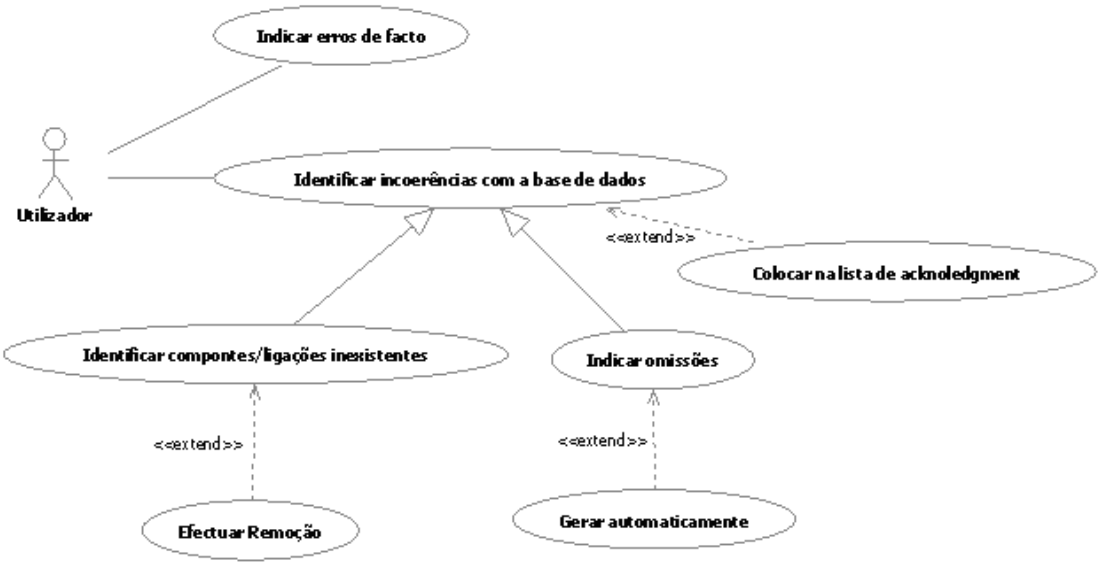


**Figura 28 - Casos de Uso de um possível módulo de identificação de erros e incoerências nos diagramas esquemáticos**

Na detecção de erros ou incoerências nos diagramas desenhados, é importante distinguir entre dois tipos de erro a considerar. Podem ocorrer erros de facto, ou seja, o utilizador coloca no diagrama informação que, por regra, está errada (por exemplo ligar dois equipamentos que não podem ser conectados), ou erros de incoerência com a informação presente na base de dados.

É importante facultar funcionalidades que lidem com os dois tipos de erro que podem surgir e comunica-los ao utilizador.

### Indicar Erros de Facto

Os erros de facto ocorrem imediatamente após uma acção do utilizador, por claro equívoco deste, que efectuou uma acção inválida. Um exemplo de um erro de facto, seria ligar dois tipos de equipamento incompatíveis. Estes erros são por isso estáticos, podendo mesmo ser catalogados.

Um erro de facto deve ser detectado imediatamente após a sua ocorrência, impedindo o utilizador de concluir uma acção errónea. Deveria, por isso, ser-lhe apresentada uma mensagem indicando-lhe o erro cometido, como apresentado na figura 29.

A detecção do erro não deveria, contudo, sobrepor-se à vontade do utilizador, deixando-o prosseguir se ainda assim fosse essa a sua intenção. Adicionalmente, poderia ainda ser perguntado ao utilizador se desejaria alterar a base de dados de erros de facto, para que semelhante erro no futuro deixasse de ser motivo de alerta.

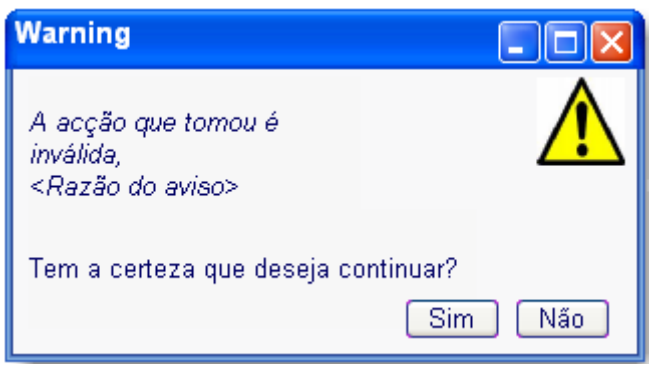


**Figura 29 - Protótipo de mensagem a apresentar em caso de ocorrência de erro de facto**

**Identificar incoerências do diagrama relativamente à base de dados real**

A detecção de incoerências no diagrama relativamente à informação da base de dados real é outra das funcionalidades que seria muito útil ao utilizador, principalmente na manutenção dos diagramas.

A solução passaria por uma ferramenta que permitisse listar as **omissões** de equipamentos/ligações no diagrama que estão presentes na base de dados real e a detecção de equipamentos/ligações presentes no diagrama que **não existem** na base de dados.

Para o efeito, é proposta a implementação de uma funcionalidade no editor de diagramas que liste as incoerências encontradas e permita a navegação para o erro, tal como a *interface* na imagem seguinte apresenta.

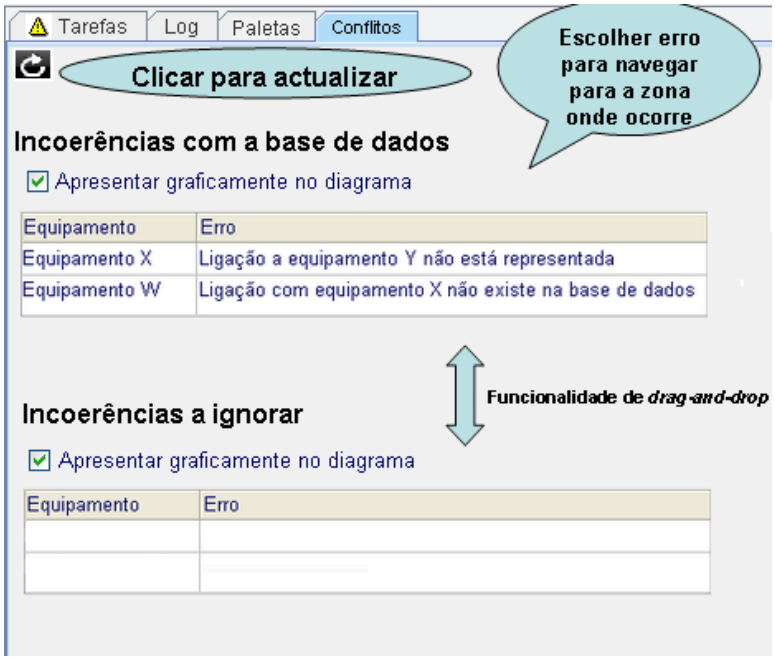


**Figura 30 - Interface que apresenta os conflitos existentes num diagrama relativamente à informação da base de dados real**

A figura 30 apresenta um protótipo da *interface* a utilizar que permitiria, não apenas obter um registo dos erros, mas também **navegar** para estes. Além de uma lista com as incoerências encontradas, deveria complementarmente estar presente uma outra com as incoerências que o utilizador decida **ignorar** (lista de ***acknowledgement***), para que não esteja constantemente a ser alertado dos mesmos problemas.

As *checkbox* presentes na *interface* sugerida para a indicação de incoerências na base de dados (figura 30) teriam o propósito de activar/desactivar a opção de, para cada uma das tabelas, apresentar ou não os erros **graficamente** no diagrama (figuras 30 e 31).

Uma apresentação gráfica dos erros, no caso de se tratar de conteúdo em falta no diagrama, apresenta-se na figura 31. Analogamente, na figura 32 apresenta-se uma proposta da *interface* gráfica a aplicar no caso do diagrama esquemático conter conteúdo não especificado na base de dados.

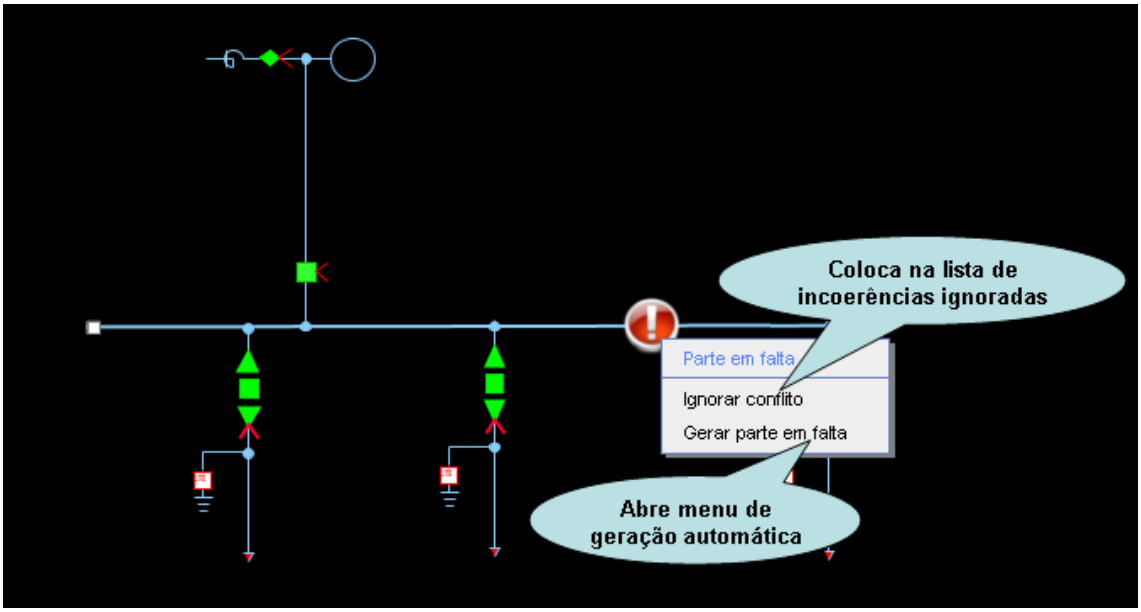


**Figura 31 - Interface gráfica de conflito em diagrama esquemático devido a informação em falta**

A geração automática de componentes, nestes casos, só seria possível mediante certas condicionantes, tornando-se inviável se existissem muitos componentes em falta.

No caso de ser possível gerar os componentes em falta, esse processo seria semelhante ao congénere aplicado sobre um equipamento cujas ligações se deseja expandir, requisito anteriormente especificado. Após a geração automática, caberia contudo ao utilizador fazer ajustes sobre a parte gerada.

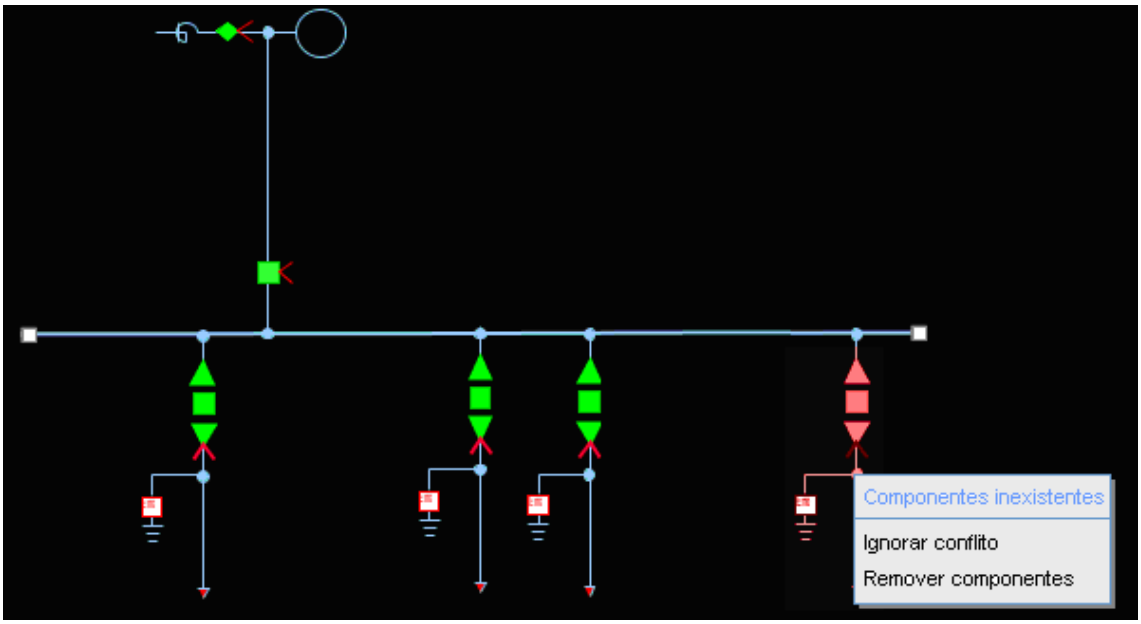


**Figura 32 - Interface gráfica de conflito em diagrama esquemático devido a conteúdo inexistente**

## 3.4 Conclusão sobre as ferramentas propostas

Neste capítulo foi feito um levantamento de alguns requisitos e respectivas *interfaces* que se crê que o editor de diagramas esquemáticos deveria apresentar. Primeiramente foi sugerida uma

funcionalidade que permitisse gerar automaticamente partes de diagramas esquemáticos. Seguidamente, foram propostas algumas ferramentas que pudessem ajudar o utilizador no processo de detecção e correcção de erros e incoerências nos diagramas.

É de crer que a funcionalidade que traria mais benefício aos utilizadores seria a de geração automática de componentes, sendo também o processo mais complexo.

A geração automática de ramos de um diagrama esquemático é um problema no âmbito da área de desenho de grafos, tema alvo de estudo no estado da arte deste documento.

Sendo o processo mais relevante para a dissertação, o processo de **geração automática de partes de um diagrama** centrará todos os estudos em capítulos posteriores deste documento. Não deixando de carecer de estudos mais aprofundados, as ferramentas de detecção de erros sugeridas parecem, à partida, ser funcionalidades mais simples de desenvolver pelo que não serão abordados possíveis métodos técnicos para a sua possível implementação.

# 4. Geração Automática de Ramificações de Diagramas Esquemáticos

## 4.1 Introdução

Como tem vindo a ser afirmado em capítulos anteriores, é objectivo principal da dissertação estudar metodologias que possibilitem a geração automática de partes de um diagrama esquemático de rede eléctrica.

Esta secção estuda concretamente essa possibilidade, apresentando um possível método técnico para conseguir gerar ramos de diagramas esquemáticos. Esse método é baseado na literatura existente na área de teoria dos grafos e está em consonância com um dos algoritmos de desenho de grafos apresentado no estado da arte, concretamente *layout* hierárquico de grafos.

Mais especificamente, foi escolhido o algoritmo de *Sugiyama* como metodologia alvo de estudo, para ser aplicada no desenho de ramos de diagramas esquemáticos de rede eléctrica. Este algoritmo, que herdou o nome do seu autor, foi descrito ao detalhe no livro *Graph drawing and applications for software and knowledge engineers* [15].

De forma a disponibilizar informação detalhada a respeito da utilização do algoritmo como forma de gerar automaticamente partes de um diagrama, este capítulo foi dividido em três subcapítulos. O primeiro faz uma pequena apresentação do algoritmo proposto por *Kozo Sugiyama*, o segundo descreve as razões que motivaram a escolha deste como possibilidade para gerar partes de diagramas esquemáticos. Finalmente, o último subcapítulo da secção mostra ao detalhe, passo a passo, os procedimentos que deveriam ser seguidos para permitir a utilização desta metodologia no desenho de ramos de diagramas esquemáticos.

## 4.2 Algoritmo de *Kozo Sugiyama*

O algoritmo de *Sugiyama* é um algoritmo de disposição automática de grafos dirigidos, que permite apresentar os nós de um grafo hierarquicamente, de acordo com o seu fluxo. [15]

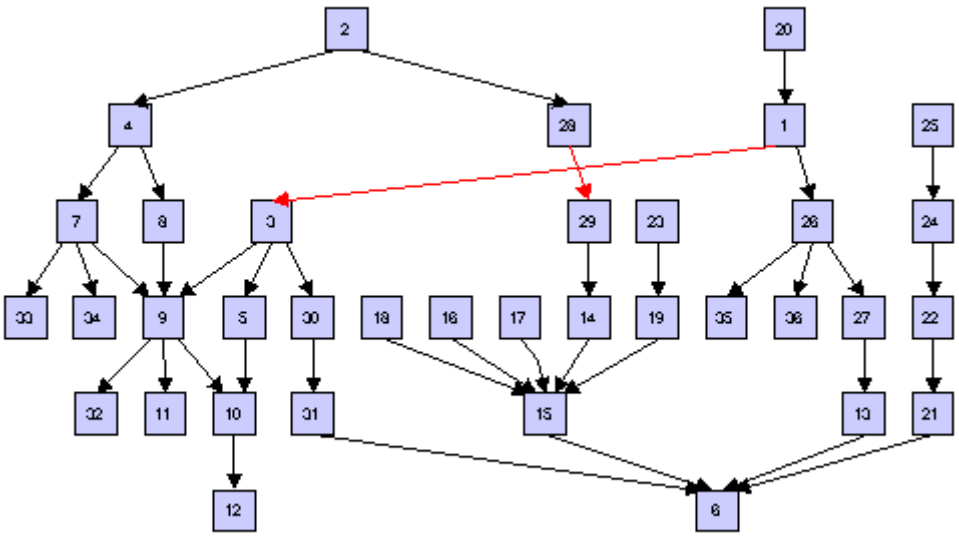


**Figura 33 - Grafo dirigido disposto hierarquicamente utilizando o algoritmo de *Sugiyama***

A figura acima apresenta o resultado da aplicação do algoritmo de *Sugiyama* sobre um grafo dirigido com disposição aleatória.

Atente-se que os nós do grafo estão colocados hierarquicamente de acordo com o fluxo do grafo, procurando-se minimizar os cruzamentos entre arestas (a única excepção está representada com as arestas a vermelho). Por outro lado, os nós estão distribuídos de forma balanceada, procurando manter-se a sua proximidade, de acordo com as suas ligações a outros nós. [15]

O algoritmo apresenta quatro passos fundamentais, tal como descrito de seguida. [15]

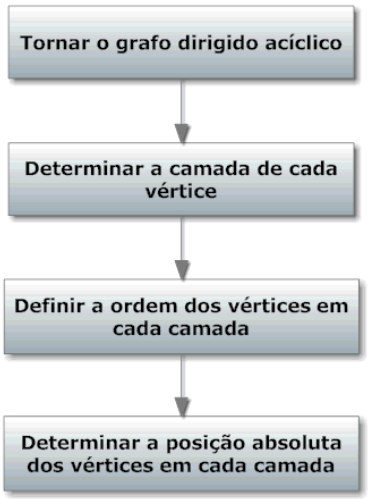


**Figura 34 - Passos do algoritmo de *Sugiyama***

Os capítulos seguintes irão apresentar ao detalhe cada um dos passos acima assinalados, onde serão também indicadas as modificações efectuadas ao algoritmo de forma a satisfazer as necessidades identificadas para o problema específico.

## 4.3 Utilização do algoritmo de *Kozo Sugiyama* na geração automática de ramos em diagramas esquemáticos

A possível aplicação de um algoritmo hierárquico nos diagramas esquemáticos apresentados, nomeadamente o algoritmo de *Sugiyama*, carece da resolução de alguns problemas. Desde logo, parece ser impossível a geração de um diagrama completo partindo apenas deste princípio, estando por isso o estudo efectuado centrado em tentar gerar apenas ramificações de um diagrama, impondo como condição máxima de paragem o surgimento de uma nova subestação ou o termo de um ramo.

No entanto, é credível que a geração de ramos correspondentes às saídas das subestações (*feeders*), constitui provavelmente a tarefa mais maçadora e repetitiva para utilizadores humanos, dado o grande número de equipamentos que uma saída pode apresentar, pelo que este estudo pode revelar-se de grande utilidade prática.

Descartada a possibilidade de gerar diagramas completos utilizando exclusivamente esta via, subsiste ainda o problema de definir uma orientação para os diagramas. É também condicionante o facto dos diagramas apresentarem equipamentos com conectores e símbolos compostos, questões não colocadas em grafos comuns e por isso não consideradas pelo autor do algoritmo.

Antes de partir para a explicação de uma possível forma de gerar ramificações de um diagrama esquemático, é importante descrever os problemas citados, bem como eventuais formas de os contornar. Por essa razão, neste capítulo é examinada cada uma das problemáticas identificadas como eventual entrave à aplicação do algoritmo de *Sugiyama* para gerar ramos de um diagrama.

A análise aos problemas identificados como condicionantes à aplicação do algoritmo será feita pela seguinte ordem:

- Facto de existirem componentes compostos;
- Problema de definir um sentido para os ramos gerados;
- Equipamentos conterem conectores.

No final do capítulo serão ainda apresentadas as razões que levaram à escolha deste algoritmo como base de partida para a geração automática de ramos de diagramas esquemáticos.

### 4.3.1 Componentes Simples e Compostos

Fazem parte de um diagrama esquemático tipos de componente simples, representados por um único símbolo e componentes compostos, cuja representação é feita utilizando mais que um símbolo pelo facto de cada um dos seus constituintes deter uma representação própria. As figuras seguintes apresentam respectivamente um componente simples e um composto usados em diagramas esquemáticos.

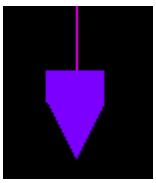

**Figura 35 - Componente simples**

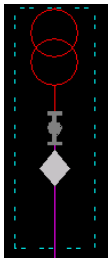

**Figura 36 - Componente composto com três símbolos**

Nos diagramas esquemáticos considerados, as representações dos componentes compostos encontram-se delimitadas a tracejado tal como o apresentado na figura 36. No exemplo indicado, o componente é decomposto em três símbolos.

A melhor forma encontrada para lidar com componentes compostos por mais que um símbolo foi tratar os mesmos como equipamentos simples para efeito de geração automática de ramos. No entanto, é necessário criar a sua representação previamente, passando o conjunto de símbolos a actuar como um único símbolo composto pela junção destes. Dado o tipo de componentes compostos ser limitado, trata-se de um procedimento simples, pelo que, na pior das hipóteses, teria de ser criada uma representação estática pré-processada de cada componente composto possível. Passa a denominar-se símbolo composto à representação de um componente composto.

Nos diagramas é ainda de salientar a existência de pontos de intersecção, não podendo estes ser catalogados como componentes, por representarem unicamente pontos de bifurcação/junção de ligações. Apesar de tudo, é de crer, que para efeitos de disposição automática estes possam ser tratados como componentes simples. A figura 37 apresenta um exemplo de um ponto de intersecção.

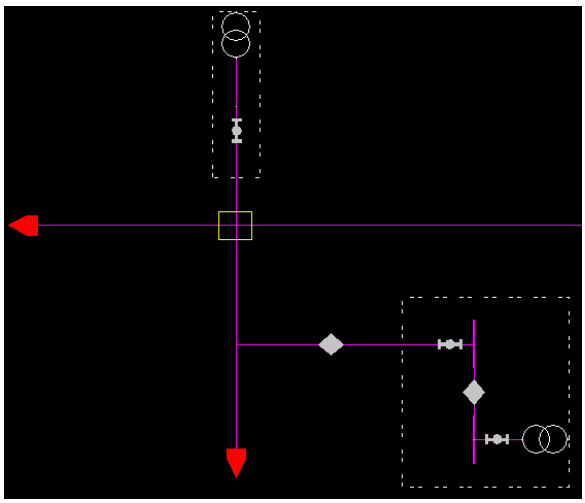

**Figura 37 - Ponto de intersecção**

### 4.3.2 Definir uma orientação

Embora os diagramas esquemáticos não se tratem de grafos dirigidos, é necessário definir uma orientação para um determinado ramo de forma a utilizar o algoritmo de *Sugiyama*. Um possível método de o conseguir, é escolhendo um conector de um componente representado no esquemático e gerar todos os componentes que aparecem ligados a este até um certo nível de profundidade ou impondo outras condições de paragem, obtendo desta forma um grafo dirigido, com o fluxo tomando o sentido do conector para os componentes gerados.

Neste caso, é importante definir o conceito de ramo. Considera-se um ramo, o conjunto de componentes e respectivas ligações que se encontram acoplados directa ou indirectamente a um conector de um componente até o seu termo, ou até encontrar uma subestação. Desta forma, pode identificar-se o ramo como o obtido a partir desse mesmo conector. Veja-se, como exemplo, a figura seguinte, que representa um ramo retirado de um diagrama esquemático.

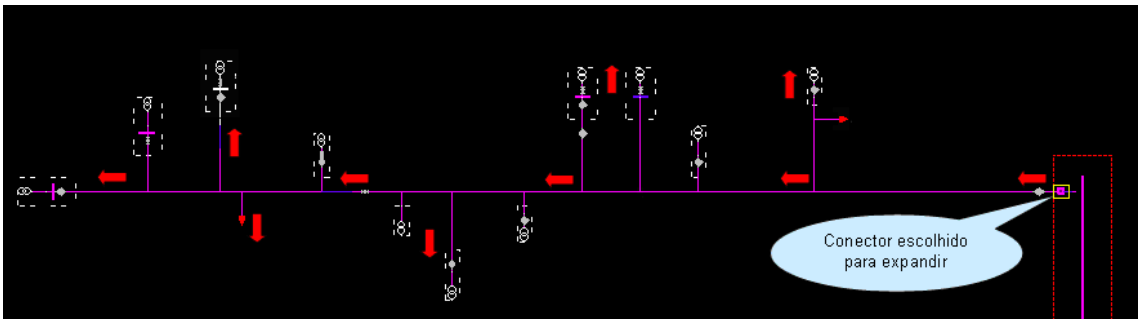


**Figura 38 -Exemplo de um ramo de um diagrama esquemático**

O exemplo da figura 38 apresenta o aspecto característico de uma ramificação de um diagrama esquemático. O ramo é obtido a partir da expansão do equipamento escolhido e toma a orientação, tal como indicado, partindo desse equipamento para os que lhe estão conectados.

Um problema que eventualmente poderá surgir, durante uma geração automática, é a possibilidade de existirem pontos de intersecção que contêm sub-ramificações disjuntas que voltam a encontrar-se (ciclos). Estas deverão ser detectadas e o fluxo deve ser mantido (ou terminar caso dessa intersecção constem apenas as duas sub-ramificações que se encontram) a partir do ponto onde foi detectada essa junção, evitando assim criar ciclos e consequentemente geração de equipamentos repetidos. Uma forma de o fazer, é verificar antes de gerar um novo equipamento se este não foi já expandido, ou seja, verificar se faz parte do "caminho" de componentes gerados. Um exemplo de dois pontos de intersecção que poderiam levar a um ciclo interminável ilustra-se na figura 39.

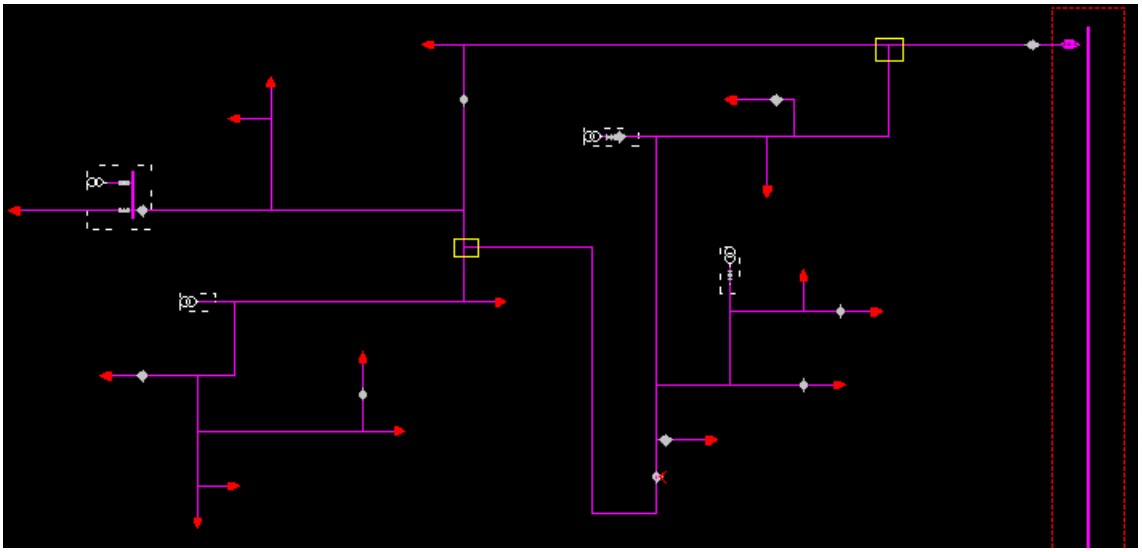

**Figura 39 - Intersecção de duas ramificações**

O ponto de reencontro entre os dois fluxos disjuntos dependeria da ordem de expansão utilizada nos equipamentos, mas deveria ocorrer entre os dois pontos indicados na figura 39 (incluindo os mesmos).

### 4.3.3 Conectividade dos componentes

A grande diferença entre um grafo dirigido comum, utilizado por *Kozo Sugiyama* nos seus estudos sobre desenho de grafos e os diagramas esquemáticos estudados, é a existência de conectores nos componentes destes, contrariamente ao que acontece num nó de um grafo. Este facto implica que as arestas deverão ligar-se especificamente a um conector e não simplesmente ao nó.

Vários problemas podem advir desta característica, nomeadamente cruzamentos entre arestas que não surgiriam em nós comuns, bem como a necessidade de encontrar um ângulo de rotação adequado para cada vértice.

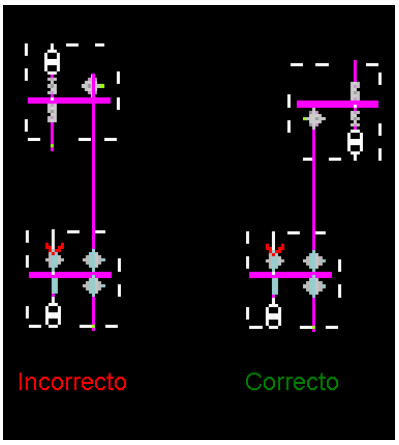


**Figura 40 - Rotação incorrecta (esquerda) de um equipamento e correcta (direita)**

Como se vê na figura anterior, é necessário definir a rotação certa para um determinado nó, caso contrário, a representação obtida poderá não ser a mais correcta, sobrepondo parte dos símbolos deste.

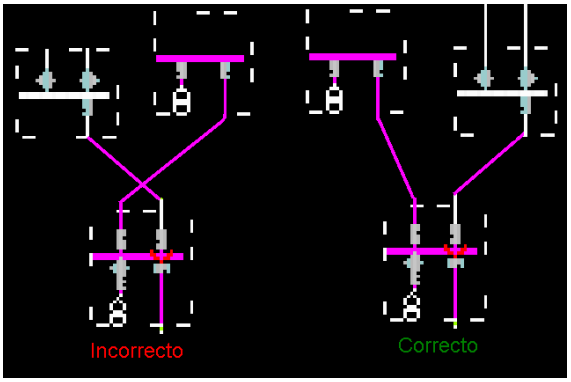


**Figura 41 - Exemplo de cruzamento devido aos conectores**

Na figura 41 verifica-se um cruzamento que ocorre devido à existência de ligações para o mesmo nó, mas para conectores distintos. Neste caso, a representação correcta, também apresentada na figura, foi obtida mudando a disposição dos nós superiores. Outra forma que poderia ter sido usada para obter uma representação igualmente certa, passaria por inverter horizontalmente o nó inferior.

É necessário também descrever o comportamento dos conectores em componentes compostos. A melhor solução parece ser considerar apenas os conectores dos elementos que

compõem o símbolo composto que têm ligações para o seu exterior. Nesse caso, o componente passaria a actuar exactamente da mesma forma que um componente simples.

Veja-se o exemplo da figura 42, em que os conectores com ligação para o interior do componente estão assinalados a amarelo. O que tem ligação para o exterior está assinalado a vermelho. Apenas este último seria considerado, passando o componente composto a actuar como um componente simples.

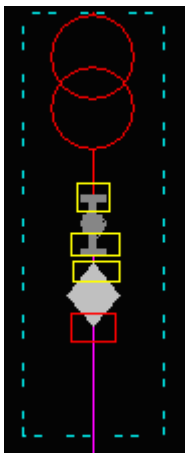

**Figura 42 - Conector de um componente composto**

### 4.3.4 Vantagens da utilização do algoritmo de *Sugiyama*

A vantagem que parece ser mais evidente na utilização deste algoritmo, é o facto das ramificações de diagramas esquemáticos, no fundo, apresentarem uma orientação de aparência hierárquica (veja-se por exemplo as figuras 38 e 43), embora por vezes o seu sentido seja invertido como acontece em alguns pontos do extracto de um diagrama apresentado na figura 39. Geralmente, subestações apresentam uma posição hierárquica superior aos restantes componentes. Por essa razão conectores das saídas de subestações (*feeders*) serão provavelmente escolhas privilegiadas na altura de expandir automaticamente componentes. Veja-se como exemplo a imagem da figura 43, que mostra as ramificações de uma subestação, dando a ideia desta se encontrar num plano superior.

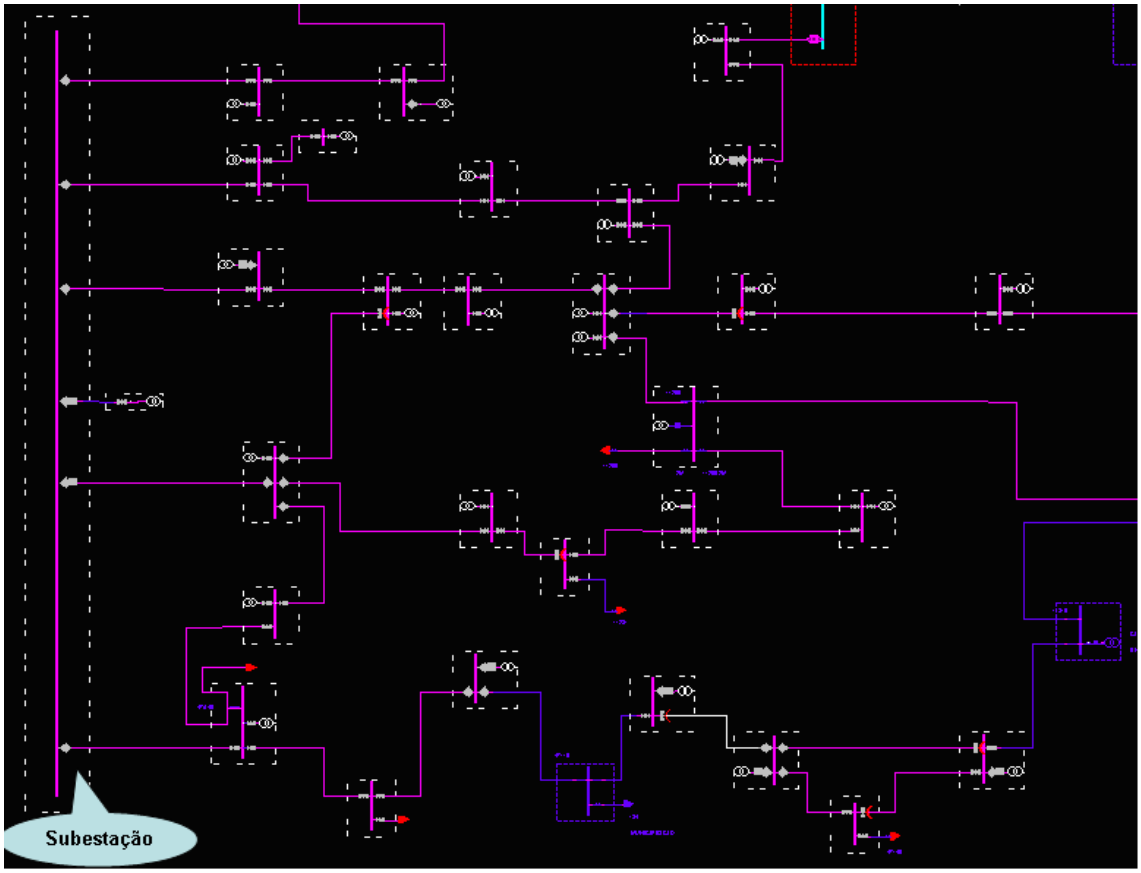


**Figura 43 - Direcção dos ramos de equipamentos que partem de uma subestação**

Na escolha do algoritmo de disposição a utilizar, nada melhor que contrapor as necessidades dos diagramas esquemáticos com os critérios estéticos que normalmente os algoritmos de disposição procuram cumprir. Recorde-se que os critérios normalmente utilizados, tal como especificado no estado da arte, são: **simetria**, **planaridade** (ausência de cruzamentos), **linearidade** (nas arestas), **uniformidade de nós**, **uniformidade de arestas** e **manter sentido do fluxo** [8]. A estes junte-se uma característica importante a manter nos diagramas esquemáticos, **ortogonalidade das arestas**. Como tal, antes de determinar o algoritmo a utilizar para estudar como possibilidade para geração de ramos de diagramas esquemáticos, consideraram-se quais os critérios importantes para os ramos de diagramas esquemáticos. Assim, construiu-se a seguinte tabela, que apresenta, para cada um dos três tipos de algoritmo apresentados no estado da arte, quais dos critérios anteriores procuram resolver. Na mesma tabela é apresentado ainda, numa escala desenvolvida para o efeito, de um (pouco importante) a três (muito importante), o grau de importância que cada critério assume na visualização dos diagramas esquemáticos. A referida escala foi elaborada por observação dos diagramas esquemáticos, transmitindo apenas um parecer e não factos concretos e por isso é susceptível de discordâncias.

| | **Algoritmo de *Sugiyama*** | **Algoritmo Ortogonal** | **Algoritmo *Force-based*** | **Importância para os diagramas esquemáticos** |
|---|---|---|---|---|
| **Simetria** | | | | 1 |
| **Planaridade** | X | X | | 3 |
| **Linearidade** | X | X | | 2 |
| **Uniformidade de Nós** | X | | X | 2 |
| **Uniformidade de Arestas** | X | | X | 2 |

| | | | | |
|---|---|---|---|---|
| **Manter sentido do fluxo** | X | | | 2 |
| **Ortogonalidade das arestas** | | X | | 3 |

**Tabela 1 - Critérios estéticos considerados por cada algoritmo de disposição estudado e respectiva importância para os diagramas esquemáticos**

Atendendo a que a probabilidade de cruzamentos entre arestas e até vértices seria relativamente alta, o que é inadmissível em diagramas esquemáticos, é de descartar a utilização de um algoritmo de princípios semelhantes ao *force based* apresentado no estado da arte. [16]

Além do mais, a execução do algoritmo de *Sugiyama* é bem mais célere. Estudos comparativos entre o algoritmo de *Sugiyama* e o algoritmo de arrefecimento simulado (princípio e resultados semelhantes aos obtidos utilizando algoritmos *force based*) comprovaram que para um mesmo diagrama, o algoritmo de *Sugiyama* levou cerca de vinte e cinco segundos a dispor os componentes enquanto que o de arrefecimento simulado demorou aproximadamente duas horas até atingir o resultado final. [16]

Subsiste no entanto a dúvida entre utilizar um algoritmo de disposição hierárquica ou de disposição ortogonal, nomeadamente o apresentado no estado da arte. Como entraves iniciais, o algoritmo ortogonal descrito tem o problema de não considerar a equidistância entre arestas. Inversamente, o algoritmo de *Sugiyama* procura manter um balanceamento ideal entre vértices e arestas com distâncias de valor próximo [15]. No entanto, teria a vantagem de automaticamente garantir ortogonalidade, não sendo necessário nenhum passo adicional para tornar as arestas ortogonais, contrariamente ao que acontece com a utilização do algoritmo de *Sugiyama.*

O estudo centrou-se, no entanto, na possibilidade de utilizar o algoritmo de *Sugiyama* na geração automática de ramos de diagramas esquemáticos, por parecer a abordagem mais lógica a tomar de acordo com os dados apresentados, não sendo no entanto de descartar uma análise futura seguindo metodologia semelhante utilizando o algoritmo de disposição ortogonal descrito.

## 4.4 Geração de Ramos de Diagramas utilizando o algoritmo de *Sugiyama*

Neste capítulo propõe-se uma metodologia para dispor os elementos de ramos de diagramas esquemáticos utilizando o algoritmo de *Sugiyama*.

Como foi referido, existem problemas que não foram considerados pelo autor do algoritmo mas que são introduzidos pelas especificidades dos diagramas, nomeadamente o factor conectividade entre terminais de equipamentos, o que induz regras que não se colocariam em grafos comuns. É assim importante, não apenas descrever os passos do algoritmo de *Sugiyama*, mas também as modificações a operar.

Este capítulo limita-se a explicar os passos tomados na disposição dos componentes de um ramo, não englobando métodos de introdução dos dados de forma a controlar a hierarquização dos componentes ou a diferenciação entre símbolos compostos e simples. Pressupõe por isso que já se detém a seguinte informação como dados de entrada:

- Conhecimento de todos os nós e respectivos conectores, assim como das ligações entre nós e sobre que conectores (terminais) ocorrem essas ligações.
- Posição relativa dos conectores em cada nó.
- Orientação das arestas nas ligações (grafo dirigido), permitindo estabelecer hierarquização dos nós. O sentido de uma aresta é sempre do nó de origem para o de destino.
- Símbolos compostos já gerados, fazendo com que actuem como qualquer símbolo simples – no fundo, o algoritmo não distingue entre símbolos simples e compostos, devendo estes ser gerados previamente.
- A distância mínima que deve existir entre dois nós.

O diagrama apresentado na figura 44 ajuda a compreender melhor os dados de entrada necessários, bem como os resultados que devem ser devolvidos pelo algoritmo, após a sua aplicação.


**Figura 44 - Dados de entrada e de saída do algoritmo aplicado na disposição de componentes**

Os resultados esperados após a aplicação do algoritmo são os seguintes:

- Para cada nó devem ser devolvidas, ordenadamente, as transformações que devem ser operadas neste na disposição final. As transformações aplicáveis são rotação sobre si

mesmo (mediante determinado ângulo) e inversão (apenas horizontal já que a inversão horizontal após rotação permite inversão vertical).
- A nova altura/largura tomada por cada nó, dados que poderão ser invertidos como resultado de uma rotação.
- A localização absoluta de cada nó, mediante coordenada vertical e horizontal. Essa localização pode ser dada usando o centro de um nó ou outro ponto de referência.
- Para cada conector existente em determinado nó, deve ser devolvida a sua posição final relativamente ao nó de que faz parte. Esta localização relativa é modificada apenas se ocorrerem rotações ou inversões.
- Para cada aresta devolvida devem ser indicadas as suas dobras, através da coordenada horizontal e vertical que define a sua localização. As dobras devem estar ordenadas do nó de origem para o de destino.

A metodologia a utilizar neste capítulo será, para cada um dos passos propostos por *Kozo Sugiyama*, começar por apresentar as metodologias previstas pelo seu algoritmo de *layout* e seguidamente as alterações e tomadas de decisão efectuadas de acordo com as especificidades dos diagramas em estudo. Serão também apresentados passos adicionais, tais como ortogonalização das arestas, questão não prevista no algoritmo de *Sugiyama*.

### 4.4.1 Tornar o grafo dirigido acíclico (Passo 1)

O primeiro passo do algoritmo é garantir que o grafo a dispor é acíclico. Este passo pode por isso ser ignorado se houver a certeza que as arestas introduzidas nos dados de entrada não contêm ciclos. [15]

Veja-se como exemplo a figura 45 que representa um grafo que contém um ciclo, sendo necessário inverter pelo menos uma aresta para o remover. Neste caso a forma mais simples de garantir um grafo acíclico é inverter qualquer aresta, por exemplo a aresta que liga os nós 3 e 1.

O objectivo deste passo do algoritmo passa por **transformar** qualquer **grafo dirigido** num **grafo dirigido acíclico removendo todos os ciclos existentes**. [15] Para tal pode ser necessário inverter a direcção de algumas arestas. [15]

Este passo irá garantir que todas as arestas do grafo irão ter o mesmo fluxo o que é necessário na sua hierarquização. [15]


**Figura 45 - Grafo com ciclo (esquerda) e com ciclo removido (direita)**

Num grafo dirigido **G,** um conjunto **R** de arestas que quando invertidas tornam o grafo **G** acíclico denomina-se conjunto das arestas de *feedback*. Denomine-se o grafo **G'** como o resultado da inversão das arestas pertencentes a um conjunto **R** no grafo **G** que o torna acíclico. O objectivo é encontrar um conjunto **R** e consequentemente **G'** através da inversão das arestas desse conjunto. O grafo **G'** deve ser utilizado no processo de *layout*, sendo que no final as arestas são repostas como estavam originalmente no grafo **G**. [15]

Na verdade podem existir vários conjuntos de arestas de *feedback*, no entanto, é preferível que este conjunto seja o mais pequeno possível, já que estas arestas na representação final irão ter sentido oposto ao do fluxo, diminuindo a visibilidade da representação. Assim, o tamanho do conjunto **R** escolhido deve ser o menor possível. [15]

O problema de encontrar um conjunto mínimo das arestas de *feedback* foi, no entanto, indicado como sendo NP-Completo. [17] Apesar disso existem algumas heurísticas que podem ser utilizadas para obter um conjunto de arestas **R,** que quando invertidas criam um grafo acíclico, não garantindo, contudo, soluções ideais. Algumas destas encontram-se descritas de seguida.

### Pesquisa em profundidade

Uma das formas mais simples de encontrar um conjunto de arestas de *feedback* é utilizar uma pesquisa em profundidade (abreviando DFS, do inglês *depth first search*) ao grafo dirigido **G** que se pretende acíclico.

O método consiste em seleccionar um vértice do grafo **G** para iniciar a pesquisa e a partir do mesmo efectuar uma busca aos restantes vértices conectados a este, sendo que sempre que for encontrada uma aresta que ligue a um vértice que já faça parte do caminho da pesquisa realizada, então a aresta que liga a esse vértice é adicionada ao conjunto das arestas de *feedback*. Se a pesquisa iniciada num dado vértice for terminada e restarem vértices por visitar, deverá ser retomada num desses vértices até que não reste nenhum nó por visitar. [18] Cada um dos vértices onde é iniciada uma pesquisa pode ser denominado raiz da pesquisa.

Assim, numa pesquisa em profundidade, começa por se pesquisar sempre os descendentes de um nó até que um destes não tenha qualquer descendente ou volte a um ponto já visitado (neste caso a aresta que liga ao ponto anterior deve ser adicionada ao conjunto **R** das arestas de *feedback*). [18] Durante a pesquisa pode começar-se pelos nós mais à direita ou mais à esquerda – pesquisa em profundidade à esquerda ou à direita – sendo que nos exemplos seguintes será utilizada pesquisa em profundidade à esquerda.

Veja-se o diagrama da figura 46, considerando-se uma pesquisa em profundidade à esquerda iniciada no vértice **a** e posteriormente retomada no vértice **g**, já que a pesquisa iniciada em **a** não permitiu atingir os nós **g** e **f**.

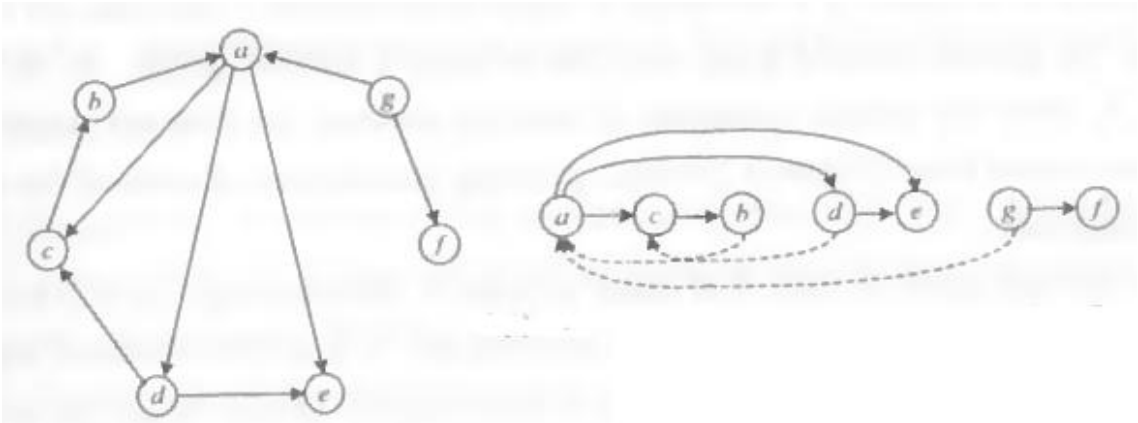


**Figura 46 - Pesquisa em profundidade a grafo G [15]**

Verifique-se a ordem de pesquisa em profundidade no grafo, partindo sempre do descendente mais à esquerda de cada nó, até ser atingido um ponto anterior da pesquisa como acontece nas arestas **A(b, a)**, **A(d, c)** e **A(g, a)** ou até um vértice não ter qualquer descendente como acontece com os vértices **e** e **f**. O conjunto **R** obtido, ou seja, as arestas a inverter de forma a obter um grafo acíclico **G'** deveriam ser então as três indicadas anteriormente, por reconduzirem a um ponto anterior. Note-se, no entanto, que a aresta **A(g, a)** não leva a qualquer ciclo, não sendo por isso necessário adicioná-la ao conjunto **R**. Tal facto deve-se a que apesar do nó **a** já ter sido

visitado anteriormente, tal ter sucedido numa pesquisa iniciada numa raiz anterior (o próprio vértice **a**) e não na que teve como começo o vértice **g** o que não constitui qualquer ciclo. Assim, embora possa levar mais tempo de processamento, seria melhor computar estritamente os componentes ligados a **a** inicialmente e posteriormente a **g**. [15] O conjunto de arestas de *feedback* seria assim constituído unicamente pelas arestas **A(b, a)** e **A(d, c)**. [15]

A escolha das raízes para iniciar ou reiniciar uma pesquisa em profundidade pode ser determinada de várias formas. Uma das formas possíveis é escolher o vértice por visitar que surja menor número de vezes como vértice de destino, neste caso o vértice **g** (surge apenas como vértice de origem, para **a** e **f**).

A figura 47 apresenta um diagrama de actividade que demonstra todo o procedimento descrito anteriormente.

Antes de aplicar a pesquisa em profundidade, deve garantir-se que o grafo não possui laços, ou seja, arestas que ligam um vértice a si mesmo. Para tal basta retirar qualquer laço existente e reintroduzi-lo no final.

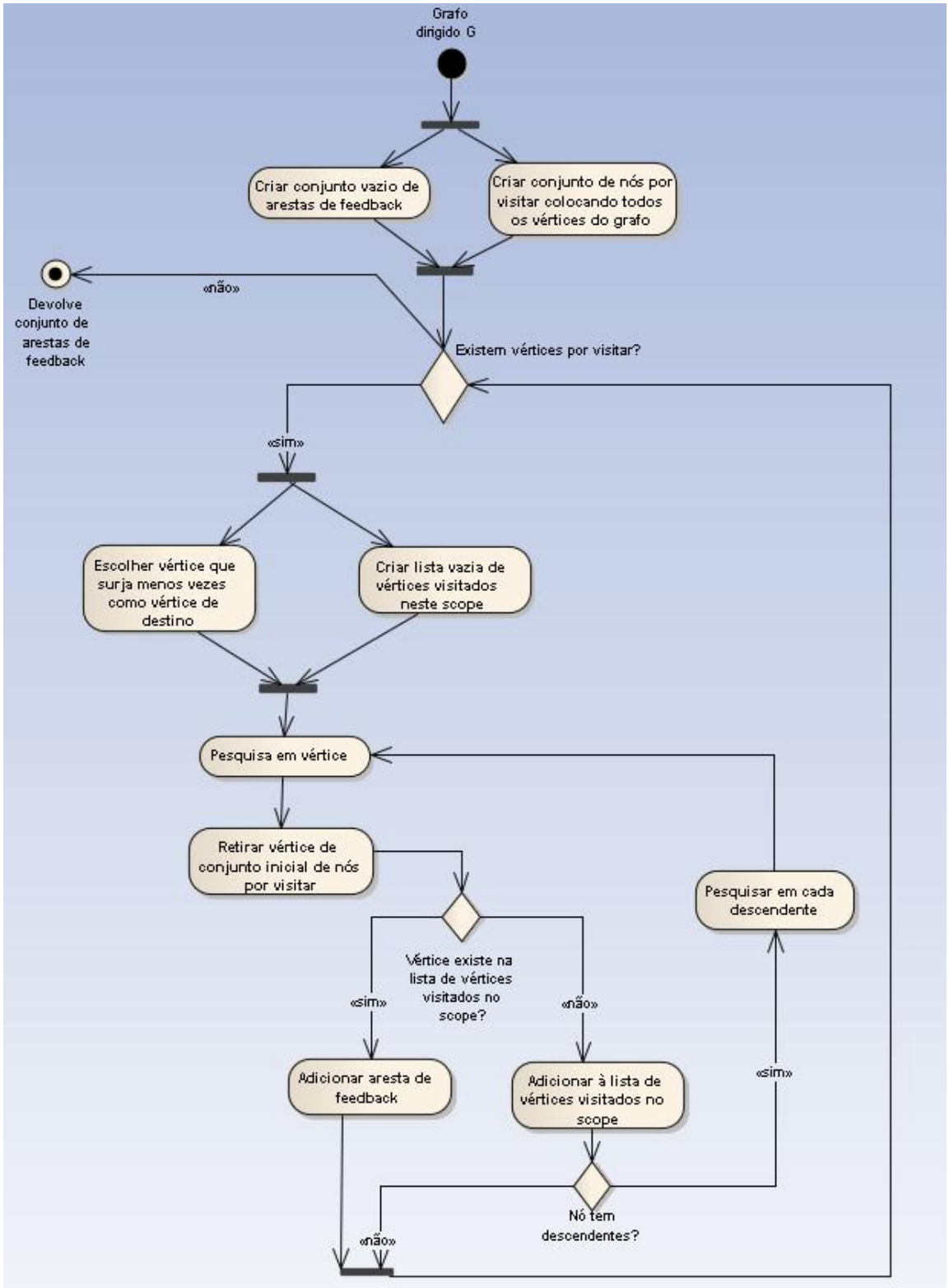


**Figura 47 - Diagrama de actividade de uma pesquisa em profundidade a um grafo dirigido para determinar o conjunto de arestas de *feedback***

**Ordenação de vértices baseada no número de descendentes**

O método consiste em partir do princípio que os vértices que têm maior número de descendentes (surgem mais vezes como nós de origem) têm mais tendência a aparecer no topo dos grafos. Assim, tenta-se atribuir um número **λ(v)** a cada vértice tanto menor quanto maior o

número de descendentes que tiver. Os nós serão posteriormente alinhados por ordem crescente desse valor. [15]

Na prática, deve seleccionar-se primeiramente o vértice do grafo que tiver maior número de descendentes e atribuir-lhe **λ(v) = 1**. Por exemplo, no grafo da figura 48 é o vértice **a**. Seguidamente deve remover-se esse vértice do grafo (subtrair o vértice) e repetir o processo até que não reste qualquer nó no grafo.

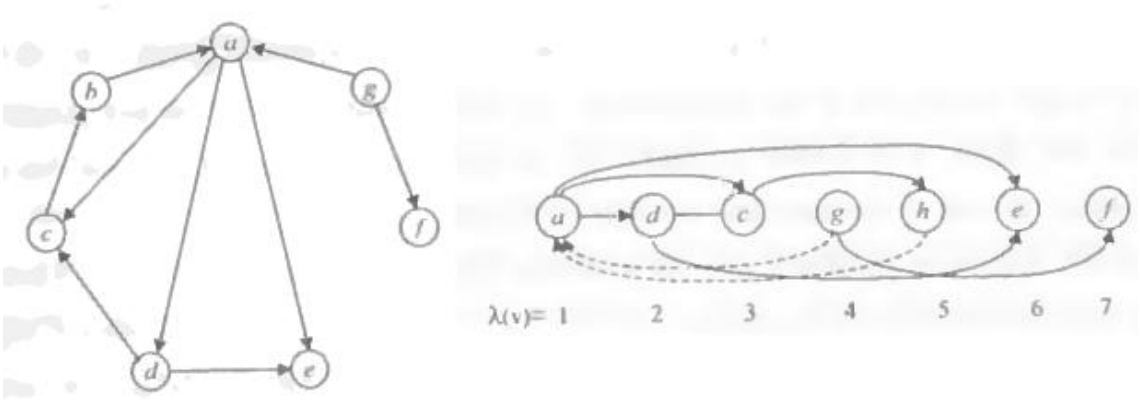


**Figura 48 - Ordenação dos nós de um grafo baseada no número de vértices descendentes [15]**

A figura 48 mostra os passos da heurística apresentada, mostrando a ordem dada aos nós do grafo consoante o número de descendentes de um dado vértice após cada remoção.

No final as arestas que tiverem um nó de destino com **λ(v)** menor que o nó de origem, devem ser adicionadas ao conjunto de arestas de *feedback*. O diagrama de actividade da figura 49 apresenta todo o processo.

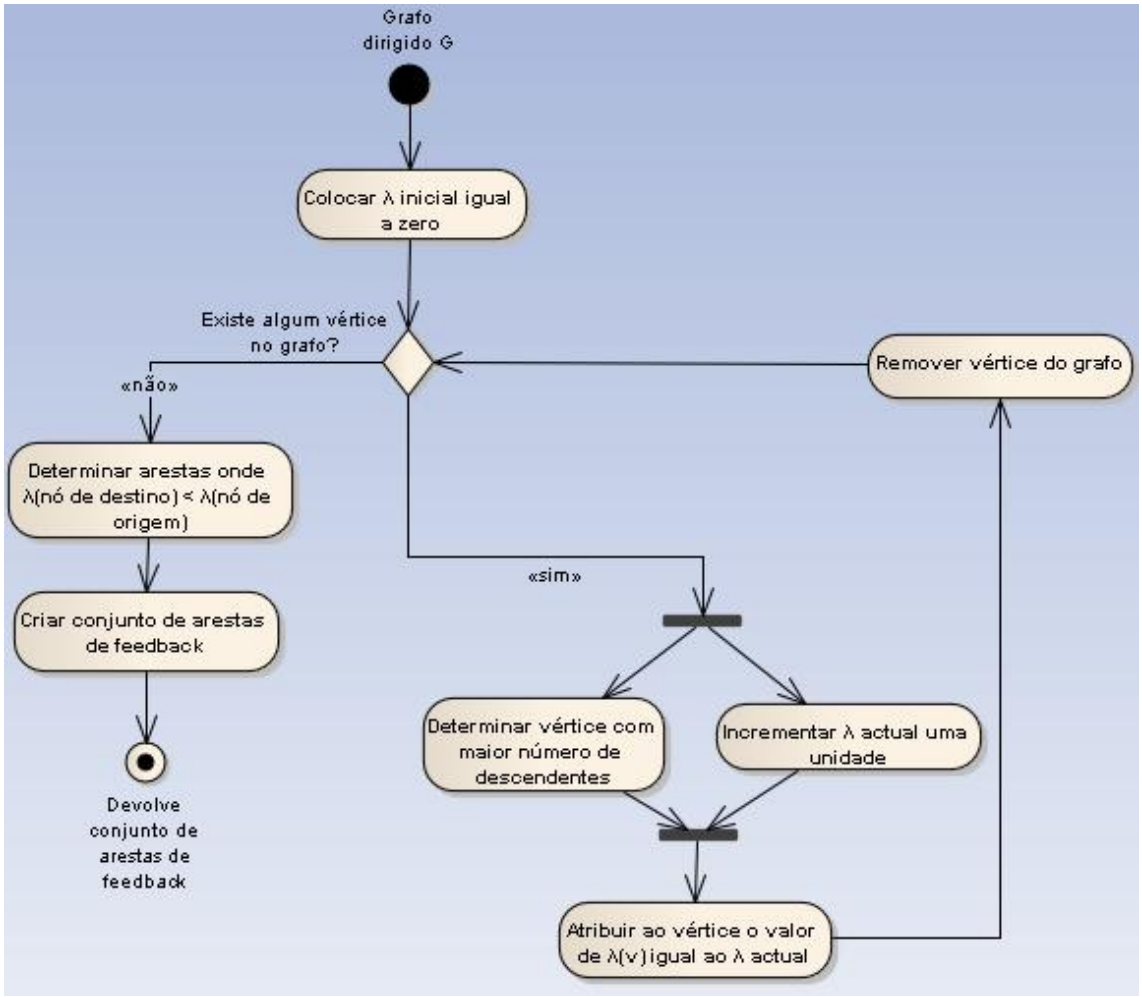


**Figura 49 - Diagrama de actividade do método de ordenação de vértices baseado no número de descendentes de cada nó**

**Pesquisa utilizando método de divisão e conquista**

Os métodos de divisão e conquista são metodologias que dividem o problema em partes, computando cada uma destas individualmente o que por vezes devolve melhores resultados. [20]

Este tipo de método no entanto é de descartar tendo em conta que é significativamente mais lento que os já apresentados, embora consiga melhores resultados, podendo diminuir em 20% o número de arestas contrárias ao fluxo. No entanto o tempo de computação é cerca de quatro vezes maior que o método de pesquisa em profundidade e uma vez que os diagramas esquemáticos podem atingir grandes dimensões, o tempo de computação é factor importante. [19]

**Método de remoção de ciclos a utilizar**

Os métodos de pesquisa em profundidade e ordenação de vértices baseada no número de descendentes apresentam resultados satisfatórios e tempo de processamento semelhantes. [15]

No entanto, dado que o segundo método apresenta resultados melhores que o primeiro, já que num maior número de casos consegue obter um conjunto de arestas de *feedback* de tamanho menor [15], este método deverá ser o melhor a considerar.

O método utilizado em testes efectuados foi, no entanto, o de pesquisa em profundidade por ser o de implementação à partida mais natural e simples, não sendo contudo de crer que os resultados finais da disposição dos diagramas sejam muito alterados com este factor.

### 4.4.2 Determinar a camada de cada vértice (Passo 2)

Após se garantir a obtenção de um grafo acíclico, podem então iniciar-se os passos de disposição dos nós de um grafo no plano. [15]

O primeiro desses passos é determinar a camada vertical que cada vértice deve ocupar. [15] A camada vertical é definida por um nível de profundidade, tal como apresentado na figura 50.

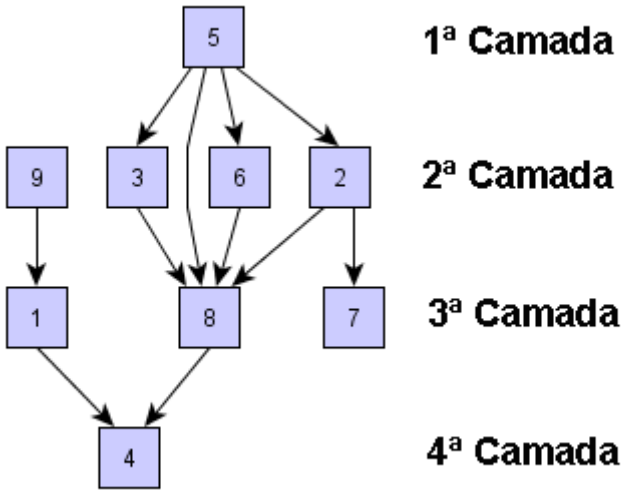


**Figura 50 - Camadas de um grafo disposto hierarquicamente**

As seguintes restrições devem ser tidas em conta na distribuição dos vértices pelas camadas [15]:

1. A cada nó corresponde unicamente uma camada e cada camada deve ter um ou mais vértices.
2. Um determinado vértice deve estar sempre numa camada maior (mais abaixo) que um antecessor seu.
3. O número total de camadas deve ser minimizado.
4. Um nó deve estar na maior camada (mais abaixo) que lhe permita cumprir as restrições anteriores.

Convém ainda dizer que um grafo pode ter uma ou mais raízes. Considera-se raiz de um grafo **G** qualquer vértice que não tenha nenhuma aresta dirigida para este. Por exemplo no diagrama da figura 50 os nós 5 e 9 são raízes. Qualquer grafo acíclico, como os tratados neste passo, apresenta no mínimo uma raiz.

Assente no algoritmo descrito por *Kozo Sugiyama* [15], que propunha um modelo baseado no caminho mais longo para efectuar este passo, procurando manter as arestas com o menor comprimento possível (restrição 4), foi desenvolvida uma heurística que permite distribuir os vértices pelas camadas cumprindo as restrições anteriores. A metodologia sugerida encontra-se de seguida.

#### Heurística proposta para atribuir uma camada a cada vértice

O método implica começar por criar listas de todos os caminhos possíveis que se pode tomar num dado grafo acíclico iniciados nas raízes. Tomando como exemplo o grafo da figura 50, e efectuando uma pesquisa em profundidade à esquerda, as listas de caminhos obtidas seriam as seguintes:

| Posição na lista | Lista 1 | Lista 2 | Lista 3 | Lista 4 | Lista 5 |
|---|---|---|---|---|---|
| 1 | 9 | 5 | 5 | 5 | 5 |
| 2 | 1 | 3 | 8 | 6 | 2 |
| 3 | 4 | 8 | 4 | 8 | 7 |
| 4 | - | 4 | - | 4 | - |
| **Comprimento** | **3** | **4** | **3** | **4** | **3** |

**Tabela 2 - Caminhos possíveis do grafo da figura 50**

Repare-se que uma nova lista surge a cada bifurcação de um nó (quando tem mais que um descendente).

Seguidamente, seria então possível atribuir uma camada a cada nó, escolhendo a posição máxima que cada vértice ocupa nas listas em que se encontra (caminho máximo), tal como apresentado de seguida.

| Vértice | Camada |
|---|---|
| 1 | **2** |
| 2 | 2 |
| 3 | 2 |
| 4 | 4 |
| 5 | 1 |
| 6 | 2 |
| 7 | 3 |
| 8 | 3 |
| 9 | **1** |

**Tabela 3 - Camadas atribuídas a cada nó**

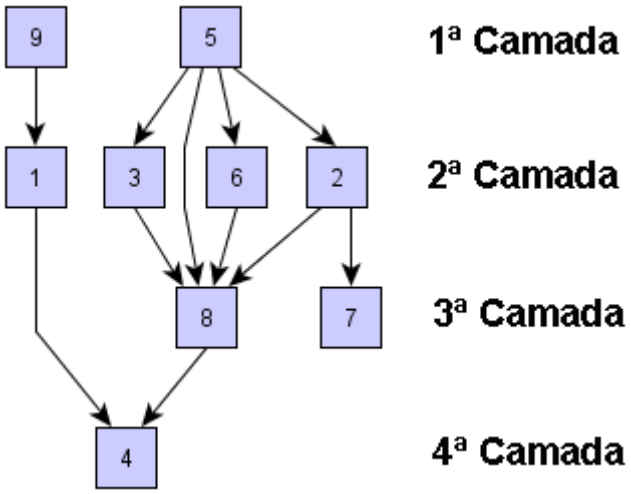


**Figura 51 - Camadas atribuídas a cada nó segundo a tabela anterior**

No entanto subsiste ainda o problema da quarta restrição nem sempre ser cumprida, dado que para tal, no exemplo apresentado o vértice 1 deveria estar na terceira camada (veja-se a figura 51) e consequentemente o vértice 9 deveria estar na segunda. Assim, resta fazer um ajuste quando casos como este surgirem, uma etapa que compacte os nós, garantindo o menor tamanho possível das arestas.

Uma forma de contornar o problema apresenta-se de seguida:

- Para cada nó de camada **N** com **pelo menos um descendente**, determinar a camada em que se encontra o seu descendente com menor camada (poderão haver vários com a mesma camada).
- Se a camada **ND** desse descendente for superior a **N + 1**, então mudar a camada **N** do vértice tratado para **ND – 1** (camada imediatamente antes do seu descendente de maior camada)
- Modificando-se a camada de um vértice deverão ser revistos os valores das camadas dos seus antecessores.

Continuando com o exemplo anterior, existiria apenas um vértice de camada **N** em que a camada do seu descendente de maior camada seria superior a **N + 1**, trata-se do vértice 1. De facto este encontra-se na segunda camada enquanto que o seu descendente de menor camada (o vértice 4) se encontra na quarta. Por essa razão, o valor da sua camada deve ser modificado para o imediatamente antes da camada do vértice quatro, ou seja a terceira camada. Modificando-se o valor do vértice 1, devem ser revistos todos os nós antecessores (neste caso apenas o 9) e pelo mesmo procedimento, este deveria passar para a segunda camada. Além dos vértices 1 e 9, nenhum outro vértice mudaria a sua camada.

Após esta última etapa, estaria então concluído o processo de atribuição de camadas aos nós, obtendo-se para o grafo exemplo, o resultado final expresso na figura 50.

### Determinar rotação de cada vértice

Dado a especificidade já referida dos diagramas esquemáticos tratados, além de determinar a camada de cada vértice é necessário também definir a rotação que cada nó deve tomar de forma a efectuar as suas ligações. Este processo pode ser efectuado mal se conheçam as camadas de cada vértice.

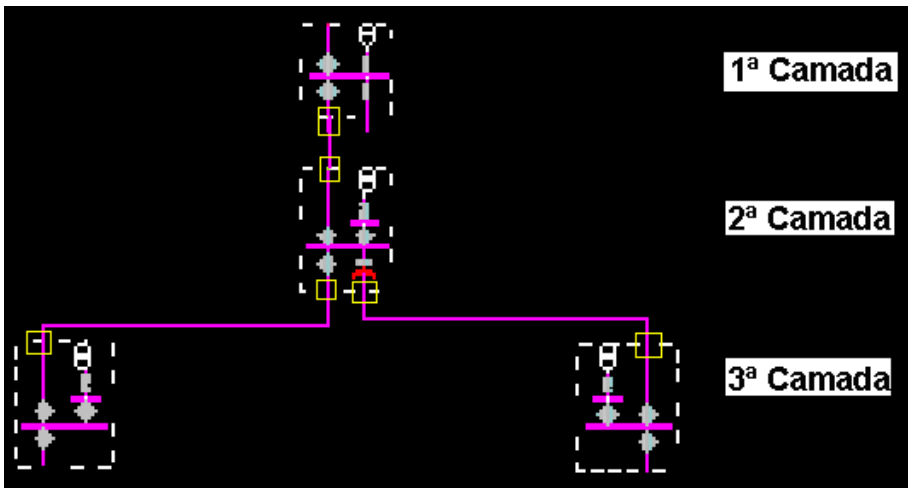


**Figura 52 - Rotação de cada vértice de acordo com as ligações dos seus conectores**

A figura 52 representa parte de um grafo esquemático com três camadas. Repare-se que cada vértice está com um ângulo de rotação de acordo com as ligações dos seus conectores (assinalados a amarelo). Por exemplo, se os conectores têm ligações a camadas superiores, o nó toma uma rotação de acordo com esse factor.

O procedimento para determinar a rotação que se sugere consiste em verificar para cada conector com ligação, se liga a uma camada superior ou inferior e procurar que a rotação a efectuar ao nó permita o maior número de conectores possível orientados para a camada a que ligam (para cima ou para baixo) e o menor número possível de ligações de conectores que se

encontrem na extremidade oposta do nó relativamente à camada a que liguem (por exemplo ligarem a um vértice de camada acima e estarem na extremidade inferior do nó).

Veja-se como exemplo o vértice da segunda camada do exemplo da figura 52. Existem três conectores com ligações neste vértice. Dois vértices na mesma extremidade do nó têm ligações para uma camada que se encontra abaixo e outro que se encontra na extremidade oposta do nó para uma camada acima. A rotação a efectuar ao nó foi feita de forma que permitisse um maior número de conectores na extremidade mais favorável à ligação, neste caso o número de conectores com a orientação mais favorável é três, o que significa que todos os conectores estão na extremidade mais adequada do nó, de acordo com a sua ligação.

Uma nota para referir, que em caso de não ser possível que todos os conectores fiquem voltados para a camada a que liguem, é preferível como critério de "desempate", que estes fiquem colocados lateralmente do que na extremidade oposta do nó.

### 4.4.3 Definir a ordem dos vértices em cada camada (Passo 3)

O terceiro passo do algoritmo de *Sugiyama* passa por definir a ordem relativa dos vértices em cada camada, determinada no passo anterior. [15] O objectivo é que haja o **mínimo de cruzamentos** possível entre arestas.

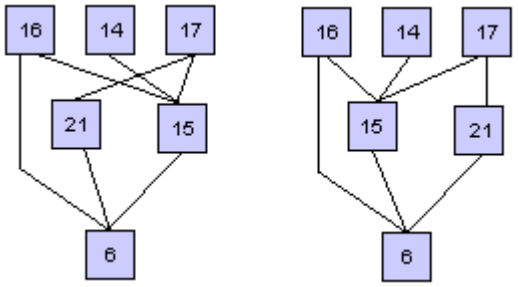


**Figura 53 -Vértices respectivamente não ordenados e ordenados num grafo**

Veja-se o exemplo da figura 53, que representa duas vezes o mesmo grafo, numa delas com os vértices ordenados o que evita cruzamento entre arestas. A imagem demonstra a necessidade de executar um passo que defina a ordem dos vértices.

Este capítulo servirá para demonstrar como pode ser determinada a ordem dos vértices em cada camada, processo que compreende algumas etapas intermédias.

#### Introdução de vértices falsos

Visto que podem existir arestas que atravessem várias camadas num mesmo grafo, como por exemplo a aresta que liga os vértices 16 e 6 do grafo da figura 53, antes de definir a ordem relativa dos vértices em cada camada é necessário introduzir nós "falsos" sempre que tal aconteça. [15]

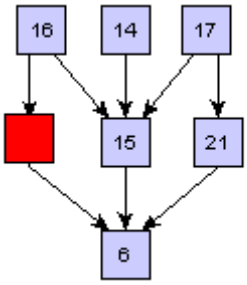


**Figura 54 - Vértice "falso" adicionado a aresta que atravessa consecutivamente mais que uma camada**

A introdução de vértices "falsos", que se passarão a designar vértices *dummy*, deverá ocorrer sempre que uma aresta atravesse mais que uma camada [15], tal como mostrado na figura 54. Nesse caso um vértice deve ser adicionado em cada camada intermédia.

A figura 55 apresenta outro exemplo em que é necessário adicionar vértices *dummy*.

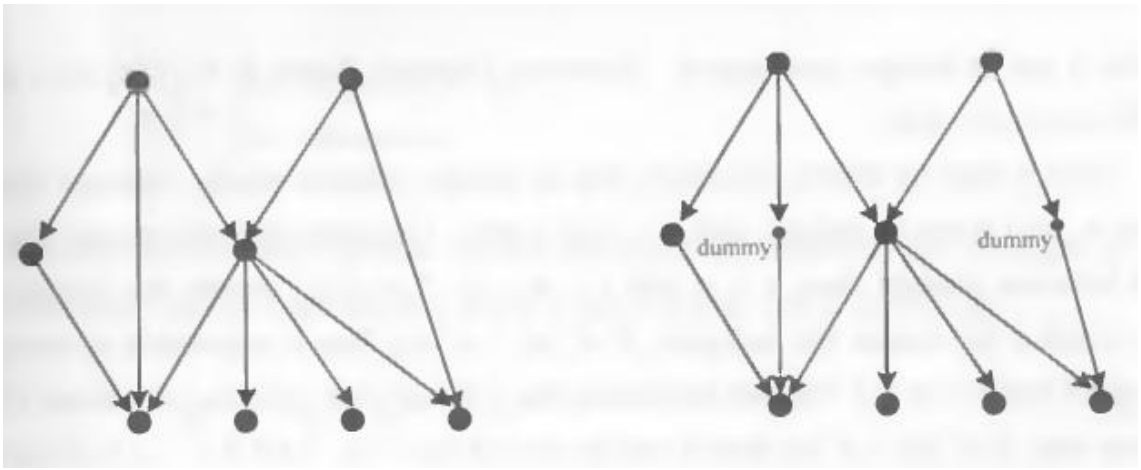


**Figura 55 - Adição de vértices *dummy* [15]**

No final os nós "falsos" adicionados funcionarão como dobras nas arestas entre os vértices. [15]

Este passo é indispensável antes de efectuar a ordenação relativa dos vértices dentro das camadas, já que durante essa fase, os vértices *dummy* irão ser tratados como quaisquer outros nós. [15]

### Ordenação entre duas camadas utilizando heurística do centro de massas

O passo base da ordenação dos vértices é operado entre duas camadas consecutivas. [15] O procedimento consiste em aplicar o algoritmo de ordenação entre cada par de camadas, modificando a ordem dos vértices nestas, de forma a existirem o mínimo de cruzamentos possíveis entre arestas. [15]

Este passo intermédio, necessário para efectuar a ordenação dos vértices nas camadas é descrito nesta secção.

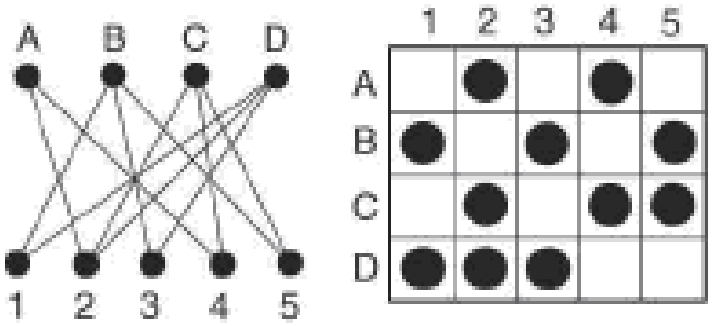


**Figura 56 - Criação da matriz de adjacência num grafo com duas camadas [21]**

O primeiro procedimento a tomar é criar a matriz de adjacência entre duas camadas, tal como exemplificado na figura 56. Para tal devem colocar-se os **vértices da camada de nível superior representados nas linhas** e os **da camada de nível inferior representados nas colunas** da matriz, tal como exemplificado. A matriz de adjacência do grafo de duas camadas apresentado na figura 56 seria a seguinte:

**0 1 0 1 0**
**1 0 1 0 1**
**0 1 0 1 1**
**1 1 1 0 0**

Tendo a matriz de adjacência, deve então mudar-se a ordem das linhas, se for pretendido mudar a ordem dos vértices da camada superior, ou a ordem das colunas para mudar a ordem dos vértices da camada inferior. A troca entre linhas ou colunas na matriz representa assim, para efeito de ordenação, uma troca das posições entre vértices. [16]

A heurística a seguir para ordenar os vértices na camada superior passa por colocar as **linhas ordenadas de forma crescente consoante o valor do seu centro de massas**. Por outro lado se for pretendido ordenar os vértices da camada inferior devem ser colocadas as **colunas de forma crescente de acordo com o valor do seu centro de massas**. [16]

A fórmula para calcular o centro de massas de um vector, que deve ser usada na ordenação das linhas e colunas da matriz de adjacência encontra-se de seguida. Refira-se que para aplicar esta fórmula é necessário dar um sentido às linhas e colunas (de forma a que actuem como vectores), o sentido é assim de cima para baixo (colunas) e da esquerda para a direita (linhas). [16]

Centro de massas **Dy** de um vector binário **y = (y1, …, ym)** [16]:

$$\frac{\sum_{j=1}^{m} j * y_j}{\sum_{j=1}^{m} y_j}$$

**Expressão 3 - Centro de massas de um vector [16]**

- ***j*** representa a posição no vector
- $y_j$ é o valor que o vector toma na posição ***j*** (zero ou um)

A tabela seguinte apresenta para a matriz de adjacência dada como exemplo anteriormente, o valor do centro de massas de cada uma das linhas e colunas.

| **Vértice** | 1 | 2 | 3 | 4 | 5 | **Centro de massas da linha** |
|---|---|---|---|---|---|---|
| A | 0 | 1 | 0 | 1 | 0 | (2 + 4) / 2 = 3 |
| B | 1 | 0 | 1 | 0 | 1 | (1+3+5) / 3 = 3 |
| C | 0 | 1 | 0 | 1 | 1 | (2 + 4 +5) / 3 ≈ 3,67 |
| D | 1 | 1 | 1 | 0 | 0 | (1 + 2 + 3) / 3 = 2 |
| **Centro de massas da coluna** | (2 + 4) / 2 = 3 | (1 + 3 + 4) / 3 ≈ 2,67 | (2 + 4) / 2 = 3 | (1 + 3) / 2 = 2 | (2 + 3) / 2 = 2,5 | |

**Tabela 4 - Centros de massas de cada uma das linhas e colunas da matriz de adjacências**

Aplicando a heurística às linhas da matriz, ou seja, modificando a ordem dos nós da camada superior, para ordenar as linhas de forma crescente de acordo com o centro de massas, obtém-se a seguinte matriz (veja-se os valores obtidos na tabela):

**D → 1 1 1 0 0**
**B → 1 0 1 0 1**
**A → 0 1 0 1 0**
**C → 0 1 0 1 1**

O resultado obtido significa que deve ser modificada a ordem dos vértices da camada superior de **A**, **B**, **C**, **D** para **D**, **B**, **A**, **C** pois tal deverá diminuir o número de cruzamentos.

O mesmo procedimento pode também ser aplicado aos vértices da camada inferior, devendo, para tal, colocar-se de forma crescente pelo seu centro de massas a ordem das colunas, de acordo com os valores calculados na tabela seguinte, criada seguindo os mesmos princípios da anterior mas após a modificação da ordem das linhas da matriz (e consequentemente dos vértices da camada superior).

| **Vértice** | 1 | 2 | 3 | 4 | 5 | **Centro de massas da linha** |
|---|---|---|---|---|---|---|
| D | 1 | 1 | 1 | 0 | 0 | (1 + 2 + 3) / 3 = 2 |
| B | 1 | 0 | 1 | 0 | 1 | (1+3+5) / 3 = 3 |
| A | 0 | 1 | 0 | 1 | 0 | (2 + 4) / 2 = 3 |
| C | 0 | 1 | 0 | 1 | 1 | (2 + 4 +5) / 3 ≈ 3,67 |
| **Centro de massas da coluna** | (1 + 2) / 2 = 1,5 | (1 + 3 + 4) / 3 ≈ 2,67 | (1 + 2) / 2 = 1,5 | (3 + 4) / 2 = 3,5 | (2 + 4) / 2 = 3 | |

**Tabela 5 - Centros de massas de cada uma das linhas e colunas da matriz de adjacências após alteração da ordem dos vértices da camada superior**

Os valores obtidos na nova tabela, permitem já modificar a ordem das colunas da matriz de adjacência, obtendo-se a seguinte matriz:

**1 3 2 5 4**
**↓ ↓ ↓ ↓ ↓**
**1 1 1 0 0**
**1 1 0 1 0**
**0 0 1 0 1**
**0 0 1 1 1**

A figura seguinte apresenta o grafo da figura 56 com os vértices ordenados, de acordo com a matriz apresentada, depois de terem sido colocados por ordem, respeitando os valores de centro de massas calculados nas tabelas anteriores, primeiramente as linhas e depois as colunas.

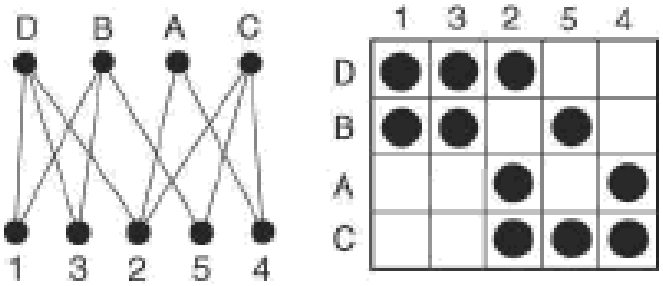


**Figura 57 - Grafo com linhas e colunas ordenadas de acordo com o centro de massas [21]**

Repare-se nos resultados das figuras 56 e 57. Ambas representam o mesmo grafo, a primeira antes de ser aplicada a heurística de ordenação de acordo com o centro de massas e a segunda após a aplicação da heurística. Como pode ser visto, no segundo caso, embora se mantenham alguns cruzamentos entre arestas, o seu número é inferior ao que acorria antes da aplicação da heurística.

### Utilização da heurística do centro de massas em nós com conectores

Também neste passo é preciso ter em conta as especificidades que os conectores de cada nó introduzem. O problema surge pelo facto de ser necessário considerar a sequência numa camada de dois nós que liguem ao mesmo vértice, o que não aconteceria em grafos comuns. Veja-se o exemplo da figura seguinte, em que ocorre um cruzamento entre duas arestas por não se considerar este factor.

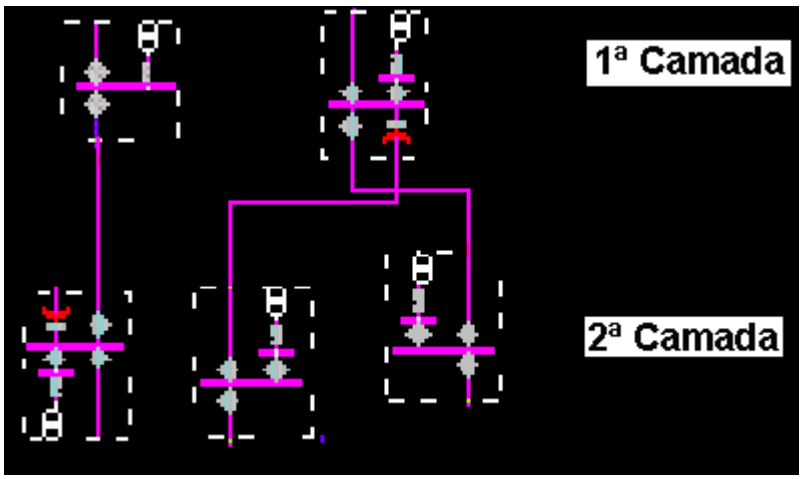


**Figura 58 - Cruzamento entre duas arestas que ligam ao mesmo nó**

Duas arestas ligam ambas ao mesmo nó levando a um cruzamento não considerado pelo método descrito. É necessário, por isso, efectuar algumas alterações que tenham em conta esta possibilidade.

Desde logo a solução deve passar por inverter horizontalmente o vértice com duas ligações (figura 59) ou trocar a ordem dos vértices que ligam aos seus conectores (figura 60). Há por isso a necessidade de fazer algumas modificações na heurística do centro de massas utilizada que

contemple a **inversão horizontal** de nós e **considere a ordem de dois nós que liguem a um mesmo vértice**.

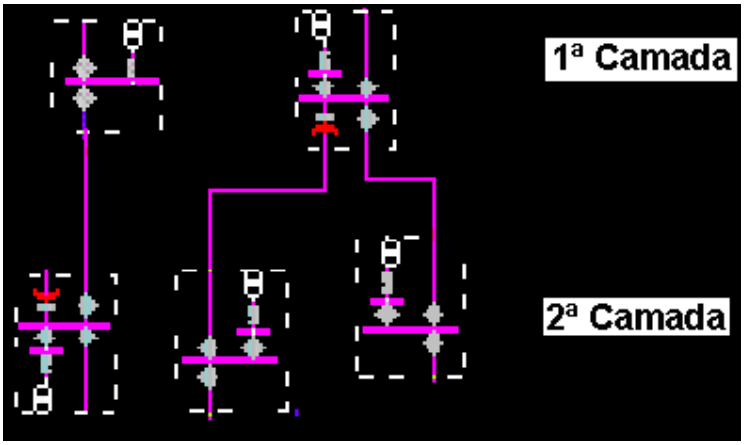


**Figura 59 - Inversão de vértice de forma a evitar cruzamento entre arestas**

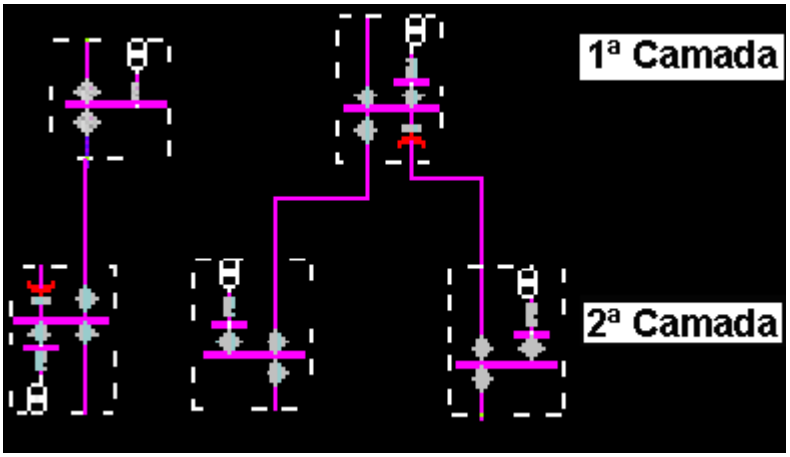


**Figura 60 - Troca entre vértices que ligam ao mesmo nó de forma a evitar cruzamento entre arestas**

A alteração proposta para resolver o problema passa por fazer algumas modificações na matriz de adjacência a criar antes de aplicar a heurística do centro de massas, **considerando na criação da matriz cada conector com ligação como um nó distinto**. No fundo é o mesmo que dizer que a matriz de adjacências passa a considerar os conectores e não os vértices.

Veja-se a figura 61, que apresenta um exemplo de como poderiam ser decompostos os vértices para criar a matriz de adjacência, também apresentada anexada à imagem.

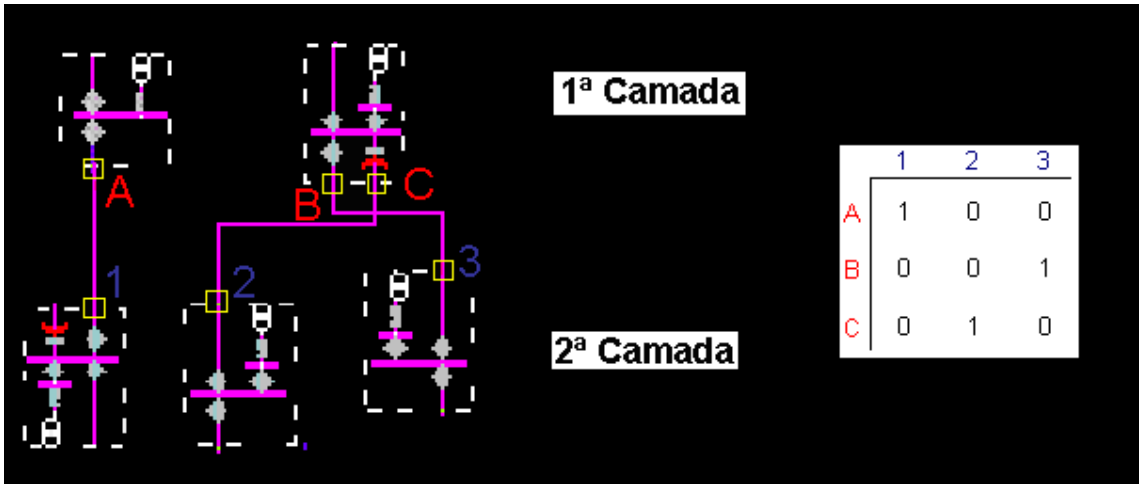


**Figura 61 - Criação de uma matriz de adjacência para utilizar na heurística do centro de massas num diagrama esquemático**

Obtida a matriz, utilizando os conectores em vez dos nós, poderia então ser aplicado o mesmo método descrito anteriormente, tal como se vê na tabela seguinte, em que se calcula o **centro de massas para as colunas** da matriz de forma a **ordenar os conectores da camada inferior**.

| **Conector** | 1 | 2 | 3 | **Centro de massas da linha** |
|---|---|---|---|---|
| A | 1 | 0 | 0 | 1 |
| B | 0 | 0 | 1 | 3 |
| C | 0 | 1 | 0 | 2 |
| **Centro de massas da coluna** | 1 | 3 | 2 | |

**Tabela 6 - Cálculo do centro de massas das colunas de uma matriz de adjacência que considera conectores**

O resultado do cálculo do **centro de massas** de cada uma **das colunas** da matriz determina que a melhor ordenação para a camada inferior do diagrama é a sequência de conectores "**1**, **3**, **2**" (figura 60). Uma vez que os nós dessa camada apenas têm um conector cada, a sequência dos vértices será a mesma que a dos conectores.

Aplicando o mesmo procedimento para a camada de vértices superior e por isso **ordenando as linhas** da matriz de adjacência segundo o seu **centro de massas**, obtém-se a sequência de conectores "**A**, **C**, **B**". Como neste caso os conectores não coincidem com os vértices tal significaria que um dos vértices poderia ser invertido horizontalmente (aquele que tem dois conectores com ligação) de forma a manter a sequência obtida para os conectores (figura 59).

No entanto este princípio tem um problema, que ocorre pelo facto dos **conectores do mesmo vértice terem de manter a sua sequência** (podem apenas reverte-la quando invertem horizontalmente). Como exemplo, veja-se a figura seguinte, que apresenta o mesmo extracto de diagrama usado acima, mas com os vértices dispostos de forma diferente, ao que corresponde outra matriz de adjacência. Na tabela 7 apresenta-se o cálculo dos centros de massas para as linhas e colunas da nova matriz.

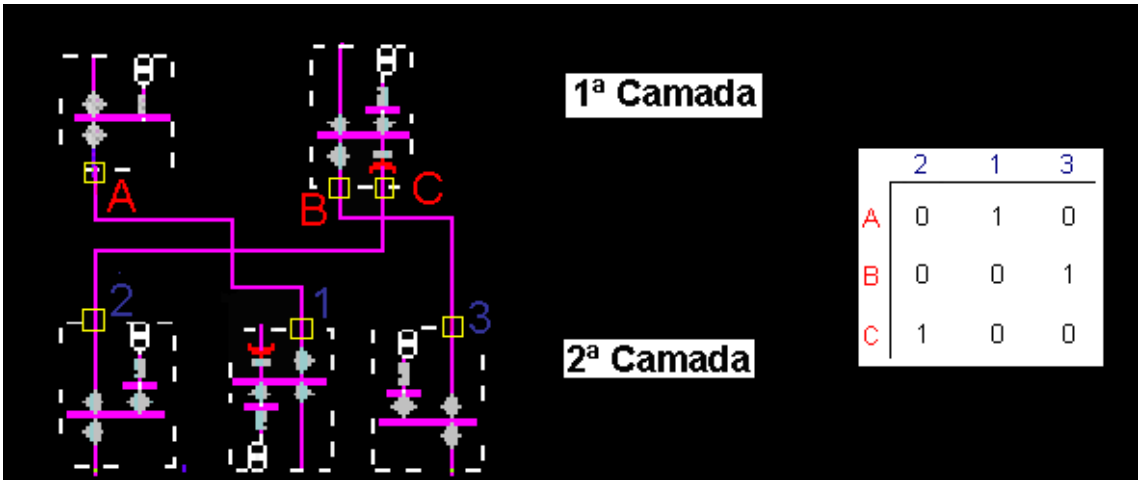


**Figura 62 - Extracto de grafo esquemático e respectiva matriz de adjacência utilizando conectores**

| Conector | 2 | 1 | 3 | Centro de massas da linha |
|---|---|---|---|---|
| A | 0 | 1 | 0 | 2 |
| B | 0 | 0 | 1 | 3 |
| C | 1 | 0 | 0 | 1 |
| **Centro de massas da coluna** | 3 | 1 | 2 | |

**Tabela 7 – Cálculo do centro de massas para nova matriz de adjacência (considerando conectores)**

Neste caso, se fosse pretendido obter a melhor sequência de nós para a camada inferior, daria a sequência de conectores (e a mesma em termos de vértices já que têm apenas um conector com ligação cada) "**1**, **3**, **2**", o que resolveria todos os cruzamentos (figura 60). Mas se se tentar ordenar as linhas da matriz segundo os centros de massas, obtém-se a sequência de vértices para a camada superior "**C**, **A**, **B**". Ora acontece que tal não é possível de representar, já que pelo facto dos vértices **C** e **B** pertencerem ao mesmo nó não podem ter um outro conector (neste caso o conector **A**) entre eles. Desta forma, este caso representaria algo **impossível de apresentar**.

Detectado este problema, a única solução encontrada para o contornar foi **considerar apenas os conectores como entradas na matriz de adjacências em vez dos vértices apenas quando actuam estaticamente**, ou seja, se for pretendido mudar a ordem dos vértices da camada oposta que consta da matriz. Por outras palavras, a **matriz de adjacências** deve ser construída **simultaneamente usando nós,** se for pretendido determinar a ordem desses nós e **usando conectores** para ajudar a determinar a melhor ordem para os nós da outra camada. Na prática quando se pretende utilizar a **heurística do centro de massas nas colunas da matriz devem considerar-se os vértices nas colunas e os conectores nas linhas** e por outro lado quando se pretender usar a **heurística nas linhas devem considerar-se vértices nas linhas e conectores nas colunas**. A figura seguinte poderá ajudar a compreender a forma como deve ser determinada a matriz de adjacências, dependendo da camada a operar.

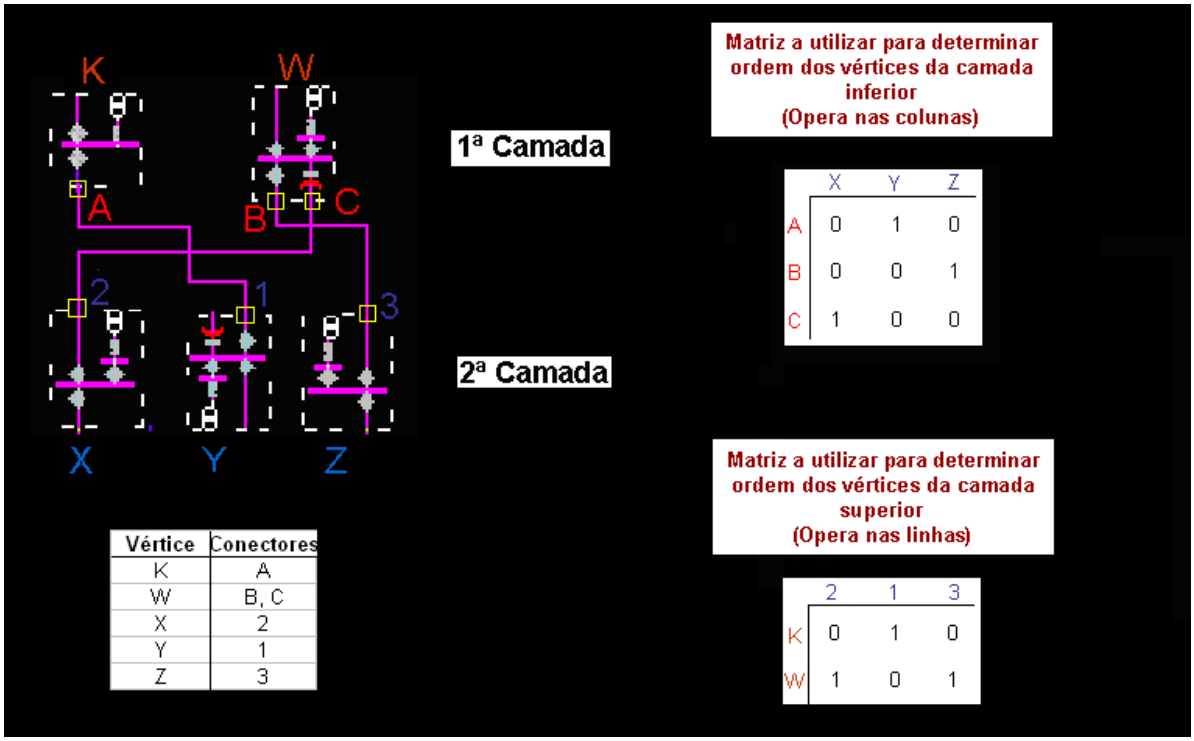


**Figura 63 - Definição da matriz de adjacência a utilizar dependendo da camada sobre a qual se pretende operar**

Claro que este procedimento **leva a que nunca surjam inversões horizontais** de vértices, não utilizando este recurso para evitar cruzamentos. No entanto, tal não invalida que **após a aplicação da heurística se escolha, para cada vértice, a sequência horizontal (inversão ou não inversão) que leve a menos cruzamentos**, bastando para tal uma heurística que detecte o número de cruzamentos existente entre as ligações de um dado equipamento e restantes arestas entre duas camadas e mediante isso, escolha a posição horizontal que leve a um menor número de cruzamentos. Tal heurística deve ser aplicada imediatamente a seguir à do centro de massas.

Voltando a usar o exemplo da figura 63, já se concluiu que operando a heurística do centro de massas aos vértices da camada inferior se consegue resolver todos os cruzamentos do grafo, uma vez que estes têm apenas um conector. No entanto, atente-se na seguinte tabela, obtida considerando vértices nas linhas e conectores nas colunas da matriz de adjacência, operando a heurística do centro de massas sobre a camada superior (linhas da matriz).

| **Conector / Vértice** | 2 | 1 | 3 | **Centro de massas da linha** |
|---|---|---|---|---|
| K | 0 | 1 | 0 | 2 |
| W | 1 | 0 | 1 | (1+3) / 2 = 2 |

**Tabela 8 - Cálculo do centro de massas para nova matriz de adjacência considerando simultaneamente conectores e vértices**

O resultado determina que a sequência de vértices da camada superior deve ser, indiferentemente "**K**, **W**" ou "**W**, **K**", já que possuem o mesmo centro de massas. Operar uma modificação na matriz não traz, por isso, qualquer vantagem neste caso. Continuam no entanto a existir **dois cruzamentos**, sendo que um deles poderia ser evitado invertendo o vértice **W**. Por essa razão, é vantajoso após cada aplicação da heurística do centro de massas, verificar para cada vértice com mais de um conector com ligação se o facto de o inverter leva a uma diminuição no número de cruzamentos e se for esse o caso, aplicar a inversão horizontal do nó. Esse passo deve

funcionar de forma independente à heurística do centro de massas e logo após aplicação desta. Aplicando este princípio a este caso em concreto o resultado para a camada superior seria, indiferentemente "**K**, **W**" ou "**W**, **K**" (obtido pela heurística do centro de massas) com o vértice **W** invertido, tal como na figura seguinte. É por isso importante identificar uma forma de detectar o número de cruzamentos entre as ligações de um conector e as restantes ligações entre duas camadas de forma a escolher a melhor sequência horizontal possível para um vértice.

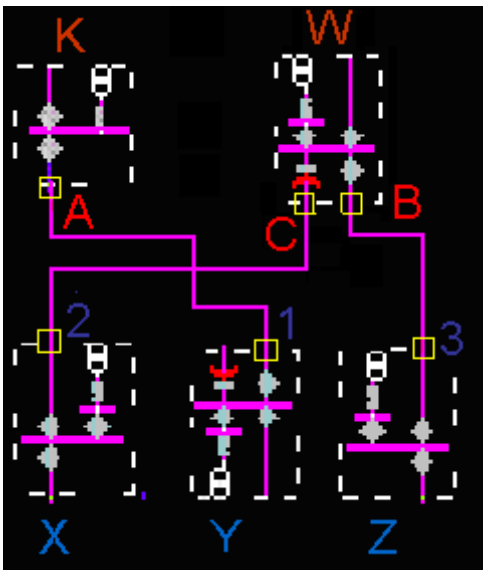


**Figura 64 - Melhor sequência possível para os conectores da camada superior, determinada usando heurística do centro de massas seguida de escolha de melhor inversão horizontal**

### Determinar o número de cruzamentos entre ligações

Já foi referida a necessidade de ter um algoritmo que determine o número de cruzamentos entre as ligações de um vértice numa dada camada. Será também importante ter uma heurística que determine o número de cruzamentos entre duas camadas.

Baseada na fórmula proposta por *Kozo Sugiyama* para o efeito, desenvolveu-se uma heurística simples para detectar o número de **cruzamentos entre as ligações de um vértice e dos restantes vértices entre duas camadas.** [15] O algoritmo desenvolvido não difere muito da fórmula proposta, mas facilita a interpretação desta. **O procedimento** no caso dos diagramas esquemáticos **deve ser feito considerando conectores em vez de vértices**, mas não apresenta diferenças nos procedimentos a seguir, bastando considerar cada conector como sendo um vértice isolado, mantendo a sua posição relativa.

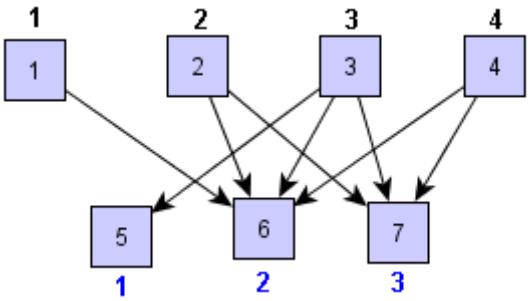


**Figura 65 - Camadas consecutivas de grafo com cruzamentos entre arestas**

De seguida apresenta-se a heurística proposta para determinar o **número de cruzamentos das ligações de um vértice entre duas camadas**. Complementarmente aos passos apresentados

irão ser dados exemplos, tomando como base determinar os cruzamentos do vértice 6 do grafo apresentado na figura 65. Repare-se, que as arestas deste vértice têm cinco cruzamentos. Assim:

1. Para o vértice que se pretende determinar o número de cruzamentos das suas arestas, determinar os vértices na sua camada que estão à sua esquerda.
   No grafo exemplo, o único vértice que está à esquerda do 6 é o vértice 5.

2. Para o vértice que se pretende determinar o número de cruzamentos das suas arestas, determinar os vértices na sua camada que estão à sua direita.
   No grafo exemplo, o único vértice que está à direita do 6 é o vértice 7.

3. Na camada oposta à do vértice tratado, determinar os seus vértices adjacentes e respectivas posições destes na camada.
   Os vértices adjacentes ao 6 na camada oposta são os vértices 1, 2, 3 e 4 (com posições iguais aos números).

4. Para cada um dos vértices determinados no primeiro passo, determinar o número de vértices adjacentes na camada oposta que estão à direita de cada um dos vértices devolvidos no terceiro passo.
   No exemplo, da lista de vértices à esquerda do nó, determinada no primeiro passo, constava apenas o vértice 5. Este tem ligação com o vértice 3 que se encontra à direita dos vértices (determinados no terceiro passo) 1 e 2. Logo foram detectados dois cruzamentos.

5. Para cada um dos vértices determinados no segundo passo, determinar o número de vértices adjacentes na camada oposta que estão à esquerda de cada um dos vértices devolvidos no terceiro passo.
   No exemplo, da lista de vértices à direita do nó constava o vértice 7 (primeiro passo), com ligação para os vértices 2, 3 e 4. O vértice 2 está à esquerda dos vértices 3 e 4 (determinados no terceiro passo), enquanto o vértice 3 está apenas à esquerda do 4 (determinado no terceiro passo). Logo, encontraram-se três cruzamentos.

6. Somar os valores obtidos nos passos 4 e 5.
   No exemplo, para as arestas incidentes ao vértice 6, foram detectados cinco cruzamentos.

A metodologia explica como obter o número de cruzamentos entre as ligações de um vértice (ou conector se for o caso) e as restantes ligações entre duas camadas. Continua contudo por clarificar como **determinar o número de cruzamentos entre duas camadas**.

O processo é semelhante, bastando para tal escolher um vértice para iniciar a contagem de cruzamentos utilizando a heurística acima. Cada vez que se detecte o número de cruzamentos das ligações de um dado vértice este deve ser removido da camada (e juntamente as arestas incidentes) e calculados os cruzamentos das ligações nos nós seguintes da mesma camada até que restasse apenas um nó (com apenas um vértice numa camada não é possível haver cruzamentos), somando o número de cruzamentos à medida que se avançava entre os nós. O número final de cruzamentos seria o resultante da soma efectuada.

Para o exemplo seguido, poderia começar-se por detectar o número de cruzamentos do nó 5 (daria três cruzamentos) removendo-se o nó e suas ligações de seguida. O nó seguinte seria o 6 onde seriam encontrados três cruzamentos (já com o vértice 5 removido), devendo remover-se o vértice. Passaria a restar apenas o vértice 7 que por si só não traria qualquer cruzamento entre arestas. O número final de cruzamentos encontrado seria, portanto, seis.

O mesmo processo poderia ser utilizado, no caso dos **diagramas esquemáticos**, para **determinar o número total de cruzamentos de um nó com dois ou mais conectores** entre duas camadas, bastando para isso determinar primeiramente os cruzamentos nas ligações de um dos conectores, removendo-o em seguida e passando a calcular o número de cruzamentos nas ligações

do conector seguinte, repetindo o processo até não restar qualquer conector no vértice. No final o número de cruzamentos nas arestas incidentes do vértice seria o somatório do número de cruzamentos de cada conector.

Finalmente resta dizer que para se detectar o **número de cruzamentos em todo o grafo** bastaria **somar o número de cruzamentos** existente **entre cada par de camadas** consecutivas.

## Aplicação da heurística do centro de massas a um grafo com várias camadas

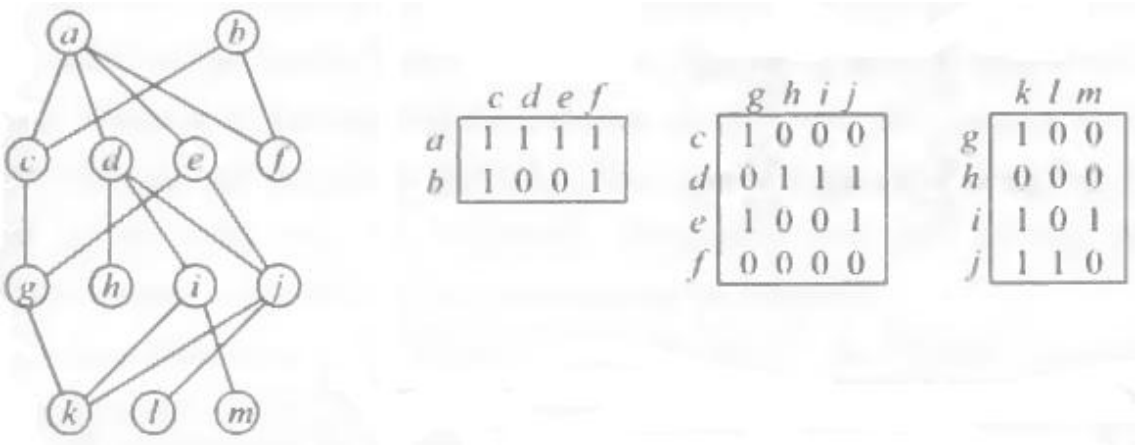


**Figura 66 - Matrizes de adjacência entre cada par de camadas num grafo [15]**

Os procedimentos anteriores explicam como obter a melhor ordenação dos vértices **entre duas camadas consecutivas**, aplicando a sequenciação de vértices numa camada superior ou inferior. Falta no entanto demonstrar como **ordenar os vértices em grafos com três ou mais camadas**.

Várias metodologias podem ser seguidas sendo que todas estas têm em comum apresentar soluções que passam por aplicar várias vezes a heurística do centro de massas entre duas camadas consecutivas, aplicando o princípio ***top-down*** e ***down-top***, ou seja aplicar a heurística do centro de massas várias de vezes, entre cada par de camadas consecutivas, de forma ascendente e descendente.

Por ser um dos procedimentos mais claros encontrados, além de ser aquele que envolve menos passos, seguidamente apresenta-se a metodologia proposta para dispor os vértices de um grafo com número de camadas superior a três [16].

Para tal, considere-se um grafo com **N** camadas, sendo **N>=3**. Do algoritmo para dispor os vértices em cada camada constam duas fases, sendo que a segunda fase usa a primeira como subalgoritmo. Antes da aplicação da primeira fase é imperativo atribuir uma ordem inicial a cada camada. [16]

**Fase 1** [16]**:**

1. Seja $i := 1$ (primeira camada).
2. Aplicar a heurística do centro de massas às colunas da matriz de adjacência obtida pelas camadas $i$ e $i + 1$ (ou seja, obter a melhor sequência para a camada $i + 1$).
3. Se $i < N - 1$, então $i := i + 1$ e voltar ao passo 2. De outra forma prosseguir para o passo 4.
4. Aplicar a heurística do centro de massas às linhas da matriz de adjacência obtida pelas camadas $i$ e $i + 1$ (ou seja, obter a melhor sequência para a camada $i$).
5. Se $i > 1$, então $i := i - 1$ e voltar ao passo 4. De outra forma terminar.

O passo dois da primeira fase ocorre no sentido descendente, ou seja inicia-se nas camadas superiores partindo para as inferiores, sendo aplicada a heurística do centro de massas nas colunas das camadas. O processo quatro é ascendente, sendo aplicada a heurística do centro de massas nas linhas. A **fase 1 deve ser iterada um número de vezes previamente definido**. [16] Os resultados finais obtidos (minimização de cruzamentos) serão tanto melhores quanto o número de iterações efectuadas, no entanto um grande número de iterações irá levar mais tempo a processar.

**Fase 2** [16]**:**

A fase 2 deve ser aplicada depois da anterior, mas apenas no caso de ainda subsistirem cruzamentos entre arestas no grafo. O principal processo introduzido na segunda fase é a **reversão da ordem de linhas e colunas em matrizes de adjacência** que detenham o mesmo valor de centro de massas. Este processo por vezes poderá ajudar a remover alguns cruzamentos.

1. Seja $i := N - 1$ (penúltima camada).
2. Reverter as linhas com igual centro de massa da matriz de adjacência obtida pelas camadas $i$ e $i + 1$ (muda a sequência dos vértices da camada $i$).
3. Aplicar fase 1.
4. Se $i > 1$, então $i := i - 1$ e voltar ao passo 2. De outra forma prosseguir para o passo 5.
5. Reverter as colunas com igual centro de massa da matriz de adjacência obtida pelas camadas $i$ e $i + 1$ (muda a sequência dos vértices da camada $i + 1$).
6. Aplicar fase 1.
7. Se $i < N - 1$, então $i := i + 1$ e voltar ao passo 5. De outra forma concluir.

Na fase 2 o segundo passo ocorre no sentido ascendente, começando por aplicar-se às camadas de maior profundidade. Já o passo 5 ocorre no sentido descendente. Tal como a fase 1, a segunda fase deve ser iterada um dado número de vezes. [16]

Para ambas as fases convém introduzir critérios de paragem entre iterações, já que se entre duas iterações consecutivas não existirem alterações é conveniente parar de forma a optimizar o processo. [16] Outra optimização que deve ser considerada é aceitar a primeira solução que surja sem qualquer cruzamento.

Uma nota final nesta etapa de remoção de cruzamentos entre arestas: por vezes existem vértices que entre duas camadas não têm qualquer ligação, por exemplo o vértice **h** no grafo da figura 66 (último exemplo) não tem qualquer ligação entre as camadas 3 e 4, o que leva a que o **centro de massas** seja **nulo** na linha que representa o vértice na matriz de adjacência entre estas camadas (ver também matriz de adjacência na figura, com uma linha só com zeros). Nestes casos o melhor é considerar que a linha nula (ou coluna se for o caso) que representa o vértice na matriz de adjacência entre camadas se mantém na mesma posição após qualquer aplicação da heurística do centro de massas. O mesmo significa dizer que aplicando o centro de massas a uma matriz com uma linha ou coluna nula faz com que a posição dessa linha ou coluna deva manter-se inalterada no final da aplicação do algoritmo. Como exemplo para esta questão, distribuam-se os vértices da terceira camada do grafo da figura 66, de acordo com os centros de massa calculados na tabela seguinte:

| **Vértice** | k | l | m | **Centro de massas da linha** |
|---|---|---|---|---|
| g | 1 | 0 | 0 | 1 |
| h | **0** | **0** | **0** | **Nulo** |
| i | 1 | 0 | 1 | (1 + 3) / 2 = 2 |
| j | 1 | 1 | 0 | (1 + 2) / 2 = 1.5 |

**Tabela 9 - Centros de massa de uma matriz com uma linha nula**

Ou seja de acordo com o que foi dito se fosse aplicada a heurística do centro de massas às linhas da matriz representada na tabela, esta manteria a segunda linha na mesma posição. A matriz tomaria a seguinte disposição:

| | | | |
|---|---|---|---|
| **g** | →1 | 0 | 0 |
| **h** | →**0** | **0** | **0** |
| **j** | →1 | 1 | 0 |
| **i** | →1 | 0 | 1 |

Tal significaria a permuta na terceira camada entre os nós **i** e **j**. Aplicando este princípio o vértice representado na linha nula permanece na mesma posição.

### 4.4.4 Definir posição horizontal absoluta dos vértices (Passo 4)

O passo anterior devolve um grafo com os vértices dispostos em camadas, sabendo-se a sequência que estes devem tomar em cada camada (posição relativa). No entanto ainda não foi determinada a **posição horizontal absoluta** de cada vértice no plano. Este passo visa devolver a posição horizontal absoluta que cada vértice deverá tomar no plano e compreende o último passo do algoritmo proposto por *Kozo Sugiyama*. [15]

No livro em que expõe o seu algoritmo de *layout*, *Kozo Sugiyama* deixa em aberto a possibilidade de utilização de programação quadrática para resolver o problema, tendo desenvolvido em conjunto com outros autores uma solução nesses moldes para o problema. [8] Foi também colocado um método alternativo baseado numa heurística para permitir resolver o problema, denominada método das prioridades. [23]

Dado a maior facilidade em implementar a solução baseada na heurística e pelo facto de necessitar de um tempo de processamento menor na sua aplicação [15], foi escolhido o **método das prioridades** para definir a posição horizontal dos vértices.

#### Método de *layout* baseado em prioridades

Este método é uma heurística desenvolvida que procura obter um grafo hierárquico com boa legibilidade. A ideia fundamental do método consiste, tal como no algoritmo do centro de massas, em decompor o problema em níveis e realizar o aprimoramento da distribuição horizontal dos vértices em cada nível separadamente. Pode considerar-se um nível, tal como no capítulo anterior, duas camadas consecutivas do grafo. Também nesta heurística o grafo é percorrido de forma ascendente e descendente de forma a aprimorar as posições dos vértices em cada nível. [15] De seguida apresentam-se os passos deste método.

1. O método inicia-se atribuindo uma coordenada inicial a cada vértice (normalmente considerando o seu centro), mediante a seguinte expressão [16]:

$$x_k^i = x_{k-1}^i + d + \frac{w_{k-1}^i + w_k^i}{2}$$

**Expressão 4 - Posição horizontal inicial de cada vértice [16]**

- **d** é um valor constante inicialmente atribuído como sendo a distância mínima entre dois vértices;
- **w** representa a largura de um vértice;
- ***i*** é o número da camada de um vértice;
- ***k*** é a posição horizontal de um vértice numa camada.
- **x** é o valor horizontal absoluto do centro do vértice

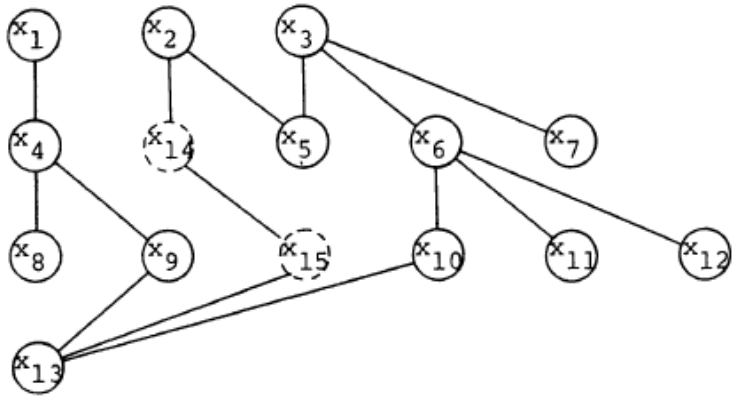


**Figura 67 - Estado de um grafo hierárquico após primeiro passo do método das prioridades [15]**

2. Percorrer de forma descendente → ascendente → descendente as várias camadas do grafo e realizar os procedimentos que se descrevem de seguida. Considera-se o movimento entre as camadas **2 e N como sendo descendente** e entre as **camadas N-1 e 1 como movimento ascendente**. [15]

2.1 Atribuir uma prioridade a cada vértice na camada tratada. No caso de se estar a percorrer o caminho **descendente** a prioridade deve ser dada pelo número de nós na **camada acima adjacentes ao vértice**. Se o grafo estiver a ser percorrido de forma **ascendente** a prioridade deve ser dada pelo número de nós na **camada abaixo** da tratada que **ligam ao vértice**. A qualquer vértice ***dummy*** **atribuir prioridade máxima**.

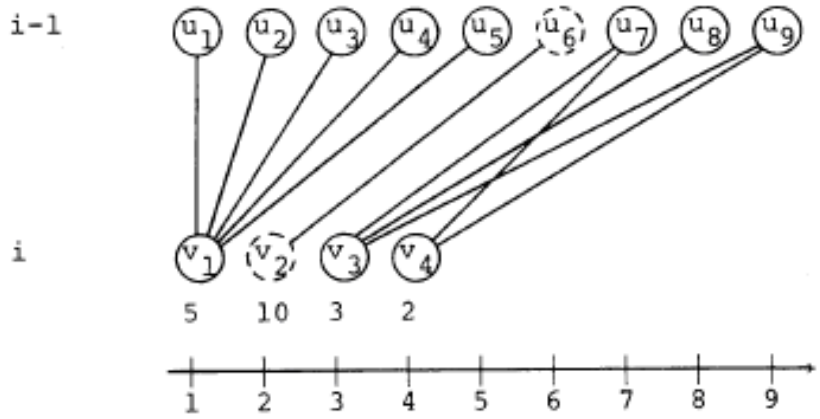


**Figura 68 - Prioridades dos vértices da camada *i* determinadas a partir do número de vértices adjacentes na camada superior *i* - 1 [15]**

2.2 Já com as prioridades de cada vértice definidas, executar os seguintes procedimentos para cada vértice **por ordem decrescente de prioridade**.

2.2.1 Quando se estiver a percorrer o grafo de forma **descendente**, deve tentar-se **aproximar** tanto quanto possível **o valor do vértice tratado** ao **centro de massas** dos seus **vértices adjacentes** na camada superior. O valor do **centro de massas** neste caso **deve ser dado pela soma das posições dos vértices adjacentes na camada superior a dividir pelo número de vértices adjacentes na camada superior**. Por outro lado quando o grafo estiver a ser percorrido de forma **ascendente** o processo deve ser semelhante, mas considerando os **vértices**

**adjacentes da camada inferior**. Neste processo, as seguintes restrições devem ser respeitadas:

1) As coordenadas horizontais de cada vértice devem ser valores inteiros. Um vértice não pode ter as mesmas coordenadas que outro, respeitando a largura de ambos e a distância mínima **d** entre cada vértice.
2) A ordem dos vértices na respectiva camada deve ser mantida de acordo com o determinado no passo que define a ordem dos vértices numa camada de forma a manter um número de cruzamentos entre arestas mínimo.
3) Quando para mudar a posição de um vértice para o aproximar do centro de massas dos seus vértices adjacentes for necessário mudar a posição de outros vértices, estes só deverão alterar a sua posição se tiverem prioridade inferior ao vértice tratado.

De seguida apresenta-se um exemplo, partindo do grafo da figura 68, de como calcular o centro de massas dos vértices adjacentes (tabela 10), bem como posicionar os vértices de acordo com o seu centro de massas (figura 69).

| Vértice | Centro de massas dos vértices adjacentes na camada oposta |
|---|---|
| v1 | (1+2+3+4+5) / 5 = 3 |
| v2 | 6 / 1 = 6 |
| v3 | (7 + 8 + 9) / 3 = 8 |
| v4 | (7 + 9) / 2 = 8 |

**Tabela 10 – Centros de massa dos vértices adjacentes aos nós da camada inferior do grafo da figura 68**

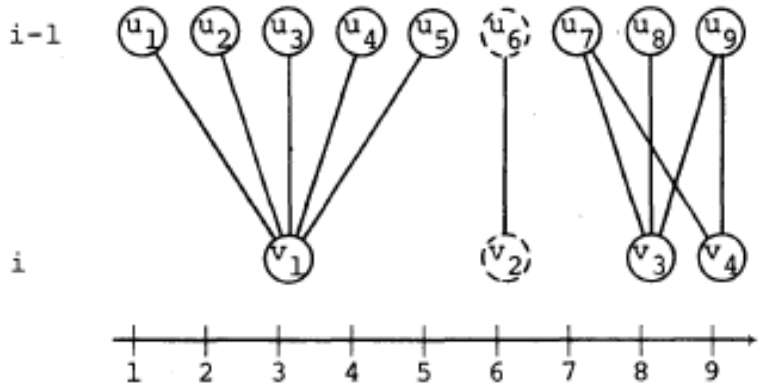


**Figura 69 - Vértices da camada *i* colocados de acordo com o centro de massas dos seus vértices adjacentes na camada *i-1* [15]**

Repare-se que no grafo anterior, o vértice 4 não assume a posição do centro de massas dos seus vértices adjacentes na camada oposta. Tal deve-se ao facto de nessa posição já se encontrar o vértice 3, de maior prioridade.

**Utilização do Método de *layout* baseado em prioridades em grafos esquemáticos de rede eléctrica**

O facto dos diagramas esquemáticos terem conectores nos vértices leva a que sejam necessárias alterações ao algoritmo de *layout* baseado em prioridades.

Nos exemplos anteriores, para calcular o centro de massas dos vértices adjacentes a um nó foram usados os centros destes vértices, uma vez que por norma, em grafos se encaminham as arestas para o centro dos nós.

Nos **diagramas esquemáticos**, as **ligações** são **encaminhadas** para os **conectores**, sendo que para obter arestas verticais rectilíneas, o alinhamento dos vértices deve ser efectuado considerando os conectores e não o centro dos vértices.

A modificação ao algoritmo é simples, neste caso. Basta que no cálculo do **centro de massas dos vértices adjacentes se considere a posição horizontal dos conectores** em vez da dos centros dos vértices, de forma que os conectores fiquem o mais próximo possível das suas ligações. No fundo procura-se aproximar horizontalmente os conectores ligados entre si em vez do centro dos vértices. O valor do centro de massas deve ser dado considerando a **posição entre si de ambos os conectores ligados** e não do centro dos vértices.

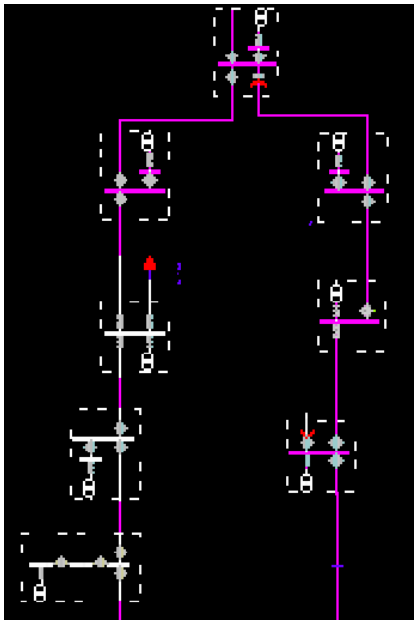

**Figura 70 - Vértices alinhados segundo o centro de massas dos conectores**

### 4.4.5 Passos adicionais ao algoritmo de *Sugiyama*

O passo anterior é o último considerado pelo algoritmo de *Sugiyama* para ordenação de grafos hierarquicamente. Foram apresentados todos os passos propostos pelo algoritmo juntamente com algumas modificações de forma a poder ser utilizado em diagramas esquemáticos de rede eléctrica. Para informação mais detalhada sobre o algoritmo utilizado, é conveniente consultar a bibliografia.

No entanto, embora terminados os passos do algoritmo, ainda há alguns problemas a resolver, principalmente o facto dos diagramas esquemáticos serem constituídos quase exclusivamente por arestas ortogonais e o algoritmo de *Sugiyama* considerar apenas arestas poligonais.

A figura seguinte apresenta o exemplo de um pequeno ramo de um diagrama esquemático gerado utilizando o algoritmo contrastando com a representação do mesmo ramo criada por um utilizador humano.

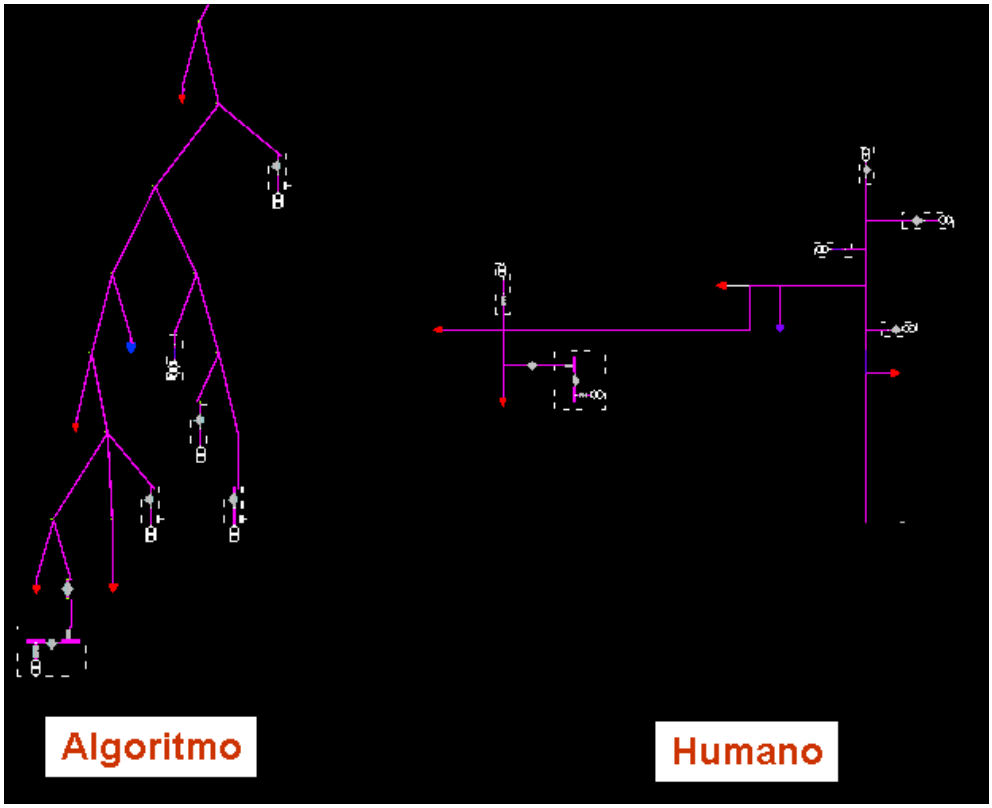


**Figura 71 - Ramo de esquemático desenvolvido pelo algoritmo em contraste com semelhante desenvolvido por humano**

Como se vê as diferenças entre uma representação produzida pelo algoritmo e por um utilizador humano ainda são muito significativas no que diz respeito à forma. Como diferenças fundamentais destacam-**se as linhas não ortogonais** e o facto do resultado produzido pelo algoritmo deter **todos os equipamentos colocados com a mesma orientação**, mesmo quando tal não se justificaria principalmente quando um equipamento surge num ramo mas não inclui qualquer sub-ramificação.

Foram então desenvolvidos dois passos adicionais que melhoram a visualização de ramos auto-gerados de diagramas esquemáticos, tornando-os mais parecidos com os produzidos por utilizadores humanos. O primeiro permite que equipamentos que surjam isolados num ramo não sejam forçados pelo algoritmo a mudar de camada tendo sido denominado **normalização de vértices**. O segundo passo adicional proposto faz a **ortogonalização** das arestas do diagrama. Estes passos adicionais ao algoritmo de *Sugiyama* encontram-se descritos de seguida.

### Normalização de vértices

Alguns vértices nos diagramas esquemáticos aparecem isoladamente numa ramificação principal, sendo que geralmente o utilizador humano quando desenha o diagrama os coloca **perpendicularmente** ao **sentido dessa ramificação principal**. Outros aparecem em pares, mas dado que são também uma sub-ramificação muito simples, normalmente também são colocados paralelamente ao sentido da ramificação principal. A figura seguinte apresenta exemplos de vértices colocados perpendicularmente numa ramificação.

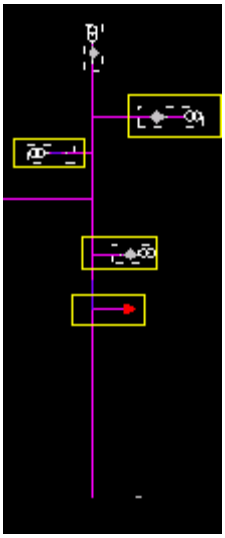

**Figura 72 - Vértices colocados perpendicularmente numa ramificação de diagrama esquemático produzido por utilizador humano**

O passo de **normalização** passa então por **evitar que vértices simples** que surjam isoladamente numa ramificação principal **dêem origem a uma nova camada de vértices**, colocando-os paralelamente à ramificação principal, na mesma camada da intersecção onde surgem. O processo de normalização foi estendido para sub-ramificações de apenas dois vértices.

A metodologia do processo passa pelos passos seguintes:

1. Para **todas as intersecções** do diagrama **determinar** aquelas que têm sub-ramificações com dois ou menos vértices. Escolher **no máximo** duas dessas sub-ramificações por cada intersecção.

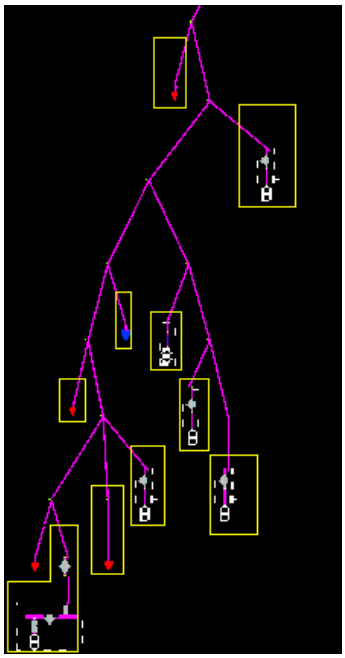

**Figura 73 - Sub-ramificações de um ramo de um diagrama esquemático com dois ou menos vértices**

2. Colocar essas ramificações lateralmente à intersecção, fazendo-as subir uma camada. Inserir essas ramificações **preferencialmente** na lateral (esquerda ou direita) mais de acordo com a posição na camada em que se encontravam (de acordo com o determinado no passo três do algoritmo de *Sugiyama*). Por exemplo se a intersecção está na posição 3 na respectiva camada e o primeiro nó da sub-ramificação surge na posição 4 na camada abaixo, então a sub-ramificação deve ser colocada preferencialmente à direita da intersecção.

2.1 **Todos os vértices da sub-ramificação** devem **subir para a camada da intersecção** onde surge a sub-ramificação. A sequência dos vértices a colocar na camada da intersecção depende do facto de serem colocados na lateral esquerda ou direita.

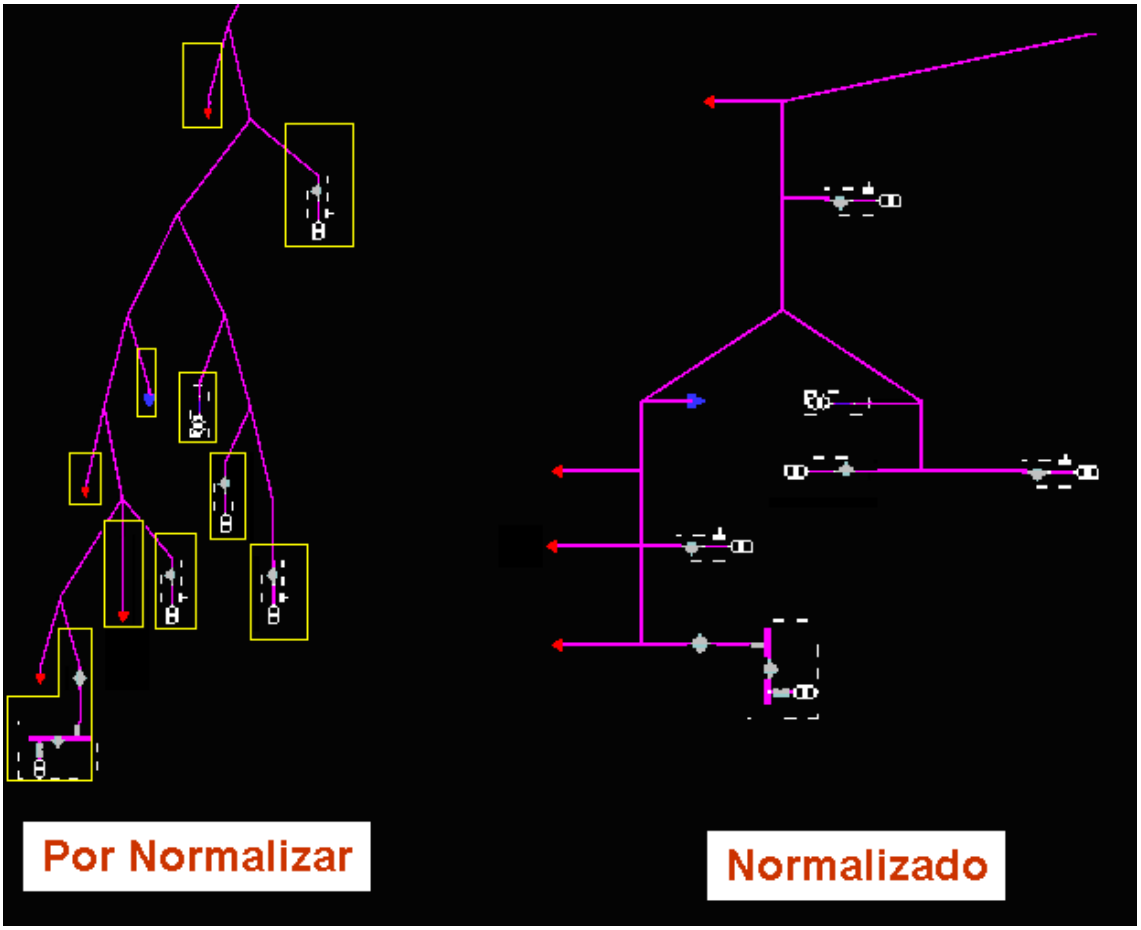


**Figura 74 - Ramificação de diagrama esquemático antes e após aplicar o passo de normalização**

O processo de normalização **diminui** consideravelmente o **número de camadas** de uma ramificação e **faz um melhor aproveitamento do espaço disponível**. Além do mais **torna o diagrama mais parecido com os produzidos por humanos**. Outra vantagem da normalização é que diminui consideravelmente as arestas poligonais dando um aspecto mais ortogonal aos diagramas.

A aplicação deste passo deve surgir imediatamente **antes do passo quatro** (definir posições horizontais absolutas dos vértices) do algoritmo de *Sugiyama* e **imediatamente após o passo três** (determinar sequência de vértices em cada camada) do mesmo algoritmo. É portanto um passo intermédio aos apresentados.

A aplicação do processo de normalização leva a que haja algumas **diferenças no passo quatro**. As alterações a operar nesse passo são as seguintes:

1. Os vértices colocados lateralmente a intersecções (sub-ramificações) devem ter prioridade mínima no método das prioridades.
2. Em vez de ser aplicado o princípio do centro de massas para os vértices adjacentes noutras camadas, para ajustar a posição dos vértices colocados lateralmente a intersecções (sub-ramificações normalizadas), estes tentarão simplesmente aproximar-se o máximo possível da intersecção onde foram colocados. No fundo vão passar a estar sempre "anexados" a essa intersecção.
3. A intersecção onde foi lateralizada uma sub-ramificação nunca irá considerar os vértices dessa sub-ramificação para determinar a sua posição horizontal absoluta. Passará portanto a ignorar esses vértices ao aplicar o princípio de ajuste de posição através do centro de massas dos vértices adjacentes.

**Ortogonalização das arestas do diagrama**

Um diagrama esquemático desenvolvido por um humano geralmente apresenta **poucas arestas poligonais**, sendo a sua maioria ortogonais. Com o passo de **normalização**, os ramos produzidos pelo algoritmo já **apresentam poucas arestas poligonais**, contudo existem ainda algumas **arestas poligonais que podem ser ortogonalizadas**.

Os cruzamentos entre arestas, não sendo frequentes, são possíveis e por isso desde logo se exclui a ortogonalização de arestas com cruzamentos. Tal procedimento poderia tornar ainda mais difícil a sua remoção à posteriori por parte de um utilizador humano.

Como a maioria das arestas poligonais surgem junto a intersecções, é conveniente começar por ortogonalizar arestas ligadas a intersecções (ver figura 74). Mas antes, é necessário ter em conta determinados factores. Atente-se na seguinte figura.

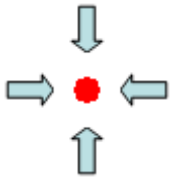

**Figura 75 - Intersecção com quatro ligações**

Dado tratar-se de um ponto, uma intersecção tem um **limite máximo de quatro arestas** ortogonais que pode acolher, uma em cada ponto cardeal. Desde logo é impossível no caso de haverem mais que quatro arestas a ligarem a uma intersecção mantê-las todas ortogonais.

O problema não é unicamente esse. Se as arestas que ligam à intersecção provierem todas do mesmo sentido, menos pontos cardeais de "embate" na intersecção vão restar. Por essa razão cada um dos pontos cardeais da intersecção deve destinar-se primeiramente a ligações específicas, tal como apresentado na figura seguinte.

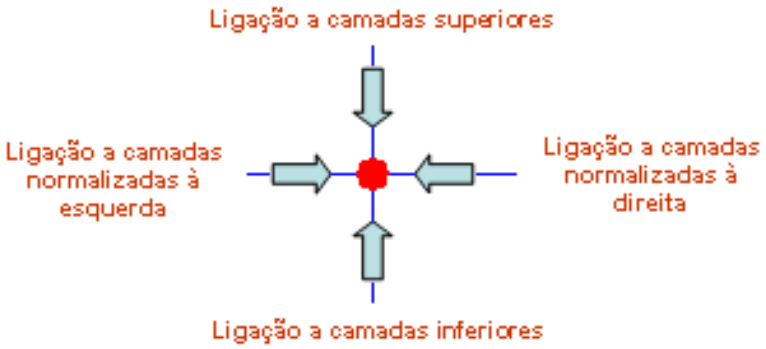


**Figura 76 - Ligações a pontos de intersecção**

Baseado nas observações acima, desenvolveram-se alguns métodos que permitem ortogonalizar a maioria das arestas de um grafo hierárquico que liguem a intersecções. Desde logo não se tenta ortogonalizar arestas adjacentes a vértices com mais de duas ligações a camadas superiores ou inferiores (figura 77) ou com duas ligações a vértices superiores e duas ligações a vértices inferiores (figura 78). Há contudo uma excepção que permite ortogonalizar arestas que liguem a um nó com três ligações a vértices de camadas superiores ou inferiores, no caso de uma das ligações (a intermédia) ser feita através de uma aresta rectilínea vertical (figura 79). Os exemplos seguintes mostram os casos em que se deve ou não proceder à ortogonalização de arestas.

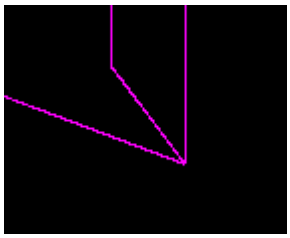

**Figura 77 - Ponto de intersecção com três ligações a vértices de camadas superiores. Neste caso não se tenta proceder a ortogonalização.**

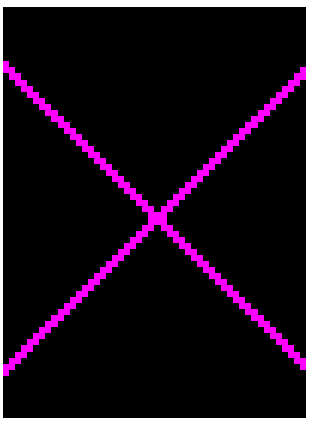

**Figura 78 - Ponto de intersecção com duas ligações a vértices de camada superior e duas ligações a vértices de camada inferior. Neste caso (muito raro) também não se tenta proceder a ortogonalização**

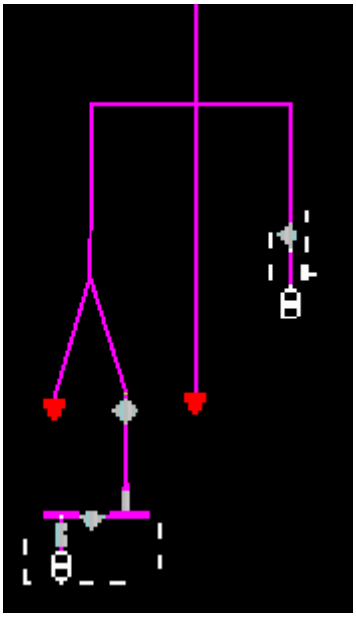

**Figura 79 - Caso excepcional onde é possível ortogonalizar arestas que ligam a intersecção com três ligações a camada inferior**

Nos restantes casos detectados é possível ortogonalizar as arestas de um ramo de um diagrama esquemático que estão ligadas a intersecções. Para tal basta **modificar a posição vertical do ponto de intersecção** para uma posição entre a camada em que se encontra e aquela para a qual quer ortogonalizar as arestas. Seguidamente é só **desenhar a aresta**, começando pela sua **componente horizontal** e **seguidamente a vertical**, tal como no exemplo seguinte.

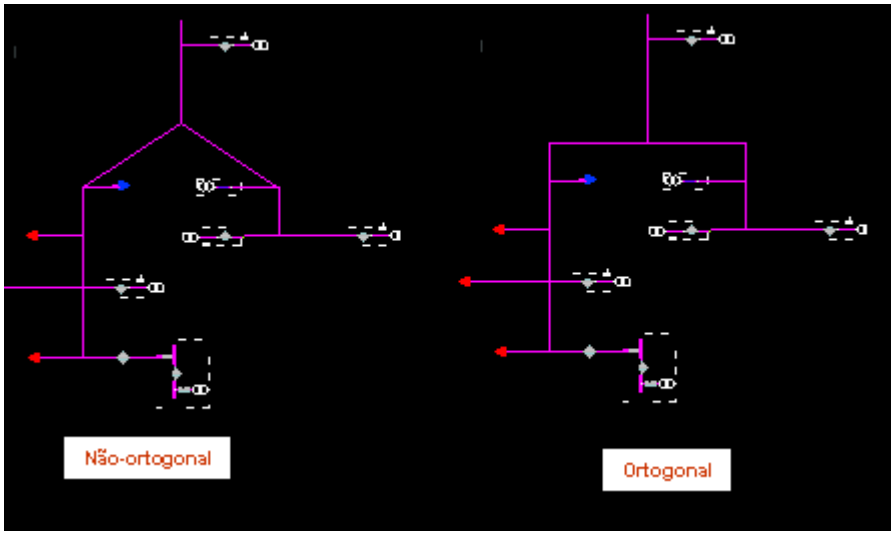


**Figura 80 - Ramo após ortogonalização de arestas**

O exemplo acima é o **caso mais frequente de ortogonalização**, permitindo **eliminar praticamente todas as arestas não ortogonais**. A intersecção tem três ligações, uma com a camada superior e duas com a inferior, uma vez que existem duas arestas a ligar à camada inferior, no processo de ortogonalização a intersecção deve ser colocada na posição vertical entre as duas camadas.

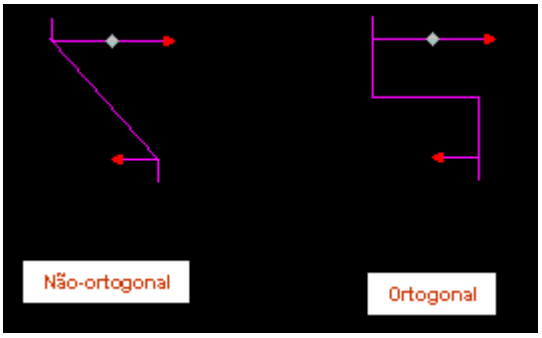


**Figura 81 - Ortogonalização simples de aresta**

O exemplo da figura 81 apresenta o caso mais simples de ortogonalização de arestas. Cada uma das intersecções ligadas por uma aresta detém apenas uma ligação para com a camada oposta. Assim, devem ser colocadas duas dobras intermédias entre ambos os pontos de forma a desenhar a componente horizontal e vertical da aresta.

A ortogonalização deverá ser feita também ao nível dos restantes símbolos do diagrama e não apenas às intersecções. Nesse caso não há casos especiais, uma vez que cada conector tem apenas uma ligação e cada equipamento tem, no máximo, duas ligações para outra camada. A razão é que se um conector se ligar fisicamente a dois equipamentos, então, na prática o que se faz, é sair no diagrama apenas uma ligação do conector que depois deriva (por meio de um ponto de intersecção) em duas arestas.

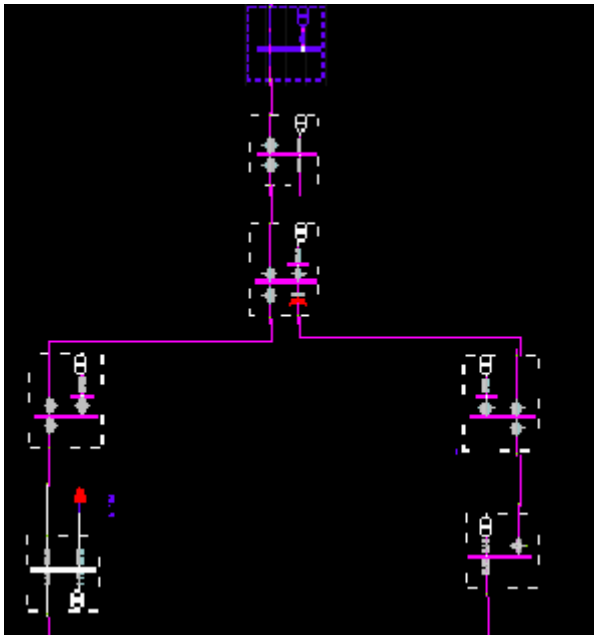

**Figura 82 - Ortogonalização entre equipamentos do diagrama**

# 5. Resultados finais

## 5.1 Introdução

No capítulo anterior foi descrito um possível método para gerar automaticamente ramificações de diagramas esquemáticos de rede eléctrica. Falta contudo provar a possibilidade de utilizar o método descrito, apresentando resultados conclusivos.

Este capítulo serve assim para apresentar resultados da aplicação do algoritmo de *Sugiyama* modificado em exemplos reais de ramos de diagramas esquemáticos. Para tal, foi desenvolvida uma *Framework* de testes que permite aplicar o algoritmo de *layout* em ramos de diagramas esquemáticos previamente introduzidos.

De forma a apresentar a *Framework* desenvolvida e ao mesmo tempo os resultados finais devolvidos por esta, o capítulo foi dividido em três subcapítulos. O primeiro irá apresentar a *Framework* de testes, seguidamente serão dados detalhes de como devem ser introduzidos os dados nesta (particularmente sentido das arestas) e finalmente, na última parte da secção, será mostrado o resultado da aplicação do algoritmo de disposição a um ramo de um diagrama esquemático.

## 5.2 *Framework* de testes

A *Framework* de testes desenvolvida teve como propósito provar o método, baseado no algoritmo de *Sugiyama*, proposto para efectuar o *layout* a ramos de diagramas esquemáticos de rede eléctrica.

Trata-se de uma aplicação independente que segue a metodologia descrita anteriormente para produzir disposições visualmente apelativas ao utilizador.

Da *Framework* consta uma área de desenho que permite ao utilizador colocar os equipamentos que pretende dispor, assim como as respectivas ligações. Novos equipamentos podem ser carregados para a *Framework* através de imagens.

A *Framework* permite ainda activar/desactivar os passos intermédios, adicionais ao algoritmo de *Sugiyama* que foram desenvolvidos.

Alguns dos exemplos já apresentados em capítulos anteriores foram desenvolvidos na *Framework*, da qual se apresenta um *print screen* na figura seguinte.

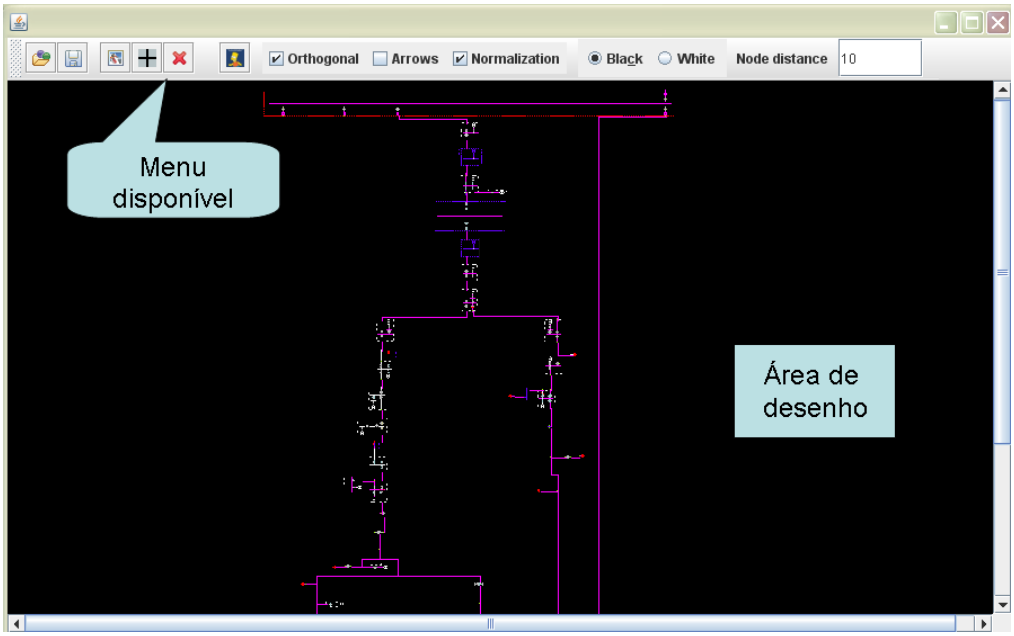


**Figura 83 - *Framework* de testes**

Foi desenvolvido um menu na *Framework* de forma a permitir ao utilizador dispor das funcionalidades supracitadas. Quanto à área de desenho, tem um comportamento *standard*: requer a existência de um rato ou dispositivo de entrada semelhante do qual a tecla principal permite adicionar um novo equipamento ou escolher um equipamento já desenhado, a direita tem funcionalidades de navegação e a tecla de *scroll* é requerida para *zoom*.

A framework foi desenvolvida utilizando uma versão experimental de uma biblioteca (*yFiles)* existente para desenho de grafos, que apresenta um conjunto de funcionalidades que facilitam operações sobre grafos. [10]

Na figura seguinte apresentam-se as funcionalidades do menu da *Framework* mais detalhadamente.

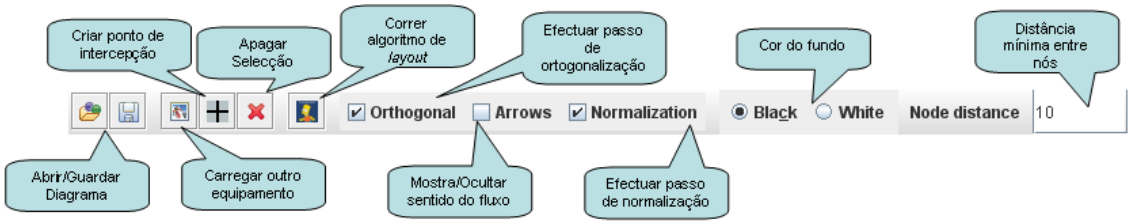


**Figura 84 - Opções do menu da *Framework* de testes**

### Abrir/Guardar diagrama

Compreende as funcionalidades de abrir um diagrama previamente criado ou guardar o diagrama representado na área de desenho. Para ambos os casos é apresentada uma janela de navegação que permite a escolha da área local onde se pretende guardar/abrir um diagrama.

### Carregar outro equipamento

Permite mudar o símbolo a ser criado na área de desenho sempre que premida a tecla principal do dispositivo de entrada (tipicamente a tecla esquerda do rato) num local vazio da área de desenho. O símbolo será usado até que seja carregado outro, o que significa dizer que poderão ser criadas várias instâncias deste na área de desenho.

Um símbolo é carregado através de uma imagem no formato GIF, onde devem estar identificados os conectores do equipamento, através de um pixel de cor especial, previamente definida. Na imagem seguinte apresenta-se um exemplo de uma imagem que cumpre com os requisitos citados para que seja um símbolo válido na *framework*.

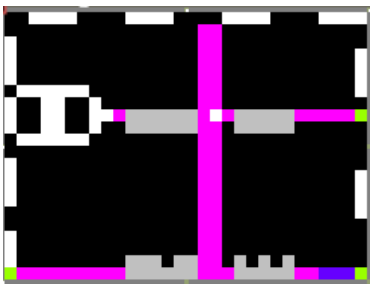

**Figura 85 - Exemplo de símbolo utilizado na *Framework***

Repare-se que na imagem os conectores estão representados por pequenos pontos verdes (representam apenas um pixel na imagem) nas extremidades do símbolo.

### Criar ponto de intersecção

Permite que a partir do momento de selecção da opção até que seja carregado um equipamento (ver funcionalidade anterior) o símbolo a ser criado na área de desenho seja um ponto de intersecção.

Uma intersecção é, em termos teóricos, um ponto e portanto não tem dimensões. No entanto para facilitar a sua representação foi-lhe atribuído um símbolo com apenas um pixel da mesma cor que os conectores, tal como se apresenta na imagem seguinte.

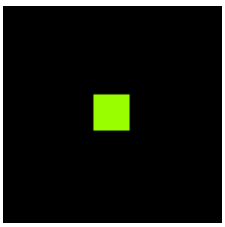

**Figura 86 - Símbolo de um ponto de intersecção**

### Apagar Selecção

Apaga todos os símbolos, pontos de intersecção, ou ligações que estejam seleccionados na área de desenho no momento em que for premida a opção.

### Correr algoritmo de *layout*

Aplica o algoritmo de *layout* hierárquico de *Sugiyama* de acordo com os princípios indicados previamente neste documento. O resultado da aplicação do algoritmo dependerá das opções que se activem nas *checkbox* que se apresentam de seguida.

### Efectuar ortogonalização das arestas

Permite activar/desactivar a opção de executar um passo adicional ao algoritmo de *Sugiyama* que permita tentar ortogonalizar as arestas do diagrama.

### Efectuar passo de normalização

Activa ou desactiva a opção de executar o passo adicional ao algoritmo de *Sugiyama* de normalizar sub-ramificações com dois ou menos vértices.

**Mostrar/Ocultar sentido do fluxo**

Permite activar/desactivar a opção de mostrar o fluxo (orientação) das arestas no diagrama. Serve apenas para efeitos de teste, uma vez que nos diagramas esquemáticos as arestas são não dirigidas.

**Alterar cor de fundo**

Permite alternar a cor de fundo entre preto e branco, consoante a preferência que leve a melhor visualização.

**Distância mínima entre nós**

Permite escolher a distância mínima (em pixéis) entre nós (de acordo com o previsto no algoritmo de *Sugiyama*) que o algoritmo de *layout* deverá respeitar.

**Marcar arestas com cruzamentos**

Uma funcionalidade extra que é accionada após aplicação do algoritmo é indicar as arestas onde não foi possível evitar cruzamentos, apresentando-as a cor vermelha.

**Determinar tempos de processamento**

Após a aplicação do *layout* automático, a *Framework* apresenta o tempo de processamento que foi necessário para o obter.

As funcionalidades criadas na *framework* foram determinantes não apenas para provar, mas também para desenvolver partes do método descrito neste documento para disposição de ramos de diagramas esquemáticos. De facto, grande parte dos exemplos e conclusões apresentadas nos capítulos anteriores tiveram como base estudos efectuados utilizando a *framework* criada.

## 5.3 Introdução de dados na *Framework*

Como já foi referido, o algoritmo de *Sugiyama* para disposição hierárquica de grafos apenas admite grafos dirigidos. Por essa razão é necessário definir um sentido das arestas, que tipicamente parte do equipamento expandido para os restantes, sendo que a ordem de expansão determina o fluxo do grafo (sentido principal das arestas).

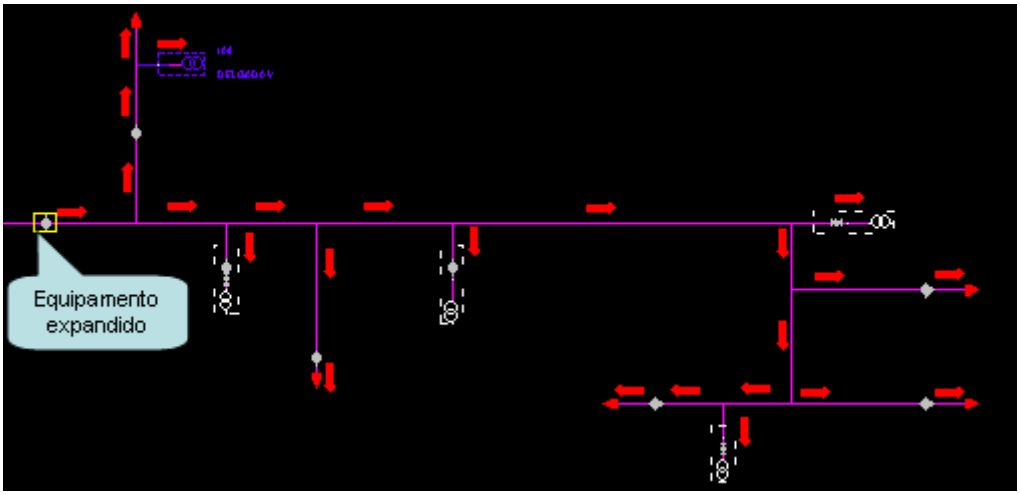


**Figura 87 - Sentido do fluxo num ramo de um diagrama esquemático sem ciclos**

Como se verifica na figura 87, tal é bastante simples em casos em que a estrutura seja acíclica, já que basta efectuar uma pesquisa simples sobre cada equipamento encontrado até que se atinja o termo de um ramo, até que se encontre uma subestação, ou até que o limite máximo de profundidade imposto pelo utilizador seja atingido.

No entanto, por vezes podem surgir ramos com ciclos que levam a um mesmo equipamento por dois caminhos distintos. Nesse caso, além dos ciclos deverem ser detectados para que não se entre num processamento de dados interminável, o sentido das arestas depende do método de pesquisa nos equipamentos efectuado a partir do equipamento seleccionado para expandir. Veja-se um exemplo de um ciclo na figura seguinte.

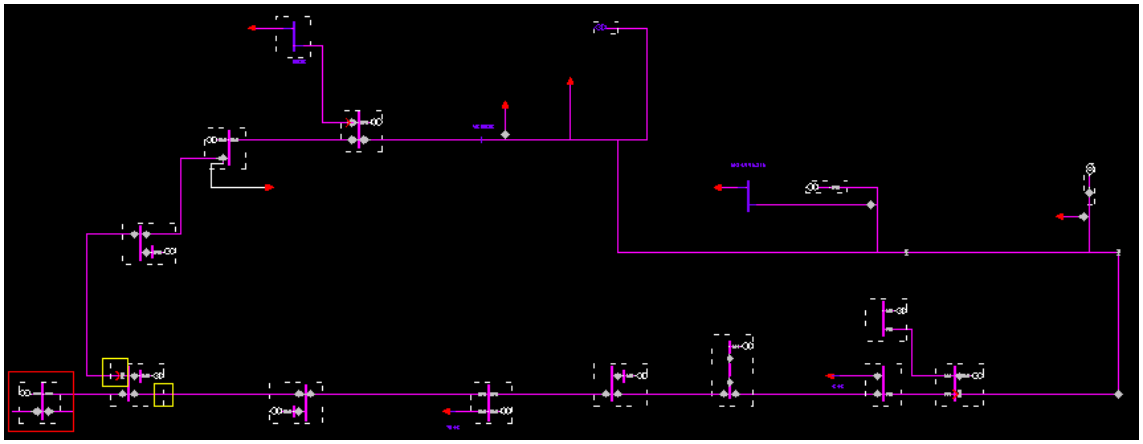

**Figura 88 - Ciclo de diagrama esquemático**

A figura apresenta o exemplo de um equipamento (rodeado por rectângulo vermelho) que se expandido poderia levar, dependendo do critério de pesquisa utilizado, a diferentes sentidos do fluxo.

Nos capítulos seguintes vai-se desenvolver esta matéria de forma a determinar qual o critério de pesquisa a utilizar. Serão apresentados respectivamente dois métodos de pesquisa distintos, a pesquisa em profundidade e a pesquisa em largura.

### 5.3.1 Expansão utilizando pesquisa em profundidade

A pesquisa em profundidade é uma das formas mais simples de expandir os equipamentos a gerar e ao mesmo tempo determinar o fluxo tomado pelas arestas destes.

O método consiste em seleccionar o equipamento que se quer expandir para iniciar a pesquisa e a partir do mesmo efectuar uma busca aos restantes componentes conectados a este, sendo que sempre que for encontrada uma aresta que ligue a um componente que já faça parte do caminho da pesquisa realizada (detectado ciclo), então a pesquisa deve ser concluída. Outra forma de concluir a pesquisa é encontrar um equipamento conectado que não tem mais nenhuma ligação, além da que levou à sua detecção. [18]

Numa pesquisa em profundidade, começa por se pesquisar sempre os descendentes de um nó até que um destes não tenha qualquer descendente ou volte a um ponto já visitado. [18] Durante a pesquisa pode começar-se pelos nós mais à direita ou mais à esquerda – pesquisa em profundidade à esquerda ou à direita – sendo que a escolha dos nós para iniciar a pesquisa leva a fluxos diferentes.

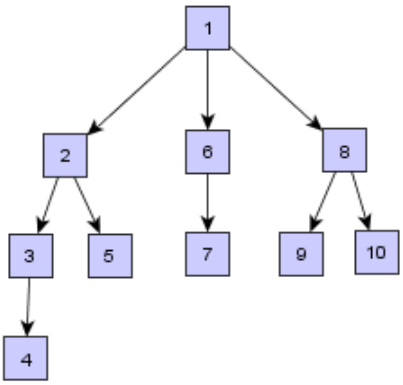


**Figura 89 - Vértices numerados de acordo com ordem de pesquisa em profundidade à esquerda**

A figura anterior apresenta a ordem de expansão que deveria ser seguida numa pesquisa em profundidade à esquerda aos nós de um grafo. Cada um dos vértices está numerado de acordo com a sua ordem de visita.

Na figura 90 apresenta-se o sentido que seria tomado pelo fluxo do ciclo apresentado na figura 88, se fosse realizada uma pesquisa em profundidade à esquerda iniciada no componente que se decidira expandir (rodeado a vermelho).

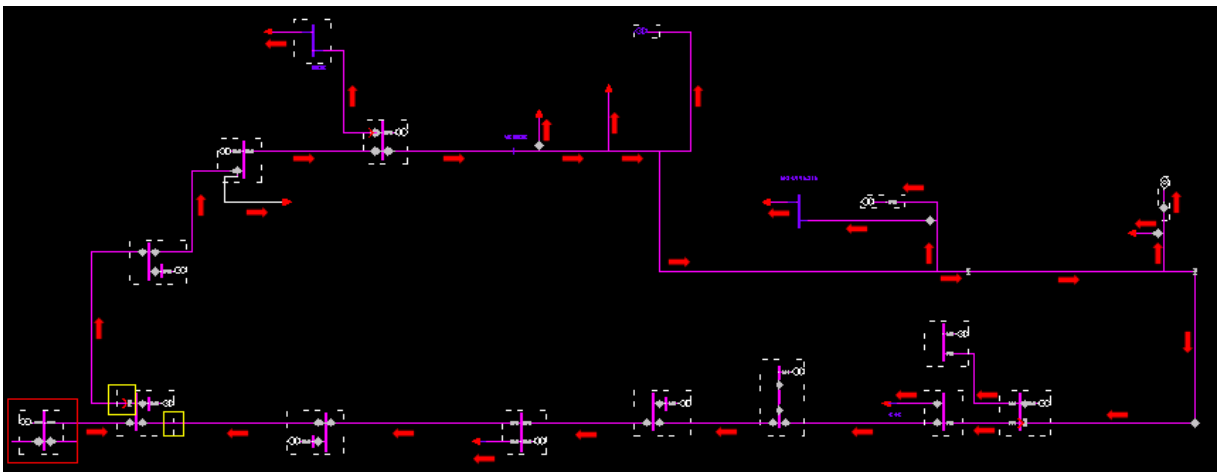

**Figura 90 - Sentido do fluxo de um ciclo num diagrama esquemático aplicando pesquisa em profundidade à esquerda**

Repare-se que o equipamento detectado directamente conectado ao expandido apresenta dois conectores (delimitados a amarelo) com ligações distintas que mais tarde vão convergir formando um ciclo. As setas a vermelho (que representam o fluxo da pesquisa) apresentam a ordem de expansão que seria tomada numa pesquisa em profundidade à esquerda.

O resultado obtido na *Framework* para disposição da parte do diagrama representada na figura 90, com o sentido das arestas colocado tal como indicado, é apresentado na imagem seguinte.

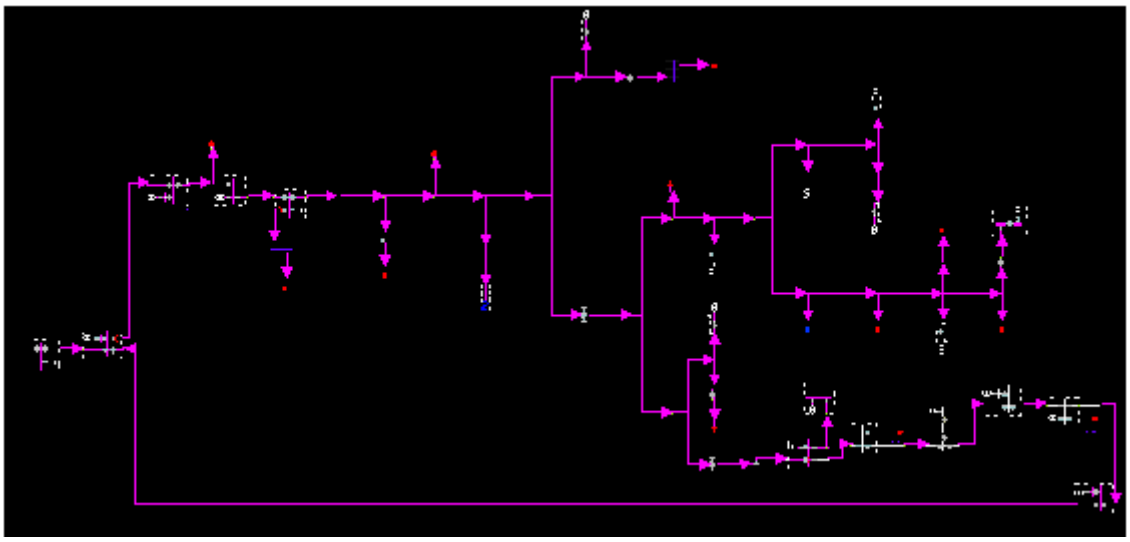

**Figura 91 - *Layout* automático do um ramo de um diagrama esquemático com fluxo obtido através de pesquisa em profundidade**

No extracto de diagrama da figura anterior foi colocado um sentido nas arestas de acordo com a ordem de expansão seguido. Estas servem no entanto apenas para efeito de testes, já que nos diagramas esquemáticos as arestas não têm orientação.

Repare-se que existe apenas uma aresta contrária ao fluxo e que esta foi a última a ser encontrada pelo algoritmo de pesquisa.

O resultado de disposição obtido, para esta parte do diagrama, teve **tempo de processamento** de **1,2 segundos**. Este valor, pode contudo variar dependendo da capacidade de cálculo do equipamento onde esteja a correr a *framework* De facto, o dispositivo utilizado para testes, encontra-se já algo obsoleto.

### 5.3.2 Expansão utilizando pesquisa em largura

Uma busca utilizando o algoritmo de pesquisa em largura poderá ser uma possibilidade para que os resultados de *layout* automático apresentem uma forma mais uniforme.

Uma pesquisa em largura iniciada na raiz (neste caso o equipamento expandido) deve começar por explorar todos os seus nós adjacentes. Seguidamente, para cada um desses nós, devem ser explorados os seus nós vizinhos por visitar e assim por diante até que seja atingida uma subestação, encontrado um equipamento já visitado ou se atinja o termo de um ramo. [24]

Deve começar-se por pesquisar sempre os nós mais próximos à raiz que se encontrem por visitar. [24] A figura seguinte exemplifica a ordem de busca que deveria ser levada a cabo numa pesquisa em largura aos vértices de um grafo.

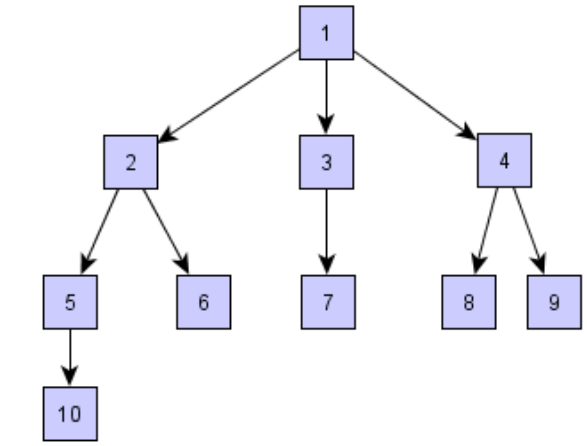


**Figura 92 - Vértices numerados de acordo com ordem de pesquisa em largura à esquerda**

Tal como na pesquisa em profundidade, numa pesquisa em largura pode optar-se por iniciar a pesquisa nos vértices mais à esquerda ou mais à direita. Nos exemplos dados a pesquisa é iniciada à esquerda.

Usando o mesmo exemplo das figuras 88 e 90, mas aplicando uma expansão aos equipamentos do ciclo utilizando o algoritmo de pesquisa em largura à esquerda, o sentido que o fluxo da pesquisa tomaria, com as arestas direccionadas de acordo com a ordem de visita seguida no algoritmo de pesquisa, apresenta-se na imagem seguinte.

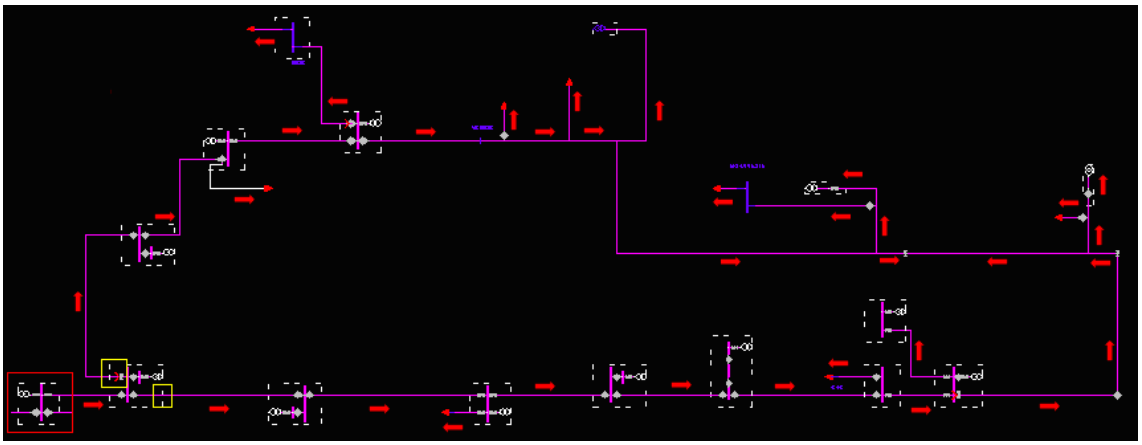

**Figura 93 - Sentido do fluxo de um ciclo num diagrama esquemático aplicando pesquisa em largura à esquerda**

O resultado obtido na *Framework* para disposição da parte do diagrama representada na figura 93, com o sentido das arestas colocado tal como indicado, é apresentado na figura seguinte. Ressalve-se, mais uma vez, que o facto das arestas apresentarem uma orientação serve apenas para efeitos de teste, não devendo estas constar de uma geração automática real.

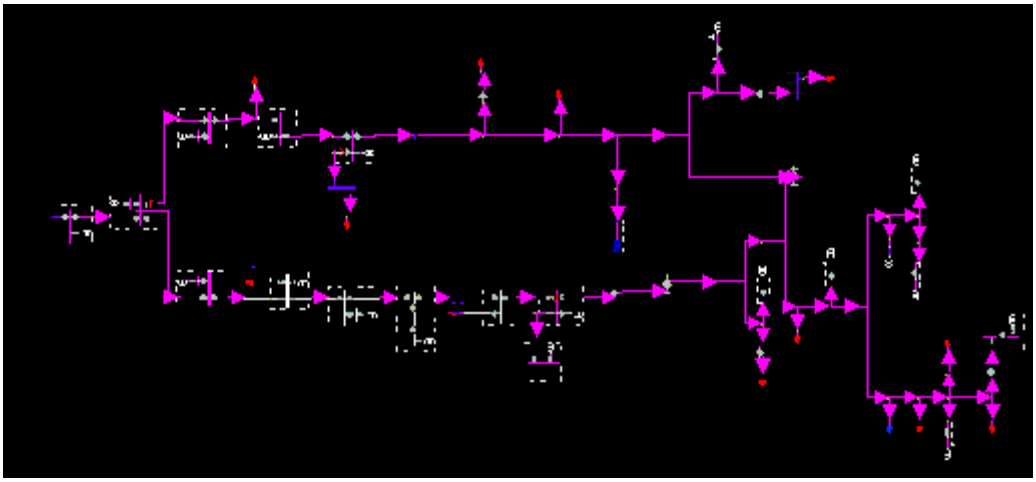

**Figura 94 - *Layout* automático do um ramo de um diagrama esquemático com fluxo obtido através de pesquisa em largura**

Neste caso, dado que a orientação das arestas foi obtida através de uma pesquisa em largura, o resultado do *layout* automático aos componentes apresenta uma representação uniforme dos dois fluxos que partem do mesmo nó.

O tempo de processamento necessário para obter a **disposição dos equipamentos** foi de **0,2 segundos**. O dispositivo utilizado para efectuar o teste foi o mesmo que havia sido usado no exemplo utilizando pesquisa em profundidade.

Contrariamente ao que acontecia quando foi utilizado o algoritmo de pesquisa em profundidade, neste caso não existe qualquer aresta em direcção contrária ao fluxo, havendo maior equidistância entre os tamanhos das arestas.

### 5.3.3 Conclusão sobre o algoritmo de pesquisa a utilizar

Nos dois subcapítulos anteriores foram apresentadas duas alternativas para efectuar a pesquisa de equipamentos a expandir numa base de dados de equipamentos real. Foi também sugerido que o sentido das ligações (arestas) fosse colocado de acordo com a sequência de equipamentos e respectivas ligações encontradas, procedimento necessário para aplicar o algoritmo de disposição sugerido.

O resultado de ambos os métodos de pesquisa devolve os mesmos equipamentos encontrados. No entanto, sendo que a ordem de detecção destes é diferente dependendo do método utilizado, o sentido das ligações (arestas) é alterado, levando a resultados finais de *layout* diferentes.

Para os métodos de pesquisa testados, respectivamente algoritmo de pesquisa em profundidade e em largura, o segundo levou a que os resultados de disposição automática se apresentassem com arestas mais equidistantes, com os equipamentos melhor distribuídos e com todas as arestas do fluxo na mesma direcção.

Quanto aos tempos de processamento, utilizando pesquisa em largura, o primeiro passo do algoritmo de *Sugiyama* (remoção de ciclos) apresenta-se muito mais rápido, uma vez que o sentido determinado para as arestas evita que surjam ciclos. De facto a disposição automática demorou seis vezes mais para o mesmo ramo quando foi utilizada pesquisa em profundidade para expandir os equipamentos.

Pelas razões citadas o algoritmo de pesquisa em largura parece ser o mais adequado para a pesquisa a efectuar na expansão dos equipamentos.

## 5.4 Resultado do *layout* automático a um ramo de um diagrama esquemático

Para concluir o estudo sobre a possibilidade de utilizar os métodos descritos como forma de gerar automaticamente os componentes de um ramo de um diagrama esquemático de rede eléctrica, falta apresentar o resultado de uma disposição automática a um ramo real de um diagrama esquemático e fazer a respectiva análise do resultado obtido.

Assim, atente-se na figura seguinte que apresenta um ramo de um diagrama esquemático desenhado por um utilizador humano.

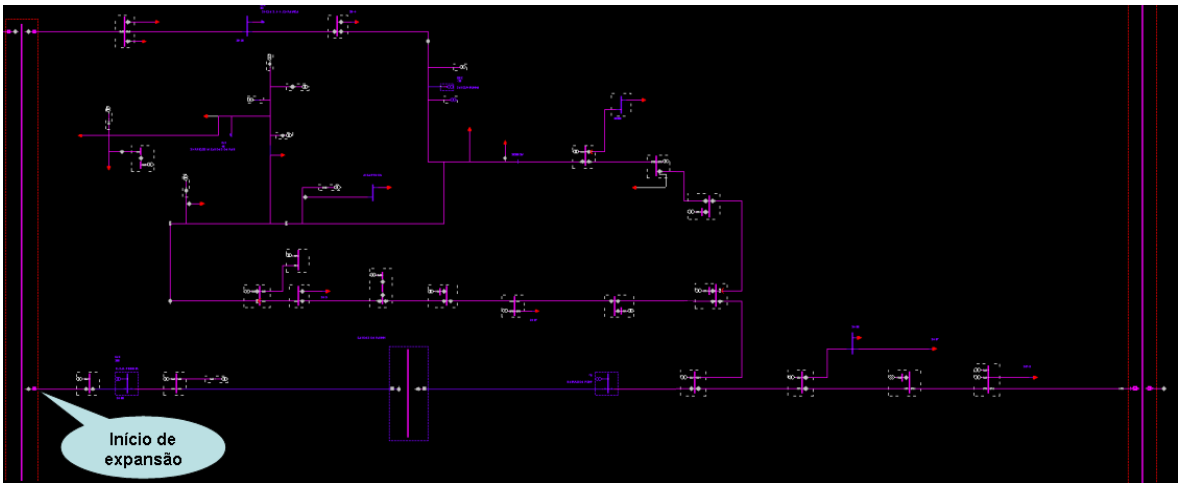


**Figura 95 - Ramo de diagrama esquemático entre duas subestações**

Os equipamentos existentes no ramo exemplo apresentado foram colocados na *Framework* de testes, aplicando os princípios da pesquisa em largura para direccionar as arestas, de forma a aferir da qualidade de uma geração automática que se obteria para o ramo. Considerou-se que o início da expansão automática ocorreria no local assinalado na figura 95.

Assim, a figura 96 apresenta o resultado obtido para uma disposição automática iniciada no ponto de expansão indicado, sendo efectuada pesquisa em largura para detectar os restantes componentes e colocando as ligações com o sentido de acordo com a ordem de pesquisa. Considerou-se que não havia restrições de paragem impostas pelo utilizador (níveis máximos de profundidade), sendo por isso este o caso teoricamente mais complexo para uma expansão automática.

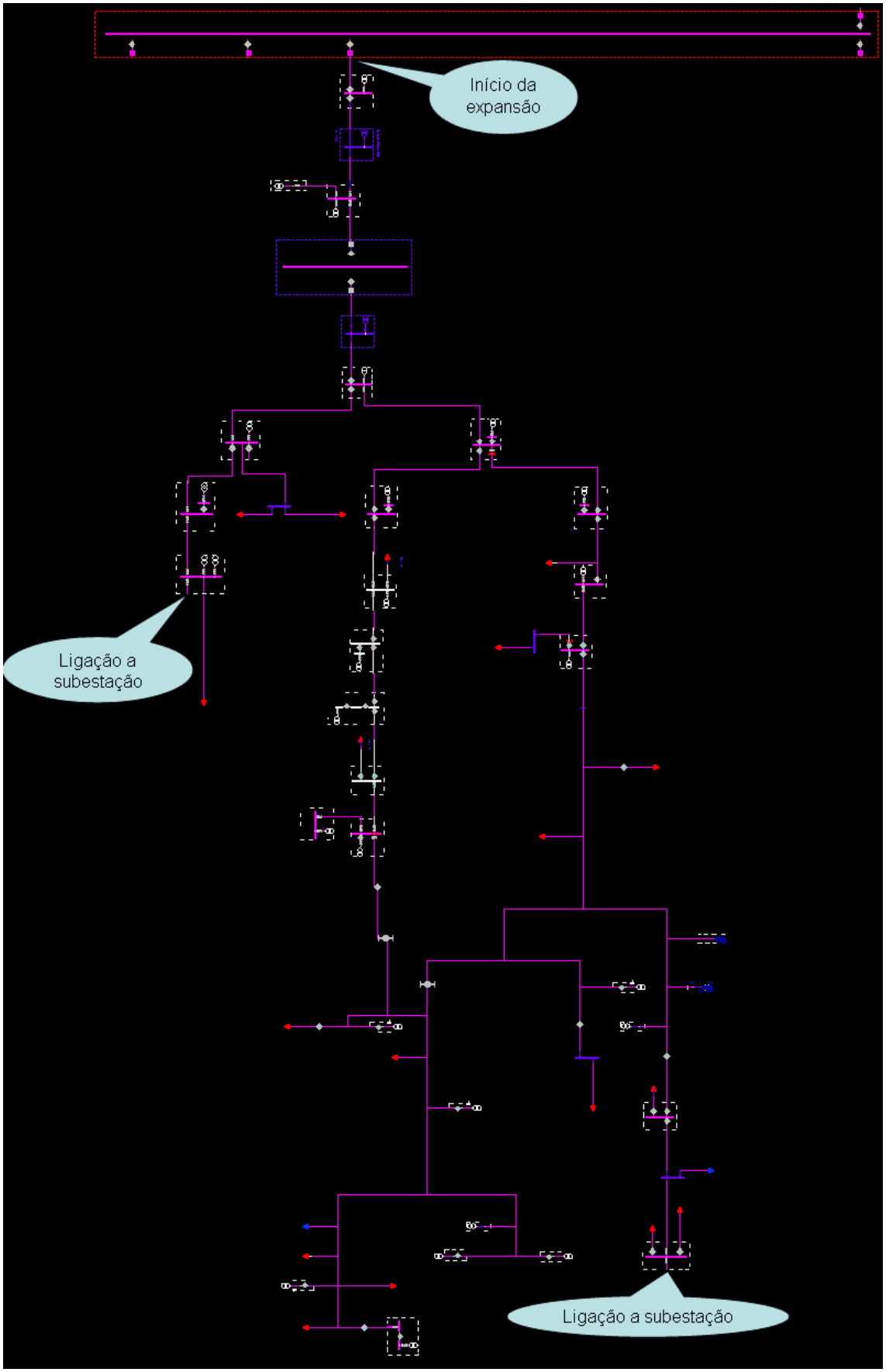


**Figura 96 - Disposição automática do ramo desenhado manualmente da figura 95**

Para facilitar a visualização do resultado obtido, a geração automática foi apresentada na figura verticalmente. Será contudo de esperar que seja mais frequente um hipotético utilizador preferir expansões na horizontal, dado o aspecto geral apresentado nos diagramas esquemáticos desenhados por utilizadores humanos privilegiar essa orientação.

A expansão foi interrompida nos locais indicados dado terem surgido ligações a subestações. No editor de diagramas, uma vez decidido incorporar a funcionalidade de geração automática nos moldes sugeridos, deve ser dada a informação ao utilizador das interrupções efectuadas a uma geração automática.

Nas opções para o *layout* automático incluiu-se uma distância mínima entre equipamentos de dez pixéis, tendo levado aproximadamente **1,8 segundos** para obter o resultado apresentado.

O resultado final pode ser considerado satisfatório, pese embora as claras diferenças existentes entre a representação produzida pelo utilizador e a automática que podem ser vistas comparando as figuras 95 e 96.

O *layout* automático apresenta os componentes dispostos uniformemente, neste caso sem qualquer cruzamento entre arestas embora por vezes possam surgir e nesse caso o utilizador deve ser alertado para os cruzamentos existentes. A visibilidade geral é boa pese embora alguns pontos devessem ser alterados por um utilizador humano de forma a conferir um aspecto mais natural. O tempo de processamento necessário, embora um pouco elevado, não é comprometedor e pode ser melhorado visto que não foi uma das principais preocupações durante o desenvolvimento da *Framework*.

Com o resultado obtido é possível provar a utilidade de uma ferramenta de geração automática baseada no algoritmo de *Sugiyama*. No entanto, não é credível que um utilizador aceitasse os resultados obtidos sem lhe operar qualquer alteração já que embora os resultados obtidos mostrem uma disposição com boa visibilidade, em alguns pontos apresentam pouca naturalidade. Ainda assim, pouparia certamente muito tempo ao utilizador, bastando-lhe operar algumas modificações rápidas no resultado final de forma a obter uma melhor visualização.

# 6.Conclusões e Trabalho Futuro

Este capítulo destina-se a apresentar as conclusões finais aos estudos efectuados, entretanto apresentados no decorrer deste documento.

Do trabalho realizado constou a apresentação de um conjunto de ferramentas a introduzir no editor de diagramas esquemáticos de rede eléctrica alvo de estudo. Esse grupo de funcionalidades teria como objectivo aumentar a produtividade de um utilizador humano no desenho de diagramas esquemáticos, evitando-lhe processos repetitivos e indicando-lhe alguns erros que possivelmente poderia cometer. Foram também indicadas ferramentas que poderiam facilitar a manutenção de diagramas esquemáticos já que deveriam permitir detectar as incoerências existentes entre estes e uma base de dados de equipamentos.

Sendo que da principal funcionalidade proposta constava a geração automática de partes de diagramas esquemáticos, o estudo desta dissertação passou a concentrar-se em apresentar um possível método que permitisse gerar e dispor automaticamente componentes de ramificações de diagramas.

Concluída a explicação sobre os métodos utilizados, foram apresentados os resultados de *layout* automático seguindo os procedimentos indicados aplicados a um ramo real de um diagrama esquemático produzido por um utilizador humano e contrastando os resultados obtidos. Foi também apresentada a *Framework* desenvolvida para permitir efectuar testes aos métodos de *layout* utilizados.

Este capítulo começará por contrapor os resultados alcançados em relação aos objectivos previamente colocados antes de expor possíveis melhorias e estudos adicionais ao trabalho desenvolvido.

## 6.1 Satisfação dos objectivos

Foram colocados como objectivos iniciais a apresentação e respectiva especificação de um conjunto de funcionalidades a introduzir no editor de diagramas esquemáticos que permitisse eliminar processos repetitivos ao utilizador no desenho de diagramas esquemáticos, bem como facilitar a manutenção de diagramas já criados, assim como evitar erros do utilizador durante a criação destes. Dentro das funcionalidades propostas, desde cedo se assumiu que a mais importante deveria ser a geração automática de partes de diagramas pelo que o estudo se centrou especialmente nessa possibilidade. Nunca foi, contudo, colocada como realista a possibilidade de gerar automaticamente a totalidade de um diagrama sem qualquer interferência do utilizador.

As funcionalidades propostas foram divididas em dois grupos: um deles referia-se a ferramentas de geração automática de partes do diagrama e o segundo a ferramentas que possibilitassem evitar erros por parte do utilizador na criação de diagramas e detectar incoerências com a base de dados. Não sendo possível quantificar o quão importante se tornariam estas ferramentas no desenho e manutenção dos diagramas (e assim determinar se os objectivos foram cumpridos), crê-se que seriam uma mais valia no editor, permitindo reduzir o tempo necessário no seu desenho e manutenção.

Uma vez que a geração automática de partes de um diagrama foi colocada como prioritária, desenvolveu-se um possível método baseado no algoritmo de *Sugiyama* para disposição hierárquica de grafos que provou obter bons resultados para geração automática de ramificações de diagramas esquemáticos, tendo os testes sido desenvolvidos numa f*ramework* criada para o efeito. Ficou contudo claro que, utilizando exclusivamente o método proposto, não seria possível gerar a totalidade de um diagrama esquemático e que os resultados devolvidos deveriam sofrer algumas revisões do utilizador.

Perante os resultados apresentados e dado a subjectividade dos objectivos colocados é impossível falar em cumprimento de objectivos uma vez que tal só poderá ser feito conclusivamente após integrar as ferramentas no editor. Ainda assim, julga-se que foi apresentado um conjunto de funcionalidades que pouparia muito tempo a um utilizador no desenho e manutenção de diagramas. Como resultado conclusivo, ficou provado que uma geração automática nos moldes indicados seria possível através dos testes efectuados e é praticamente certo que tal funcionalidade reduziria drasticamente o tempo necessário para desenhar um novo diagrama.

Há, por isso, um nível de satisfação elevado com os dados obtidos resultantes do estudo, já que se acredita que uma vez integradas as ferramentas indicadas no editor de diagramas esquemáticos, tais resultariam em mais valias para o mesmo, reduzindo o tempo necessário para o desenho e manutenção dos diagramas.

## 6.2 Trabalho futuro

O trabalho desenvolvido poderá ter uma continuidade que facilite ainda mais a criação e manutenção de diagramas esquemáticos de rede eléctrica.

Trabalhos futuros poderão concentrar-se em especificar novas ferramentas completamente diferentes das sugeridas, que permitam aumentar a produtividade dos utilizadores no desenho de diagramas ou aprimorar as indicadas.

Um exemplo de novas ferramentas que poderiam ser estudadas que certamente trariam grande utilidade, seriam funcionalidades que permitissem lidar melhor com os componentes já existentes nos diagramas, que possibilitassem abrir espaços na área de desenho para novas gerações automáticas ou mesmo para que o utilizador tivesse espaço para inserir manualmente equipamentos.

No âmbito da geração automática de partes de um diagrama, poderão ser estudados métodos que integrem vários algoritmos de disposição de forma a permitir desenvolver maiores áreas de diagramas esquemáticos que os estudos levados a cabo, tornando o processo de desenvolvimento de diagramas mais iterativo e rápido.

Poderão ser estudados também métodos de *layout* alternativos ao apresentado como possibilidade para geração automática e verificar se tais procedimentos produzem melhores resultados que os obtidos, tais como *layout* ortogonal utilizando o algoritmo *topology-shape-metrics* referido na revisão bibliográfica.

No método apresentado como possibilidade para *layout*, baseado no algoritmo de *Sugiyama*, poderão ser estudadas soluções alternativas para alguns dos passos deste, nomeadamente o passo dois (remoção de cruzamentos entre arestas) e o passo quatro (determinar posição horizontal absoluta dos vértices) baseadas em programação quadrática e verificar se tais alterações levam a melhores resultados, não só no aspecto final, mas também nos tempos de processamento, que se julga, seriam superiores. Durante uma geração automática, poderão também ser encontrados

métodos que lidem melhor com a gestão do espaço disponível na área de desenho, passando a considerar áreas indisponíveis para colocar componentes. Uma sugestão é começar por definir ferramentas que permitam fazer encaminhamento de arestas entre componentes já desenhados, funcionalidade normalmente designada *autorouting* e por especificar funcionalidades de colocação imediata de componentes no diagrama gerindo o espaço livre.

Importa ressalvar que possíveis trabalhos futuros devem ter como princípio fundamental estar centrados em promover estudos que levem à introdução de funcionalidades no editor que permitam reduzir o tempo de criação de diagramas esquemáticos e facilitar a sua manutenção.

# Referências


[1] Grupo EFACEC, *Quem somos?*, disponível em: http://www.efacec.pt/, acedido dia 17 de Junho de 2010

[2] Michael Leslie, *Computer Applications in Power Systems – SCADA*, Ottawa, IEEE Ottawa section, 2003

[3] Quintas, António Rocha, *Sistemas SCADA*, Porto, 2004

[4] Peter M. Fonash, *Supervisory Control and Data Acquisition (SCADA) Systems* – Technical Information Bulletin 04-1, Virginia, National Communications System, 2004

[5] Walski, Thomas M., et. al., *Advanced Water Distribution Modelling and Management*, Haestad Press, January 2003

[6] Gonzalez, A. Emmanuel, *Introductory Graph Theory for Electrical and Electronics Engineers*, IEEE, 2007

[7] Picado, Jorge, *Teoria dos Grafos*, Departamento de Matemática da Universidade de Coimbra

[8] Cruz, Isabel F. e Tamassia, Roberto, *Graph Drawing Tutorial*, Worcester Polytechnic Institute e Brown University

[9] Tzoumakas, Vasileios e Theodoulidis, Babis, *Force Based Visualizations for Instructor Support*, School of Informatics University of Manchester, 2005

[10] yWorks GmbH, *yFiles for Java Developer's Guide*, http://www.yworks.com/products/yfiles/doc/developers-guide/, consultado dia 25 de Janeiro, 2010

[11] Bertolazzi, Paola; Di Battista, Giuseppe e Didimo Walter, *Computing Orthogonal Drawings with the Minimum Number of Bends*, IEEE Computer Society, 2000

[12] Diesendruck, Liana, *Graph Drawing: Algorithms for the Visualization of Graphs*

[13] Nishizeki, Takao e Rahman, Saidur, *Planar Graph Drawing*, World Scientific Publishing, Singapore, 2004

[14] Laohavaleeson, Ekarat, *Orcad Layout Plus Tutorial*, http://www.docstoc.com/docs/2144173/Orcad-Layout-Plus-Tutorial, consultado a 29 de Janeiro de 2010, 2006

[15] Sugiyama, Kozo, *Graph drawing and applications for software and knowledge engineers*, Vol. 11, Japan advanced institute of science and technology, 2002

[16] Roger Van Dalen e Mike Spaans, *An algorithmic and a declarative approach designed and implemented in the functional system clarity*, On automated Graph layout, 2001

[17] Michael R. Garey e David S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W.H. Freeman, 1979

[18] Tarjan, R., *Depth first search and linear graph algorithms*, SIAM J. Computing, Vol.1, no.2 146/159, 1982

[19] Eades, P. e Sugiyama, *How to draw a directed graph*, J. of Information processing, vol.13, no.4, 424/437, 1990

[20] Karatsuba, Anatolii A.; Yuri P. Ofman (1962). *множение многозначных чисел на автоматах*. Doklady Akademii Nauk SSSR 146: 293–294. Translated in Physics-Doklady 7: 595–596, 1963.

[21] Erkki Makinen, *The Barycentre heuristic and the reorderable Matrix*, Department of Computer Sciences, Universidade de Tampere, 2005

[22] Sugiyama, K., Tagawa, S., e Toda, M., *Effective representations of ISM hierarchies*, Proc. of Int. Conf. on Cybernetics and Society, Denver, 413/418, 1979

[23] Sugiyama, K., Tagawa, S., e Toda, M., *Research in the visual representation of structural information – layered structure models and automatic planar drawing algorithms and applications*, Fujitsu International Research Center for Social Information Science, 2nd Research report, 1/41, 1981

[24] Knuth, Donald E., *The Art Of Computer Programming*, Vol. 1. 3rd ed., Boston: Addison-Wesley, 1997

[25] Mendes, João, *Editor gráfico de Rede Eléctrica – Conectividade na EFACEC, Sistemas de Electrónica, S.A.*, Relatório de Estágio Curricular, 2006

# Anexo A: Exemplos de Diagramas

## A. 1 Diagramas Esquemáticos

O objectivo do estudo foi identificar possíveis ferramentas de suporte ao desenho de diagramas esquemáticos de rede eléctrica. No decorrer do documento foram facultados alguns extractos de diagramas, contudo não foram apresentados exemplos de diagramas completos ou mesmo partes significativamente grandes destes.

Este anexo procura colmatar essa carência, apresentando exemplos retirados do visualizador de diagramas esquemáticos. Assim, está composto unicamente por várias figuras que apresentam exemplos de diagramas esquemáticos.

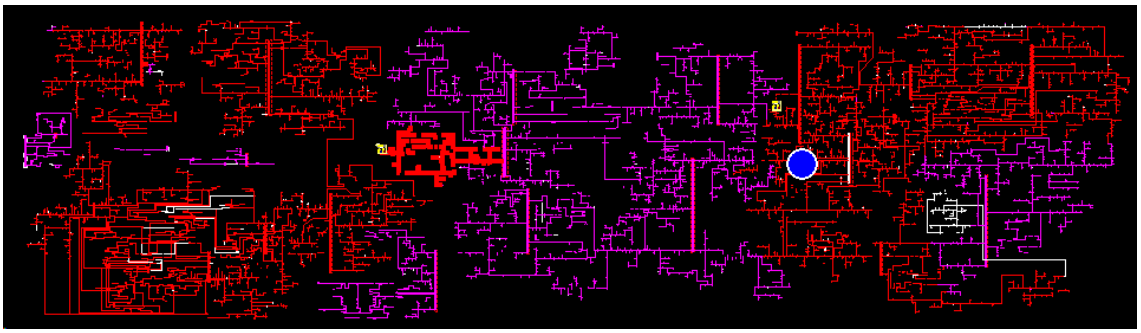

**Figura 97 - Diagrama esquemático completo (Exemplo 1)**

**Figura 98 - Diagrama esquemático completo (Exemplo 2)**

**Figura 99 - Diagrama esquemático completo (Exemplo 3)**

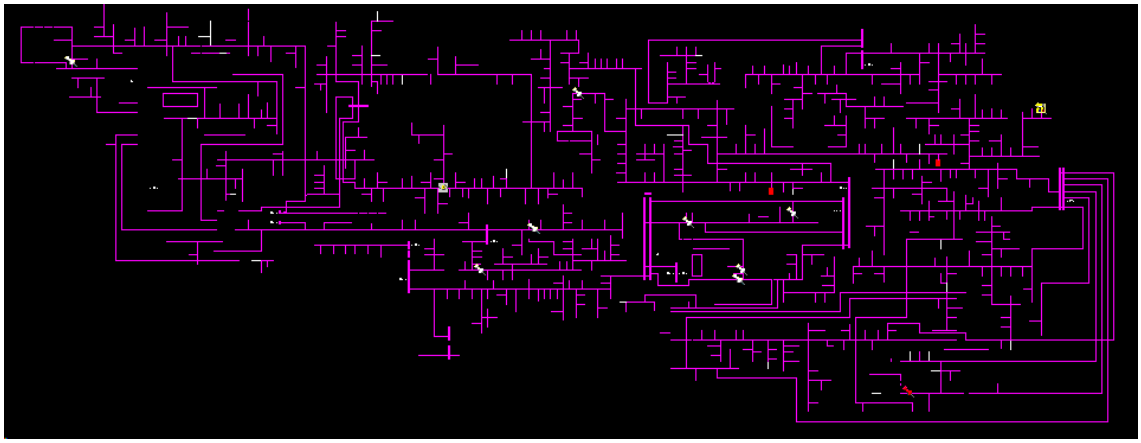

**Figura 100 - Parte isolada de diagrama esquemático**

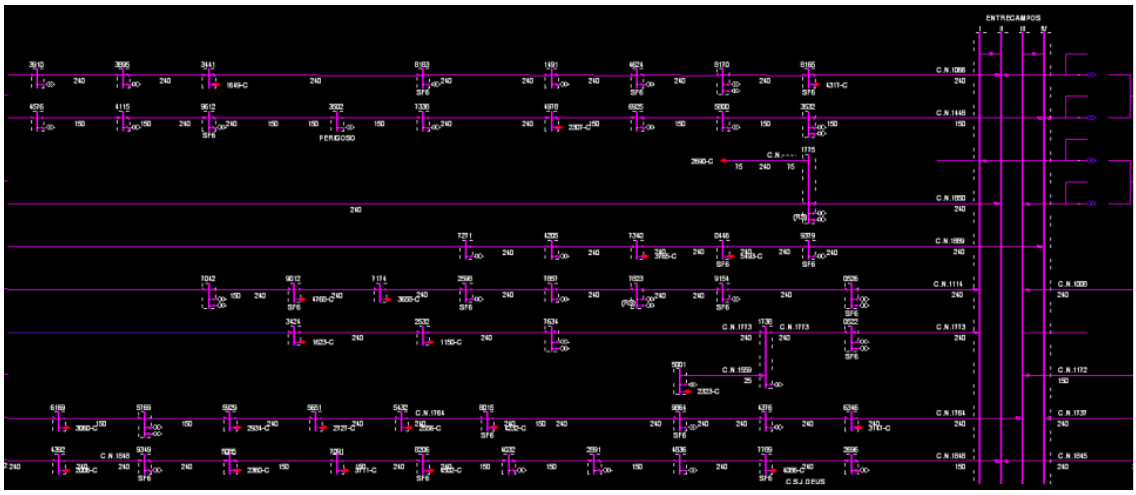

**Figura 101 - Extracto de diagrama esquemático junto a subestação (Exemplo 1)**

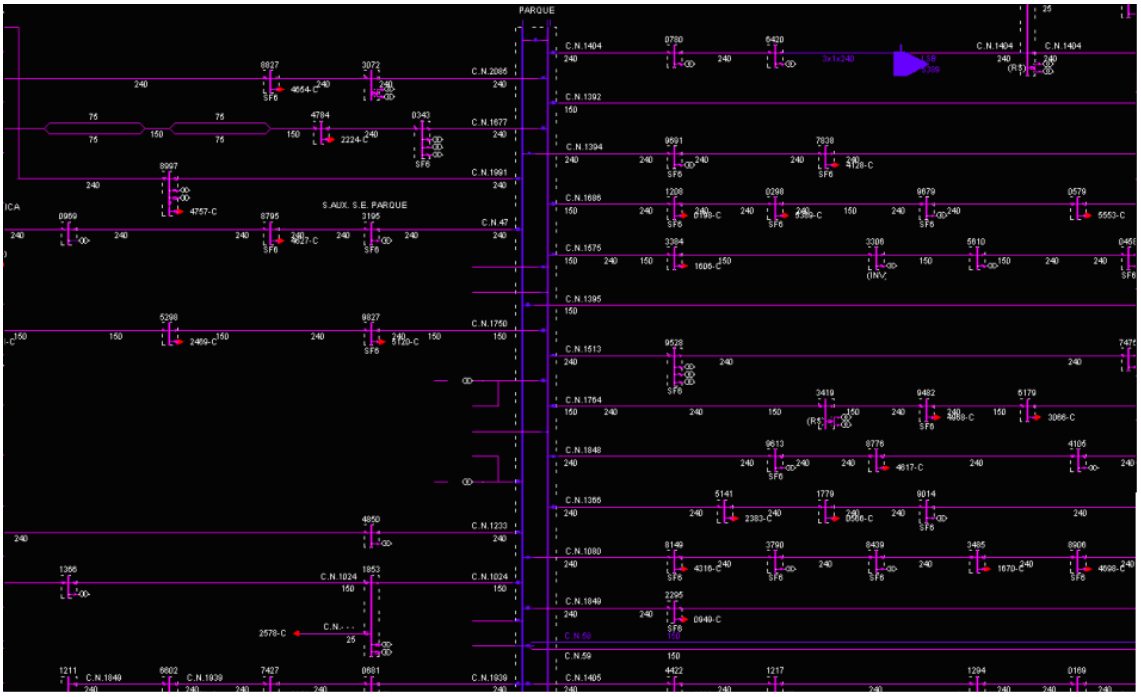


**Figura 102 - Extracto de diagrama esquemático junto a subestação (Exemplo 2)**

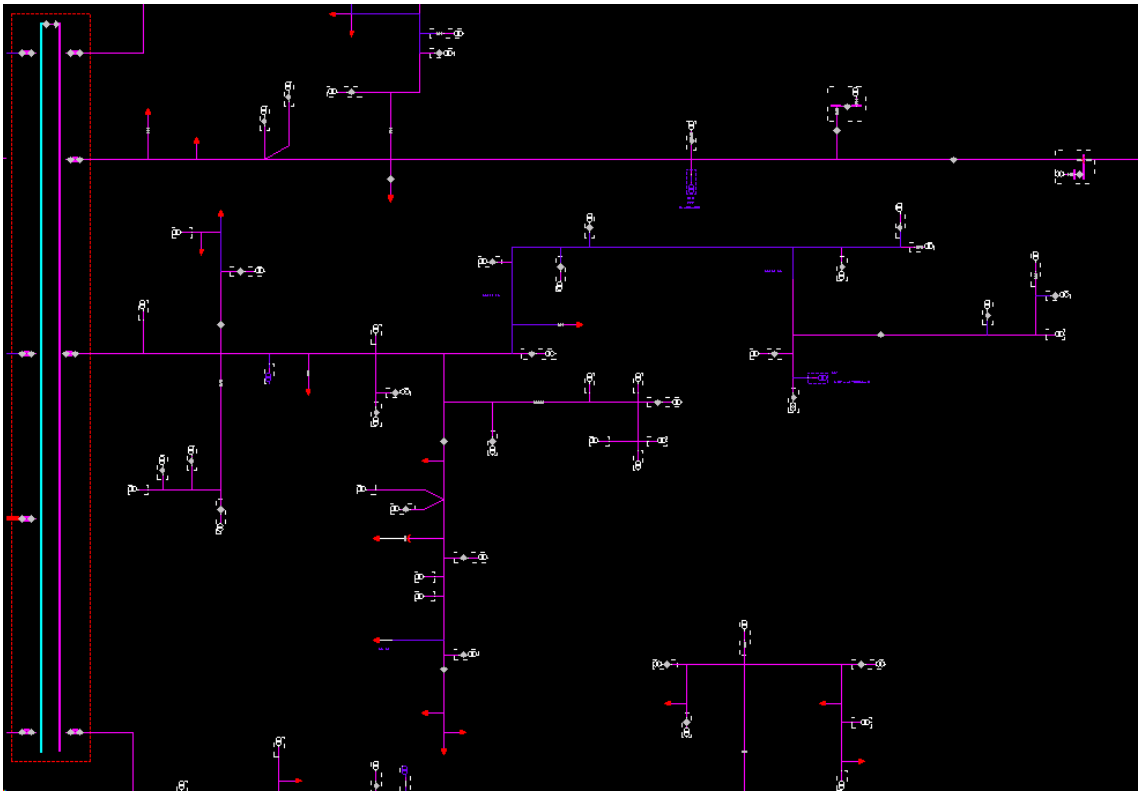

**Figura 103 - Extracto de diagrama esquemático junto a subestação (Exemplo 3)**

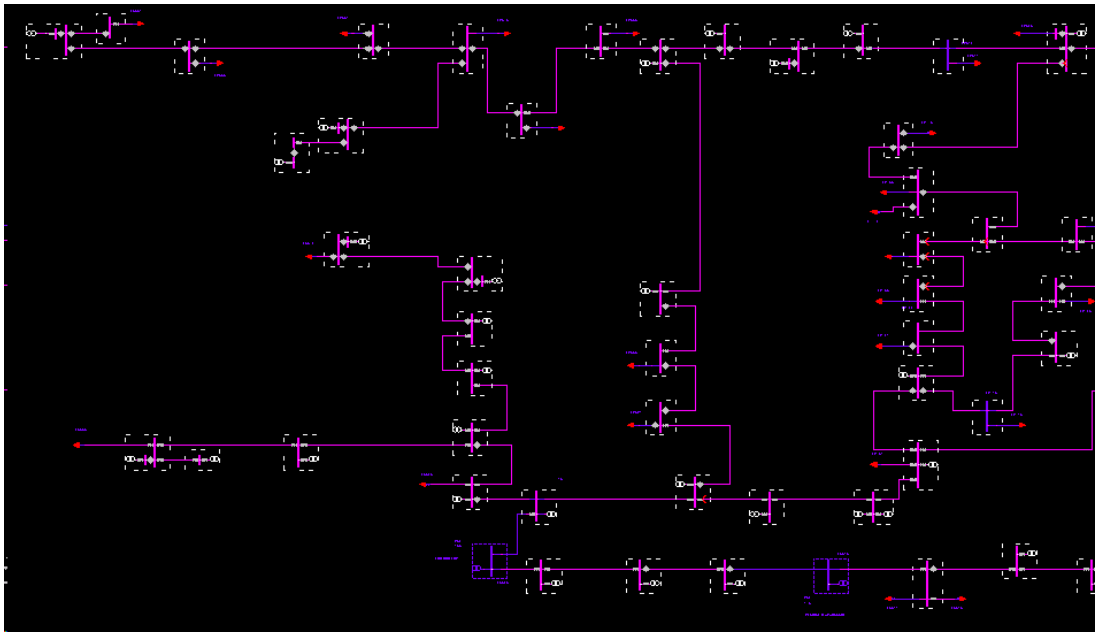

**Figura 104 - Ligações entre equipamentos em diagrama esquemático**

## A. 2 Diagramas Geográficos

Os diagramas esquemáticos são desenvolvidos complementarmente a diagramas geográficos que apresentam uma perspectiva de acordo com a localização real dos equipamentos.

Nesta secção apresentam-se alguns exemplos de diagramas geográficos da rede eléctrica que mostram o quão difícil se poderia tornar gerir operações utilizando unicamente esta perspectiva.

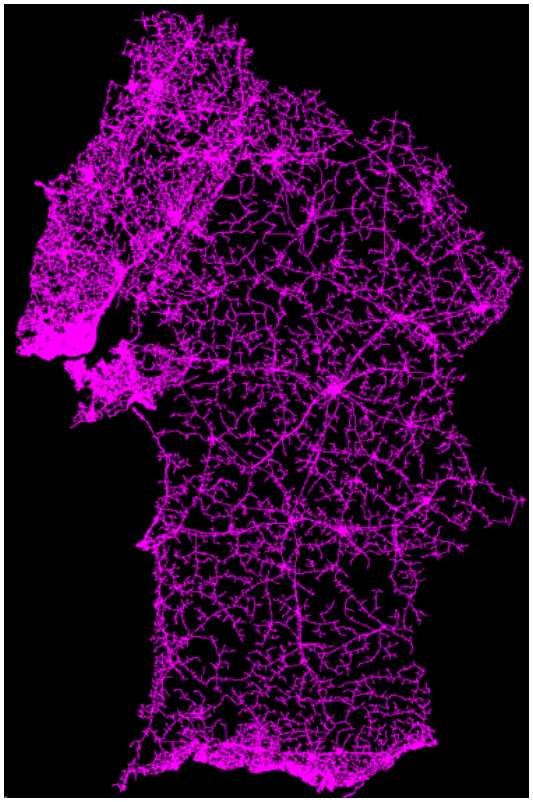

**Figura 105 - Diagrama geográfico da região sul de Portugal**

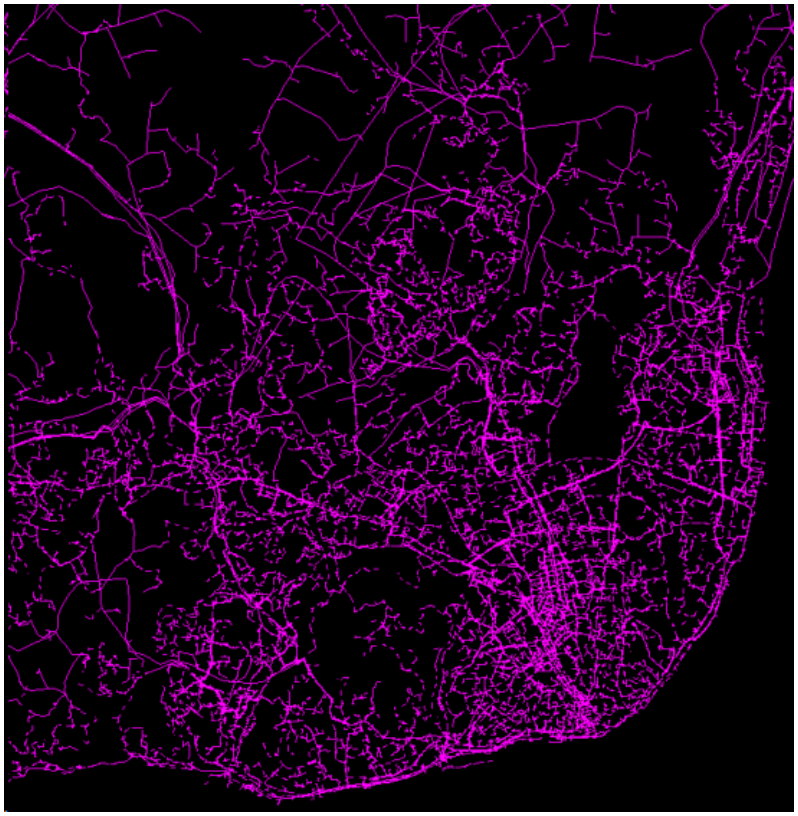

**Figura 106 - Parte de diagrama geográfico**

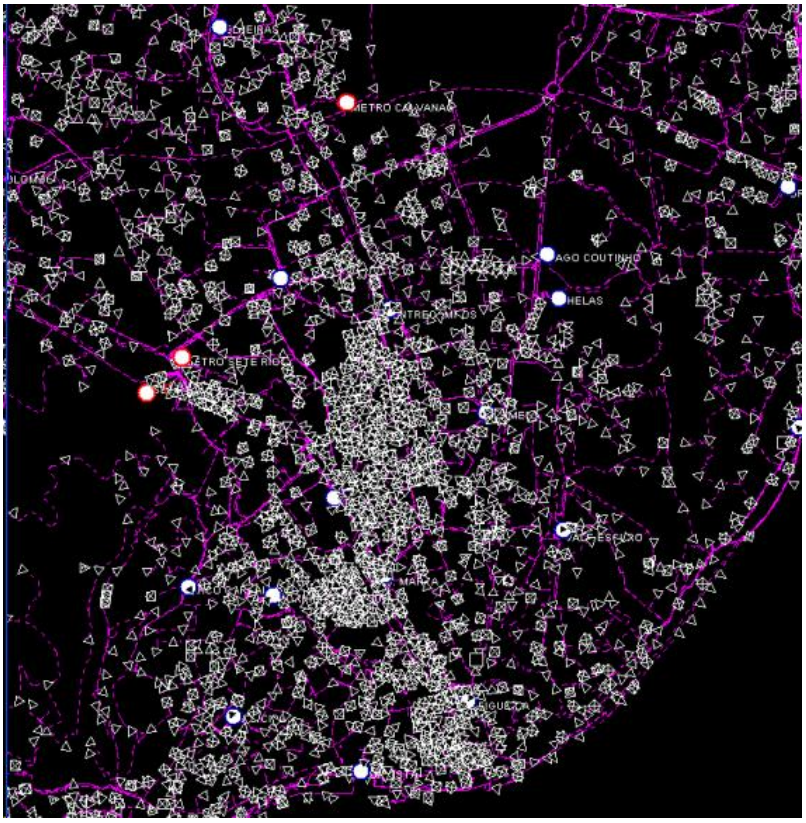


**Figura 107 - Extracto localizado de diagrama Geográfico (Exemplo 1)**

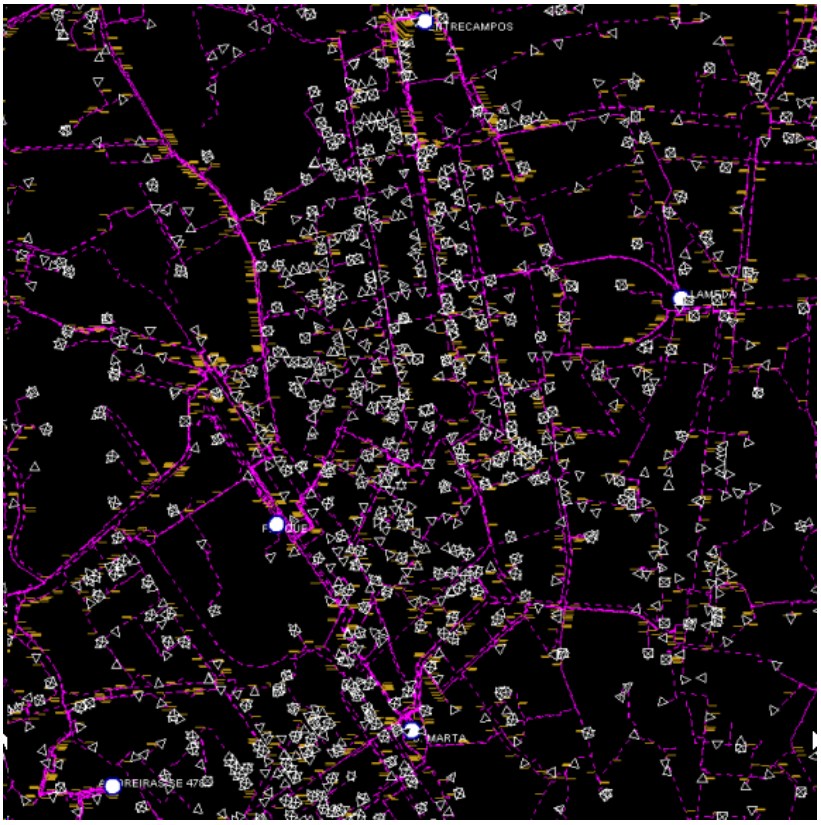

**Figura 108 - Extracto localizado de diagrama Geográfico (Exemplo 2)**

# Anexo B: Ferramentas de auxílio à construção de diagramas esquemáticos

O editor de diagramas esquemáticos alvo de estudo carece de ferramentas de auxílio ao utilizador na construção e visualização de diagramas esquemáticos de redes de distribuição eléctrica.

A inclusão de algumas ferramentas simples de assistência à construção de diagramas poderá permitir ao utilizador aumentar substancialmente a sua produtividade, diminuindo o tempo perdido em tarefas repetitivas.

Outro dos problemas prende-se com a navegação nos diagramas, que por vezes apresentam milhares de componentes. Actualmente não existe qualquer ferramenta de acesso rápido a partes do diagrama o que obriga o utilizador a procurar manualmente locais do diagrama que pretenda visualizar.

Durante o corpo principal da dissertação foram já apresentadas algumas das principais ferramentas a incorporar no editor de diagramas. No entanto algumas ferramentas de interacção bastantes simples mas que não estão presentes no editor, não foram ainda referidas.

Nesta secção, apresentam-se algumas ferramentas elementares de visualização e edição rápida que poderão ajudar o utilizador a aumentar a sua produtividade, procurando minimizar os problemas de navegação existentes e introduzindo algumas ferramentas auxiliares.

## B. 1 Tipos de Ferramenta

As ferramentas propostas, para além das já incluídas no corpo principal da dissertação, dividem-se em dois tipos:

1. Ferramentas de auxílio à navegação no diagrama;
2. Ferramentas de edição rápida.

De acordo com os tipos de ferramenta considerados, na figura seguinte apresentam-se os módulos a considerar nos casos de uso a propor. Todos os casos de uso propostos estarão distribuídos por cada um dos módulos representados.

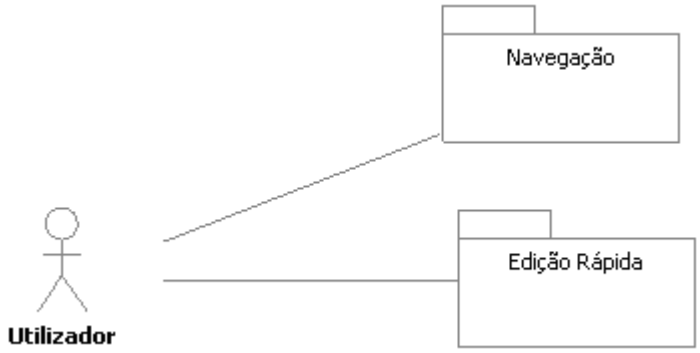


**Figura 109 - Módulos dos casos de uso das ferramentas auxiliares propostas**

Seguidamente serão abordadas propostas de solução para os tipos de ferramenta citada acompanhadas de respectiva exemplificação de funcionamento. Não serão abordados conceitos técnicos de uma possível implementação, apenas os presumíveis resultados da sua aplicação e procedimento a tomar por parte do utilizador (interface) para utilizar as ferramentas.

## B. 1.1 Ferramentas de auxílio à navegação

Entende-se por ferramenta de auxílio à navegação, um determinado utilitário que permita ao utilizador aceder rapidamente uma parte do diagrama que pretenda.

Também se entende como parte deste módulo funcionalidades que permitam colocar o *zoom* em modos óptimos de visualização para o utilizador.

Nesta secção, serão propostas ferramentas que poderão ser úteis no que concerne à navegação e visualização dos diagramas.

A figura seguinte apresenta os casos de uso propostos para o módulo de navegação, que serão descritos de seguida.

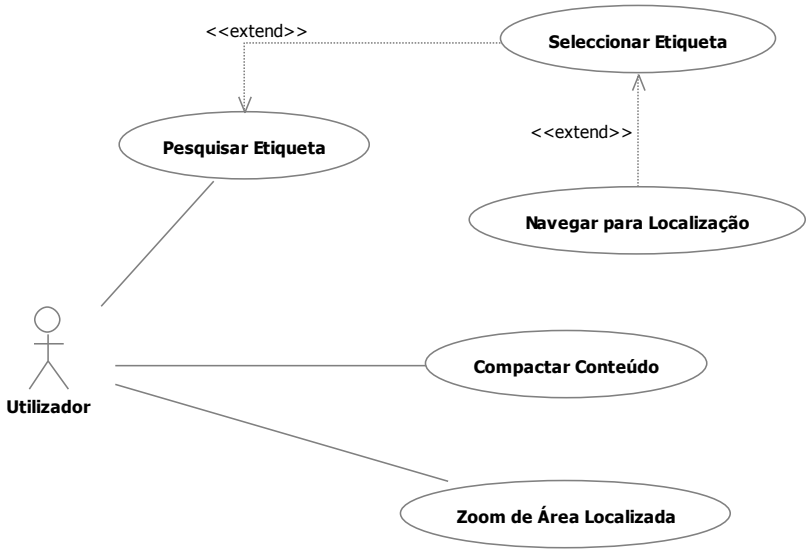


**Figura 110 - Diagrama de casos de uso do módulo de navegação**

### Pesquisa de etiqueta

Uma ferramenta de navegação que seria certamente de grande utilidade, principalmente quando se tratassem diagramas de grandes dimensões, seria mover automaticamente a posição de visualização de um diagrama para um ponto indicado pelo utilizador.

Uma forma possível do utilizador indicar o ponto pretendido seria efectuando uma pesquisa prévia de equipamentos ou localizações, através de texto, em que lhe seriam devolvidos resultados possíveis da localização pretendida, de acordo com as etiquetas de texto presentes no diagrama que contivessem as palavras do campo de pesquisa. Dentro dos resultados possíveis, o utilizador escolheria o pretendido e automaticamente a posição de visualização central do diagrama seria alterada para o ponto escolhido.

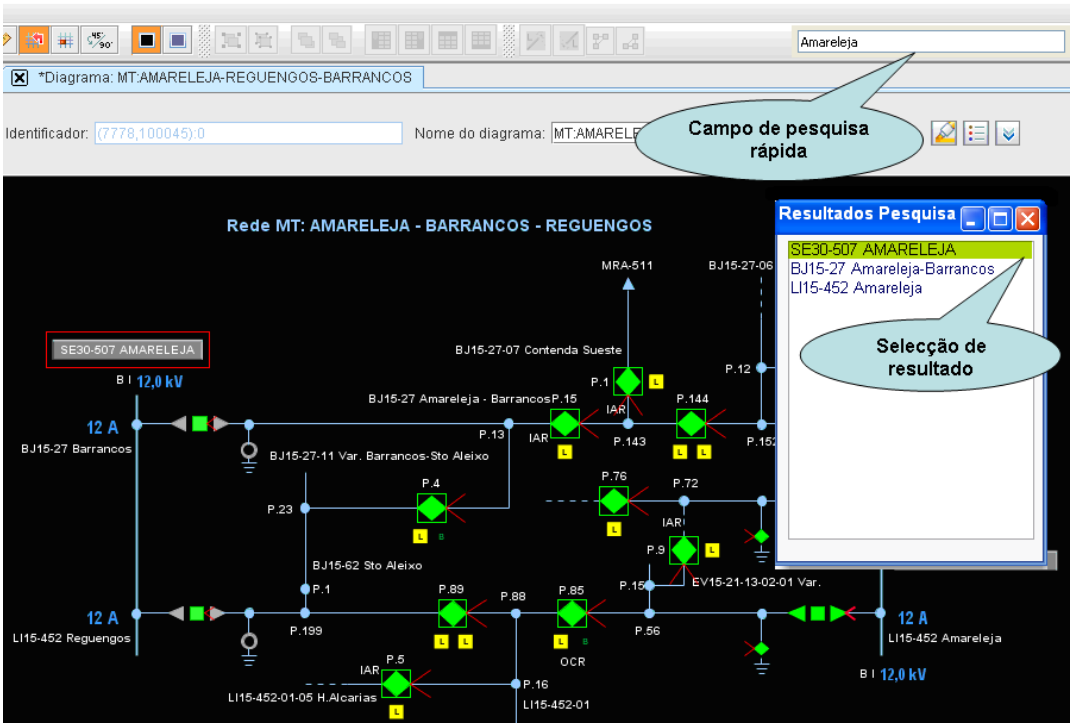


**Figura 111 - Campo de pesquisa rápida**

Na figura anterior pode verificar-se como seria utilizada a ferramenta proposta. Deve ser indicado no campo de pesquisa uma ou mais palavras que possam identificar a zona que se pretende visualizar.

Suponha-se por exemplo que se pretende aceder a zona circundada a vermelho, etiquetada com o nome **SE30-507 AMARELEJA**. Neste caso, poderia pesquisar-se através do campo de pesquisa rápida a entrada **"Amareleja"**, sendo que deveriam ser devolvidos três resultados, representantes das três etiquetas existentes no diagrama com esta entrada presente. Nesse momento, o utilizador poderia escolher o local pretendido na *listbox* apresentada na imagem devendo o ponto central de visualização ser alterado para esse local.

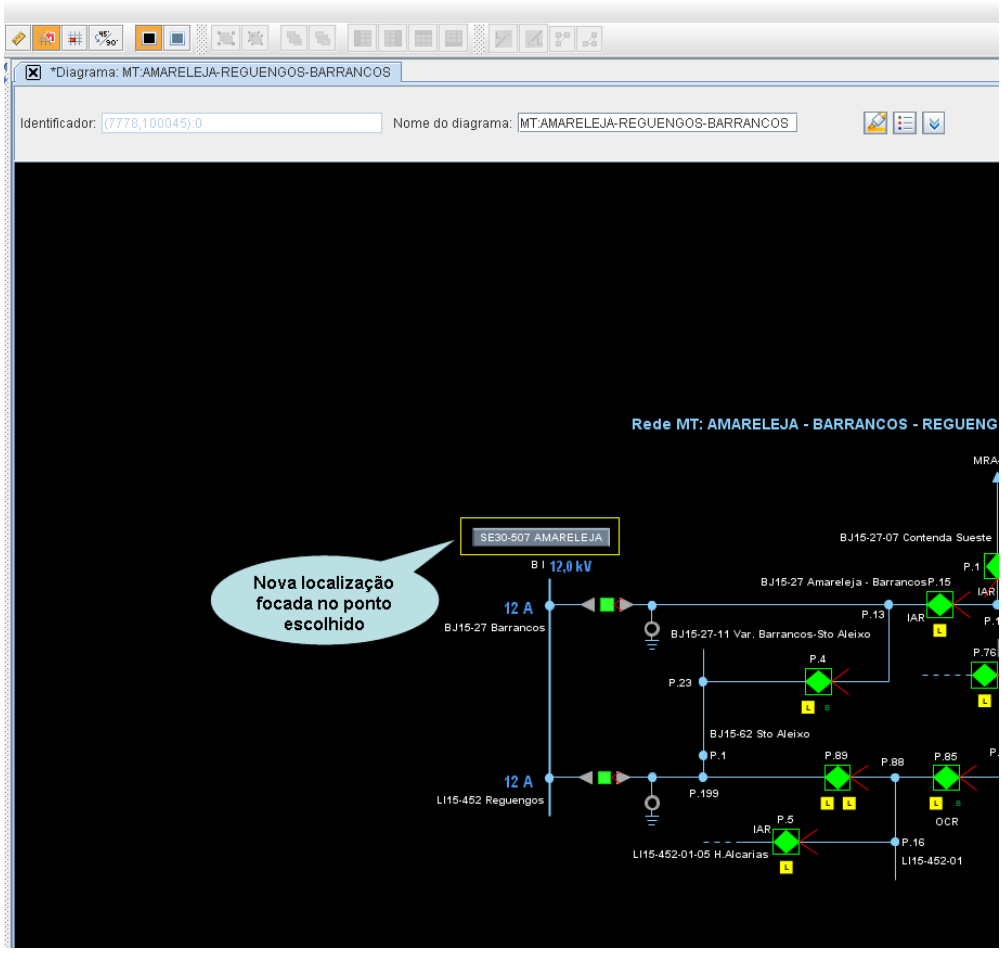


**Figura 112 - Visualização focada no ponto escolhido**

A figura acima expressa o resultado do procedimento efectuado anteriormente. Como se pode verificar foi mudado o ângulo de visualização da imagem, estando neste momento focada no ponto seleccionado, **SE30-507 AMARELEJA**.

## Compactar conteúdo

Tratar-se-ia de uma ferramenta de visualização simples que permitiria colocar o *zoom* do diagrama de tal forma que seria possível ver todo o conteúdo deste na área de visualização, aproveitando o espaço do ecrã disponível da melhor forma possível, dependendo das dimensões do conteúdo do diagrama. Geralmente a ferramenta é conhecida como *fit content to window*, sendo largamente usada noutras aplicações.

A funcionalidade deveria ajustar o *zoom* e centrar o diagrama de forma adequada a proporcionar a melhor visualização possível ao utilizador de todo o diagrama.

Para realizar esta funcionalidade seria necessário unicamente um botão que quando seleccionado, tomaria o procedimento mencionado.

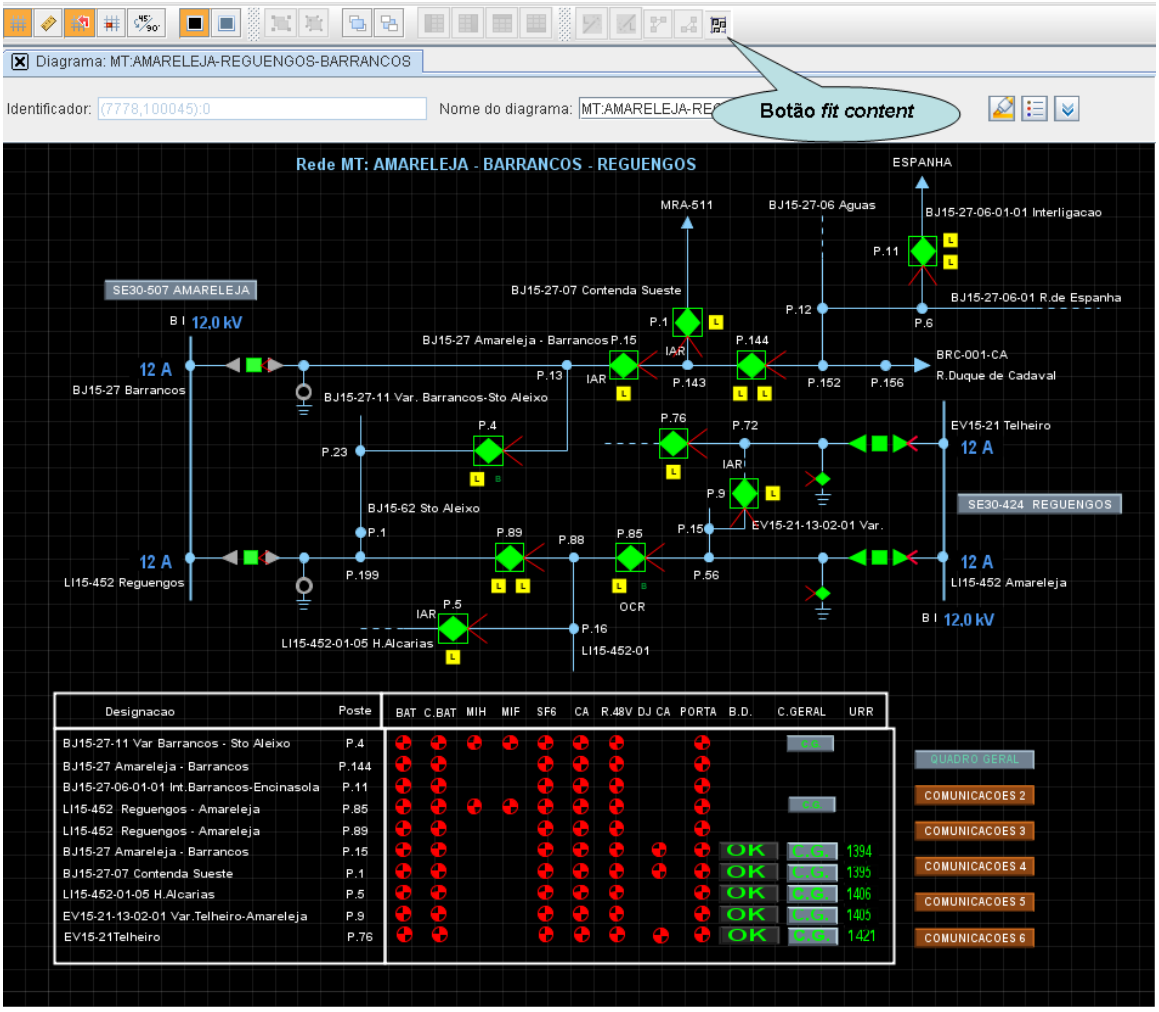


**Figura 113 - Diagrama com conteúdo compactado**

Na figura anterior pode ver-se como seria esperada a visualização num diagrama após aplicação da funcionalidade indicada. Como se verifica, o diagrama encontra-se no *zoom* máximo possível em que é possível ver todo o diagrama, proporcionando a melhor visualização ao utilizador se este pretender ter todo o diagrama visível.

A selecção da ferramenta, de forma a aplicar a funcionalidade mencionada, seria feita através de um novo botão no menu superior, tal como mostrado na figura.

### *Zoom* automático de área localizada

Funcionalidade que permitiria ao utilizador obter uma focagem ideal, ajustada ao ecrã de visualização, de uma área previamente seleccionada por este.

Na prática, o utilizador começaria por seleccionar a opção através de um botão, indicaria a área que pretendia focar através do cursor e obteria uma focagem da área pretendida.

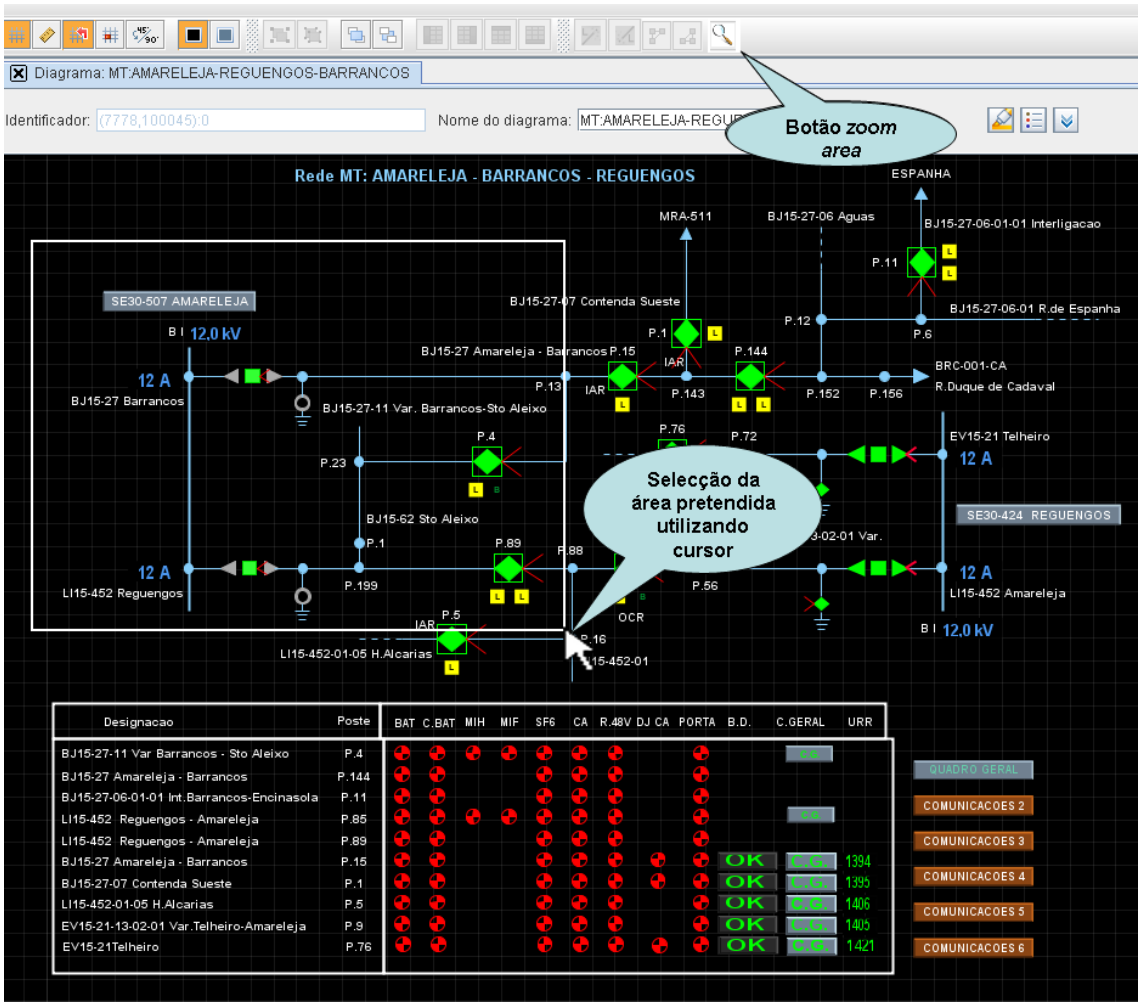


**Figura 114 - Método de utilização da funcionalidade de *zoom* automático a área seleccionada**

A figura acima exemplifica como se procederia para utilizar a ferramenta proposta. O utilizador começaria por seleccionar a funcionalidade no menu superior, seguidamente deveria desenhar um rectângulo na área de visualização do diagrama da área que pretendesse visualizar, utilizando o cursor.

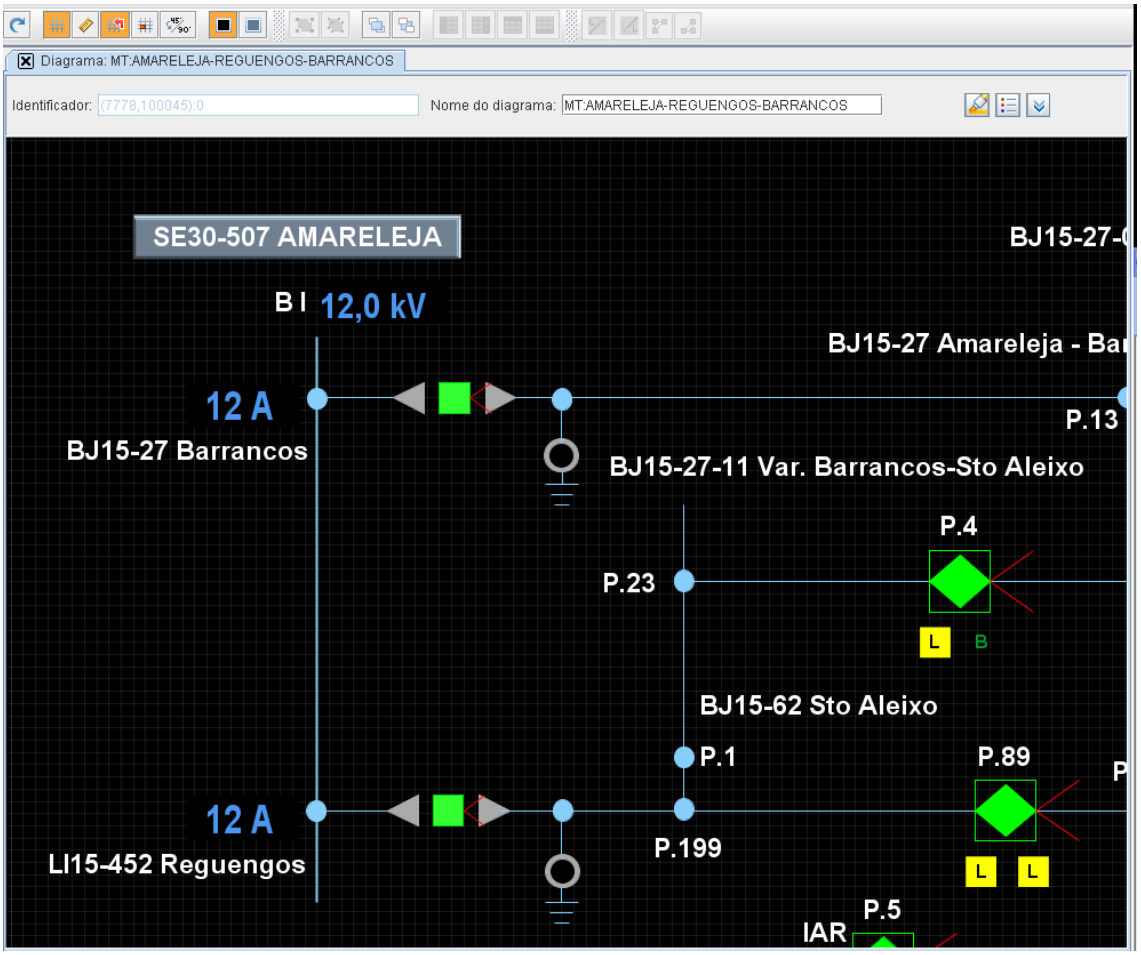


**Figura 115 - Presumível resultado da aplicação da funcionalidade à área seleccionada**

A figura acima é uma aproximação do resultado previsto após aplicação de *zoom* à área indicada na figura 114. Como se pode verificar a área de visualização visível passou a ser a área indicada pelo utilizador através do cursor.

É de crer que esta funcionalidade seja de grande utilidade no auxílio à navegação em diagramas, tendo sido já testada, com receptividade por parte dos utilizadores noutras aplicações.

## B. 1.2 Ferramentas de Edição Rápida

Embora as funcionalidades básicas de edição de diagramas estejam já disponíveis na aplicação, esta carece ainda de algumas ferramentas úteis deste tipo. De entre as funcionalidades já existentes encontram-se a área de transferência (*clipboard*) e voltar atrás/avançar (*undo* e *redo*).

Ainda assim, foram identificadas funcionalidades que poderão ser úteis para efectuar edições rápidas ao diagrama. Certas funcionalidades já existentes parecem carecer de melhorias, algumas também identificadas. O diagrama da figura seguinte apresenta os casos de uso que se propõe introduzir.

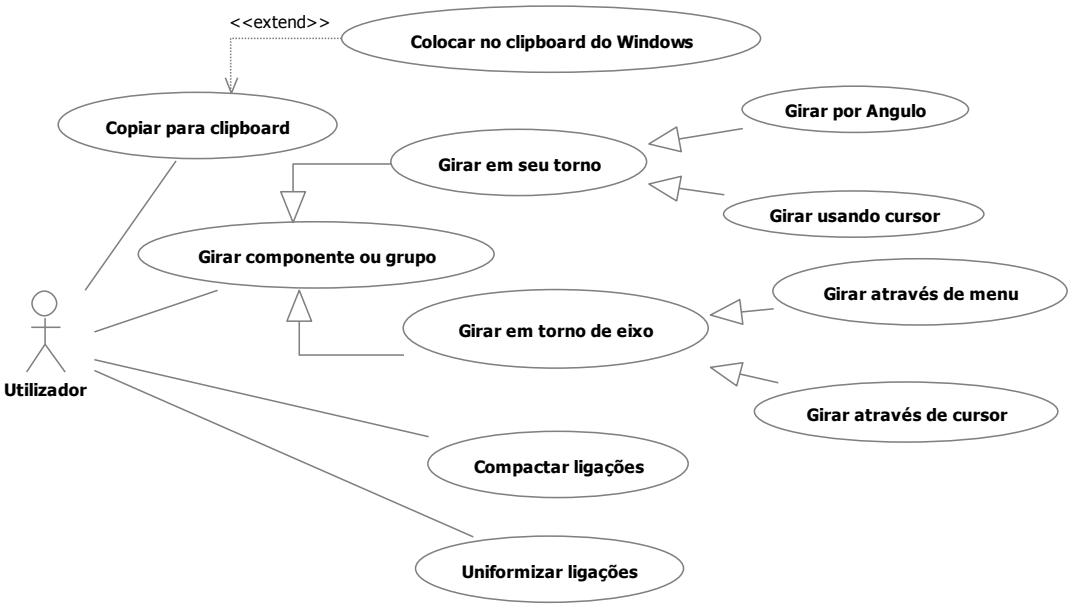


**Figura 116 - Diagrama de Casos de uso do módulo de Edição Rápida**

As funcionalidades propostas devem aumentar a produtividade ao utilizador, evitando que perca algum tempo com certas tarefas que poderão ser feitas rapidamente com acesso às ferramentas indicadas.

A informação detalhada da aplicação prática de cada uma das ferramentas expostas no diagrama de casos de uso da figura anterior encontra-se de seguida.

### Copiar para *clipboard*

A funcionalidade de copiar dados para a área de transferência já existe, no entanto apenas pode ser aplicada internamente na aplicação. Seria útil para o utilizador poder colocar na área de transferência do seu sistema operativo, em vez de apenas na da própria aplicação, partes do diagrama que pudesse colar facilmente como imagens noutras aplicações. A ferramenta seria ainda mais útil se a aplicação pudesse distinguir caixas de texto e até tabelas dos restantes objectos, o que poderia permitir ao utilizador editar os dados mesmo quando os colasse noutras aplicações que também interagissem com a área de transferência do sistema operativo.

Este utilitário seria particularmente útil quando o utilizador pretendesse produzir documentação sobre os seus diagramas, que certamente seria desenvolvida numa aplicação exterior ao editor.

Repare-se ainda que grande parte das aplicações de edição de imagem, diagramas ou mesmo texto detêm esta funcionalidade pelo que esta ferramenta representaria uma forma de "ligação com o mundo".

### Girar componente ou grupo de componentes

A funcionalidade de rotação de um objecto em torno de si mesmo já existe na aplicação. No entanto é de crer que ainda não estão exploradas todas as formas de interacção que podem facilitar este processo.

Actualmente não é possível a rotação de um grupo de objectos, apenas de cada um individualmente, utilizando exclusivamente o cursor. Além disso o objecto pode rodar apenas em torno de si mesmo, não sendo possível a rotação em torno de um eixo, o que por vezes facilita a edição. As ferramentas de que se irá falar procuram acrescentar estas possibilidades à aplicação.

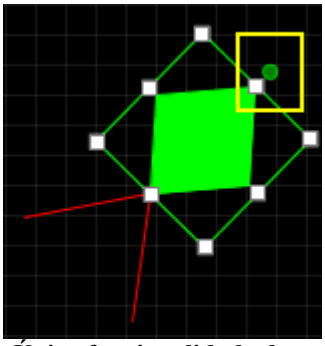

**Figura 117 - Única funcionalidade de rotação actual**

A figura anterior apresenta a única funcionalidade de rotação actualmente disponível. Pressionando-se utilizando o cursor o ponto verde rodeado a amarelo, pode efectuar-se a rotação do objecto.

## Girar componente ou grupo em torno de si mesmo

Esta funcionalidade de rotação encontra-se já implementada para o caso de um único objecto como exemplificado na figura 117. Não se encontra no entanto disponível para um grupo de objectos. Nesta funcionalidade ocorre a rotação de um objecto em torno do seu centro.

A funcionalidade extra a propor neste caso seria unicamente garantir a mesma possibilidade para um grupo seleccionado de objectos. Outra melhoria a fazer seria possibilitar a rotação através de introdução de um valor numérico (tipicamente em graus) em vez de o permitir unicamente utilizando o cursor.

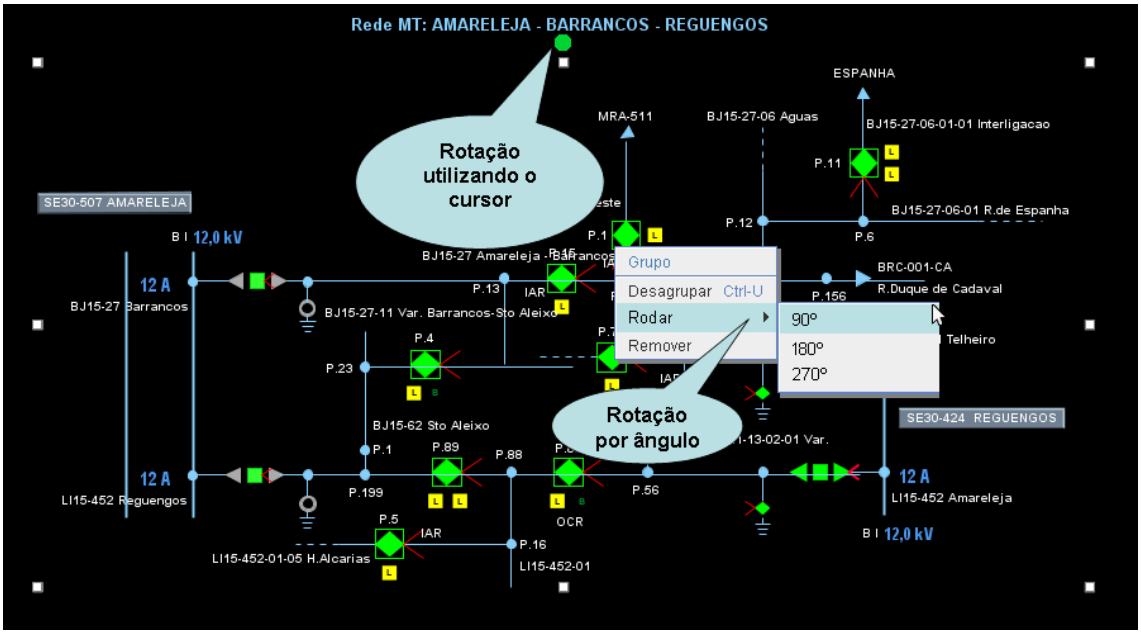


**Figura 118 - Protótipo de novas funcionalidades de rotação**

Na figura anterior apresenta-se a forma como seria utilizada a rotação de um grupo de objectos sobre si mesmos, utilizando o cursor ou através de um ângulo. A rotação indicando um ângulo deveria também ser aplicável da mesma forma sobre um objecto simples. Para efectuar a rotação indicando um ângulo, o utilizador deveria dispor de uma nova entrada no menu.

Esta funcionalidade está normalmente presente na maioria das ferramentas de edição de diagramas e imagens, podendo revelar-se útil para efectuar mudanças rápidas em partes do diagrama.

### Girar componente ou grupo em torno de um eixo (inverter)

Uma ferramenta que permitisse a rotação em torno de um eixo poderia também ser útil na edição de partes do diagrama. Não existe nenhuma funcionalidade semelhante na aplicação actual. Trata-se no entanto de um utilitário frequentemente encontrado em aplicações de edição de diagramas e imagem.

Esta funcionalidade estaria restringida à rotação sobre os eixos horizontal e vertical e permitiria no fundo a inversão horizontal ou vertical, mediante o eixo de rotação de um objecto ou um grupo de objectos do diagrama.

A inversão horizontal representaria assim a rotação sobre o eixo vertical e vice-versa.

De seguida apresenta-se um modelo de funcionamento da ferramenta proposta, incluindo resultados previstos.

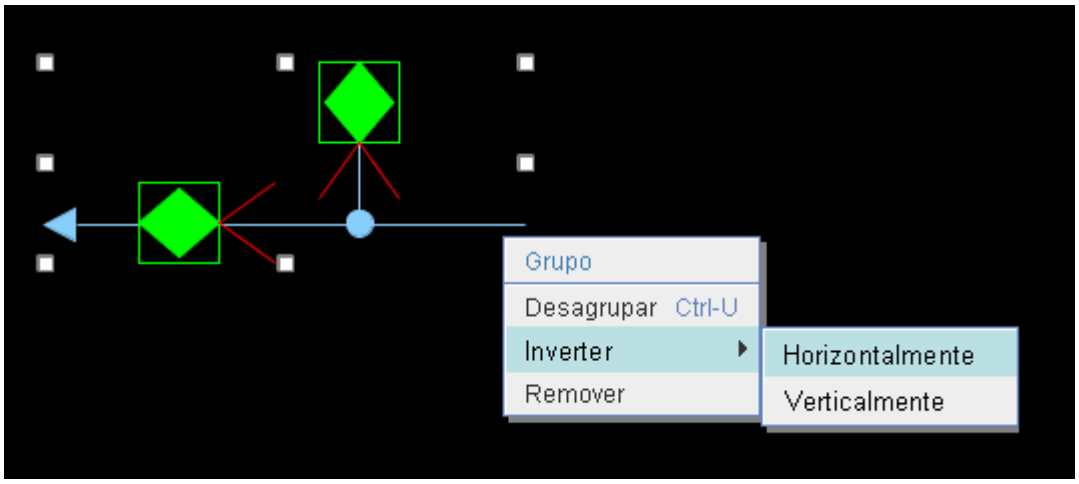


**Figura 119 - Menu de inversão sobre um grupo de objectos**

Na figura anterior apresenta-se o menu que permitiria ao utilizador rodar um grupo de objectos em torno dos eixos vertical e horizontal. Para aplicar a mesma ferramenta a um componente simples, deveria dispor-se de um menu semelhante.

Os resultados de uma inversão horizontal e vertical sobre o grupo de objectos da figura 119 encontram-me respectivamente nas figuras 120 e 121.

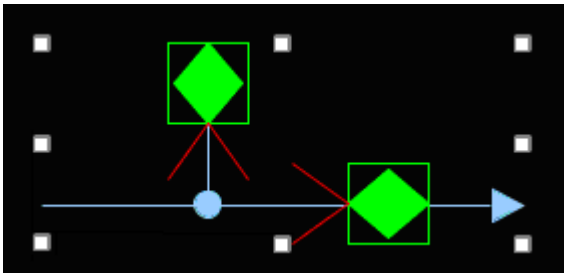

**Figura 120 - Inversão horizontal de um grupo de objectos**

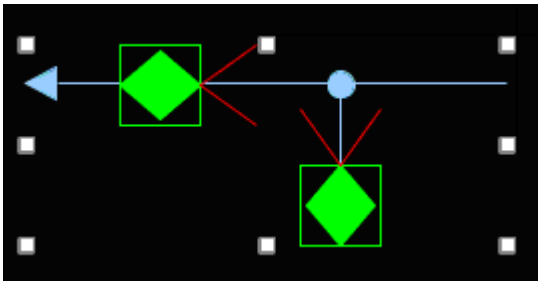

**Figura 121 - Inversão vertical de um grupo de objectos**

Outra alternativa para realizar a mesma operação utilizando unicamente o cursor pede ser vista na figura 122. Este método consistiria em utilizar a ferramenta de redimensionamento de um componente ou um grupo para inverter o mesmo, o que permitiria inversões rápidas aos componentes.

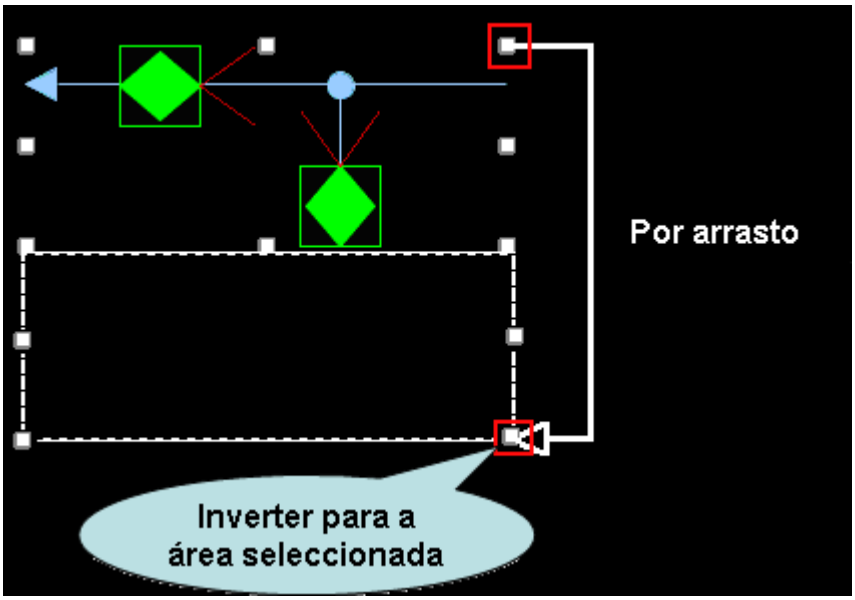


**Figura 122 - Inversão de um grupo de componentes utilizando o cursor**

O exemplo da figura anterior, faria com que a imagem fosse invertida (verticalmente) e ao mesmo tempo redimensionada para a área indicada pelo utilizador. Um procedimento semelhante poderia ser aplicado para inverter a imagem horizontalmente.

Uma previsão do procedimento tomado encontra-se na figura 123. Repare-se que nesta figura o grupo além de estar invertido se encontra mais achatado, resultante do redimensionamento que ocorreu simultaneamente com a inversão.

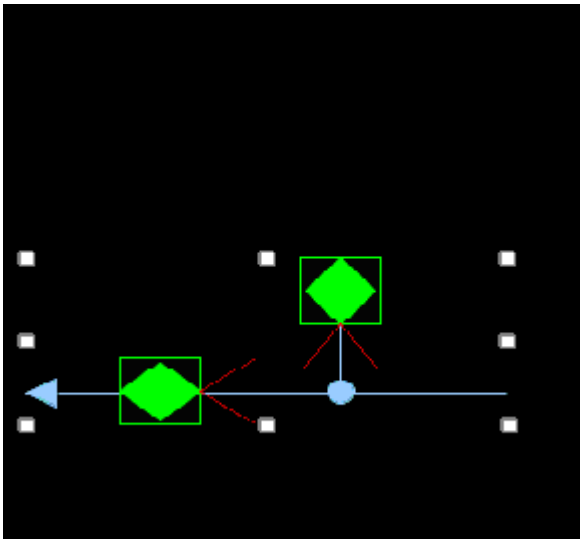

**Figura 123 - Resultado da inversão e redimensionamento do grupo seleccionado**

### Compactar conjunto de ligações em selecção

A funcionalidade de diminuir o tamanho das ligações num conjunto de componentes seleccionados seria de interesse para o utilizador conseguir fazer uma boa gestão do espaço disponível para desenho, ajudando-o muitas vezes a libertar espaço. Uma funcionalidade para compactar ligações passaria por uma ferramenta que permitisse redimensionar um conjunto de ligações seleccionadas. Desta forma estaria disponível não apenas um cenário de diminuição do tamanho das ligações, mas também um aumento.

Quando objectos seleccionados actuam em grupo, existe já a possibilidade de redimensionar o tamanho das ligações, no entanto, tal processo implica também a mudança de tamanho dos restantes componentes do grupo (por exemplo equipamentos ou etiquetas de texto). Não existe, no entanto, nenhuma funcionalidade que permita redimensionar exclusivamente o tamanho das ligações.

Uma ferramenta deste tipo poderia ser aplicada aquando da selecção de um conjunto de componentes.

A proposta de interacção para a funcionalidade, actuando sobre um conjunto de objectos escolhidos, ilustra-se na figura 124. Repare-se na adição de uma nova funcionalidade, que permite o redimensionamento de ligações utilizando o cursor.

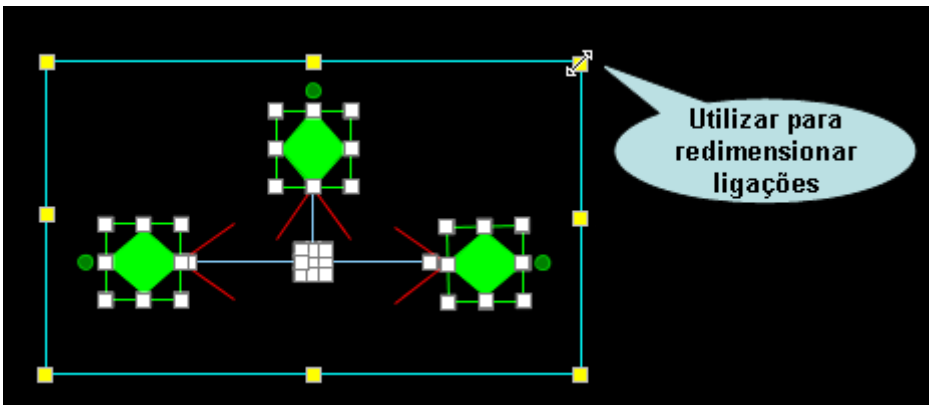


**Figura 124 - Método proposto para redimensionar ligações**

### Uniformizar distância entre componentes num barramento

A funcionalidade serviria para num dado barramento previamente seleccionado se aplicar com uma acção simples uma uniformização a todos os componentes que permitisse que ficassem dispostos a distâncias equidistantes.

A figura 125 apresenta o menu de interacção que permitiria efectuar a acção num dado barramento e a figura 126 os resultados dessa acção.

Refira-se que cada componente seria arrastado com todas as suas ligações até um determinado nível de profundidade. No entanto, esta ferramenta estaria mais indicada para barramentos onde os seus componentes não contivessem vários equipamentos ligados (tivesse pouco nível de profundidade).

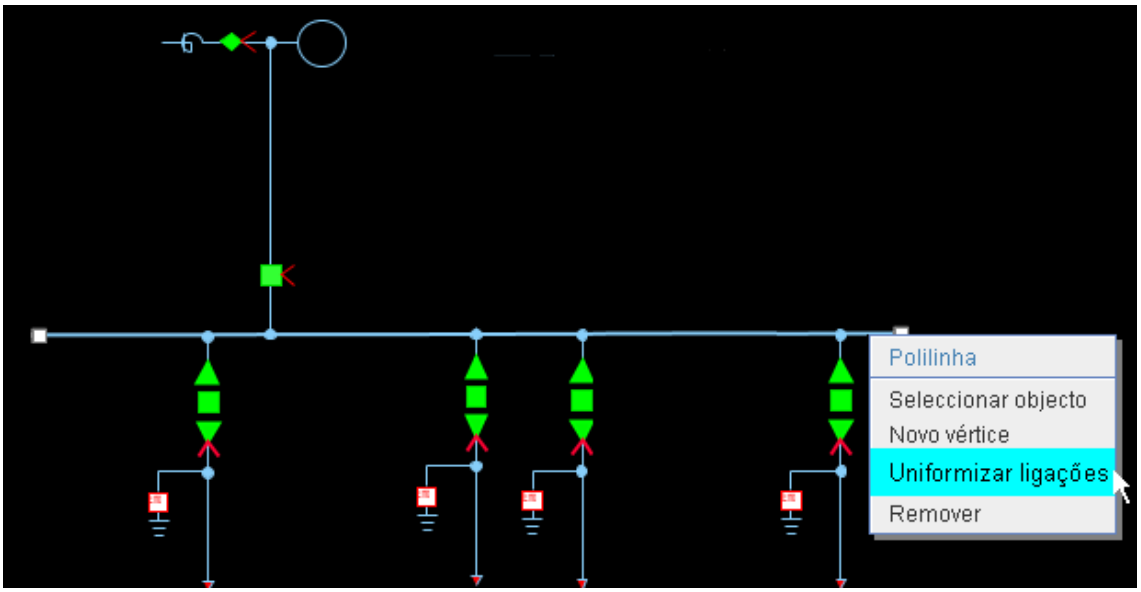


**Figura 125 - Menu sobre um barramento que permite uniformizar as distâncias entre componentes**

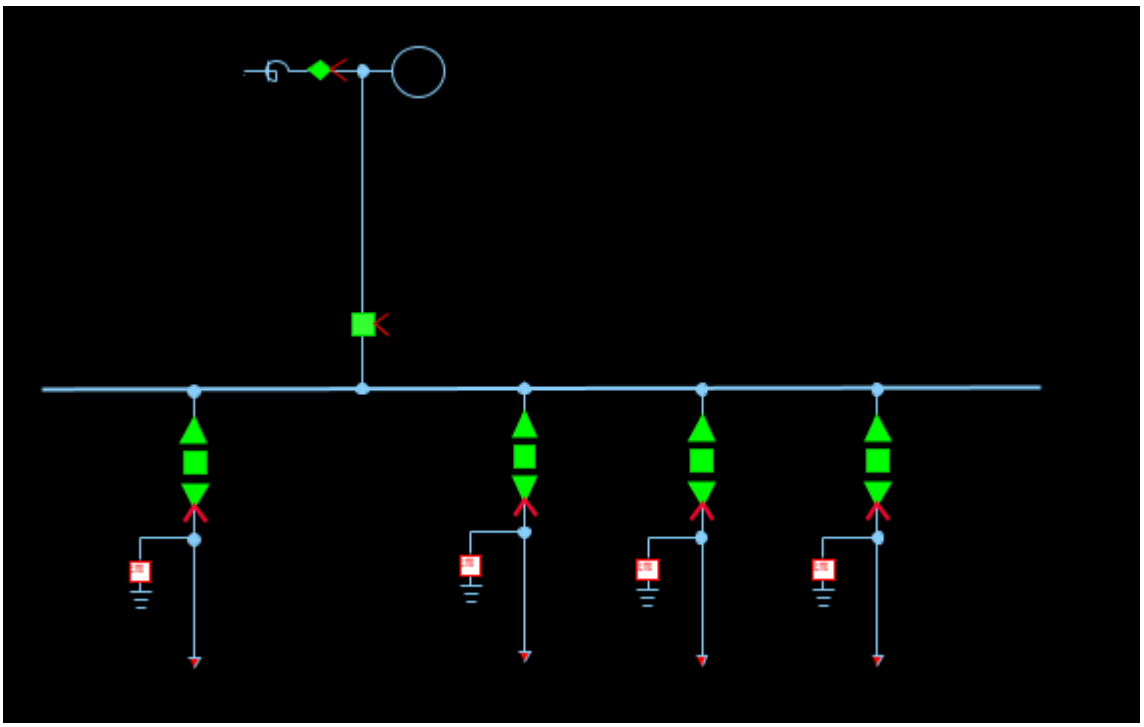

**Figura 126 - Resultado da aplicação da funcionalidade de uniformização sobre um barramento**